

Ciencia de Datos

TÉCNICAS ANALÍTICAS Y APRENDIZAJE ESTADÍSTICO
EN UN ENFOQUE PRÁCTICO

Jesús García - José M. Molina - Antonio Berlanga -
Miguel Á. Patricio - Álvaro L. Bustamante - Washington R. Padilla

BIG DATA



Alfaomega
Empresas del Grupo

Colombia

Alfaomega Colombiana S.A.
Calle 62 N° 20-46 esquina, Bogotá
Teléfono (57-1) 746 0102
cliente@alfaomegacolombiana.com

México

Alfaomega Grupo Editor S.A. de C.V.
Calle Doctor Olvera N° 74, Colonia Doctores,
Delegación Cuauhtemoc Ciudad de México
C.P. 06720 · Teléfono (52-55) 5089 7740
Fax (52-55) 5575 2420 - 5575 2420
Sin costo 01-800-020-4396
libreriapitagoras@alfaomega.com.mx

Argentina

Alfaomega Grupo Editor Argentino S.A.
Av. Córdoba 1215 piso 10
Capital Federal, Buenos Aires
Teléfono/Fax: (54-11) 4811 7183 / 8352 / 0887
ventas@alfaomegaeditor.com.ar

Chile

Alfaomega Grupo Editor S.A.
Av. Providencia 1443. Oficina 24, Santiago
Teléfonos (56-2) 2235 4248 / 2947 9351 / 2235 5786
agechile@alfaomega.cl

www.alfaomega.com.co

Publicaciones Altaria, S.L.
Tarragona, España
Teléfono (+34) 93 516 19 66
info@altariaeditorial.com

Ciencia de datos. Técnicas analíticas y aprendizaje estadístico
Bogotá, mayo de 2018

© Jesús García, José M. Molina, Antonio Berlanga, Miguel
A. Patricio, Álvaro L. Bustamante y Washington R. Padilla
© Alfaomega Colombiana S.A. 2018
© De la edición: PUBLICACIONES ALTARIA, S.L.

Derechos reservados. Esta publicación no puede ser
reproducida total ni parcialmente. No puede ser registrada
por un sistema de recuperación de información, en ninguna
forma ni por ningún medio, sea mecánico, fotoquímico,
electrónico, magnético, electroóptico, fotocopia o cualquier
otro, sin el previo permiso escrito de la editorial.

Revisado por: Sonia Vives y Carlos Martínez

ISBN: 978-958-778-425-1 (Edición Colombia)
ISBN: 978-84-947319-6-9 (Edición España)

Impreso en Colombia
Printed and made in Colombia

Ciencia de datos

Técnicas analíticas y aprendizaje estadístico

Un enfoque práctico

Jesús García

José M. Molina, Antonio Berlanga, Miguel A. Patricio, Álvaro
L. Bustamante y Washington R. Padilla



Prefacio

La ciencia de datos es hoy en día la herramienta fundamental para la explotación de datos y la generación de conocimiento. Entre los objetivos que persigue se encuentra la búsqueda de modelos que describan patrones y comportamientos a partir de los datos con el fin de tomar decisiones o hacer predicciones. Es un área que ha experimentado un enorme crecimiento al extenderse el acceso a grandes volúmenes de datos e incluso su tratamiento en tiempo real, requiriendo de técnicas sofisticadas que puedan tratar con los problemas prácticos como escalabilidad, robustez ante errores, adaptabilidad con modelos dinámicos, etc. Abarca a numerosos grupos de investigación de diferentes áreas (computación, estadística, matemáticas, ingeniería, etc.) que trabajan en la propuesta de nuevos algoritmos, técnicas de computación e infraestructuras para la captura, almacenamiento y procesamiento de datos, etc.

El objetivo del libro es proporcionar una visión general de las principales técnicas de la ciencia de datos y de las aplicaciones que las implementan, permitiendo comprender los conceptos fundamentales sobre los que se basan y los resultados al aplicarlas a problemas reales. Existen muchos libros que se centran en aspectos teóricos de técnicas específicas, y otros que mantienen un nivel introductorio omitiendo detalles para centrarse en las aplicaciones. Este libro busca un equilibrio entre estos extremos: se lleva a cabo una presentación que permita comprender los fundamentos de cada familia de técnicas y simultáneamente desarrolla ejemplos de datos reales y el análisis de resultados con las diferentes técnicas presentadas.

Se parte de las técnicas clásicas basadas en modelos probabilísticos, su aplicación en los problemas de regresión y clasificación, para presentar después su generalización mediante la incorporación del aprendizaje automático en diferentes familias de técnicas. Los últimos capítulos presentan dos áreas de creciente interés por las particularidades de los datos cuando forman series temporales, como es el caso de sensores en la Internet de las cosas, o datos geolocalizados que permiten hacer regresión espacial. El libro presenta una estructura combinada en la que se van alternando la presentación de los fundamentos teóricos con su aplicación en un problema real utilizando la implementación en el entorno R. Existe actualmente una gran pugna entre varios lenguajes en los que implementar los algoritmos que manipularán y extraerán la información de los datos. Sin lugar a dudas, los más destacados con diferencia, son Python y R frente a alternativas como Julia, C++, Java, Scala o MATLAB. El flujo del proceso típico de análisis de datos es aplicar una técnica o algoritmo una sola vez. Esto significa que se ha creado una regla general entre los científicos de datos a la hora de decidir el lenguaje a utilizar. Para un proceso que se va a ejecutar una sola vez, ¿qué es mejor: utilizar un lenguaje que puede llevar treinta minutos en desarrollarlo y ejecutarlo en diez segundos o usar un lenguaje en el que se tarda diez minutos en desarrollar y un minuto en ejecutar? Éste es el motivo por el que lenguajes de alto rendimiento, como C++, no tienen un

gran uso en este contexto de problemas. La diferencia entre Python y R es más sutil. R es un lenguaje de dominio específico, es decir, orientado a un uso específico, el análisis estadístico y, por tanto, su construcción está pensada y dirigida a ese uso. Sin embargo, Python es un lenguaje de propósito general, más rápido y seguro que R, y cualquier algoritmo que no sea del ámbito del análisis de datos será más fácil de desarrollar. Pero, para el análisis estadístico y de datos, R es imbatible, ningún lenguaje posee la variedad de librerías para tratar, analizar y representar datos, ni una comunidad tan activa que le permite actualizarse para ir incorporando las últimas técnicas del estado del arte en el análisis y representación de datos.

Se ha intentado centrar la aplicación con un único ejemplo que se desarrolla en la mayor parte del libro, el análisis de calidad del aire en la ciudad de Madrid. Hay varias razones que han motivado esta decisión. Hemos querido ilustrar la aplicación de las diferentes técnicas sobre un conjunto de datos que nadie antes haya utilizado, debido a que se acerca a la situación real a la que el futuro científico de datos se enfrentará. Tener que realizar la exploración de cero, sin conocimiento previo de qué se va a encontrar. Enfrentarse a la tarea de preparar los datos y realizar las exploraciones estadísticas que permiten atisbar las técnicas que después han de usarse. Aplicar diferentes algoritmos, algunos no aportarán resultados, otros funcionarán y entonces habrá que ajustar sus parámetros y analizar y representar los resultados. Ésta será la rutina habitual de un científico de datos. Habrá algoritmos que, al aplicarlos a estos datos, no arrojarán buenos resultados. Esto no es un fracaso, entender por qué un algoritmo no es adecuado puede proporcionar una información muy valiosa acerca de la información que contienen y cuáles deben ser los siguientes pasos de análisis que deben seguirse. Finalmente, también se quiere poner de manifiesto la utilidad que puede suponer que se liberen datos al escrutinio público mediante iniciativas *open data*.

Por último, el libro refleja la actividad docente e investigadora de los autores, miembros del grupo de investigación GIAA (Grupo de Inteligencia Artificial Aplicada) de la Universidad Carlos III de Madrid, en las líneas de investigación de aprendizaje automático, análisis de datos y optimización para el apoyo a la toma de decisiones en entornos complejos. Esperamos que el texto sea de utilidad para los lectores y alumnos que se acerquen a la ciencia de datos, y en particular para aquellos que tengan interés en aplicar los conocimientos teóricos en el campo de la práctica.

Los autores

Índice general

Prefacio	5
----------------	---

Capítulo 1

Introducción	13
1.1 Introducción.....	13
1.2 Los datos.....	19
1.3 Etapas en los procesos de <i>big data</i>	20
1.4 Minería de datos.....	21
1.5 Estructura de un proyecto de análisis de datos..	22
1.6 Aplicaciones	25
1.6.1 <i>Marketing</i>	25
1.6.2 Compañías de seguros.....	26
1.6.3 Banca	26
1.6.4 Telecomunicaciones	27
1.6.5 Medicina	27
1.6.6 Industria farmacéutica.....	27
1.6.7 Biología	28
1.6.8 Minería de textos	28
1.6.9 Minería de datos web	29
1.6.10 Redes sociales.....	30
1.7 Modelos y tareas	31
1.7.1 Tareas descriptivas	32
1.7.1.1 Agrupamiento	32
1.7.1.2 Correlaciones y factorizaciones	32
1.7.1.3 Reglas de asociación.....	33
1.7.1.4 Dependencias funcionales.....	33
1.7.2 Tareas predictivas.....	34
1.7.2.1 Clasificación.....	34
1.7.2.2 Clasificación suave	35
1.7.2.3 Categorización	35
1.7.2.4 Preferencias o priorización.....	35
1.7.2.5 Regresión	36
1.8 Métodos y técnicas.....	36
1.8.1 Técnicas algebraicas y estadísticas.....	36
1.8.2 Técnicas bayesianas	37
1.8.3 Técnicas basadas en conteos de frecuencias y tablas de contingencia	37

1.8.4 Técnicas basadas en árboles de decisión y sistemas de aprendizaje de reglas.....	37
1.8.5 Técnicas relacionales, declarativas y estructurales.....	37
1.8.6 Técnicas basadas en redes neuronales artificiales	37
1.8.7 Técnicas basadas en núcleo y máquinas de soporte vectorial	38
1.8.8 Técnicas estocásticas y difusas	38
1.8.9 Técnicas basadas en casos, en densidad o distancia	38

Capítulo 2

Análisis estadístico de datos.....39

2.1 Introducción.....	39
2.2 Análisis de una variable. Estadística descriptiva e inferencia.....	40
2.2.1 Estadísticos de variable continua.....	41
2.2.2 Histograma.....	42
2.2.3 Estadísticos de variables nominales	44
2.3 Contrastes de hipótesis	46
2.3.1 Distribuciones de probabilidad.....	46
2.3.1.1 Distribución normal	47
2.3.2 Inferencia	49
2.3.3 Evaluación de hipótesis.....	51
2.4 Análisis de relaciones entre variables. Evaluación de hipótesis.....	54
2.4.1 Relación entre variables nominales-nominales.....	54
2.4.2 Relaciones numéricas-nominales.....	56
2.4.2.1 Comparación de dos medias.....	56
2.4.2.2 Análisis de la varianza	58
2.4.3 Relaciones numéricas-numéricas.....	61

Capítulo 3

Introducción al lenguaje R. Lectura, procesamiento y visualización de datos:

***data wrangling*.....63**

3.1 Carga y transformaciones de datos	63
3.1.1 Estructura básica de datos	64
3.1.2 Lectura de fichero.....	65

3.2 Estadística descriptiva	68
3.2.1 Variables categóricas.....	84
3.2.2 Correlación	88
3.2.2.1 Visualización	88
3.2.3 Test de hipótesis	97
3.2.4 Representación de datos.....	100

Capítulo 4

Predicción y clasificación con técnicas

numéricas.....117

4.1 Técnicas numéricas de predicción.....	117
4.1.1 Regresión lineal	117
4.1.1.1 Regresión lineal simple.....	118
4.1.1.2 Regresión lineal múltiple.....	119
4.1.1.3 Regresión lineal ponderada localmente	121
4.1.1.4 Atributos nominales	123
4.1.2 Evaluación del modelo de regresión	124
4.1.2.1 Error de regresión y selección de variables	126
4.1.3 Regresión no lineal.....	130
4.1.3.1 Transformaciones sencillas	131
4.1.3.2 Otras transformaciones.....	133
4.1.4 Ejemplos de regresión lineal.....	133
4.2 Técnicas numéricas de clasificación	136
4.2.1 Clasificación mediante regresión lineal	138
4.2.2 Clasificación mediante regresión logística	139
4.2.3 Clasificación bayesiana.....	141
4.2.3.1 Clasificación bayesiana de atributos numéricos	141
4.2.3.2 Clasificación bayesiana con atributos nominales.....	146
4.2.4 Ejemplos de clasificación bayesiana	147

Capítulo 5

Predicción y clasificación con R153

5.1 Regresión.....	153
5.1.1 Regresión lineal	153
5.1.2 Selección de atributos	165
5.1.3 Regresión no lineal.....	169
5.1.4 Regresión de atributos no continuos	172
5.1.5 Modelos lineales generalizados	179

5.2 Algoritmos de clasificación.....	185
5.2.1 Detección de valores atípicos.....	186
5.2.2 LDA, <i>Linear Discriminant Analysis</i>	195
5.2.3 Clasificadores probabilísticos.....	200
5.2.3.1 <i>Naïve</i> bayesiano.....	201
5.2.3.2 Redes bayesianas.....	202

Capítulo 6

Técnicas de minería de datos.....205

6.1 Técnicas de minería de datos	205
6.2 <i>Clustering</i>	207
6.2.1 <i>Clustering</i> numérico (k-medias).....	209
6.2.2 <i>Clustering</i> conceptual (COBWEB).....	210
6.2.3 <i>Clustering</i> probabilístico (EM)	214
6.3 Reglas de asociación	217
6.4 Predicción numérica	220
6.4.1 Predicción no lineal con árboles de regresión.....	220
6.4.2 Estimador de núcleos.....	225
6.4.2.1 Aplicación a problemas multivariantes.....	228
6.4.2.2 Aplicación a problemas de clasificación.....	229
6.5 Clasificación	231
6.5.1 Tabla de decisión	231
6.5.2 Árboles de decisión.....	233
6.5.3 Reglas de clasificación	245
6.5.4 Clasificación bayesiana.....	251
6.5.5 Aprendizaje basado en ejemplares	257
6.5.5.1 Algoritmo de los k-vecinos más próximos.....	258
6.5.5.2 Algoritmo k-estrella.....	260
6.5.5.3 Probabilidad de transformación para los atributos permitidos.....	261
6.5.5.4 Combinación de atributos	262
6.5.5.5 Selección de los parámetros aleatorios	262
6.5.5.6 Clasificación de un ejemplo	264
6.5.6 Máquinas de vectores de soporte (SVM)	265
6.5.6.1 SVM lineal.....	266
6.5.6.2 SVM lineal de margen blando (<i>soft margin</i>)	270
6.5.6.3 SVM no lineal. Funciones <i>kernel</i>	272
6.5.6.4 Clasificación multiclase	276
6.5.7 Redes de neuronas	277
6.5.7.1 Estructura de las redes de neuronas.....	278
6.5.7.2 Proceso de entrenamiento (retropropagación).....	279
6.5.8 Lógica borrosa (<i>fuzzy logic</i>)	281
6.5.9 Técnicas genéticas: algoritmos genéticos (<i>genetic algorithms</i>)	282

Capítulo 7

Técnicas de minería de datos en R.....285

7.1 Agrupamiento. <i>Clustering</i>	285
7.1.1 Agrupamiento jerárquico	286
7.1.2 Número óptimo de agrupaciones	289
7.1.3 Agrupamiento por particionamiento	299
7.1.4 Agrupamiento basado en modelos.....	305
7.1.5 Agrupamiento borroso (<i>fuzzy</i>).....	312
7.1.6 Otras técnicas de agrupamiento.....	314
7.1.7 Representación y análisis de las clases.....	323
7.1.8 Validación de resultados	326
7.2 Clasificación	329
7.2.1 Selección de atributos	329
7.2.2 Reducción de la dimensionalidad.....	340
7.2.3 Árboles de decisión.....	347
7.2.3.1 RPART (<i>Recursive Partitioning and Regression Trees</i>).....	347
7.2.3.2 Árboles de inferencia condicional, CTREE	350
7.2.3.3 C5.0	351
7.2.4 Metaalgoritmos.....	354
7.2.4.1 AdaBoost (<i>ADaptative BOOSTing</i>).....	355
7.2.4.2 GBM (<i>Gradient Boosting Machine</i>)	356
7.2.4.3 <i>Random forest</i>	358
7.2.5 SVM, máquinas de vectores de soporte	359
7.2.6 K vecinos próximos. k-NN (<i>k-Nearest Neighbors</i>).....	361
7.2.7 Redes de neuronas artificiales	363

Capítulo 8

Internet de las cosas y análisis de

series temporales.....369

8.1 Internet de las cosas	369
8.2 Thingier.io IoT	371
8.2.1 <i>Hardware</i>	372
8.2.2 Configuración de la plataforma	374
8.2.3 <i>Software</i> del dispositivo	376
8.2.4 Visualización y exportación de la información.....	379
8.3 Series temporales	381
8.3.1 Predicción con series temporales	382
8.3.1.1 Predicción lineal (autorregresión).....	382
8.3.1.2 Error de predicción.....	384
8.3.1.3 Predicción no lineal	385
8.3.2 Análisis y descomposición de series.....	386
8.3.2.1 Tendencia y estacionariedad	386
8.3.2.2 Modelos ARMA/ARIMA	389

8.4 Análisis de series con R	390
8.4.1 Componentes de la serie temporal	392
8.4.2 Modelos de predicción	400
8.4.3 Detección de anomalías	408

Capítulo 9

Análisis de datos espaciales.....411

9.1 Introducción.....	411
9.1.1 Datos de tipo espacial.....	412
9.1.2 Latitud, longitud	413
9.1.3 La clase de datos Spatial en RStudio	414
9.1.4 Datos	414
9.2 Tipos de datos.....	416
9.2.1 Creación de objetos SpatialPoints	416
9.2.2 Creación de objetos SpatialGrid.....	420
9.3 Visualización de datos espaciales.....	421
9.4 Análisis estadístico (interpolación).....	423
9.4.1 Análisis exploratorio de datos.....	424
9.4.1.1 Crear <i>grid</i> Ávila-Madrid	424
9.4.2 Interpolación IDW (<i>Inverse Distance Weighted</i>).....	427
9.4.2.1 Presentación de resultados IDW	428
9.4.3 Correlación espacial (variograma).....	428
9.4.3.1 Selección modelo de variograma.....	428
9.4.3.2 Presentación de resultados.....	430

Bibliografía.....431

1.1 Introducción

A finales del siglo XX, la cantidad de datos que fue almacenada en las bases de datos ya excedía la habilidad de los operadores para reducir y analizar los datos sin el uso de técnicas de análisis automatizadas. Muchas bases de datos comerciales transaccionales y científicas estaban creciendo de forma desproporcionada. A finales del siglo XX apareció por primera vez el término KDD (*Knowledge Discovery in Databases*), que es el proceso completo de extracción de información que se encarga además de la preparación de los datos y de la interpretación de los resultados obtenidos [PSF91]. Se trataba de interpretar grandes cantidades de datos y encontrar relaciones o patrones. Para conseguirlo harían falta técnicas de aprendizaje automático, estadísticas, bases de datos, técnicas de representación del conocimiento, razonamiento basado en casos, razonamiento aproximado, adquisición de conocimiento, redes de neuronas y visualización de datos. Tareas comunes en KDD son la inducción de reglas, los problemas de clasificación y *clustering*, el reconocimiento de patrones, el modelado predictivo, la detección de dependencias, etc. KDD era un campo creciente donde había muchas metodologías en desarrollo tanto genéricas como de dominio específico.

En los últimos tiempos, se ha producido una eclosión en la disponibilidad de datos. Según un informe de IBM [IBM16], el 90 % de los datos disponibles actualmente en el mundo se han creado en los dos últimos años. Las nuevas posibilidades que ofrecen las infraestructuras y las redes de comunicación, unido a la amplia disponibilidad de dispositivos inteligentes (teléfonos móviles), Internet de las cosas (electrodomésticos inteligentes), metadatos, etc., generan un flujo de información que crece a un ritmo vertiginoso [Spencer16].

La ingente cantidad de datos disponibles, generados de forma continua e incesante por entidades, usuarios, servicios o dispositivos, ha implicado el desarrollo de nuevos métodos científicos e ingenieriles para disponer de sistemas y procedimientos capaces de almacenar, procesar y analizar dichos datos, generando de esta manera información y conocimiento en sectores tan dispares como la industria, banca, finanzas, *marketing*, seguros, redes sociales, comercio electrónico, salud, gobierno electrónico, ciudades inteligentes, biología, medicina o ciencias de la tierra, por citar algunos.

Big data es un término de origen inglés cuya traducción equivale a “datos masivos”. Muchas son las definiciones que entidades y organizaciones han dado para el término *big data*, pero todas ellas se pueden resumir en el conjunto de datos cuyo tamaño supera considerablemente la capacidad de captura, almacenado, gestión y análisis del *software* convencional de bases de datos.

Sin embargo, el concepto no hace referencia simplemente al tamaño de la información, sino también a la variedad del contenido y a la velocidad con la que los datos se generan, almacenan y analizan. Estas dimensiones son las “3V” con las que la empresa Gartner describió *big data*, es decir **volumen, velocidad y variedad de los datos**¹:

- **Volumen.** Como su propio nombre indica, *big data* corresponde al gran volumen de datos que se generan diariamente en las empresas y organizaciones de todo el mundo. Por ejemplo, la cadena de supermercados americana Walmart almacena más de un millón de transacciones comerciales cada hora identificando los productos que compran sus clientes; más de cien mil gigas de información almacena la red social Facebook diariamente, así como setenta y dos millones de descargas se realizan en la tienda *online* App Store.
- **Velocidad.** Se trata de los flujos de datos, la creación de registros estructurados y la disponibilidad para el acceso y la entrega. Es decir, cómo de rápido se están produciendo los datos, así como la rapidez en la que se trata de satisfacer la demanda de éstos. La tecnología *big data* ha de ser capaz de almacenar y trabajar en tiempo real con las fuentes generadoras de información tales como sensores, redes sociales, blogs, páginas webs, etc., fuentes que generan millones de datos al segundo; y, por otro lado, capacidad de analizar dichos datos con la suficiente rapidez reduciendo los largos tiempos de procesamiento que presentaban las herramientas tradicionales de análisis.
- **Variedad.** *Big data* ha de tener la capacidad de combinar una gran variedad de información digital en los diferentes formatos en los que se puedan presentar. Las empresas líderes en TIC siempre han tenido un problema para traducir grandes volúmenes de información transaccional en decisiones. Puesto que ahora hay más tipos de información para analizar, provenientes principalmente de los medios sociales, la complejidad aumenta. Esta variedad en los datos incluye datos

¹ <http://www.gartner.com/newsroom/id/1731916>

estructurados (bases de datos) y no estructurados, datos jerárquicos, documentos, correo electrónico, datos de medición, vídeo, imágenes fijas, audio, datos de cotizaciones, transacciones financieras, etc., entre otras clases de fuentes generadoras de diferentes tipos de información.

Sin embargo, algunas organizaciones incluyen una cuarta V, ya sea para referirse a la **veracidad** de los datos, cuya valía exacta es vital para el negocio, o para referirse al **valor** de la información que proporcionan estos análisis:

- **Veracidad.** *Big data* ha de ser capaz de tratar y analizar inteligentemente este gran volumen de datos con la finalidad de obtener una información verídica y útil que nos permita mejorar la toma de decisiones basada en los datos más exactos.
- **Valor.** Hace referencia a los beneficios que se desprenden del uso de *big data* (reducción de costes, eficiencia operativa, mejoras de negocio).

Por tanto, se define a la tecnología *big data* como el conjunto de arquitecturas y herramientas informáticas destinadas a la manipulación, gestión y análisis de grandes volúmenes de datos desde todo tipo de fuentes, diseñadas para extraer valor y beneficio de los mismos, con una amplia variedad en su naturaleza, mediante procesos que permitan capturar, descubrir y analizar información a alta velocidad y con un coste reducido.

Comparativamente, estas nuevas tecnologías de *big data* no pueden ser equiparadas con las herramientas informáticas tradicionales. Actualmente las arquitecturas clásicas de tratamiento de datos no soportan el procesamiento de grandes cantidades de datos a costes asequibles para la mayoría de las empresas.

Por consiguiente, el objetivo fundamental de *big data* es dotar de una infraestructura tecnológica a las empresas y organizaciones con la finalidad de poder almacenar, tratar y analizar de manera económica, rápida y flexible la gran cantidad de datos que se generan diariamente, para ello es necesario el desarrollo y la implantación tanto de *hardware* como de *software* específicos que gestionen esta explosión de datos con el fin de extraer valor y así obtener información útil para el negocio. La tecnología *big data* tiene por objetivo esta gestión de los datos e información de manera inteligente que ayude a una correcta toma de decisión.

En el auge de esta era tecnológica, el sistema más utilizado en la industria para ofrecer capacidades analíticas avanzadas ha sido Hadoop, un *software* de código abierto cuyo desarrollo lo coordina la organización Apache Foundation. Hadoop es un *framework* de *software* que soporta aplicaciones distribuidas y facilita el almacenamiento y procesamiento de la información. Hadoop y su ecosistema han permitido el procesamiento de grandes cantidades de datos de manera asequible. Pero todavía no es una tecnología que se encuentre al alcance de cualquier empresa. La razón de esto es que las tecnologías *big data* todavía están en fase de evolución y maduración.

Dentro de los entornos *big data* se suelen integrar herramientas de análisis de datos que permiten extraer nuevo conocimiento a partir de la ingente cantidad de datos almacenada. Actualmente hay un conjunto de herramientas comerciales que llevan a cabo procesos de análisis de datos, como son:

- **Weka.** Es una aplicación de código abierto, disponible de forma gratuita bajo Licencia Pública General de GNU. Soporta prácticamente todas las tareas estándar de *data mining*. Los algoritmos pueden ser aplicados directamente sobre un conjunto de datos o llamados desde código Java. Weka también proporciona acceso a bases de datos SQL gracias a que tiene conexión JDBC (*Java Database Connectivity*) y puede procesar el resultado devuelto por una consulta hecha a la base de datos. No permite realizar minería de datos multirrelacional, pero existen aplicaciones que pueden convertir una colección de tablas relacionadas de una base de datos en una única tabla que ya puede ser procesada con Weka. Es una herramienta amigable para el usuario, provista de una interfaz gráfica que facilita a los usuarios inexpertos identificar información oculta en bases de datos y sistemas de archivos, utilizando simplemente las opciones de sus interfaces visuales.
- **Orange.** Es una *suite* de *software* para aprendizaje automático y minería de datos basada en componentes, desarrollada en el Laboratorio de Bioinformática de la Facultad de Ciencias de la Computación e Informática de la Universidad de Liubliana, Eslovenia, junto con la comunidad de código abierto. Es un *software* libre, que puede ser redistribuido o modificado bajo los términos de la Licencia Pública General de GNU, y es distribuido por Orange, Data Mining Fruitful & Fun, web: <http://orange.biolab.si>, sin ninguna garantía. Orange incluye un amplio rango de técnicas de preproceso, modelado y exploración de datos. Está basada en componentes C++, a las que se puede acceder directamente a través de *scripts* Python (mejor y más fácil), o a través de objetos GUI llamados *Orange Widgets*.
- **RapidMiner.** Anteriormente llamada YALE (*Yet Another Learning Environment*), es un entorno que contiene procedimientos de *data mining* y aprendizaje automático. El proceso de *data mining* puede hacerse mediante operadores arbitrariamente anidados, descritos en ficheros XML y creados con la interfaz gráfica de usuario de RapidMiner. RapidMiner está escrito en el lenguaje de programación Java. También integra esquemas de aprendizaje y evaluadores de atributos del entorno Weka y esquemas de modelización estadística de R-Project. RapidMiner puede ser utilizado para minería de texto, minería multimedia, minería de flujo de datos, desarrollo de métodos de conjunto y minería de datos distribuida.
- **Tanagra.** Es un *software* gratuito de *data mining* para propósitos académicos y de investigación. Propone varios métodos de *data mining*, desde análisis exploratorio de datos, aprendizaje estadístico, aprendizaje automático y del área de bases de datos. Tanagra contiene algo de aprendizaje supervisado, pero también otros paradigmas como *clustering*, análisis factorial, estadística paramétrica y no

paramétrica, reglas de asociación, selección de características y algoritmos de construcción.

- **KNIME (Konstanz Information Miner).** Es una plataforma de código abierto para la integración de datos, procesamiento, análisis y exploración desarrollada por la cátedra de Bioinformática y Minería de Información de la Universidad de Konstanz, Alemania, usando prácticas de ingeniería de *software*, y actualmente está siendo utilizada por más de seis mil profesionales en todo el mundo, tanto de la industria como a nivel académico. Integra todos los módulos de análisis del entorno Weka, y *plugins* adicionales permiten que se ejecuten *R-scripts*, ofreciendo acceso a una vasta librería de rutinas estadísticas.
- **Oracle Data Mining (ODM).** Es una opción de sistema de gestión de base de datos relacional (RDBMS) de Oracle Database Enterprise Edition (EE). Contiene varios algoritmos de minería de datos y análisis de datos para clasificación, predicción, regresión, asociaciones, selección de características, detección de anomalías, extracción de características y análisis especializado. Estas implementaciones se integran en el núcleo de la base de datos Oracle, y operan de forma nativa sobre los datos almacenados en las tablas de bases de datos relacionales. El sistema está organizado en torno a unas pocas operaciones genéricas que proporcionan una interfaz unificada general de las funciones de minería de datos.
- **IBM SPSS Modeler.** Originalmente llamada SPSS Clementine de SPSS Inc., después fue renombrada PASW Modeler, pero cuando en 2009 IBM adquirió SPSS Inc. fue denominada IBM SPSS Modeler. IBM SPSS Modeler es una aplicación de *software* de *data mining* de IBM. Es una herramienta de *data mining* y de análisis de texto, utilizada para construir modelos predictivos. Tiene una interfaz visual que permite a los usuarios utilizar algoritmos estadísticos y de *data mining* sin programar.
- **SAS Enterprise Miner.** Es una potente herramienta de apoyo en el proceso de minería de datos con un diseño abierto y extensible con un amplio conjunto de capacidades. Dispone de una interfaz de usuario fácil de usar, que permite a los usuarios empresariales construir y valorar los mejores y más avanzados modelos predictivos y descriptivos de manera rápida y fácil, mejorando la precisión de las predicciones, y compartiendo información fiable para que los analistas de negocio puedan mejorar la calidad de sus decisiones disponiendo de conclusiones e ideas de forma rápida, autosuficiente y automatizada.

Específicamente, en España, las tecnologías *big data* arrancaron definitivamente en 2012, con gran aceleración del número de iniciativas, productos y servicios en 2013. Sin embargo, la casi totalidad de las aún pocas iniciativas comerciales existentes en España se basan en servicios a medida para la construcción de soluciones *big data* (en algún caso alineado a la representación comercial de algún producto), o en soluciones “verticales” sobre plataformas ya preimplementadas (soluciones para casos de uso

específicos como los sistemas de búsqueda, minería web, procesamiento de logs, etc.). Dentro de las iniciativas creadas desde 2013 cabe resaltar:

- **Daedalus.** Creada en España en 1998, ofrece una gama de productos pensada para ofrecer soluciones avanzadas de procesamiento semántico. También merece destacar su participación en varios proyectos nacionales e internacionales de I+D. Sus productos:
 - **Textalytics.** Ofrece, en modo SaaS, distintas API de alto nivel y aplicación específica para diversos sectores y escenarios de uso.
 - **K-Site.** Es una familia de módulos de tecnología lingüística, semántica y de gestión de contenido multimedia, pensada para ofrecer soluciones personalizadas.
 - **Stilus.** Es una herramienta para la revisión de textos multiidioma. Además de la corrección de textos ortográfica y gramaticalmente, permite realizar una revisión de estilo y recibir sugerencias y explicaciones didácticas. Este producto se ofrece a través del portal web: mystylus.com.
 - **Sentimentalytics.** Es un *plugin* para navegadores que analiza y etiqueta semánticamente los *timelines* que aparecen en redes y herramientas sociales.
- **Pragsis.** Erigida en España en 2004, es actualmente el principal *partner* de Cloudera en España, con fuerte actuación en el campo de la formación. En 2012 Pragsis integró una amplia gama de herramientas formando su plataforma para *big data* basada en Hadoop: Bidoop.
 - **Bidoop Layer.** Es una capa de componentes *big data* para Hadoop que integra herramientas analíticas y de visualización. La arquitectura de Bidoop Layer se divide en seis componentes principales: motor de *workflow*, motor de eventos en tiempo real, gestor de usuarios y accesos, Bidoop Manager, *dashboards* y herramientas de visualización.
- **Datasalt.** Fundada en España en 2011, ofrece asistencia en la integración y el desarrollo de soluciones *big data* y *cloud computing*, principalmente a través de servicios de consultoría y formación. Ofrece una serie de productos verticales para sectores específicos (banca, comercio electrónico, publicidad...) basados en dos herramientas *big data* creadas por sus fundadores:
 - **Splout.** Es un *software* libre que provee una vista SQL Big Data con latencias por debajo del segundo y alto rendimiento, integración con Hadoop, escalabilidad, flexibilidad y una interfaz REST.
 - **Pangool.** Es una API Java mejorada de bajo nivel para Hadoop basada en el paradigma Tuple MapReduce, gracias a su soporte nativo de los patrones de

desarrollo más comunes: *joins*, ordenación secundaria y registros compuestos.

- **Treelogic.** Fundada en España en 1996, cuenta actualmente con SMYZER, una herramienta para la monitorización de la información contenida en redes y medios sociales en tiempo real. Treelogic no quiere faltar en el apogeo de desarrollos de plataformas tecnológicas *big data*, de modo que creó en 2014 una versión *open source* de un *middleware* que permitía de forma opaca para el cliente tratar datos en tiempo real en combinación con datos históricos. Comercialmente la plataforma se llamaba Lambdoop.
- **Paradigma Tecnológico.** Establecida en 2007 y dedicada a tecnologías web, creó a finales de 2013 una *start-up* filial, Stratio. Con Stratio, Paradigma Tecnológico ha centrado sus esfuerzos en el desarrollo y comercialización de una plataforma *big data* a nivel internacional.

Entre las empresas y productos destacados anteriormente, todos ellos tenían en común el enfoque de abstraerse de las distintas tecnologías que componen un entorno *big data* principalmente a través de soluciones basadas en API, estas ofertas actuales están casi siempre limitadas a un determinado tipo de funcionalidad o a soluciones específicas.

En paralelo al crecimiento del *big data*, del número y tamaño de los datos, de las aplicaciones, de las plataformas, ha evolucionado una nueva área de conocimiento que da respuesta a las nuevas necesidades de explotación de los mismos. Conocida globalmente como *data science* o *ciencia de los datos*, se trata de un campo interdisciplinar que combina *machine learning*, estadística, análisis avanzado, minería de datos, *big data* y programación, con el objetivo de extraer conocimiento oculto y útil a partir de los datos, mediante procesos de descubrimiento o de formulación y verificación de hipótesis ([IBM17], [NIST15]).

1.2 Los datos

Los datos recogen un conjunto de hechos (una base de datos, BD) y los patrones son expresiones que describen un subconjunto de los datos (un modelo aplicable a ese subconjunto). BD involucra un proceso iterativo e interactivo de búsqueda de modelos, patrones o parámetros. Los patrones descubiertos han de ser válidos, novedosos para el sistema (para el usuario siempre que sea posible) y potencialmente útiles.

Se han de definir medidas cuantitativas para los patrones obtenidos (precisión, utilidad, beneficio obtenido...). Se debe establecer alguna medida de interés que considere la validez, utilidad y simplicidad de los patrones obtenidos mediante alguna de las técnicas de minería de datos. El objetivo final de todo esto es incorporar el conocimiento obtenido en algún sistema real, tomar decisiones a partir de los resultados alcanzados o, simplemente, registrar la información conseguida y suministrársela a quien esté interesado.

De forma esquemática el proceso de descubrimiento de conocimiento sigue los siguientes pasos:

1. Formular el problema.
2. Determinar la representación (atributos y clases). Esta determinación se puede realizar directamente a la vista de los datos, mediante la intervención de expertos o utilizando técnicas automáticas como son los filtros.
3. Identificar y recolectar datos de entrenamiento (bases de datos, ficheros, etc.).
4. Preparar datos para análisis.
5. Selección de modelo, construcción y entrenamiento.
6. Evaluar lo aprendido. La evaluación puede realizarse automáticamente, como por ejemplo la validación cruzada o mediante la intervención de un experto que valore los resultados obtenidos.
7. Integrar la base de conocimiento a la espera de nuevos datos tras acciones.

El proceso de BD se inicia con la identificación del problema y de los datos que lo representan. Para ello hay que imaginar qué datos se necesitan, dónde se pueden encontrar y cómo conseguirlos. Una vez que se dispone de datos, se deben seleccionar aquellos que sean útiles para los objetivos propuestos. Se preparan poniéndolos en un formato adecuado.

Una vez se tienen los datos adecuados se procede a la minería de datos, proceso en el que se seleccionarán las herramientas y técnicas adecuadas para lograr los objetivos pretendidos. Y tras este proceso llega el análisis de resultados, con lo que se obtiene el conocimiento pretendido.

1.3 Etapas en los procesos de *big data*

Las etapas del trabajo en *big data* incluyen muchas decisiones que deben ser tomadas por el usuario estructuradas de la siguiente manera:

- **Comprensión del dominio de la aplicación, del conocimiento relevante y de los objetivos del usuario final.**
- **Creación del conjunto de datos.** Consiste en la selección del conjunto de datos, o del subconjunto de variables o muestra de datos, sobre los cuales se va a realizar el descubrimiento.
- **Limpieza y preprocesamiento de los datos.** Se compone de operaciones, tales como recolección de la información necesaria sobre la cual se va a realizar el

proceso, decidir las estrategias sobre la forma en que se van a manejar los campos de los datos no disponibles, estimación del tiempo de la información y sus posibles cambios, etc.

- **Reducción de los datos y proyección.** Encontrar las características más significativas para representar los datos, dependiendo del objetivo del proceso. En este paso se pueden utilizar métodos de transformación para reducir el número efectivo de variables a ser consideradas o para encontrar otras representaciones de los datos.
- **Elegir la tarea de minería de datos.** Decidir si el objetivo del proceso es: regresión, clasificación, agrupamiento, etc.
- **Elección del algoritmo(s) de minería de datos.** Selección del método(s) a ser utilizado para buscar los patrones en los datos. Incluye además la decisión sobre qué modelos y parámetros pueden ser los más apropiados.
- **Minería de datos.** Consiste en la búsqueda de los patrones de interés en una determinada forma de representación o sobre un conjunto de representaciones, utilizando para ello métodos de clasificación, reglas o árboles, regresión, agrupación, etc.
- **Interpretación de los patrones encontrados.** Dependiendo de los resultados, a veces se hace necesario regresar a uno de los pasos anteriores.
- **Consolidación del conocimiento descubierto.** Consiste en la incorporación de este conocimiento al funcionamiento del sistema, o simplemente documentación e información a las partes interesadas.

El proceso de BD puede involucrar varias iteraciones y puede contener ciclos entre dos de cualquiera de los pasos. La mayoría de los trabajos que se han realizado sobre BD se centran en la etapa de minería en la búsqueda de algoritmos que extraigan relaciones y conocimiento de grandes cantidades de datos. Sin embargo, los otros pasos se consideran importantes para el éxito del proceso completo. Gran parte del esfuerzo del proceso de extraer conocimiento recae sobre la fase de preparación de los datos, fase crucial para tener éxito, como ya se comentó anteriormente.

1.4 Minería de datos

Minería de datos es un término genérico que engloba resultados de investigación, técnicas y herramientas usadas para extraer información útil de grandes bases de datos. Si bien minería de datos es una parte del proceso completo de BD, en buena parte de la literatura, los términos *minería de datos* y *BD* se identifican como si fueran lo mismo. Concretamente, el término *BD* es usado comúnmente por los estadísticos, analistas de datos y por la comunidad de administradores de sistemas informáticos como todo

el proceso de descubrimiento, mientras que el término *minería de datos* o *machine learning* es utilizado más por los especialistas en inteligencia artificial.

Hasta ahora, los mayores éxitos en *big data* se pueden atribuir directa o indirectamente a avances en bases de datos (un campo en el que los ordenadores superan a los humanos). No obstante, muchos problemas de representación del conocimiento y de reducción de la complejidad de la búsqueda necesaria (usando conocimiento *a priori*) están aún por resolver. Ahí reside el interés que ha despertado el tema entre investigadores de todo el mundo.

Los procesos de BD involucran diversas tecnologías:

- **Bases de datos que permiten almacenar los datos de forma estructurada**, tanto a nivel lógico (con la aparición en los últimos años de las bases de datos no relacionales) y a nivel de *hardware* (con la capacidad para el proceso en clúster).
- **Técnicas de visualización** que permiten realizar representaciones gráficas de los datos que facilitan la labor del usuario a la hora de entender los datos, para filtrarlos y procesarlos.
- **Técnicas estadísticas**, que permiten analizar analíticamente los datos almacenados en las bases de datos y desarrollar modelos estadísticos que los expliquen.
- **Técnicas de aprendizaje automático**, que permiten desarrollar modelos conceptuales que representan los datos almacenados en la base de datos.

1.5 Estructura de un proyecto de análisis de datos

En este apartado se repasan las fases de un proyecto de BD y se identifican los actores correspondientes. Por actores entendemos:

- **La empresa que requiere el conocimiento extraído, el cliente**, en principio ese conocimiento puede ser estático, es decir, un conjunto de reglas o relaciones que les permitan interpretar lo que está ocurriendo en un momento determinado, o puede ser dinámico, es decir, un *software* con los algoritmos adecuados para la extracción de conocimiento que puedan ejecutar sobre los datos a medida que se van almacenando.
- **La empresa especializada en desarrollo de herramientas avanzadas**, la empresa TIC, y que habitualmente ha desarrollado el sistema de información del cliente, y que desea incorporar entre las herramientas del sistema de información un núcleo de visualización y extracción de conocimiento para incrementar el valor innovador del sistema de información. Un ejemplo claro de estos sistemas son las herramientas de *business intelligence*, cuya idea es proporcionar la información procesada y el conocimiento extraído para que los directivos tomen las decisio-

nes. En muchos clientes, sobre todo en las grandes empresas, esta empresa TIC es directamente su departamento TIC, que es el responsable de dar soporte y desarrollar las herramientas del sistema de información del cliente.

- **Una empresa o universidad con personas expertas en el campo del *machine learning***, expertas en los algoritmos de extracción que pueden valorar qué tipos de algoritmos y con qué parámetros tiene sentido intentar extraer conocimiento de las bases de datos del cliente utilizando la tecnología de la empresa TIC. Estos expertos pueden formar parte de la plantilla de la empresa TIC, pero en general suelen subcontratarse sus servicios para llevar los análisis previos que permitan valorar qué algoritmos utilizar, qué parámetros e incluso si se deben desarrollar algoritmos propios para los datos de esa empresa.

Para cada fase del proceso de KDD hace falta la participación de uno o varios de estos actores, a modo de resumen:

TAREA	CLIENTE	EMPRESA TIC	EXPERTO AD
1. Comprensión del dominio de la aplicación, del conocimiento relevante y de los objetivos del usuario final.	Es muy relevante que el cliente tenga claro el objetivo del proyecto, si se trata de un conocimiento estático o uno dinámico, y sobre todo, qué quiere aprender de sus propios datos. El proceso de BD no es un automatismo que les va a generar cosas insospechadas y válidas para la empresa.	La empresa TIC tiene que hacer una valoración de la tecnología, qué soluciones puede aportar para el almacenamiento de datos, para el procesado de grandes volúmenes de datos, qué cambios en el sistema de información, y sobre todo, si lo que requiere el cliente necesita de un experto ML o es suficiente con una adecuada captación de información y procesado de la misma.	Puede participar para aclarar qué no puede ser resuelto con algoritmos de ML a la luz de la información almacenada, pero no se requiere su presencia en esta primera fase si la empresa TIC tiene claras cuáles son sus capacidades.
2. Creación del conjunto de datos. Consiste en la selección del conjunto de datos o del subconjunto de variables o muestra de datos sobre los cuales se va a realizar el descubrimiento.	En este caso, el cliente únicamente debe facilitar la recogida de la información.	La empresa TIC debe asegurar que cuenta con las tecnologías que permitan llevar a cabo la creación de ese conjunto de datos.	El experto debe validar que el conjunto de datos es completo y representativo, o al menos dar las pautas para que se genere ese conjunto de datos, a menos que la empresa TIC tenga experiencia previa y asegure que la información cumple con los requisitos para poder aplicar sobre ella diferentes algoritmos.

3. Limpieza y preprocesamiento de los datos. Se compone de operaciones tales como: recolección de la información necesaria sobre la cual se va a realizar el proceso, decidir las estrategias sobre la forma en que se van a manejar los campos de los datos no disponibles, estimación del tiempo de la información y sus posibles cambios, etc.		La empresa TIC desarrolla los procesos de <i>software</i> para llevar a cabo estas tareas. Es importante un alto grado de automatización, porque esta tarea es la más importante del proceso y se volverá a ella a medida que se descubran relaciones entre los datos, llevando a cabo limpiezas sucesivas.	El experto define cómo llevar a cabo estos procesos, cuáles son los algoritmos que deben desarrollarse para hacer esta limpieza. Estos cambios dependerán del conocimiento del experto y de los resultados de las siguientes fases.
4. Reducción de los datos y proyección. Encontrar las características más significativas para representar los datos, dependiendo del objetivo del proceso. En este paso se pueden utilizar métodos de transformación para reducir el número efectivo de variables a ser consideradas o para encontrar otras representaciones de los datos.		La empresa TIC suministra procesos de <i>software</i> para llevar a cabo el acceso a los datos de forma rápida y al mismo tiempo suele incorporar estas funciones como parte del sistema de información porque son útiles al cliente.	El experto ejecuta los algoritmos e interpreta la solución, en muchos casos se debe volver a la fase previa.
5. Elegir la tarea de ML. Decidir si el objetivo del proceso de BD es: regresión, clasificación, agrupamiento, etc.	El cliente tiene que haber dejado definido previamente qué desea obtener.		El experto valora los algoritmos posibles y las posibilidades de aplicar qué objetivo en función de lo que desea obtener el cliente.
6. Elección del algoritmo(s) de ML. Selección del método(s) a ser utilizado para buscar los patrones en los datos. Incluye además la decisión sobre qué modelos y parámetros pueden ser los más apropiados.			El experto lleva a cabo diversos experimentos, que en muchos casos deben empezar desde la fase 3 de limpieza para valorar algoritmos aplicables.

7. ML. Consiste en la búsqueda de los patrones de interés en una determinada forma de representación o sobre un conjunto de representaciones, utilizando para ello métodos de clasificación, reglas o árboles, regresión, agrupación, etc.		La empresa TIC implementa los algoritmos decididos sobre el sistema de información de la empresa.	El experto lleva a cabo el proceso de minería con los algoritmos previos sobre el conjunto de datos para extraer conclusiones concretas con ese conjunto de datos.
8. Interpretación de los patrones encontrados. Dependiendo de los resultados, a veces se hace necesario regresar a uno de los pasos anteriores.	El cliente explica la validez de los resultados y su posible aplicación futura, valora el conocimiento extraído y el éxito del proyecto.		El experto explica los resultados y su validez matemática.
9. Consolidación del conocimiento descubierto. Consiste en la incorporación de este conocimiento al funcionamiento del sistema, o simplemente documentación e información a las partes interesadas.	El cliente incorpora el conocimiento si es estático o las funciones del sistema de información si es dinámico.	La empresa TIC desarrolla las funciones específicas para el cliente en el caso de extracción de conocimiento dinámico.	

Como se puede observar, las tareas de cada actor tienen una mayor influencia en cada uno de los pasos y en muchos casos deberían intervenir los tres en reuniones de seguimiento para ir analizando los resultados obtenidos.

1.6 Aplicaciones

En este punto se presentan las principales áreas y sectores empresariales en los que se puede aplicar la minería de datos.

1.6.1 Marketing

Actualmente, con la generación de los puntos de ventas informatizados y conectados a un ordenador central y el constante uso de las tarjetas de crédito se genera gran cantidad de información que hay que analizar. Con ello se puede emplear la minería de datos para:

- **Identificar patrones de compra de los clientes.** Determinar cómo compran a partir de sus principales características, conocer el grado de interés sobre tipos de productos, si compran determinados productos en determinados momentos, etc.
- **Segmentación de clientes.** Consiste en la agrupación de los clientes con características similares, por ejemplo, demográficas. Es una importante herramienta en la estrategia de *marketing* que permite realizar ofertas acordes a diferentes tipos de comportamiento de los consumidores.
- **Predecir respuestas a campañas de *mailing*.** Estas campañas son caras y pueden llegar a ser molestas para los clientes a los que no les interesa el tipo de producto promocionado, de modo que es importante limitarlas a los individuos con una alta probabilidad de interesarse por el producto. Está, por ello, muy relacionada con la segmentación de clientes.
- **Análisis de cestas de la compra (*market-basket analysis*).** Consiste en descubrir relaciones entre productos, esto es, determinar qué productos suelen comprarse junto con otros, con el fin de distribuirlos adecuadamente.

1.6.7 Compañías de seguros

En el sector de las compañías de seguros y la salud privada, se pueden emplear las técnicas de minería de datos, por ejemplo, para:

- Análisis de procedimientos médicos solicitados conjuntamente.
- Predecir qué clientes compran nuevas pólizas.
- Identificar patrones de comportamiento para clientes con riesgo.
- Identificar comportamiento fraudulento.

1.6.8 Banca

En el sector bancario la información que puede almacenarse es, además de las cuentas de los clientes, la relativa a la utilización de las tarjetas de crédito, que puede permitir conocer hábitos y patrones de comportamiento de los usuarios. Esta información puede aplicarse para:

- Detectar patrones de uso fraudulento de tarjetas de crédito.
- Identificar clientes leales. Es importante para las compañías de cualquier sector mantener los clientes. Y es que hay estudios que demuestran que es cuatro veces más caro obtener nuevos clientes que mantener los existentes.
- Predecir clientes con probabilidad de cambiar su afiliación.

- Determinar el gasto en tarjetas de crédito por grupos.
- Encontrar correlaciones entre indicadores financieros.
- Identificar reglas del mercado de valores a partir de datos históricos.

1.6.4 Telecomunicaciones

En el sector de las telecomunicaciones se puede almacenar información interesante sobre las llamadas realizadas, así como el destino, la duración, la fecha, etc. en que se realiza la llamada, por ejemplo, para:

- Detección de fraude telefónico. Mediante el agrupamiento o *clustering* se pueden detectar patrones en los datos que permitan detectar fraudes.

1.6.5 Medicina

También en el campo médico se almacena gran cantidad de información, sobre los pacientes, tal como enfermedades pasadas, tratamientos impuestos, pruebas realizadas, evolución, etc.

Se pueden emplear técnicas de minería de datos con esta información, por ejemplo, para:

- Identificación de terapias médicas satisfactorias para diferentes enfermedades.
- Asociación de síntomas y clasificación diferencial de patologías.
- Estudio de factores (genéticos, precedentes, de hábitos, alimenticios, etc.) de riesgo para la salud en distintas patologías.
- Segmentación de pacientes para una atención más inteligente según su grupo.
- Estudios epidemiológicos, análisis de rendimientos de campañas de información, prevención, sustitución de fármacos, etc.
- Identificación de terapias médicas y tratamientos erróneos para determinadas enfermedades.

1.6.6 Industria farmacéutica

En el sector químico y farmacéutico se almacenan gran cantidad de información:

- Bases de datos de dominio público conteniendo información sobre estructuras y propiedades de componentes químicos.
- Resultados de universidades y laboratorios publicados en revistas técnicas.

- Datos generados en la realización de los experimentos.
- Datos propios de la empresa.

Los datos son almacenados en diferentes categorías y a cada categoría se le aplica un diferente trato. Se podrían realizar, entre otras, las siguientes operaciones con la información obtenida:

- **Clustering de moléculas.** Consiste en el agrupamiento de moléculas que presentan un cierto nivel de similitud, con lo que se pueden descubrir importantes propiedades químicas.
- **Búsqueda de todas las moléculas que contienen un patrón específico.** Se podría introducir una subestructura (un patrón), devolviendo el sistema todas las moléculas que son similares a dicha estructura.
- **Búsqueda de todas las moléculas que vinculan un camino específico hacia una molécula objetivo.** Realizar una búsqueda exhaustiva puede ser impracticable, de manera que se pueden usar restricciones en el espacio de búsqueda.
- **Predicción de resultado de experimentos de una nueva molécula a partir de los datos almacenados.** A través de determinadas técnicas de inteligencia artificial es posible predecir los resultados a nuevos experimentos a partir de los datos, con el consiguiente ahorro de tiempo y dinero.

1.6.6 Biología

Con la finalización en los próximos años del Proyecto Genoma Humano y el almacenamiento de toda la información que está generando en bases de datos accesibles por Internet, el siguiente reto consiste en descubrir cómo funcionan nuestros genes y su influencia en la salud. Existen nuevas tecnologías (chips de ADN, proteómica, genómica funcional, variabilidad genética individual) que están posibilitando el desarrollo de una “nueva biología” que permite extraer conocimiento biomédico a partir de bases de datos experimentales en el entorno de un ordenador, básicamente mediante técnicas de minería de datos y visualización. Estos trabajos forman parte de los desarrollos de la bioinformática.

1.6.7 Minería de textos

La minería de textos (*text mining*) surge ante el problema cada vez más apremiante de extraer información automáticamente a partir de masas de textos. Se trata así de extraer información de datos no estructurados: texto plano. Existen varias aproximaciones a la representación de la información no estructurada:

- **Bag of words.** Cada palabra constituye una posición de un vector y el valor corresponde con el número de veces que ha aparecido.
- **N-gramas o frases.** Permite tener en cuenta el orden de las palabras. Trata mejor frases negativas "... *excepto...*", "... *pero no...*", que tomarían en otro caso las palabras que le siguen como relevantes.
- **Representación relacional (primer orden).** Permite detectar patrones más complejos (si la palabra *X* está a la izquierda de la palabra *Y* en la misma frase...).
- **Categorías de conceptos.**

Casi todos se enfrentan con el *vocabulary problem*, ya que tienen problemas con la sinonimia, la polisemia, los lemas, etc. En cualquier caso, aunque aún no se ha avanzado mucho en el área de minería de textos, ya hay productos comerciales que emplean esta tecnología con diferentes propósitos.

1.6.9 Minería de datos web

La minería de datos web (*web mining*) es una tecnología usada para descubrir conocimiento interesante en todos los aspectos relacionados con la web. Es uno de los mayores retos. El enorme volumen de datos en la web generado por la explosión de usuarios y el desarrollo de librerías digitales hace que la extracción de la información útil sea un gran problema. Cuando el usuario navega por la web se encuentra frecuentemente saturado por los datos. La integración de herramientas de minería de datos puede ayudar a la extracción de la información útil.

La minería de datos web se puede clasificar en tres grupos distintos no disjuntos, dependiendo del tipo de información que se quiera extraer o de los objetivos:

- **Minería del contenido de la web (*web content mining*).** Extraer información del contenido de los documentos en la web. Se puede clasificar a su vez en:
 - *Text mining.* Si los documentos son textuales (planos).
 - *Hypertext mining.* Si los documentos contienen enlaces a sí mismos o a otros documentos.
 - *Markup mining.* Si los documentos son semiestructurados (con marcas).
- **Multimedia mining.** Para imágenes, audio, vídeo, etc.
- **Minería de la estructura de la web (*web structure mining*).** Se intenta descubrir un modelo a partir de la tipología de enlaces de la red. Este modelo puede ser útil para clasificar o agrupar documentos.

- **Minería del uso de la web (*web usage mining*).** Se intenta extraer información (hábitos, preferencias, etc. de los usuarios o contenidos y relevancia de documentos) a partir de las sesiones y comportamiento de los usuarios navegantes.

1.6.10 Redes sociales

Las redes sociales proporcionan una cantidad ingente de información, lo que supone una gran oportunidad para obtener conocimiento útil, como información sobre el comportamiento de los usuarios y la interacción entre ellos, lo que en el caso de las empresas puede redundar finalmente en una ventaja competitiva. La explotación de esta información no es sólo una oportunidad sino también un reto importante, ya que, a diferencia del análisis de textos tradicional, en el que se investigaba el contenido con el objetivo de obtener información para su clasificación, en este caso, la explotación se complica, tanto por la cantidad de datos de entrada como por su naturaleza. Por ejemplo, es preciso considerar nuevas variables, como los seguidores (en el argot de Twitter, los denominados *followers*).

En el caso específico de Twitter, la información (los *tweets*) es generada de forma masiva y a una velocidad vertiginosa propiciando la necesidad imperiosa de un almacenamiento y procesamiento óptimos y eficientes que gestionen este aumento exponencial de información a analizar en formato de texto. Las herramientas de visualización también están cambiando para adaptarse a la información rápida y con representaciones gráficas de alto nivel de percepción que ayuden al análisis y a la toma de decisiones.

Existen numerosas herramientas de análisis de los datos de redes sociales, en particular de Twitter. Generalmente, estas herramientas organizan, clasifican o filtran la información de manera que los usuarios de la misma puedan monitorizarla y extraer conocimiento que de otro modo hubiera sido muy difícil extraer del conjunto total de información. Por ejemplo, Tweet Binder² es capaz de organizar usuarios en diferentes listas, como la de “usuarios más activos”, que ordena a los usuarios según el número de *tweets* que publican, o la de “usuarios que generan mayor impacto”, que cuenta el número de *tweets* de un usuario vistos por el resto de usuarios. Otra funcionalidad de la herramienta es filtrar sólo los *tweets* que contienen enlaces a otras páginas. En la misma línea, la aplicación para iPad Tweet Category³ saca estadísticos como, por ejemplo, el índice de actividad en una conferencia, que se mide por el porcentaje de usuarios que publican *tweets* sobre el total de los asistentes. En efecto, esta información puede ser útil para los usuarios de la herramienta, por ejemplo, las empresas, que pueden identificar personas que tienen impacto o generan interés en su área de negocio.

² <http://www.tweetbinder.com/>

³ <https://itunes.apple.com/app/tweet-category/id534497714>

Otras herramientas buscan facilitar la tarea de los usuarios *filtrando* o *reorganizando* la información *procesada* mediante métodos estadísticos y de reconocimiento de patrones, extrayendo un conocimiento para el usuario. Este conocimiento no está presente en los datos crudos de entrada y sólo es posible alcanzarlo mediante este procesamiento automático (no mediante inspección de los datos ni de listas organizadas de éstos).

Un ejemplo de la utilidad de este tipo de análisis y su capacidad de mejorar y complementar las soluciones existentes es la herramienta Hashtracking⁴, que permite extraer un gráfico de la evolución del número de *tweets* publicados en relación con un *hashtag* (por ejemplo, a lo largo de una conferencia). También saca una lista con los usuarios más activos, e imprime en pantalla el número de *followers* de cada uno. De esta manera, los usuarios de la aplicación pueden identificar a las personas más influyentes. Sin embargo, la labor de inteligencia se cede al usuario de la herramienta, ya que es éste quien debe decidir cómo medir esa influencia. Lo más normal es que el usuario se limite a decantarse por los usuarios con más *followers*, ya que a primera vista no tienen más elementos para juzgar. La realidad es que estos elementos existen, pero a menudo no son accesibles para el usuario o son demasiados y demasiado abstractos como para que el usuario pueda establecer una relación inmediata entre ellos. Por ello, las técnicas de ML tratan de transferir esa parte de inteligencia a la herramienta, que es capaz de sintetizar los datos crudos en información útil para el usuario.

1.7 Modelos y tareas

Los modelos son las relaciones, reglas, patrones y resúmenes extraídos tras el análisis de los datos. Gracias a esta extracción se obtiene el conocimiento útil que se estaba buscando. Estos modelos pueden ser tanto descriptivos como predictivos:

- **Modelos descriptivos.** Su objetivo no es otro que hallar patrones o resumir los datos. No pretenden predecir nuevos datos a partir de la información recabada. Los datos se presentan como un conjunto, $\delta = \{e: e \in E\}$ sin estar ordenados ni etiquetados de manera alguna. Técnicas tales para estos modelos son el agrupamiento, las reglas de asociación y el análisis correlacional.
- **Modelos predictivos.** Tienen como principal objetivo aproximar posibles valores del futuro o desconocidos a través de los datos de los que ya se dispone. Los datos van acompañados de una salida (clase, categoría o valor numérico). La regresión y la clasificación son técnicas comúnmente usadas en este tipo de modelos.

En paralelo con los tipos de modelos se tienen las tareas, que se pueden desarrollar cuando se aborda un problema de BD.

⁴ <https://www.hashtracking.com/>

1.7.1 Tareas descriptivas

1.7.1.1 Agrupamiento

Conocido globalmente como *clustering*, aunque también puede recibir otros nombres como *segmentación*, *aglomeración* o *racimamiento*. Probablemente sea la tarea descriptiva más empleada de todas. El *clustering* consiste en formar grupos “naturales” a partir de un conjunto de datos. En contraposición a la clasificación, en vez de analizar datos etiquetados con una clase, los analiza para generar una etiqueta. Los datos se agrupan de modo que los que pertenezcan a un mismo grupo guarden muchas similitudes entre sí y los que pertenezcan a grupos distintos se diferencien lo máximo posible. Al agrupamiento o *clustering* también se le conoce por *segmentación*, como ya hemos mencionado anteriormente, ya que segmenta el conjunto de datos en diversos grupos.

Al principio se desconoce cuántos grupos de clasificación existirán ni cómo serán estos grupos. Aquí se barajan dos opciones: o “forzar” el algoritmo para obtener un número determinado de grupos o dejar que la herramienta analice la naturaleza de los datos y calcule cuántos grupos sería deseable obtener.

Ya que el *clustering* organiza la información en diferentes segmentos o grupos, tiene una gran capacidad de predicción: en cuanto aparezcan nuevos datos, podrán ser clasificados en los grupos ya existentes. Gracias a esto, sabremos que comparten una serie de características y comportamientos comunes. Además, permite la explotación *a posteriori* de nuevos algoritmos dentro de cada grupo creado. De este modo se estudiará la información de una manera más inteligente.

El *clustering* es un buen aliado en el campo de las ventas. Un ejemplo de ello es la clasificación de clientes por su comportamiento ante cierto tipo de productos. De este modo se puede orientar el lanzamiento de un producto para maximizar los beneficios.

1.7.1.2 Correlaciones y factorizaciones

Es una tarea descriptiva que analiza el porcentaje de similitud entre los valores de dos variables numéricas. Teniendo los ejemplos de un grupo $E = A_1 \times A_2 \times \dots \times A_n$, se puede analizar la correlación existente entre dos atributos de todos los elementos de ese grupo A_i y A_j . Se lleva a cabo gracias a un modelo matemático con un coeficiente de correlación r , que toma valores entre -1 y 1. En caso de que el coeficiente dé 1 o -1 significa que las variables están fuertemente correlacionadas (de modo positivo o negativo respectivamente). Si el valor obtenido es 0 las variables no guardan ninguna correlación. Esto significa que, cuando guardan correlación positiva, ambas variables crecen al mismo tiempo. Decrecen de igual modo cuando la correlación es negativa. La correlación puede ser lineal o de cualquier otro tipo. Las tareas de correlación y las

factorizaciones se pueden combinar con modelos de regresión para estudiar relaciones entre atributos de causa-efecto.

1.7.1.3 Reglas de asociación

Estas tareas han evolucionado conjuntamente con la minería de datos desde los años noventa. Son tareas descriptivas similares a las correlaciones y factorizaciones. Su función principal es hallar relaciones no explícitas entre atributos categóricos. Dicho en otros términos, su objetivo es el mismo que el de las correlaciones pero para variables nominales, no numéricas. Dado el conjunto de elementos definidos por un conjunto de atributos $E = A_1 \times A_2 \times \dots \times A_n$, una regla de asociación se escribirá del siguiente modo: “si..., $A_i = a \wedge A_j = b \wedge \dots \wedge A_k = h$ entonces... $A_r = u \wedge A_s = v \wedge \dots \wedge A_z = w$ ”. La anterior ecuación no significa otra cosa que: “si el atributo X toma el valor a entonces el atributo Y tomará el valor c ”. Esto no tiene por qué significar que los atributos estén relacionados entre sí por causa-efecto. La estructura de la anterior asociación es una regla de asociación direccional, es decir, está orientada. Por este motivo, se denominan también *dependencias de valor*. También existen las reglas de asociación bidireccionales, donde en vez de haber una implicación, existe una “coimplicación”. Por otra parte, se pueden emplear además otro tipo de reglas de asociación como las negativas (con desigualdades), las reglas de asociación secuenciales (cuando una asociación se produce a continuación de la anterior y no al mismo tiempo) o reglas de asociación multinivel (involucran ítems con diferentes niveles de abstracción).

En la actualidad, grandes cadenas de supermercados emplean este tipo de herramientas de minería de datos para conocer mejor a sus clientes. Aplicando reglas de asociación direccionales del estilo “si compra_ginebra = sí \wedge compra_tónica = sí entonces compra_hielos”, el supermercado podría aplicar descuentos especiales o incluso hacer una mejor disposición de los productos para facilitar la compra a sus consumidores. Esto no significa que si alguien compra hielos vaya a comprar ginebra, pero sí es muy probable que ocurra en el sentido inverso.

Por el contrario, si ponemos en práctica una regla de asociación bidireccional del estilo “si compra_cereales = sí compra_leche”, estaríamos afirmando que una compra no se produce sin la otra. Ofertar estos productos en el mismo *pack* o colocarlos en estanterías muy próximas en el comercio sería una buena estrategia de negocio.

1.7.1.4 Dependencias funcionales

A menudo son enmarcadas dentro del campo de las reglas de asociación. Pero cuentan con una sustancial diferencia. Las dependencias funcionales consideran todos los posibles valores expresados del siguiente modo: “dados los valores de los atributos A_i, A_j, \dots, A_k soy capaz de determinar el valor del atributo A_r ”. La explicación es que

A_i , depende o es función de los valores del resto de atributos A_i, A_j, \dots, A_k . Las dependencias funcionales, al igual que ocurría con las reglas de asociación, pueden ser orientadas o no orientadas.

Un ejemplo típico de dependencias funcionales es discretizar grupos de gente sucesivamente (primero en edad, sexo, antecedentes familiares, etc.) para determinar si finalmente alguien podría padecer cierta enfermedad en el futuro.

Directamente relacionada con los *outlier*, está la técnica detección de valores e instancias anómalas, la cual pretende localizar y aislar en el conjunto global de los datos aquellas instancias anómalas en uno o todos sus atributos. Después de hacer una agrupación de los ejemplos, aquellos que tengan baja probabilidad de entrar en cualquier grupo serán detectados como sucesos aislados y acabarán siendo rechazados.

1.7.2 Tareas predictivas

1.7.2.1 Clasificación

También conocida como *discriminación*. Es, con mucha probabilidad, la tarea más popular de *data mining*. Cada entrada de la base de datos (a la cual llamaremos *instancia*) pertenece a una clase, que se indica mediante el valor de un atributo llamado *clase de la instancia*. Este atributo toma diversos valores discretos, correspondiendo cada uno a una clase. La clasificación busca predecir la clase desconocida de nuevas instancias o, más concretamente, clasificar de modo más preciso las nuevas instancias. Esto lo consigue calculando el cociente entre las predicciones correctas y el total de todas las predicciones.

Los ejemplos se presentan como un conjunto de duplas de elementos pertenecientes a los dos conjuntos, E y $S\delta = \{(e, s) : e \in E, s \in S\}$, siendo S el conjunto de valores de salida. Los ejemplos, representados por e , suelen ir junto a un valor de S , se les llama pues *ejemplos etiquetados* (e, s) . En consecuencia, δ será el conjunto de datos etiquetado. Esta tarea busca el aprendizaje de una función llamada *clasificador*, $C: E \rightarrow S$: $E \rightarrow S$ que exponga la correspondencia presente entre los ejemplos. En otras palabras, que para cada valor de E sólo tengamos un valor de S . Esta S no toma valores numéricos ya que es un conjunto nominal. Los valores que adquiera serán los denominados *clases*. Para el caso de que sólo existan dos clases se llamará *clasificación binaria*. Si el proceso y el algoritmo se han desarrollado correctamente, la función podrá establecer la clase de nuevos ejemplos aún no etiquetados, es decir, dará un valor de S a cada valor de e .

Uno de los ejemplos más habituales en la actualidad es el algoritmo que emplean los clientes de correo electrónico para clasificar los mensajes nuevos entrantes como *spam* o no.

1.7.2.2 Clasificación suave

Se parte de los mismos conjuntos de la clasificación, E y S $\delta = \{\langle e, s \rangle : e \in E, s \in S\}$. La diferencia con la clasificación ordinaria es que aquí añadimos otra función, $CS: E \rightarrow S, p$ $\Theta: E \rightarrow \mathfrak{N}$, que evalúa el porcentaje de certeza de la función de clasificación $\lambda: E \rightarrow S$. Como es evidente, es muy aconsejable acompañar cualquier clasificación de un grado de certeza para asegurar de este modo unos buenos resultados.

Continuando con el ejemplo del *spam* expuesto en la clasificación, la clasificación suave añadiría una graduación de la posibilidad de que un correo sea basura.

1.7.2.3 Categorización

Esta tarea no pretende el aprendizaje de una función, sino el de una correspondencia. Cada ejemplo de $E\delta = \{\langle e, s \rangle : e \in E, s \in S\}$ puede pertenecer a varias categorías, por lo tanto, la función a aprender $\lambda: E \rightarrow S$ debe ser capaz de asignar varias categorías a un mismo e , mientras que la clasificación sólo es capaz de asignar una. Es decir, cada ejemplo puede tener varias categorías asignadas al mismo tiempo.

Un ejemplo para este tipo de tareas es la categorización de documentos asignando categorías según el tipo de cada uno de ellos.

1.7.2.4 Preferencias o priorización

Se trata de, teniendo dos o más ejemplos de nuestro conjunto de datos, elaborar un orden de preferencia según las características que estemos buscando. Cada ejemplo de nuestra base de datos forma una secuencia de atributos $\langle e_1, e_2, \dots, e_k \rangle : e_i \in E, k \geq 2$. El orden que lleve esta secuencia representará la preferencia. El modelo está representado por un conjunto de datos que se comparan por las preferencias en orden $\delta: \{\langle e_1, e_2, \dots, e_k \rangle : e_i \in E\}$. A pesar de que la tarea es útil para ordenar un conjunto grande de ejemplos, a menudo se emplea para calcular la prioridad entre dos elementos únicos, en otras palabras, para compararlos sólo a ellos dos.

Un ejemplo de este tipo de tareas se puede ver en la contratación inteligente en una empresa. El sistema evaluaría a los candidatos en función de, por ejemplo, puestos de trabajo anteriores, estudios, experiencia, etc. Más tarde elaboraría un orden entre los mismos para dar con el más propicio para la empresa.

Las técnicas de preferencias o priorización también están siendo muy usadas en el campo de la biomedicina para descubrir qué genes afectan más a según qué enfermedades.

1.7.2.5 Regresión

También llamada en ocasiones *interpolación* (si el valor a predecir se encuentra dentro del rango de los valores conocidos) o *estimación* (cuando la tarea es de predicción pura). Debido a la sencillez del modelo con el que trabaja, es una de las tareas más fáciles de explicar. Es similar a la clasificación, ya que su fin es aprender una función real para asignar un valor real a una instancia. Es, por tanto, una tarea de predicción. Se diferencia de la clasificación en que el valor a calcular es numérico. Será prioridad reducir el error al máximo posible entre el valor predicho y el valor real, lo que se conoce como el *error cuadrático medio*.

El conjunto de datos o ejemplos en la regresión viene representado por $e, S: E \rightarrow S$ siendo S el conjunto de salida de los valores. Como sucedía en la clasificación, los ejemplos se llaman *ejemplos etiquetados* al ir acompañados de un valor de S . El símbolo e vuelve a ser el conjunto de datos etiquetado. La función a implementar por la tarea es $R: e \rightarrow S: \lambda: E \rightarrow S$, con un matiz respecto a la clasificación: aquí S tomará valores numéricos y no nominales, en forma de números enteros o reales.

Se emplea habitualmente para modelos de predicción de ventas de empresas o en sectores de calidad para, por ejemplo, estimar el número de unidades defectuosas de una partida.

1.8 Métodos y técnicas

Los métodos suponen el modo de llevar a la práctica la resolución de las tareas mediante la aplicación de técnicas o algoritmos matemáticos. El número de métodos que se pueden hallar es muy elevado debido al ingente campo que abarca la minería de datos. De esta manera, una misma tarea se podrá resolver mediante la aplicación de variados métodos. Como era de esperar, hay un método o técnica para cada situación y su efectividad se verá recompensada con una buena elección del mismo. El conjunto de técnicas que existen es muy variado.

1.8.1 Técnicas algebraicas y estadísticas

Este tipo de técnicas expresan los modelos y patrones mediante la utilización de fórmulas algebraicas, funciones lineales, funciones no lineales, distribuciones, varianzas, correlaciones, etc. Suelen extraer los patrones con la ayuda de un modelo anterior predeterminado, a partir del cual generan unos parámetros o coeficientes. Es por ello que muchas veces reciben el nombre de *técnicas paramétricas*.

Algunas de las técnicas algebraicas y estadísticas más conocidas son las regresiones: logística, logarítmica y lineal. También se engloban dentro de esta clase los discriminantes

lineales y no lineales, basados en funciones predefinidas, los llamados *discriminantes paramétricos*.

1.8.2 Técnicas bayesianas

Utilizan el teorema de Bayes (como su nombre indica) para evaluar la probabilidad de pertenencia a una clase o grupo a través de la estimación de las probabilidades condicionales inversas o *a priori*. Los algoritmos más empleados de este tipo de técnicas son los métodos basados en máxima verisimilitud, el algoritmo EM y el clasificador bayesiano *naive*. Gracias a las técnicas bayesianas es posible representar gráficamente la interacción entre variables e interacciones probabilísticas.

1.8.3 Técnicas basadas en conteos de frecuencias y tablas de contingencia

Permiten contabilizar la frecuencia con la que dos sucesos o más se producen simultáneamente. En el caso de que la base de datos a explotar sea muy grande, hay ciertos algoritmos que empiezan por pares de sucesos y van aumentando el conjunto si y sólo si las frecuencias conjuntas alcanzan un determinado umbral. El algoritmo más famoso y ampliamente empleado en este campo es el de *a priori*.

1.8.4 Técnicas basadas en árboles de decisión y sistemas de aprendizaje de reglas

Este tipo de técnicas se representan en forma de reglas, basándose principalmente en dos tipos de algoritmos diferentes: los algoritmos del tipo “divide y vencerás” (ID3/C4.5 o CART) y los llamados “separa y vencerás” (algoritmo CN2).

1.8.5 Técnicas relacionales, declarativas y estructurales

Emplean un tipo de lenguaje declarativo, esto es, declaran un conjunto de afirmaciones, proposiciones, transformaciones y ecuaciones que describen el problema detallando del mismo modo su posible solución. Dentro de este lenguaje declarativo se encuentran los lenguajes lógicos, funcionales o lógico-funcionales. Las técnicas ILP (programación lógica inductiva) constituyen un buen ejemplo de las técnicas de esta categoría. Ellas mismas han dado nombre a la *minería de datos relacional*.

1.8.6 Técnicas basadas en redes neuronales artificiales

El aprendizaje mediante el entrenamiento es lo que caracteriza a este conjunto de técnicas. El peso de las conexiones de los nodos o neuronas, así como la topología

de la propia red que los conecta, establece el patrón aprendido. Existen multitud de variantes de organización como el perceptrón simple, redes multicapa, redes de base radial, redes de Kohonen, etc. La mayoría de ellas emplean la retropropagación o *backpropagation* como algoritmo.

1.8.7 Técnicas basadas en núcleo y máquinas de soporte vectorial

Aquellas que, mediante transformaciones basadas en la dimensionalidad, tratan de maximizar el margen de las clases y grupos formados. Las transformaciones realizadas son más conocidas como *núcleos* o *kernels*. Estas técnicas ofrecen muchas posibilidades a la hora de trabajar con los diferentes núcleos y márgenes.

1.8.8 Técnicas estocásticas y difusas

Son, junto a las redes neuronales, aquellas técnicas que se enmarcan dentro de la denominada *computación flexible* (*soft computing*). La aleatoriedad en este tipo de técnicas es un componente esencial. Muestra de ello son herramientas como el *simulated annealing*, los métodos evolutivos y genéticos o la utilización de funciones de pertenencia difusas (*fuzzy*).

1.8.9 Técnicas basadas en casos, en densidad o distancia

Están basadas, como indica su nombre, en estudiar la distancia entre el conjunto de elementos. Lo pueden hacer de forma directa o comparando estas distancias con los vecinos más próximos. Los nombres de los algoritmos más usados de este tipo de técnicas son *two-step*, COBWEB (dentro de los algoritmos jerárquicos) y *k* medias (algoritmo no jerárquico).

Análisis estadístico de datos

CAPÍTULO 2

2.1 Introducción

Este capítulo tiene como objetivo ofrecer una revisión breve de los métodos clásicos de análisis de datos que permiten obtener información básica de nuestros datos aplicando técnicas estadísticas. El capítulo tiene dos partes, en la primera se revisan los métodos de estadística descriptiva para resumir las propiedades de cada una de las variables de nuestros datos, y en la segunda se resumen las técnicas de contraste de hipótesis utilizadas para evaluar las relaciones de dependencia entre variables, organizadas según la naturaleza de las variables.

Tras un proceso de selección, recolección y preparación de nuestros datos que aquí no se detalla, en el caso ideal tendremos como punto de partida una tabla de datos organizada en columnas (V_1, \dots, V_M) para cada variable disponible, siendo las filas ($1, \dots, n$) cada una de las observaciones disponibles, también resultantes del proceso de preparación. En el caso de disponer de su evolución temporal, pueden además organizarse según una dimensión temporal tal como se muestra en la figura siguiente:

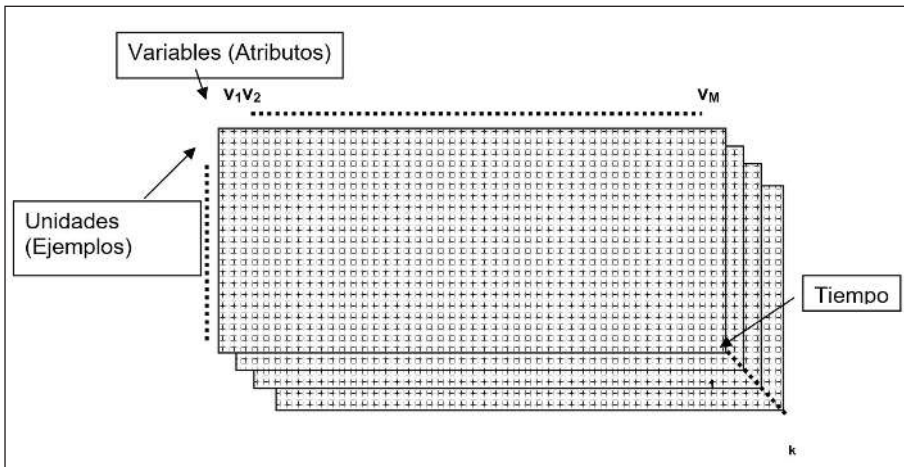


Fig. 2.1. Tabla generalizada para análisis de datos.

Las variables pueden ser de distintos tipos, en el caso más general podemos distinguir los casos principales: variables nominales o categóricas (incluyendo ordinales) y variables numéricas. Las del primer tipo tienen un conjunto discreto de valores posibles, como por ejemplo sexo, nacionalidad, raza, etc., mientras que las del segundo tipo tienen un valor continuo, como altura, potencia, peso, etc. Todas las técnicas mostradas a lo largo del libro se organizan según el tipo de variable con el que trabajemos.

2.2 Análisis de una variable. Estadística descriptiva e inferencia

Se denominan *estadísticos* a aquellos valores que resumen la información contenida en una muestra completa de datos. En general, una muestra la representamos como:

y_i ; $i = 1 \dots n$; tomando la variable y_i sus valores en un rango continuo o discreto.

Los estadísticos se clasifican según el tipo de variable en las siguientes categorías:

- Variables continuas.
 - Medidas centrales (media, moda, mediana).
 - Medidas de dispersión (rango, varianza, desviación estándar, percentiles).
 - Medidas de forma (histograma).

- Variables nominales.
 - Frecuencias relativas (probabilidades), moda.
 - Media y varianza de probabilidad estimada.

En las siguientes secciones revisamos la definición de cada tipo.

2.2.1 Estadísticos de variable continua

- **Media (esperanza) muestral.** Promedio de todos los valores.

$$media(y) = \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad \text{Ec. 2.1}$$

- **Moda.** Valor que aparece más veces.
- **Mediana.** Valor que deja el mismo número de casos a ambos lados.

$$mediana(y) = y_i | N^o \text{ casos } (y_j \leq y_i) = N^o \text{ casos } (y_k \geq y_i) \quad \text{Ec. 2.2}$$

El cálculo de la mediana equivale a ordenar el vector de datos y tomar el valor central, por tanto, es menos sensible que la media frente a valores extremos poco probables.

- **Recorrido (rango).**

$$\max(y_i) - \min(y_i) \quad \text{Ec. 2.3}$$

- **Varianza.** Promedio de desviaciones con respecto al valor medio.

$$Var(y) = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2 = \frac{1}{n-1} \left[\sum_{i=1}^n y_i^2 - n\bar{y}^2 \right] \quad \text{Ec. 2.4}$$

- **Desviación estándar (típica).** Raíz cuadrada de la varianza.

$$desv(y) = \sigma_y = \sqrt{Var(y)} \quad \text{Ec. 2.5}$$

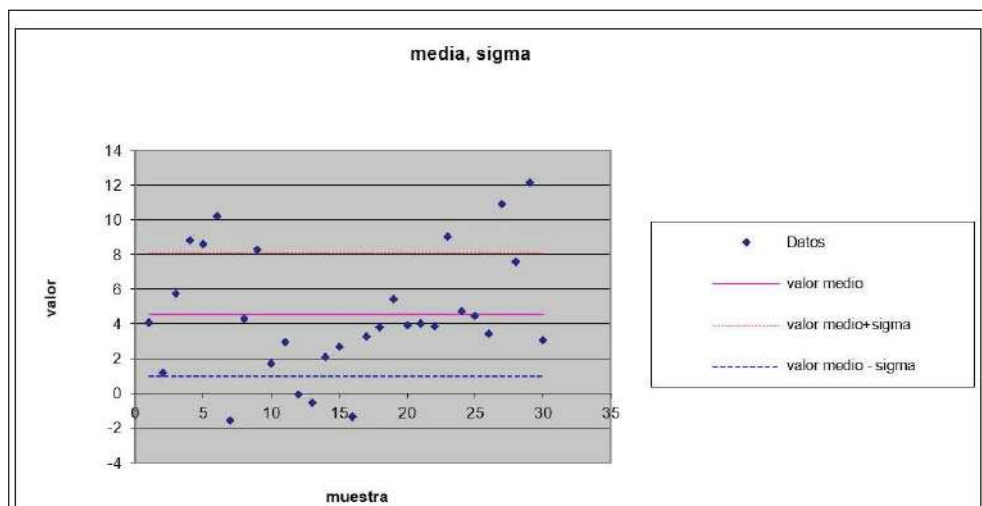


Fig. 2.2. Media y dispersión de una variable continua.

2.2.2 Histograma

El histograma estima cómo se distribuye la probabilidad de una variable a lo largo de su recorrido, y se define como la frecuencia de datos que aparecen a lo largo del recorrido de valores de la variable de y_i por unidad de intervalo. Por tanto, la suma total de frecuencias absolutas es el número de datos o, si trabajamos con frecuencias relativas, la suma deber ser uno.

El histograma acumulado se define como la suma de frecuencias relativas de casos inferiores al valor en abscisas (acumulación de histograma normalizado):

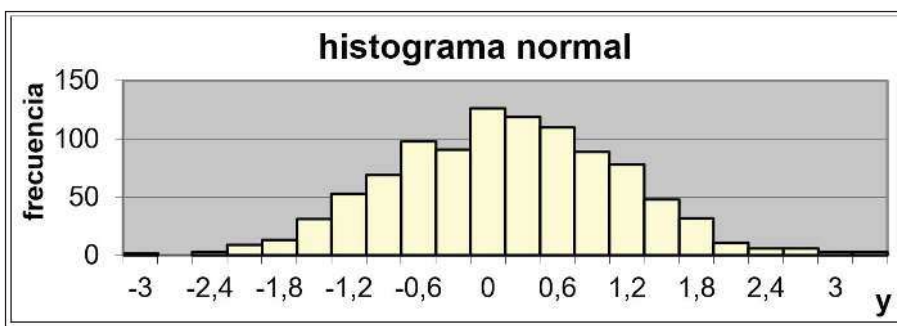


Fig. 2.3. Histograma de variable continua.

De modo que el histograma acumulado es una estimación de $\text{Prob}(Y \leq y_i)$, que, por tanto, en el extremo superior debe ser uno.

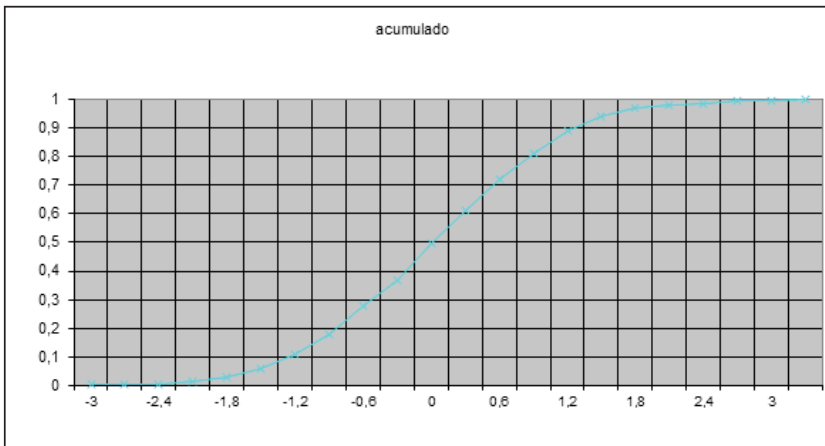


Fig. 2.4. Histograma acumulado de variable continua.

A partir de la distribución estimada por el histograma es interesante obtener los valores de los cuantiles:

- **Cuantil.** Valores que dividen el recorrido de datos en k partes de la misma frecuencia (percentiles: cien partes, cuartiles: cuatro partes, etc.).
- **Percentil p .** Valor que deja debajo al p % de los individuos, y al $(100-p)$ % por encima: se obtienen si entramos en el eje vertical del histograma acumulado y leemos el valor correspondiente en el eje horizontal.
- **Percentil 50.** Mediana (por definición).
- **Percentiles 25, 75.** Cuartiles. Abarcan al 50 % de los individuos (recorrido intercuartílico).

Por ejemplo, para una distribución normal los percentiles alcanzan los valores:

- Percentiles 25, 75: $[-0.674, 0.674]$.
- Percentiles 2.5, 97.5: $[-1.96, 1.96]$.

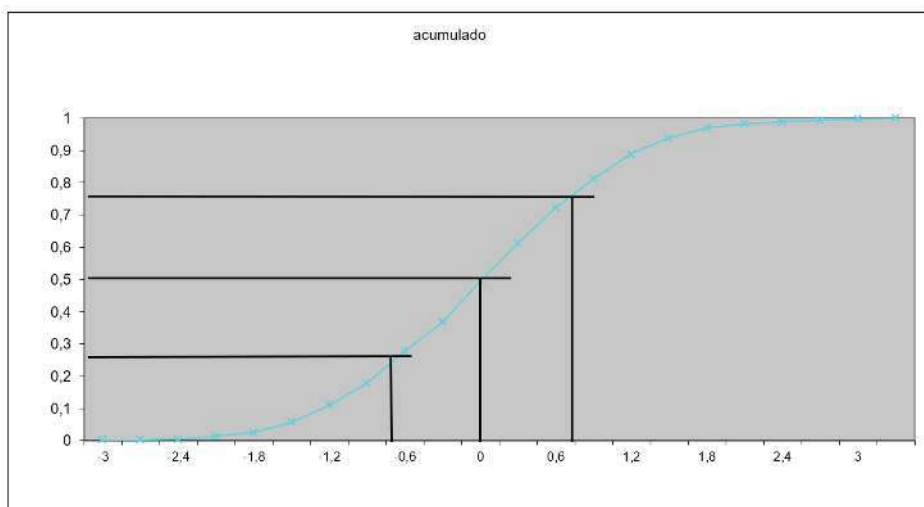


Fig. 2.5. Cuartiles de un histograma.

2.2.3 Estadísticos de variables nominales

En el caso de tener una variable nominal, y_i tomará valores de un conjunto discreto de valores: $\{v_1, \dots, v_k\}$. Podemos caracterizar el número de veces que en la muestra de n datos aparece cada valor: $\{n_1, \dots, n_k\}$, cumpliéndose $n = \sum_{j=1}^k n_j$

- **Moda.** Valor que aparece más veces:

$$v_j | j = \arg \max_j (n_j) \quad \text{Ec. 2.6}$$

- **Histograma.** En este caso se obtendría a partir de la distribución de frecuencias de los valores en las n muestras:

$$\begin{aligned} p_1 &= 100(n_1 / n)\% \\ p_2 &= 100(n_2 / n)\% \\ &\vdots \\ p_M &= 100(n_M / n)\% \\ p_k &= 100(n_k / n)\% \end{aligned} \quad \text{Ec. 2.7}$$

A continuación, se muestra un ejemplo que contiene una variable nominal y una numérica.

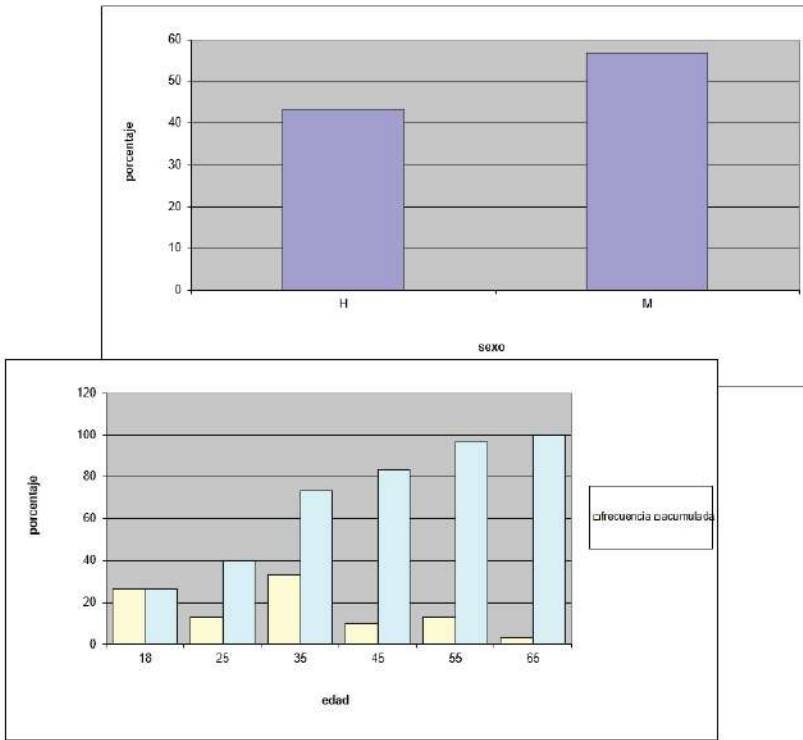


Fig. 2.6. Histograma de variable discreta.

Para las frecuencias estimadas con variables nominales, es posible obtener también una estimación de la media y varianza de frecuencias estimadas.

- **Cálculo de media de p .** Para una categoría dada, se obtiene a partir de los m casos observados en los n datos disponibles.

$$p = m/n \quad \text{Ec. 2.8}$$

Este cálculo puede verse como asignar una variable binaria con cada ejemplo del conjunto de datos:

$$p = \frac{1}{n} \sum_{i=1}^n v_i \quad \text{Ec. 2.9}$$

$v_i = 1$ cada ejemplo en la categoría

$v_i = 0$ en el resto

- **Varianza de p .** De modo análogo, podemos obtener la varianza del estimador de probabilidad p de cada categoría.

$$\begin{aligned} \text{Var}(p) &= \frac{1}{n} \sum_{i=1}^n (v_i - p)^2 = p(1 - p) \\ \sigma_p &= \sqrt{p(1 - p)} \end{aligned} \quad \text{Ec. 2.10}$$

Por tanto, puede verse que el caso de máxima varianza aparece para $p=0.5$ (con una variable binaria), o en general $p=1/k$, siendo k el número de valores que toma la variable discreta (caso de equiprobabilidad).

2.3 Contrastes de hipótesis

Aquí revisamos el procedimiento utilizado para generalizar a partir de una muestra y decidir propiedades que pueden tener los elementos de la población (inferencia de la muestra a la población). Como hemos visto en la sección anterior, los estadísticos resumen la información contenida en una muestra de datos, y el procedimiento para generalizar las conclusiones consiste en la formulación de razonamientos sobre la población que genera la muestra. Los pasos son los siguientes:

- Uso de distribuciones teóricas de probabilidad para caracterizar los estimadores (hipótesis de análisis).
- Relación de los estadísticos obtenidos (y_i) con los estimadores (Y_i).
- Cuantificación de la probabilidad de los resultados si se cumplen las hipótesis (nunca se garantiza con certeza absoluta).

El procedimiento anterior podría llevarse a cabo a la inversa y deducir propiedades esperadas de la muestra a partir de la población, aunque el interés es más teórico.

2.3.1 Distribuciones de probabilidad

La distribución de probabilidad es el modelo teórico que representa el comportamiento de una variable aleatoria, puede verse como la curva teórica a la que tiende un histograma con muchos datos haciendo los intervalos arbitrariamente pequeños. Se distingue la función de distribución y de densidad de distribución de probabilidad:

- Función distribución de probabilidad de X: $F_X(x)$

$$F_X(x) = P(X \leq x); \quad -\infty < x < \infty \quad \text{Ec. 2.11}$$

- Función densidad de probabilidad de X: $f_X(x)$

$$f_X(x) = \frac{dF_X(x)}{dx}; \quad -\infty < x < \infty$$

$$F_X(x) = \int_{-\infty}^x f_X(x)dx; \quad P(a \leq X \leq b) = \int_a^b f_X(x)dx \quad \text{Ec. 2.12}$$

2.3.1.1 Distribución normal

Es una curva de gran interés porque explica el comportamiento de los datos en muchas situaciones. Fue aplicada por primera vez como distribución por A. Quetelet en 1830 y, en general, explica bien variables en las que la aleatoriedad se debe a múltiples efectos independientes.

$$f(z) = \frac{1}{\sqrt{2\pi}} \exp \left[-\frac{1}{2} z^2 \right] \quad \text{Ec. 2.13}$$

Pueden destacarse estas propiedades de la distribución normal:

- Distribución simétrica: coinciden media y mediana en 0.
- Se dispone del valor de la distribución de probabilidad: área bajo la curva de $f_z(z)$ para cualquier valor.

Es muy habitual **tipificar o estandarizar** las variables aleatorias, tanto la distribución normal como cualquier otra. Para ello se mide el desplazamiento respecto a la media en unidades de desviación típica:

$$z_i = \frac{y_i - \bar{y}}{\sigma_i} \quad \text{Ec. 2.14}$$

z	$F_Z(z)$
-3	0,001349967
-2,5	0,00620968
-2	0,022750062
-1,5	0,066807229
-1	0,15865526
-0,5	0,308537533
0	0,5
0,5	0,691462467
1	0,84134474
1,5	0,933192771
2	0,977249938
2,5	0,99379032
3	0,998650033

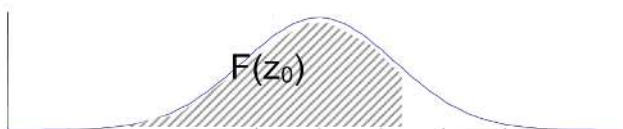


Fig. 2.7. Distribución normal de probabilidad.

A partir de las tablas de la distribución se calculan los intervalos de confianza, definidos como los intervalos con respecto a la media que se corresponden con una probabilidad, destacando el de una cola (asimétrico) y el de dos colas (simétrico).

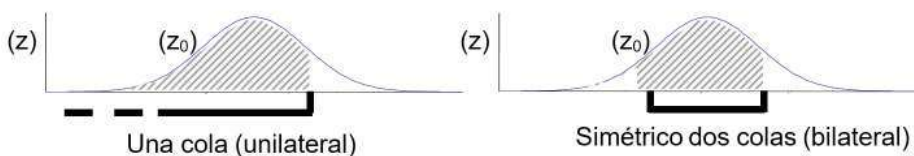


Fig. 2.8. Test de una y dos colas sobre distribución normal.

A continuación se muestra un ejemplo operativo de utilización de la distribución normal para razonar acerca de probabilidades asociadas a intervalos definidos sobre la variable aleatoria:

Se conocen estos parámetros de una población con distribución normal: media: $m=115$; desviación típica: $s=20$.

- ¿Probabilidad de observar valores inferiores a 70? $z=(70-115)/20$, $F(z)=0,012$.
- ¿Probabilidad de observar valores superiores a 150? $z=(150-115)/20$, $1-F(z)=0,04$.
- ¿Probabilidad de observaciones en intervalo 90-130? $F((130-115)/20)-F((90-115)/20)=0,667$.
- ¿Qué intervalos simétricos tienen el 80 % y 95 % de los casos (intervalos de confianza)? $z=F^{-1}(\alpha/2)$; $y=m \pm zs$.
 - 80 %: $z_{0.1}=1,28$; $115 \pm z_{0.1} \cdot 20 = [89.3, 140.6]$.
 - 95 %: $z_{0.025}=1,96$; $115 \pm z_{0.025} \cdot 20 = [75.8, 154.2]$.

2.3.2 Inferencia

Finalmente, el objetivo de la inferencia es, dado unos estadísticos obtenidos a partir de una muestra sacada al azar, razonar acerca del verdadero parámetro de la población, que habitualmente es desconocido, una variable objeto de la estimación.

Para ello, a partir de una muestra aleatoria (n datos) de la población: $y_i, i=\{1, \dots, n\}$, podemos obtener los estadísticos, que a su vez son variables aleatorias. Por ejemplo, para el estadístico de la media, \bar{y} , podemos deducir estas propiedades:

$$\begin{aligned}\bar{y} &= \frac{1}{n} \sum_{i=1}^n y_i \\ E(\bar{y}) &= \frac{1}{n} \sum_{i=1}^n E(y_i) = \bar{Y} \\ Var(\bar{y}) &= \frac{1}{n^2} \sum_{i=1}^n Var(y_i) = \frac{1}{n} Var(\bar{Y}); \quad \sigma_{\bar{y}} = \frac{1}{\sqrt{n}} \sigma_Y\end{aligned}\tag{Ec. 2.15}$$

En cuanto a la distribución que sigue, es aplicable un resultado fundamental de enorme utilidad, el **teorema del límite central**:

“Una muestra suficientemente grande de una población con distribución arbitraria tendrá estadístico media con distribución normal”.

Como consecuencia, el intervalo de confianza de la media puede obtenerse a partir de la distribución normal (mayor “normalidad” cuanto mayor sea el tamaño de las muestras):

$$\bar{y} = \bar{Y} \pm z \frac{1}{\sqrt{n}} \sigma_Y$$

Ejemplo de límite central:

- Población: mil individuos, cuatrocientas mujeres, seiscientos hombres.

$$P = 0.4; \quad \sigma = \sqrt{P(1-P)} = 0.49$$

- Muestras de diez individuos.

$$\begin{aligned}p &= \sum_{i=1}^{10} y_i; \quad E(p) = P = 0.4; \\ \sigma_p &= \frac{1}{\sqrt{10}} \sqrt{P(1-P)} = 0.155\end{aligned}$$

- Intervalo de confianza al 95 % (con distribución normal):
 - Influye:
 - ◊ Intervalo de confianza (z): “garantía” de no equivocarnos.
 - ◊ Tamaño de muestra (n).
 - ◊ Variabilidad de población (p).

$$p = \frac{1}{10} \sum_{i=1}^{10} y_i ;$$

$$E(p) = P = 0.4;$$

$$\sigma_p = \frac{1}{\sqrt{10}} \sqrt{P(1-P)} = 0.155$$

$$P \pm 1.96\sigma_p = [0.1, 0.7]$$

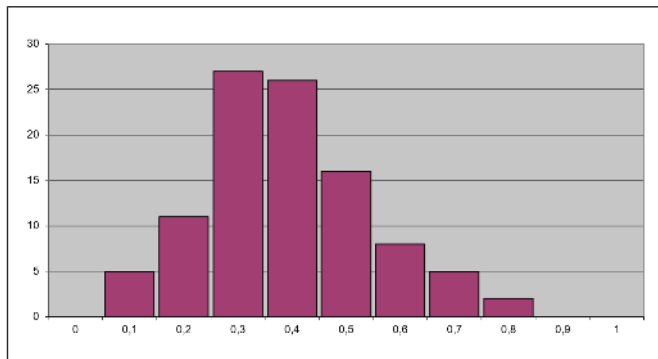


Fig. 2.9. Histograma de promedios calculados.

- Si las muestras fueran de cincuenta individuos:

$$p = \frac{1}{50} \sum_{i=1}^{50} y_i ;$$

$$\sigma_p = \frac{1}{\sqrt{50}} \sqrt{P(1-P)} = 0.069$$

$$P \pm 1.96\sigma_p = [0.26, 0.54]$$

2.3.3 Evaluación de hipótesis

La evaluación de hipótesis pretende validar o rechazar ideas preconcebidas a partir del análisis de los datos disponibles, generalizando las conclusiones.

Pasos:

- Generación de hipótesis.
- Determinar qué datos son necesarios. Recolectar y preparar.
- Evaluación de hipótesis para aceptar o rechazar.

Contrastes de hipótesis:

- Contrastar es medir la probabilidad de que el estadístico obtenido en una muestra sea fruto del azar.
- Formulación del modelo e hipótesis. Se conoce la distribución del estadístico bajo condiciones de hipótesis.
- Hipótesis nula (**H₀**). Es lo que dudamos y queremos contrastar. Ejemplo: ¿el porcentaje total es 10 %?, ¿la media de los ingresos es superior a cinco?
 - Bajo H₀, el estadístico sigue el modelo, y la diferencia observada es únicamente fruto del azar.
- Hipótesis alternativas. Alternativas que permiten rechazar la hipótesis nula: probabilidad distinta de 10 %, media menor a cinco, etc.
- Rechazar hipótesis H₀: hay evidencia para negar H₀.
- No rechazable: no hay evidencia estadística para hacerlo (no implica demostrar su veracidad).

Contrastes con normal y varianza conocida:

Uno de los casos más frecuentes de evaluación de hipótesis aparece con distribuciones de tipo normal. En el caso de estimación de frecuencias, el promedio puede suponerse normal si tenemos suficientes datos ($n > 30$), y la varianza viene dada por la expresión anterior, $\text{Var} = P(1-P)$):

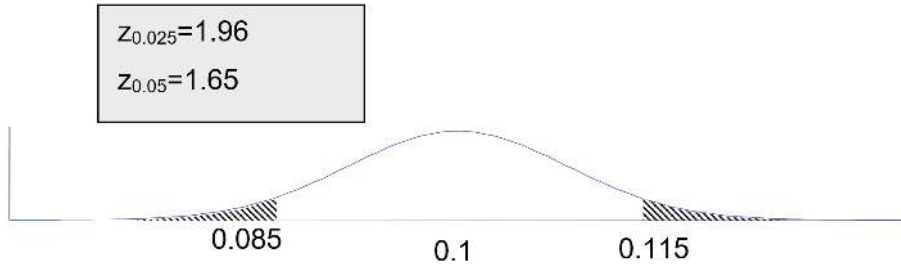
- Contraste de dos colas (bilateral): deja la mitad a cada lado, $\alpha/2$.

Ejemplo: hipótesis nula H₀: $P = 10\%$.

$$\sigma_p = \sqrt{0.1 * \frac{1 - 0.1}{1500}}; \quad p \in [0.085 \quad 0.115]$$

Hipótesis alternativa:

$P \neq 10\%$



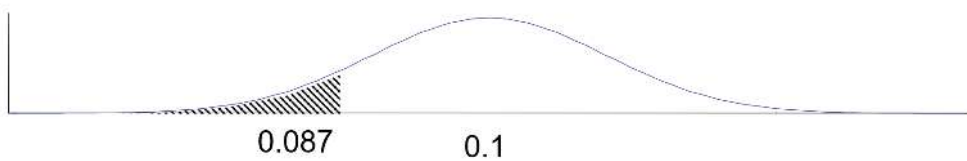
Región crítica: $-1.96 < z < 1.96$

- **Contraste de una cola (unilateral):** deja a un solo lado α .

Ejemplo: hipótesis nula H_0 :

$$p > P - 1.65\sigma_p = 0.087$$

Hipótesis alternativa: $P < 10\%$



En este caso la región crítica viene dada por: $z > 1.65$

Contraste con varianza estimada:

- La variable $(y_i - \bar{y})/s$ no es exactamente la normal tipificada (s es estimada):
- Distribución **t-Student**. Parámetro grados de libertad: $n-1$.

Como puede apreciarse se ensanchan los intervalos de confianza (sólo si hay pocos datos).

	Student (N=9)	Student (N=50)	Student (N=100)	Normal
Prob[X>z]	z			z
0,10%	4,30	3,26	3,17	3,09
0,50%	3,25	2,68	2,63	2,58
1%	2,82	2,40	2,36	2,33
2,50%	2,26	2,01	1,98	1,96
5%	1,83	1,68	1,66	1,64
10%	1,38	1,30	1,29	1,28
20%	0,88	0,85	0,85	0,84

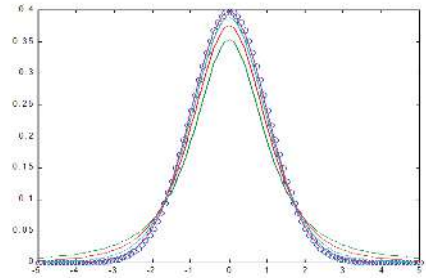


Fig. 2.10. Distribución normal y distribución t-Student.

<p>σ conocida</p> <p>estadístico</p> $\frac{\bar{y} - \mu}{\sigma/\sqrt{n}} \rightarrow N(0,1)$ <p>Int. confianza</p> $\bar{y} \pm \frac{z_{\alpha/2} \sigma}{\sqrt{n}}$	<p>σ desconocida</p> <p>estadístico</p> $\frac{\bar{y} - \mu}{\sigma/\sqrt{n}} \rightarrow t_{n-1}(0,1)$ <p>Int. confianza</p> $\bar{y} \pm \frac{t_{\alpha/2, n-1} \sigma}{\sqrt{n}}$
--	--

Ejemplo de intervalos con t-Student:

Los valores del pH de una piscina en diez determinaciones son: 6,8; 6,78; 6,77; 6,8; 6,78; 6,8; 6,82, 6,81; 6,8 y 6,79. Utilizando normal y t-Student, hallar:

- Intervalo de confianza 95 % para media poblacional.
- Intervalo de confianza 65 % para media poblacional.
- Contrastar hipótesis nula de que la media poblacional es 6,8 con niveles de significación $\alpha=0,05$ y $\alpha=0,35$.

Normal:

media 95%: [6,765, 6,825], media 65%: [6,781, 6,809]

t-Student:

media 95%: [6,761, 6,829], media 65%: [6,780, 6,801]

2.4 Análisis de relaciones entre variables. Evaluación de hipótesis

El objetivo del análisis entre los atributos que definen los datos es ver el tipo de interrelación o dependencia que existe entre los valores de dichos atributos. Este análisis se lleva a cabo haciendo uso de los datos disponibles para tener “evidencia estadística” que permita validar o refutar hipótesis que pretendan explicar las relaciones.

La herramienta o técnica que permite llevar a cabo este tipo de análisis es el denominado *test de hipótesis*, que se define de manera distinta en función del tipo de atributos con los que estemos trabajando. De esta manera, en función del tipo de atributo tenemos:

- **Nominales-nominales.** En este caso, los dos atributos toman valores de un conjunto de posibles valores (por ejemplo: norte, sur, este y oeste). La relación entre las variables se obtiene mediante las tablas de contingencia.
- **Nominales-numéricos.** En este caso, uno de los atributos toma valores discretos de un conjunto de posibles categorías y el otro toma valores numéricos. La relación entre los atributos se obtiene mediante la comparación de medias y el análisis de varianza.
- **Númericos-numéricos.** En este caso, los dos atributos toman valores numéricos. La relación entre los dos atributos se obtiene mediante el análisis de regresión y covarianza.

Más adelante se contemplarán más casos de contrastes de hipótesis.

2.4.1 Relación entre variables nominales-nominales

El objetivo es analizar la interrelación (dependencia) entre los valores de variables nominales. En este caso la herramienta de análisis para dos variables es la denominada **tabla de contingencia**. En esta tabla se calcula la distribución de los casos (las frecuencias de aparición) para las distintas combinaciones de valores de las dos variables, como se observa en la figura siguiente.

		Variable 2				totales 1
		valor 1	valor 2	...	valor p2	
Variable 1	valor 1	n_{11}	n_{12}	...	n_{1p2}	t_1
	valor 2	n_{21}	n_{22}	...	n_{2p2}	t_2

	valor p1	n_{p11}	n_{p12}	...	n_{p1p2}	t_{p1}
	totales 2	t'_1	t'_2	...	t'_{p2}	t

Fig. 2.11. Tabla de contingencia.

A partir de la tabla de contingencia podemos calcular las probabilidades marginales de los valores de la variable 1 como $P_i = t_i/t$, que representa la probabilidad de que la variable 1 tome el valor i . Del mismo modo podemos calcular las probabilidades para la variable 2 como $P_j = t'_j/t$.

A partir de las probabilidades marginales podemos calcular los casos “esperados”, bajo la hipótesis de independencia entre variables. Para calcular el valor esperado se multiplica el número total de casos por la probabilidad de que la variable 1 tome el valor i y la variable 2 tome el valor j , es decir, $E_{ij} = t(t_i/t)(t'_j/t) = t_i t'_j / t$. Obsérvese que únicamente bajo la hipótesis de independencia podemos calcular la probabilidad conjunta como un producto de probabilidades.

La técnica de análisis estadístico que se aplica para la relación entre dos variables nominales es el contraste chi-cuadrado. Las características de este test son:

- Es aplicable en análisis bivariable (normalmente clase versus atributo).
- Determina si es rechazable la hipótesis de que dos variables son independientes:
 - Bajo hipótesis H_0 se determinan los casos en el supuesto de variables independientes. Los valores esperados se determinan con probabilidades marginales de las categorías: $E_{ij} = t P_i P_j$ (valores esperados).
 - El estadístico chi-cuadrado mide la diferencia entre los valores esperados y los valores observados, de modo que su expresión es:

$$\chi^2 = \sum_{i=1}^{p1} \sum_{j=1}^{p2} (E_{ij} - O_{ij})^2 / E_{ij} \quad \text{Ec. 2.16}$$

La expresión anterior, χ^2 , bajo las condiciones de H_0 sigue una distribución conocida denominada *distribución chi-cuadrado*, caracterizada por el parámetro “grados de libertad” que es el $(n^\circ \text{ de filas} - 1)(n^\circ \text{ de columnas} - 1)$. Cuando no se cumple la hipótesis H_0 las variables son dependientes.

Por lo tanto, se formula un test de hipótesis para determinar el valor de chi-cuadrado para esa hipótesis. La distribución chi-cuadrado está tabulada:

probabilidad chi2 supera estadístico	valor estad.						
grados de libertad	5	6	7	8	9	10	11
1	0,025	0,014	0,008	0,005	0,003	0,002	0,001
2	0,082	0,050	0,030	0,018	0,011	0,007	0,004
3	0,172	0,112	0,072	0,046	0,029	0,019	0,012
4	0,287	0,199	0,136	0,092	0,061	0,040	0,027
5	0,416	0,306	0,221	0,156	0,109	0,075	0,051
6	0,544	0,423	0,321	0,238	0,174	0,125	0,088
7	0,660	0,540	0,429	0,333	0,253	0,189	0,139

El test lo que calcula es la probabilidad de que la diferencia entre el valor observado y el valor esperado supere un cierto umbral.

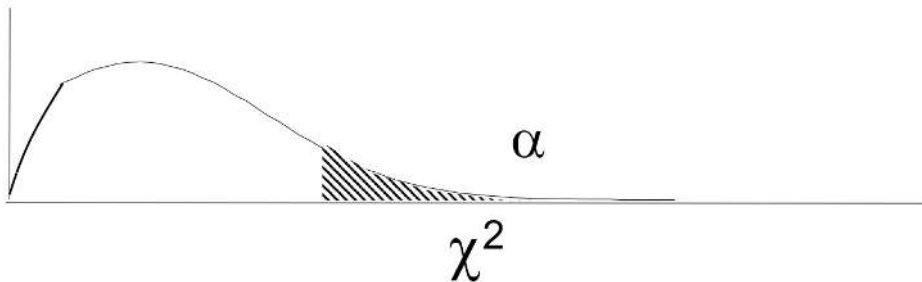


Fig. 2.12. Representación gráfica del test chi-cuadrado.

2.4.2 Relaciones numéricas-nominales

La técnica que se utiliza para establecer posibles relaciones entre dos variables, una de ellas numérica y la otra nominal (o entre dos nominales si trabajamos con proporciones), es la técnica de la comparación de medias y proporciones. Ésta mide la relación entre variables numéricas y nominales, o nominales y nominales (proporciones), determinando si es rechazable la hipótesis de que las diferencias de medias o proporciones condicionadas a las etiquetas de la variable nominal son debidas al azar. Es decir, que se calcula el impacto de la variable nominal sobre la continua.

Existen dos tipos de análisis según si tenemos dos medias o proporciones o un número mayor de dos. Si tenemos dos medias o proporciones se calcula la significatividad de la diferencia. Si tenemos más de dos valores distintos se realiza un análisis de varianza.

2.4.2.1 Comparación de dos medias

En este caso tenemos dos subpoblaciones, una para cada grupo, cada una con su media y varianza. Las hipótesis que podemos establecer son:

- H_0 . La diferencia de medias en la población es nula $D=0$.
- Hipótesis alternativa A. Las medias son distintas: $D \neq 0$.
- Hipótesis alternativa B. La media de la diferencia es mayor que 0: $D > 0$.
- Hipótesis alternativa C. La media de la diferencia es menor que 0: $D < 0$.

Como vemos, no hay una única posibilidad de hipótesis alternativa sino varias, con diferentes intervalos de rechazo en función de la información que tengamos *a priori*. En este ejemplo, si no tenemos información decimos que la hipótesis alternativa sería la hipótesis A (medidas distintas), mientras que las hipótesis B y C incluyen información

a priori de que, o bien las medidas son iguales, o bien una es mayor que la otra (hipótesis B y C). Además, para la comparación de las variables numéricas de dos clases, las situaciones posibles que podemos encontrarnos dentro de la muestra total son:

- Muestras independientes: conjuntos distintos.
- Muestras dependientes: es decir, las muestras pertenecen al mismo conjunto, con dos variables a comparar en cada ejemplo.

Cuando el número de muestras es muy elevado para cada grupo, las muestras siguen una distribución normal, de modo que las hipótesis anteriormente expuestas se evalúan mediante los valores de una gaussiana estándar. De esta manera se calcularía la media de la diferencia y su varianza y se aplicaría al cálculo de probabilidades de una gaussiana estándar. En el caso de la hipótesis A se utilizarían las dos colas de la gaussiana y en el caso de la hipótesis B utilizaríamos una única cola, como se observa en la siguiente figura.

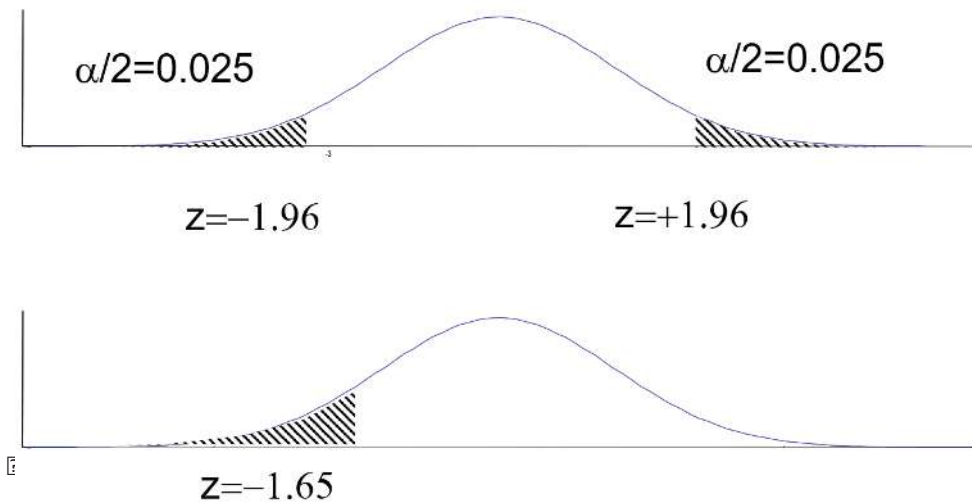


Fig. 2.13. Representación gráfica de comparación de dos medias mediante una gaussiana.

Cuando las muestras son pequeñas no es válida la hipótesis de normalidad de los estadísticos de medias y el test se realiza considerando una distribución t-Student:

$$\bar{y} \pm t_{\frac{\alpha}{2}, GL} \sigma \quad \text{Ec. 2.17}$$

El proceso para el cálculo cuando las muestras son independientes (test no pareado) es:

- En cada muestra (tamaños n_1, n_2) obtenemos las medias y varianzas:

$$\bar{y}_1, \bar{y}_2, \sigma_{\bar{y}_1}, \sigma_{\bar{y}_2}$$

- Se calcula la diferencia:

$$d = \bar{y}_1 - \bar{y}_2$$

Ec. 2.18

- Varianza de la diferencia:

$$\sigma_d^2 = \frac{\sigma_{\bar{y}_1}^2}{n_1} + \frac{\sigma_{\bar{y}_2}^2}{n_2}$$

Ec. 2.19

- Los grados de libertad de la t-Student se evalúan según la varianza:
 - Distinta varianza (heteroscedasticidad): $gl = \min(n_1, n_2)$.
 - Misma varianza (homoscedasticidad): $gl = n_1 + n_2 - 2$.

El proceso de cálculo cuando las muestras son dependientes (test pareado) se fundamenta en que se dispone de la diferencia en cada uno de los ejemplos y no en que tenemos dos variables (ejemplo: cambio en el tiempo de una variable para todos los ejemplos d_1, d_2, \dots, d_n): $d_i = d_{1i} - d_{2i}$. En este caso, todo es equivalente al caso anterior pero los cálculos son:

$$\bar{d} = \frac{1}{n} \sum_{i=1}^n d_i; \quad \sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (d_i - \bar{d})^2; \quad \sigma_{\bar{d}} = \frac{1}{\sqrt{n}} \sigma$$

Ec. 2.20

2.4.2.2 Análisis de la varianza

Esta técnica también mide la relación entre variables numéricas y nominales, pero en este caso se descompone la variabilidad del resultado en varios componentes:

- Efectos de factores representados por otras variables.
- Efectos de error experimental.

La técnica del análisis de la varianza simple (ANOVA) considera un solo factor con varios niveles nominales. Para cada nivel se tiene una serie de observaciones y el modelo: $Y_{ij} = \mu_i + u_{ij}$ representa ruido con la misma varianza por nivel, donde i varía entre 1 y el número de niveles (variable nominal) y j varía entre 1 y el número de datos por nivel. Además de esta técnica existe la técnica MANOVA, que es un modelo multifactorial de la varianza. En este modelo se definen I niveles, cada uno de ellos representado por un conjunto de muestras, como se puede observar en la siguiente figura, y donde cada nivel está representado por una media y una varianza.

	Factor B	1	2	...	r
Factor A					
1		X_{111}	X_{121}	...	X_{1r1}
		X_{112}	X_{122}	...	X_{1r2}
	
		X_{11n1}	X_{12n1}	...	X_{1rn1}
2		X_{211}	X_{221}	...	X_{2r1}
		X_{212}	X_{222}	...	X_{2r2}
	
		X_{21n1}	X_{22n1}	...	X_{2rn1}
...	
t		X_{t11}	X_{t21}	...	X_{tr1}
		X_{t12}	X_{t22}	...	X_{tr2}
	
		X_{t1n1}	X_{t2n1}	...	X_{trn1}

Fig. 2.14. Descomposición en niveles con la técnica MANOVA.

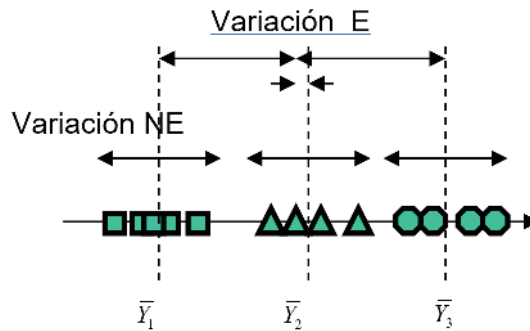


Fig. 2.15. Representación gráfica de los niveles de la técnica MANOVA.

El análisis MANOVA evalúa las siguientes variables:

- Número total de elementos:

$$n = \sum_{i=1}^I n_i$$

Ec. 2.21

- Media por nivel:

$$\bar{Y}_i = \frac{1}{n_i} \sum_{j=1}^I Y_{ij} \quad \text{Ec. 2.22}$$

- Media total:

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^I \sum_{j=1}^{n_i} Y_{ij} \quad \text{Ec. 2.23}$$

- Relación entre “cuadrados”:

$$\sum_{i=1}^I \sum_{j=1}^{n_i} (Y_{ij} - \bar{Y})^2 = \sum_{i=1}^I \sum_{j=1}^{n_i} (Y_{ij} - \bar{Y}_i)^2 + \sum_{i=1}^I n_i (\bar{Y}_i - \bar{Y})^2 \quad \text{Ec. 2.24}$$

Y realiza una estimación de varianzas de la siguiente manera:

- Varianza intergrupo (*between*) (I-1 grados de libertad):

$$S_b = \frac{1}{I-1} \sum_{i=1}^I n_i (\bar{Y}_i - \bar{Y})^2 \quad \text{Ec. 2.25}$$

- Varianza intragrupo (*within*) (n-I grados de libertad):

$$S_w = \frac{1}{n-I} \sum_{i=1}^I \sum_{j=1}^{n_i} (Y_{ij} - \bar{Y}_i)^2 \quad \text{Ec. 2.26}$$

- Varianza total (n-1 grados de libertad):

$$S = \frac{1}{n-1} \sum_{i=1}^I \sum_{j=1}^{n_i} (Y_{ij} - \bar{Y})^2 \quad \text{Ec. 2.27}$$

La hipótesis que planteamos o la pregunta que queremos responder es: ¿es significativamente mayor que la unidad la relación entre la varianza intergrupo e intragrupo, $f=S_b/S_w$? Para ello, debemos realizar un contraste de hipótesis de cociente de varianzas muestrales, que sigue una distribución **F de Fisher-Snedecor**: $F(x, I-1, n-I)$, como se ve en la figura siguiente.

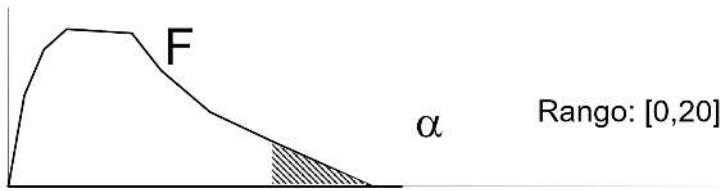


Fig. 2.16. Representación de la distribución F de Fisher-Snedecor.

Al igual que en los casos anteriores, este test permite rechazar o no la hipótesis de que el cociente entre varianzas estimadas se deba al azar.

2.4.3 Relaciones numéricas-numéricas

En general, se pueden calcular factores de correlación entre cualquier par de variables numéricas: indica el grado de relación lineal existente. La **matriz de covarianza** indica el grado de relación lineal existente. Para ello se calcula de la siguiente manera, tomando todas las variables existentes en el vector \vec{X} :

$$C_{\vec{X}} \equiv \frac{1}{n} \sum_{i=1}^n (\vec{X}_i - \bar{\mu})(\vec{X}_i - \bar{\mu})^t = \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1, x_2) & \cdots & \text{cov}(x_1, x_I) \\ \text{cov}(x_1, x_2) & \text{var}(x_2) & & \vdots \\ \vdots & & \ddots & \vdots \\ \text{cov}(x_1, x_I) & \cdots & \cdots & \text{var}(x_I) \end{bmatrix} \quad \text{Ec. 2.28}$$

Donde las medias se sustraen de las variables para calcular la covarianza:

$$\bar{\mu} = \frac{1}{n} \sum_{i=1}^n \vec{X}_i \quad \text{Ec. 2.29}$$

La **matriz de correlación** es similar, en realidad, es la matriz de covarianza normalizada con las varianzas de cada par de variables:

Ec. 2.30:

$$\text{corr}_{\vec{X}} \equiv \begin{bmatrix} \frac{\text{var}(x_1)}{\sigma_{x_1}^2} & \frac{\text{cov}(x_1, x_2)}{\sigma_{x_1} \sigma_{x_2}} & \cdots & \frac{\text{cov}(x_1, x_I)}{\sigma_{x_1} \sigma_{x_I}} \\ \frac{\text{cov}(x_1, x_2)}{\sigma_{x_2} \sigma_{x_1}} & \frac{\text{var}(x_2)}{\sigma_{x_2}^2} & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\text{cov}(x_1, x_I)}{\sigma_{x_I} \sigma_{x_1}} & \cdots & \cdots & \frac{\text{var}(x_I)}{\sigma_{x_I}^2} \end{bmatrix} = \begin{bmatrix} 1 & \rho_{x_1 x_2} & \cdots & \rho_{x_1 x_I} \\ \rho_{x_1 x_2} & 1 & & \vdots \\ \vdots & & \ddots & \vdots \\ \rho_{x_1 x_I} & \cdots & \cdots & 1 \end{bmatrix}$$

$$\text{Siendo } \sigma_{x_i}^2 = \frac{1}{n-1} \sum_{j=1}^n (x_j^i - \mu^i)^2 \quad \text{Ec. 2.31}$$

En esta sección se han presentado las técnicas de análisis estadístico que permiten evaluar las relaciones entre variables según su naturaleza. Estas técnicas sirven de base para desarrollar los procedimientos analíticos de clasificación y regresión que se presentarán en capítulos posteriores.

Introducción al lenguaje R. Lectura, procesamiento y visualización de datos: *data wrangling*

3.1 Carga y transformaciones de datos

Gran parte del proceso de análisis de datos se consume en las transformaciones que tienen que aplicarse a los datos para que puedan ser procesados y en la presentación de los resultados de la forma más inteligible y clara posibles. Ambos procesos son clave para el éxito del análisis y en parte pueden guiar las técnicas que pueden emplearse.

Al proceso de transformación se le denomina *data wrangling* o *data munging*, el primer término (de uso más extendido) significa literalmente “peleando o riñendo con los datos”, mientras que el segundo hace referencia a los cambios irreversibles que se realiza a un conjunto de datos para transformarlos en otros distintos. Esta etapa, típicamente, consume el 70 % del tiempo y del presupuesto de los proyectos.

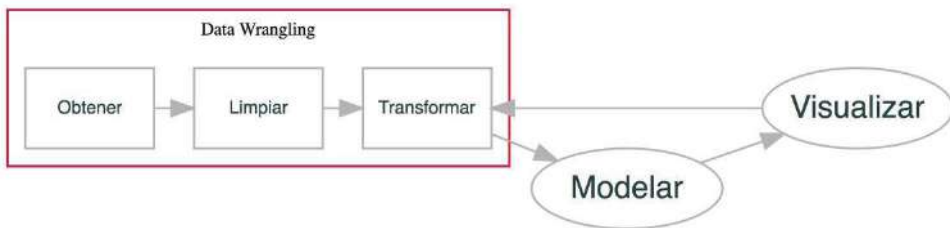


Fig. 3.1. Esquema de proceso de datos.

El *data wrangling* es bastante similar a un proceso ETL (*Extract, Transform and Load*) en cuanto a las transformaciones que se pueden aplicar, pero difieren en algunos aspectos. El ETL actualiza la información de un *data warehouse*, la preparación y transformación de los datos tienen sólo este fin, incrementar o actualizar la cantidad de datos disponibles. Por contra, el objetivo del *data wrangling* es adecuar los datos a las técnicas de

extracción de información que se aplicarán sobre ellos. A partir de los datos “en crudo” que pueden proceder de diferentes fuentes, que tienen distintas estructuras, se aplican algunas de las siguientes transformaciones:

- Muestrear datos.
- Crear nuevos atributos.
- Discretizar atributos cuantitativos continuos.
- Fusionar y reordenar datos y atributos.
- Completar datos.
- Reescalar magnitudes.
- Creación de variables *dummy*.
- Reducción de dimensionalidad.
- Eliminación de datos espurios.

Para algunas de estas transformaciones se podrían dedicar capítulos completos, aquí se darán algunas pautas generales para realizar muchas de ellas.

3.1.1 Estructura básica de datos

El primer paso consiste en la carga de datos “crudos”. El formato tabular es el más común para tratar los datos, organizando por filas los diferentes registros, cuyas dimensiones están ordenadas en las columnas. Esta organización de los datos es útil para los algoritmos de clasificación, agrupamiento, extracción de reglas, regresiones, redes de neuronas artificiales, donde la entrada de datos requiere de un vector n -dimensional de atributos, homogéneo para todos los registros.

Las estructuras de datos básicas que pueden manejarse en R son:

- Vectores.
- Matrices.
- Tablas.
- Listas.
- Hoja de datos (*data frame*).
- Factores.

Un vector es un conjunto de elementos del mismo tipo. Los tipos básicos de R son: numéricos (enteros o reales), lógicos, complejos y cadenas de caracteres. Los datos en un vector pueden estar en forma de fila o columna. La extensión natural de los vectores son las matrices (*matrix*), estructura bidimensional de datos con todos sus elementos

del mismo tipo. Para extender el número de dimensiones, hay que utilizar la estructura de tablas (*array*).

Cuando se quiere tener un conjunto de datos de tipos heterogéneos, entonces hay que utilizar listas y *data frames* (se va a utilizar el término en inglés dado que es el que habitualmente se usa). Las listas son en esencia estructuras de datos unidimensionales, aunque puede “expandir” su dimensión creando una lista cuyos elementos son también listas. Los *data frames* pueden contener datos heterogéneos por columnas, pero todos los elementos de una misma columna tienen el mismo tipo. Es la estructura más utilizada para organizar los datos que serán procesados por otros algoritmos.

Por último, los factores, empleados para las variables de tipo nominal o categórica, son de especial interés para el modelado de estadístico. Internamente se tratan como un vector de enteros que referencia al conjunto de factores de la variable. Por ejemplo, los días de la semana o el género. Este tipo de estructura de datos es muy conveniente cuando se abordan técnicas de aprendizaje automático supervisado como la extracción de reglas o árboles de decisión entre otras.

3.1.2 Lectura de fichero

Hay una gran variedad de formas para adquirir los datos a procesar. Se puede conectar con bases de datos, descarga de datos de webs de Internet, extraer de hojas de cálculo, consultas en redes sociales o ficheros con multitud de formatos. Se han desarrollado librerías con funciones específicas para cada uno de estos tipos de operaciones de carga de datos. Utilizar las funciones adecuadas puede simplificar mucho el proceso de lectura de datos.

Los datos que se van a utilizar como ejemplo de la aplicación de algoritmos y técnicas de extracción de información se han tomado de ficheros públicos del ayuntamiento de Madrid sobre la calidad del aire. Portal de datos abiertos del ayuntamiento de Madrid: “<http://datos.madrid.es/portal/site/egob>”. Y de la web de *open data* de la Agencia Estatal de Meteorología: “http://www.aemet.es/es/datos_abiertos/AEMET_OpenData”. Los datos se encuentran en un fichero con un formato tabular fijo.

A continuación, se describe el formato:

- Los ocho primeros dígitos identifican el código de la estación de medida.
- Los siguientes dos dígitos determinan el valor del parámetro físico a medir.
- Dos dígitos para codificar la técnica analítica.
- Dos dígitos que indican si son valores diarios u horarios.
- La fecha se codifica con cuatro dígitos, los dos primeros para el año (todos corresponden al año 2015) y los dos finales para el mes.

- Después están los valores registrados para cada uno de los días del mes del parámetro físico en la estación. Los valores se codifican en cinco dígitos, con una V al final si el valor es válido o una N en caso contrario.

La lectura de un fichero con formato se puede realizar de muchas formas diferentes. Es habitual utilizar `read.csv` cuando los valores están separados por un carácter delimitador: coma, punto y coma, etc. En este caso, el formato está determinado por la posición que ocupa en la columna, de modo que se va a utilizar `read.fwf`.

Posteriormente habrá que procesar los datos de medida para validarlos.

```
setwd("/Users/aberlan/Google Drive/Libro_DATA_SCIENCE/Datos/")
dfMedidas<-read.fwf("datos15.txt"
widths=c(8
rep(2
5)
rep(6
31)))
```

Las columnas de la técnica, la que indica que es un valor diario y la correspondiente al año, no son de interés. Por tanto, se suprimen del *data frame*. Además, se van a renombrar las columnas.

```
dfMedidas <- dfMedidas[, -(3:5), drop = FALSE] nombreDia <- paste0("Dia_",sprintf("%02s",
seq(1:31)))
colnames(dfMedidas) <- c("Estación", "Magnitud", "Mes",nombreDia)
```

Las medidas que no terminan con una V no son válidas y deben ser eliminadas para no introducir sesgos en las medidas agregadas. Hay varias alternativas en R para tratar estos datos no válidos. Se pueden convertir a NA, (*Not Available*, “no disponible”), NAN (*Not A Number*, “no es un número”) o NULL (valor nulo). Cada una de las alternativas tiene su contexto de aplicación. En esta situación es preferible utilizar NA, un valor lógico.

El algoritmo para realizar las modificaciones es simple: recorrer las columnas de datos, de la 4ª a la 34ª, tomando el valor numérico si termina en V o poner el valor NA en caso contrario. La forma habitual de proceder sería con dos bucles anidados tal como se muestra a continuación. Se va a calcular el tiempo para realizar esta transformación con el fin de poder compararla con la forma recomendada de operar con datos en R.

```

start.time1 <- Sys.time()
dfMedidas1 <- data.frame(matrix(ncol = 34, nrow = nrow(dfMedidas))) dfMedidas1[, 1:3]
<- dfMedidas[, 1:3]

for (i in 1:nrow(dfMedidas1)){
  for (j in 4:34){
    if (grepl("V", dfMedidas[i,j]) == TRUE){
      dfMedidas1[i, j] <- as.numeric(substr(dfMedidas[i, j], 1, 5))
    }
  }
}

end.time1 <- Sys.time()
time.taken1 <- end.time1 - start.time1 time.taken1

## Time difference of 5.688286 secs

```

El mismo resultado puede obtenerse si se utiliza la familia de funciones que se aplican a vectores; *apply*, *mapply*, *lapply*, *tapply*. El algoritmo anterior puede simplificarse con el siguiente código.

```

start.time2 <- Sys.time()
dfMedidas2 <- cbind(dfMedidas[1:3], apply(dfMedidas[4:34], 2,
function(x) {ifelse (grepl("V", x) == 0, NA,
as.numeric(substr(x, 1, 5))}))
end.time2 <- Sys.time()
time.taken2 <- end.time2 - start.time2 time.taken2

## Time difference of 0.03562903 secs

```

Efectivamente, en tan sólo una línea de código se puede obtener el mismo resultado, con un *speedup* superior al 100 %. Sin duda, siempre que sea posible, deberá evitarse el uso de bucles en beneficio de las operaciones en su versión vectorizada.

Para guardar los datos procesados en un nuevo fichero se puede utilizar el formato propio de R con *save* o funciones específicas para otros formatos. Por ejemplo, *write.csv* para datos separados por comas, formato muy extendido y que incorporan la inmensa mayoría de programas para el tratamiento de datos.

Vamos a leer los datos salvados y cargarlos directamente a un *data frame*. Si se utiliza *read.csv* por defecto aplica la conversión de datos de tipo cadena a factores.

```
rm(list = ls())
dfMedidas <- read.csv("meteo_calidad_2015.csv", sep = ";", dec = ",", header = TRUE)
attach(dfMedidas)
typeof(dfMedidas$Mes)
class(dfMedidas$Mes)
[1] "integer"
[1] "factor"
```

La función *typeof* aplicada a la columna “Mes” muestra que es de tipo entero. Como ya se ha mencionado, el tipo factor se trata internamente como un valor entero. Sin embargo, la función *class* devuelve el tipo de dato desde el punto de vista de jerarquía interna de clases definidas en R, es decir, “Mes” es de clase “factor”.

Se va a añadir una columna con la fecha, esto será útil para posteriores procesos.

```
meses <- c("ENE", "FEB", "MAR", "ABR", "MAY", "JUN", "JUL", "AGO", "SEP", "OCT", "NOV",
"DIC")
fecha <- as.data.frame(paste0("2015-", formatC(match(dfMedidas$Mes, meses),
width = 2, flag = 0), "-", formatC(dfMedidas$Dia_mes,
width = 2, flag = 0)))
names(fecha) <- "date"
dfMedidas <- data.frame(c(dfMedidas[1:5], fecha, dfMedidas[6:33])) rm(fecha)
dfMedidas$date <- as.POSIXct(dfMedidas$date, tz = "Europe/Madrid")
```

3.2 Estadística descriptiva

El primer paso para analizar un conjunto de datos consiste en aplicar las medidas cuantitativas de la estadística descriptiva, comenzando con el análisis univariado y pasando después al multivariado. Se pueden aplicar las funciones básicas para calcular la media, rango, cuartiles, etc. sobre un atributo concreto o sobre el conjunto de datos.

```

mean(dfMedidas$T_MAX) #Calcula la media
median(T_MAX) #Mediana
sd(T_MAX) #Desviación estándar
range(T_MAX) #Valor máximo y mínimo
quantile(T_MAX) #Cuartiles
sd(T_MAX) /sqrt(length(T_MAX)) #Error estándar de la media

[1] 22.13233
[1] 21.4
[1] 9.01446
[1] 3.5 39.9
  0%  25%  50%  75% 100%
  3.5 14.5 21.4 29.5 39.9
[1] 0.4718384

```

Hay funciones que simplifican la aplicación de un conjunto de estadísticos, la más conocida es *summary*.

vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	seX	
1	1	365	22.13	9.01	21.4	21.91	10.67	3.5	39.9	36.4	0.21	-1.02	0.47
vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis		
Dia	1	365	183.00	105.51	183.00	183.00	134.92	1.00	365.00	364.00	0.00	-1.21	
Dia_mes	2	365	15.72	8.81	16.00	15.71	11.86	1.00	31.00	30.00	0.01	-1.20	
Mes*	3	365	6.50	3.46	7.00	6.50	4.45	1.00	12.00	11.00	0.00	-1.23	
Dia_sem*	4	365	3.99	2.00	4.00	3.99	2.97	1.00	7.00	6.00	0.01	-1.26	
Lluvia	5	365	0.73	3.07	0.00	0.04	0.00	0.00	26.60	26.60	6.04	40.04	
date*	6	365	NaN	NA	NA	NaN	NA	Inf	-Inf	-Inf	NA	NA	
T_MAX	7	365	22.13	9.01	21.40	21.91	10.67	3.50	39.90	36.40	0.21	-1.02	
T_MIN	8	365	11.66	6.86	11.30	11.52	7.71	-1.50	26.00	27.50	0.16	-0.98	
Viento_MAX	9	365	31.51	11.65	31.00	31.04	11.86	10.00	75.00	65.00	0.53	0.52	
Viento_MED	10	365	14.22	4.88	14.00	14.10	4.45	4.00	28.00	24.00	0.23	-0.32	
SO2	11	365	6.86	2.40	6.00	6.43	1.33	3.78	15.60	11.82	1.58	1.99	
CO	12	365	0.37	0.17	0.31	0.34	0.10	0.18	0.99	0.81	1.59	2.10	
NO	13	365	26.72	34.35	11.12	18.68	9.39	2.33	168.00	165.67	2.04	3.52	
NO2	14	365	40.97	19.48	37.08	38.88	18.29	9.58	104.88	95.29	0.92	0.42	
PM2.5	15	365	11.59	6.28	10.40	10.74	5.34	2.80	36.17	33.37	1.31	1.73	
PM10	16	365	21.09	10.70	19.27	19.77	9.36	5.73	69.00	63.27	1.35	2.35	
O3	17	365	50.60	25.47	53.54	50.54	29.59	7.43	108.36	100.93	-0.06	-1.02	
TOL	18	365	3.15	2.26	2.48	2.77	1.41	0.48	14.17	13.68	1.84	3.96	
BEN	19	365	0.76	0.48	0.60	0.68	0.30	0.16	2.63	2.47	1.52	2.01	
EBE	20	365	0.50	0.41	0.35	0.42	0.22	0.10	2.45	2.35	2.00	4.31	

TCH	21	365	1.51	0.12	1.49	1.50	0.08	1.24	2.08	0.84	1.50	4.24
MMCH	22	365	0.26	0.14	0.21	0.25	0.11	0.06	0.66	0.61	0.87	-0.27
SO2_MAX	23	365	13.57	4.96	12.00	12.90	2.97	7.00	39.00	32.00	1.49	2.66
CO_MAX	24	365	0.57	0.23	0.50	0.54	0.15	0.30	1.50	1.20	1.38	2.17
NO_MAX	25	365	67.47	74.12	35.00	51.10	25.20	8.00	337.00	329.00	1.87	2.65
NO2_MAX	26	365	64.91	25.01	59.00	61.76	19.27	26.00	165.00	139.00	1.26	1.73
PM2.5_MAX	27	365	13.96	7.00	13.00	13.02	5.93	4.00	40.00	36.00	1.28	1.62
PM10_MAX	28	365	28.65	13.38	26.00	27.02	11.86	8.00	95.00	87.00	1.36	2.63
O3_MAX	29	365	66.45	24.82	69.00	66.67	28.17	16.00	134.00	118.00	-0.10	-0.81
TOL_MAX	30	365	6.38	4.32	5.20	5.67	2.52	0.90	29.10	28.20	2.00	5.17
BEN_MAX	31	365	1.77	0.83	1.60	1.71	0.74	0.30	4.70	4.40	0.73	0.25
EBE_MAX	32	365	1.05	0.92	0.70	0.87	0.44	0.10	5.60	5.50	2.03	4.59
TCH_MAX	33	365	1.71	0.18	1.69	1.70	0.12	1.33	2.44	1.11	1.11	2.59
MMCH_MAX	34	365	0.52	0.35	0.40	0.48	0.20	0.08	1.69	1.61	0.90	-0.11
se												
Dia	5.52											
Dia_mes	0.46											
Mes*	0.18											
Dia_sem*	0.10											
Lluvia	0.16											
date*	NA											
T_MAX	0.47											
T_MIN	0.36											
Viento_MAX	0.61											
Viento_MED	0.26											
SO2	0.13											
CO	0.01											
NO	1.80											
NO2	1.02											
PM2.5	0.33											
PM10	0.56											
O3	1.33											
TOL	0.12											
BEN	0.03											
EBE	0.02											
TCH	0.01											
MMCH	0.01											
SO2_MAX	0.26											
CO_MAX	0.01											
NO_MAX	3.88											
NO2_MAX	1.31											
PM2.5_MAX	0.37											
PM10_MAX	0.70											
O3_MAX	1.30											
TOL_MAX	0.23											
BEN_MAX	0.04											
EBE_MAX	0.05											
TCH_MAX	0.01											

Se pueden realizar cálculos sólo sobre los datos que cumplen un determinado criterio. Por ejemplo, la temperatura máxima media en enero sería:

```
mean(dfMedidas[which(Mes == "ENE"), "T_MAX"])

[1] 11.02581
```

Para calcular el valor medio de la temperatura máxima por mes, se puede utilizar la función *tapply* (hay otras posibilidades como *aggregate*), que pertenece a la familia de funciones que aplican funciones sobre columnas o filas de un conjunto de datos.

```
tapply(T_MAX, Mes, mean)
```

```
ABR      AGO      DIC      ENE      FEB      JUL      JUN      MAR      MAY      NOV
20.58000 32.51935 13.74194 11.02581 10.84643 36.76774 31.35667 17.25484 27.41935 16.76000
OCT      SEP
19.95806 26.48000
```

Se aprecia que el valor se presenta con los meses ordenados alfabéticamente. Ésta no es una forma muy conveniente y habitual, para estos datos es más razonable la ordenación cronológica. Para lo cual se reordenan las categorías de la variable "Mes".

```
Mes <- factor(Mes, levels = c("ENE", "FEB", "MAR", "ABR", "MAY", "JUN", "JUL", "AGO", "SEP",
"OCT", "NOV", "DIC"))

tapply(T_MAX, Mes, mean) #Ahora la salida es en el orden cronológico
```

```
ENE      FEB      MAR      ABR      MAY      JUN      JUL      AGO      SEP      OCT
11.02581 10.84643 17.25484 20.58000 27.41935 31.35667 36.76774 32.51935 26.48000 19.95806
NOV      DIC
16.76000 13.74194
```

De nuevo, hay funciones en librerías que pueden simplificar y hacer más legible el código para hacer cálculos sobre subconjuntos de datos.

```
library(FSA)

Summarize(T_MAX ~ Mes, data = dfMedidas)
```

Mes	n	mean	sd	min	Q1	median	Q3	max
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
ABR	30	20.58000	3.056299	11.9	18.825	21.35	22.800	24.8
AGO	31	32.51935	3.286479	25.1	30.900	33.30	34.800	37.6
DIC	31	13.74194	1.724872	10.0	12.600	14.10	14.950	16.6
ENE	31	11.02581	2.778605	3.5	9.300	11.70	12.900	15.5
FEB	28	10.84643	3.417437	3.5	8.800	11.70	13.375	16.8
JUL	31	36.76774	2.134852	31.0	35.600	37.10	38.200	39.9
JUN	30	31.35667	5.213820	20.6	28.325	32.65	34.100	39.3

Mes	n	mean	sd	min	Q1	median	Q3	max
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
MAR	31	17.25484	5.335719	8.7	12.950	17.40	21.900	27.0
MAY	31	27.41935	4.096293	21.1	23.700	28.00	30.750	35.8
NOV	30	16.76000	3.086522	9.2	15.825	16.35	19.150	21.5
OCT	31	19.95806	2.574331	15.3	18.600	20.20	21.450	25.4
SEP	30	26.48000	2.780523	19.8	24.550	27.10	28.525	30.3

```
library(Rmisc)
summarySE(data = dfMedidas, "T_MAX", groupvars = "Mes", conf.interval = 0.95)
```

Mes	N	T_MAX	sd	se	ci
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
ABR	30	20.58000	3.056299	0.5580014	1.1412409
AGO	31	32.51935	3.286479	0.5902691	1.2054903
DIC	31	13.74194	1.724872	0.3097961	0.6326881
ENE	31	11.02581	2.778605	0.4990522	1.0192005
FEB	28	10.84643	3.417437	0.6458349	1.3251437
JUL	31	36.76774	2.134852	0.3834307	0.7830699
JUN	30	31.35667	5.213820	0.9519090	1.9468724
MAR	31	17.25484	5.335719	0.9583233	1.9571574
MAY	31	27.41935	4.096293	0.7357159	1.5025322
NOV	30	16.76000	3.086522	0.5635193	1.1525264
OCT	31	19.95806	2.574332	0.4623636	0.9442724
SEP	30	26.48000	2.780523	0.5076518	1.0382645

```
library(psych)
describeBy(T_MAX, group = Mes, digits= 4)
```

Descriptive statistics by group

group: ENE

```
vars  n  mean    sd median trimmed  mad min  max range  skew kurtosis  se
X1    1  31  11.03  2.78   11.7   11.28 2.22  3.5  15.5   12 -0.81    0.02 0.5
```

group: FEB

```
vars  n  mean    sd median trimmed  mad min  max range  skew kurtosis  se
X1    1  28  10.85  3.42   11.7   10.97 3.48  3.5  16.8  13.3 -0.47   -0.79 0.65
```

group: MAR

```
vars  n  mean    sd median trimmed  mad min  max range  skew kurtosis  se
X1    1  31  17.25  5.34   17.4   17.13 6.82  8.7   27  18.3 0.14   -1.22 0.96
```

```

-----
group: ABR
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 30 20.58 3.06  21.35   20.85 3.48 11.9 24.8  12.9 -0.78    0.14 0.56
-----

group: MAY
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 31 27.42 4.1   28    27.33 5.19 21.1 35.8  14.7 0.14   -1.25 0.74
-----

group: JUN
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 30 31.36 5.21  32.65   31.61 3.34 20.6 39.3  18.7 -0.58   -0.68 0.95
-----

group: JUL
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 31 36.77 2.13  37.1   36.93 2.08  31 39.9   8.9 -0.75   -0.07 0.38
-----

group: AGO
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 31 32.52 3.29  33.3   32.78 2.37 25.1 37.6  12.5 -0.67   -0.62 0.59
-----

group: SEP
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 30 26.48 2.78  27.1   26.73 2.97 19.8 30.3  10.5 -0.68   -0.39 0.51
-----

group: OCT
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 31 19.96 2.57  20.2   19.88 1.93 15.3 25.4  10.1  0.1   -0.51 0.46
-----

group: NOV
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 30 16.76 3.09  16.35   17.05 3.26  9.2 21.5  12.3 -0.69   -0.11 0.56
-----

group: DIC
vars  n  mean    sd median trimmed  mad  min  max range  skew kurtosis  se
X1    1 31 13.74 1.72  14.1   13.86 1.48  10 16.6   6.6 -0.56   -0.63 0.31

```

Una medida muy importante relacionada con la media es el cálculo de intervalos de confianza. Permite establecer una cota máxima y mínima al intervalo en el que se encuentra un valor, por ejemplo, la media, con una determinada probabilidad. De momento se asumirá, esto será discutido posteriormente, que las temperaturas en un mes siguen una distribución normal y se realizarán los cálculos de intervalos de confianza para los valores medios.

```
library(Rmisc)
```

```
CI(T_MAX, ci = 0.95)
```

```
      upper      mean      lower
23.06020 22.13233 21.20446
```

```
group.CI(T_MAX ~ Mes, data = dfMedidas, ci = 0.95)
```

es	T_MAX.upper	T_MAX.mean	T_MAX.lower
<fctr>	<dbl>	<dbl>	<dbl>
BR	21.72124	20.58000	19.438759
GO	33.72485	32.51935	31.313864
IC	14.37462	13.74194	13.109247
NE	12.04501	11.02581	10.006606
EB	12.17157	10.84643	9.521285
UL	37.55081	36.76774	35.984672
UN	33.30354	31.35667	29.409794
AR	19.21200	17.25484	15.297681
AY	28.92189	27.41935	25.916823
OV	17.91253	16.76000	15.607474
CT	20.90234	19.95806	19.013792
EP	27.51826	26.48000	25.441736

Cuando los datos no siguen una distribución normal, el intervalo de confianza se puede estimar por remuestreo de los datos *bootstrapping*. Dado que es una técnica estocástica, de una ejecución a otra cambiará el valor, conviene fijar la semilla de la secuencia pseudoaleatoria para poder reproducir los resultados.

```
library(boot)
set.seed(1234)
T_MAX_boot <- boot(T_MAX, function(x,i) mean(x[i]), R = 10000)
mean(T_MAX_boot$t[,1]) #La media calculada por bootstrapping

boot.ci(T_MAX_boot, conf = 0.95, type = "all")
```

```
[1] 22.12566
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates

CALL :
boot.ci(boot.out = T_MAX_boot, conf = 0.95, type = "all")

Intervals :
Level      Normal              Basic
95%      (21.21, 23.06 )      (21.21, 23.09 )

Level      Percentile          BCa
95%      (21.18, 23.05 )      (21.19, 23.06 )
Calculations and Intervals on Original Scale
```

Las medidas de dispersión estiman cómo se distribuyen los datos alrededor de la media. Dan una visión de la función que podría expresar una variable. La desviación absoluta media (o desviación media absoluta), varianza, rango, rango medio o la desviación típica son medidas típicas de dispersión.

```
mad(dfMedidas[which(Mes == "JUL"), "T_MAX"]) #desviación absoluta media
sd(dfMedidas[which(Mes == "JUL"), "T_MAX"]) #desviación estándar
IQR(dfMedidas[which(Mes == "JUL"), "T_MAX"]) #rango intercuartil
```

```
[1] 2.07564
[1] 2.134852
[1] 2.6
```

Si representamos un histograma con los valores de la temperatura máxima para el mes de julio, junto con la función de distribución normal, nos podemos hacer una idea de cómo los datos se ajustan a la función.

```
library(rcompanion)
plotNormalHistogram(dfMedidas[which(Mes == "JUL"), "T_MAX"]
```

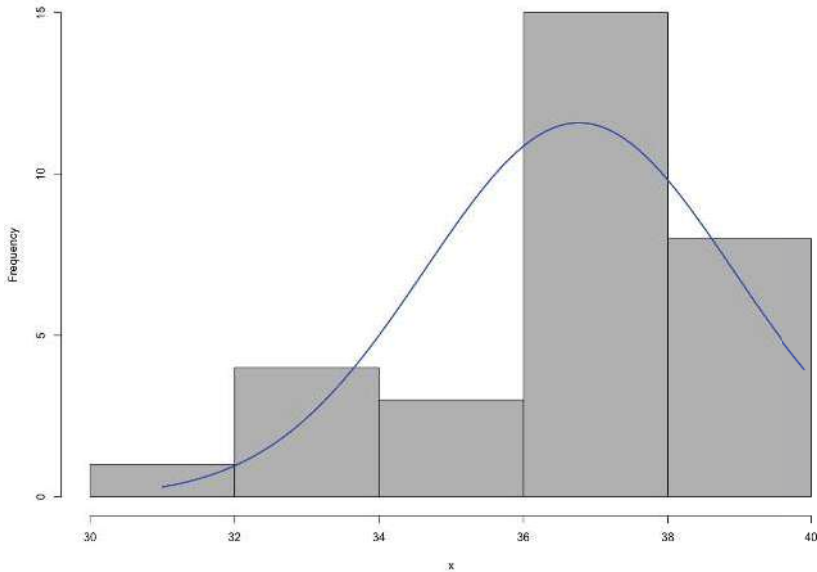


Fig. 3.2. Histograma de temperaturas máximas del mes de julio.

Pero, de forma general, es difícil hacer una valoración precisa a partir del histograma del ajuste a la curva teórica. Una medida común que puede dar indicio del grado de similitud es la asimetría estadística, *skewness*.

```
library(e1071)

op <- par(mar = c(3, 3, 4, 2) + 0.1)

hist(dfMedidas[which(Mes == "JUL"), "T_MAX"], col = "light blue", probability=TRUE,
main = paste("skewness =", round(skewness(dfMedidas[which(Mes == "JUL"), "T_MAX"]),
digits=2)),
xlab="", ylab="")

lines(density(dfMedidas[which(Mes == "JUL"), "T_MAX"]), col = "red", lwd = 5)
rug(dfMedidas[which(Mes == "JUL"), "T_MAX"], col = "green", lwd = 3)

par(op)
```

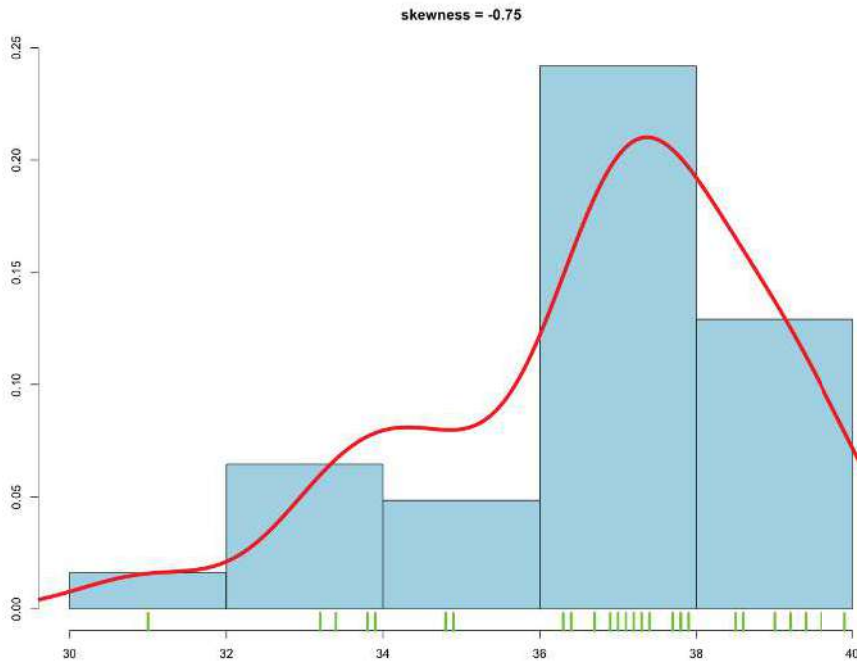


Fig. 3.3. Gráfico de la asimetría de las temperaturas máximas del mes de julio.

```
skewness(dfMedidas[which(Mes == "JUL"), "T_MAX"])
```

```
[1] -0.7539249
```

El valor negativo indica que la función está más desplazada a la izquierda que la normal.

La medida de curtosis permite establecer si los datos se extienden más o menos que la normal en las colas y si hay mayor densidad alrededor del valor medio.

```
T_MAX_JUL <- dfMedidas[which(Mes == "JUL"), "T_MAX"]
qqnorm(T_MAX_JUL, main = paste("curtosis =", round(kurtosis(T_MAX_JUL), digits = 2),
"(gaussian)"))
qqline(T_MAX_JUL, col = "red")
op <- par(fig=c(0.02, 0.5, 0.5, 0.98), new = TRUE)
hist(T_MAX_JUL, probability=T, col="light blue", xlab = "", ylab = "", main = "",
axes = FALSE)
lines(density(T_MAX_JUL), col = "red", lwd=2)
box()
par(op)
```

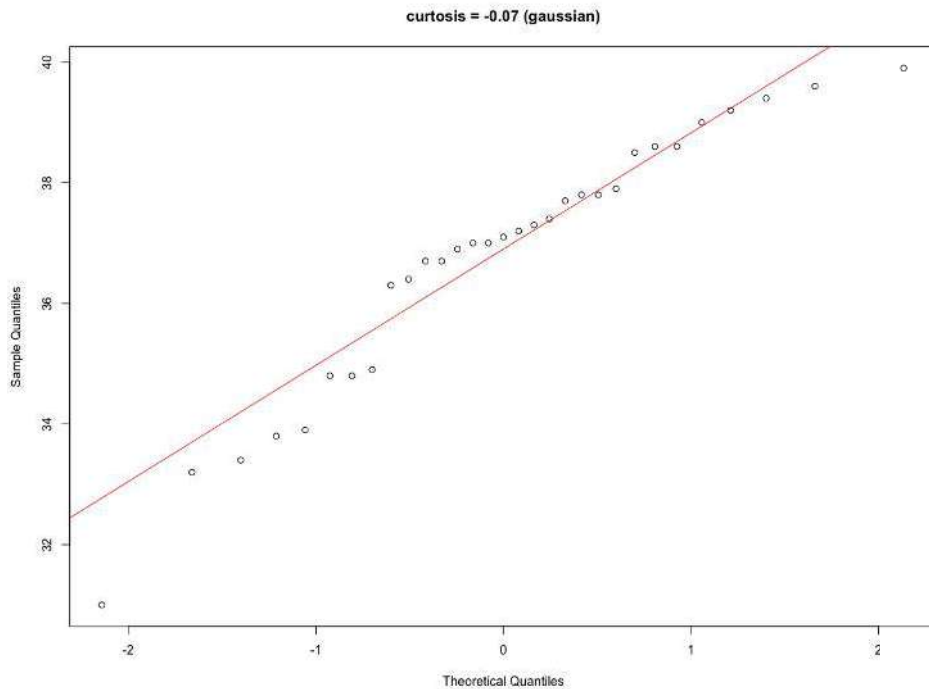


Fig. 3.4. Gráfico de kurtosis.

El valor negativo de kurtosis indica un “aplanamiento” mayor de los datos comparados con los distribuidos según una distribución normal.

Con los valores de asimetría y kurtosis se puede determinar lo próximo que están los datos a las funciones de distribución más conocidas. Hay una representación gráfica que permite visualizar fácilmente la comparación.

```
library(fitdistrplus)
descdist(T_MAX_JUL, discrete = FALSE)
```

```
summary statistics
-----
min:  31    max:  39.9
median:  37.1
mean:  36.76774
estimated sd:  2.134852
estimated skewness:  -0.8327837
estimated kurtosis:  3.379631
```

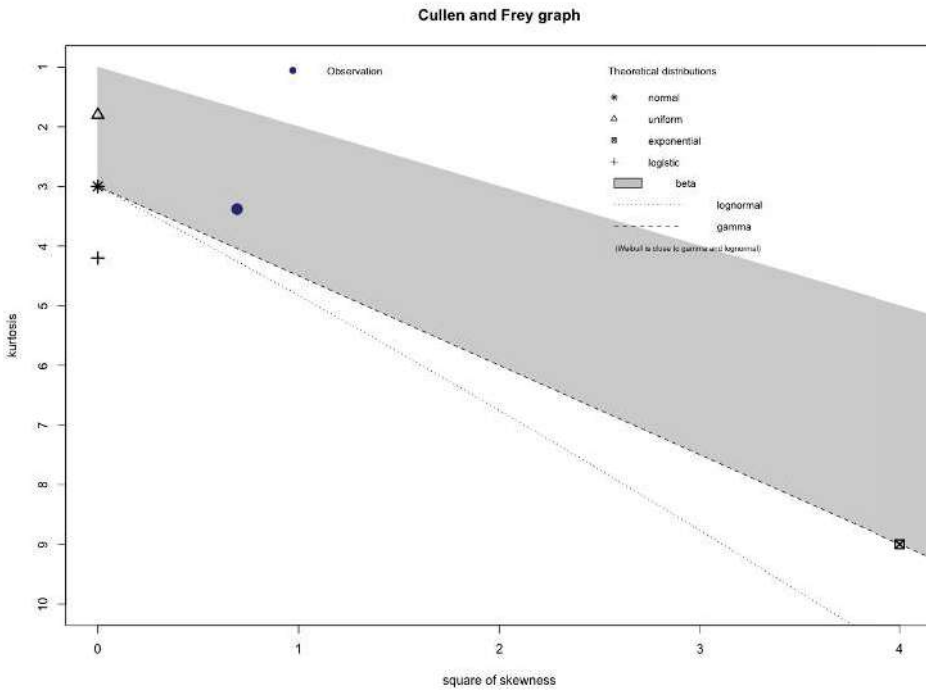


Fig. 3.5. Gráfico de Cullen-Frey.

Se puede observar que los datos de la temperatura máxima del mes de julio se podrían aproximar a una distribución gamma, beta o normal. Esta representación complementa los test que permiten comparar la bondad del ajuste de un conjunto de datos a una distribución dada. Como por ejemplo con el test de Shapiro-Wilk.

```
shapiro.test(T_MAX_JUL)
shapiro.test(T_MAX)
```

```
Shapiro-Wilk normality test

data:  T_MAX_JUL
W = 0.93655, p-value = 0.06625

Shapiro-Wilk normality test

data:  T_MAX
W = 0.96403, p-value = 8.115e-08
```

Del test Shapiro-Wilk se puede concluir que la temperatura máxima en un año no sigue una distribución normal ($p < 0.05$), por contra, en la temperatura máxima en el mes de julio, el valor de p es muy ligeramente mayor a 0.05, que es el umbral para considerar la hipótesis de que la distribución de los datos no es significativamente diferente de la distribución normal.

Realizando un gráfico Q-Q, se observa que los valores de temperatura máxima para junio están dentro del intervalo de confianza en el que se puede asumir una distribución normal.

```
library(ggpubr)
ggqqplot(T_MAX_JUL) #gráfica Q-Q
```

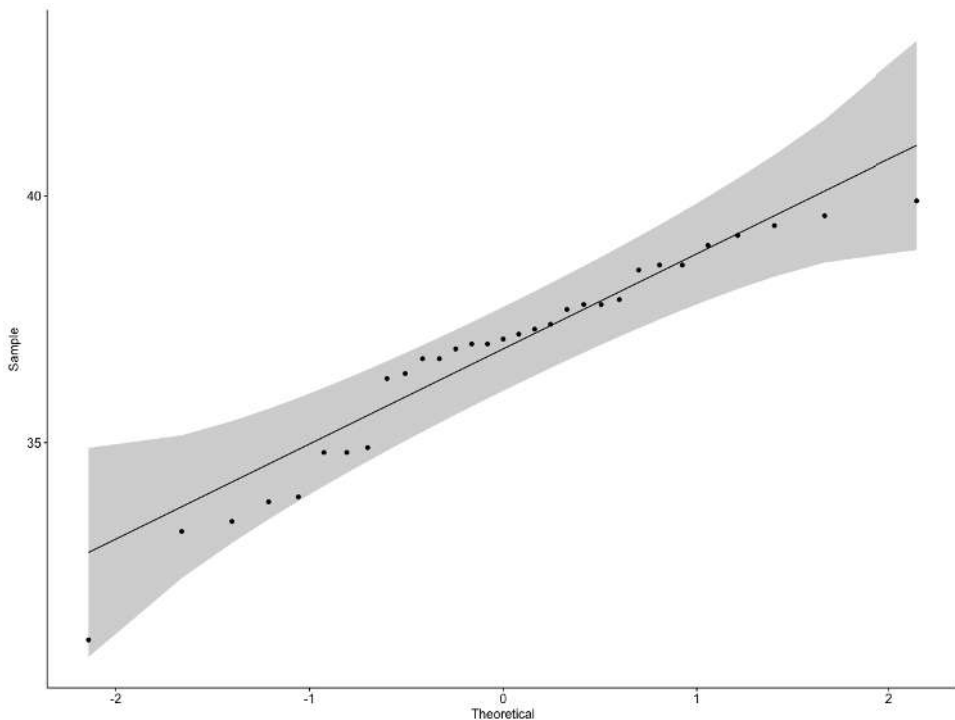


Fig. 3.6. Gráfico Q-Q, temperatura máxima del mes de julio.

No ocurre igual al representar los valores de la temperatura máxima en un año.

```
library("car")
qqPlot(T_MAX) #La misma gráfica Q-Q pero con otra librería
```

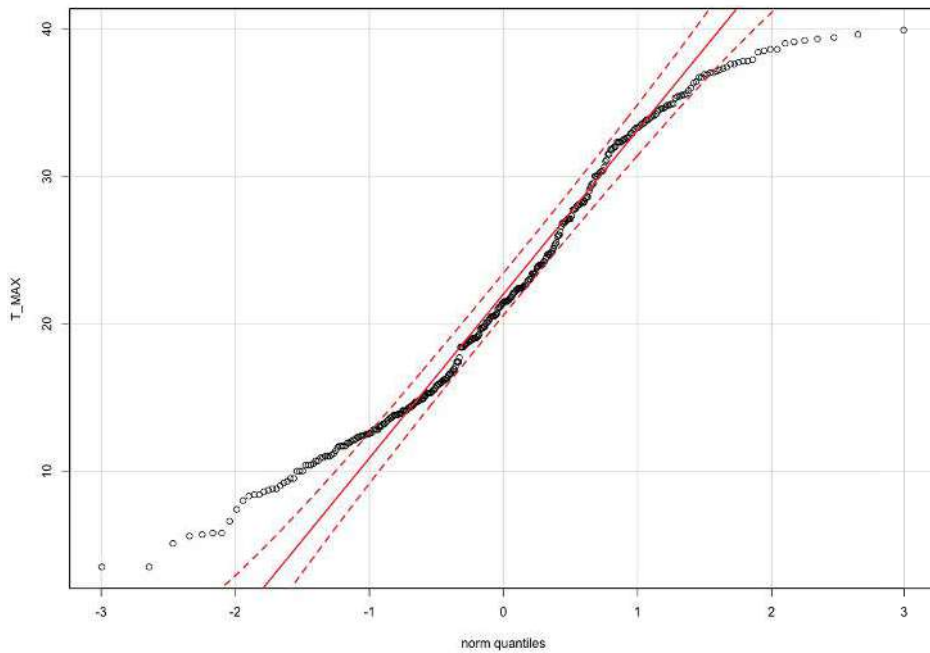


Fig. 3.7. Gráfico Q-Q. Temperaturas máximas durante todo un año.

Claramente las colas difieren sustancialmente de la distribución teórica.

Se pueden aplicar otros test para comprobar la normalidad de la temperatura máxima en julio, como la prueba Anderson-Darling, la de Cramér-von Mises o de Kolmogorov-Smirnov.

```
library(nortest)
ad.test(T_MAX_JUL)
cvm.test(T_MAX_JUL)
lillie.test(T_MAX_JUL)
```

```
Anderson-Darling normality test
```

```
data: T_MAX_JUL
```

```
A = 0.71185, p-value = 0.0568
```

```
Cramer-von Mises normality test
```

```
data: T_MAX_JUL
```

```
W = 0.13183, p-value = 0.03894
```

```
Lilliefors (Kolmogorov-Smirnov) normality test
```

```
data: T_MAX_JUL
```

```
D = 0.16476, p-value = 0.0316
```

El resultado muestra que, para las temperaturas máximas de julio, dos test aceptarían la normalidad (Lilliefors y Anderson-Darling) y Cramér-von Mises la descartaría. Sin embargo, para la temperatura máxima durante un año, todos los test descartan rotundamente la normalidad.

```
ad.test(T_MAX)
```

```
cvm.test(T_MAX)
```

```
lillie.test(T_MAX)
```

```
Anderson-Darling normality test
```

```
data: T_MAX
```

```
A = 3.9664, p-value = 6.835e-10
```

```
Cramer-von Mises normality test
```

```
data: T_MAX
```

```
W = 0.59717, p-value = 3.824e-07
```

```
Lilliefors (Kolmogorov-Smirnov) normality test
```

```
data: T_MAX
```

```
D = 0.080909, p-value = 5.263e-06
```

Finalmente, antes de pasar a los cálculos de correlación entre variables, se va realizar un cálculo de utilidad para magnitudes univariadas continuas. Si se quisiera realizar una tabla de frecuencias habría que enfrentar la decisión del número y rango de intervalos en los que discretizar los valores continuos. La solución habitual, la más simple, consiste en dividir en intervalos constantes. Pero esta solución, en determinadas circunstancias, no es la más adecuada. Por ejemplo, las encuestas de satisfacción de servicios que usan el *net promoter score* dividen el rango de puntuación en tres rangos: detractores (1-5), pasivos (6-7) y promotores (8-10). O el caso de la concentración de ozono, donde por legislación se fijan dos umbrales, uno de información y otro de alerta o las conocidas alertas de AEMET, tres umbrales, amarillo, naranja y rojo, para diferentes situaciones meteorológicas. Podrían asumirse rangos en función de los cuartiles o de medidas de densidad basadas en varios algoritmos de agrupamiento. Fijados los intervalos, ya podrían construirse las tablas de frecuencias.

```
library(classInt)
T_MAX_JUL_aemet <- classIntervals(T_MAX_JUL, n = 4, style = "fixed", fixedBreaks = c(0,
36, 39, 42, 60), intervalClosure = "left") # Frecuencias en rangos aemet
plot(T_MAX_JUL_aemet, pal = c("white", "yellow", "orange", "red"), main = "T_MAX Julio")
```

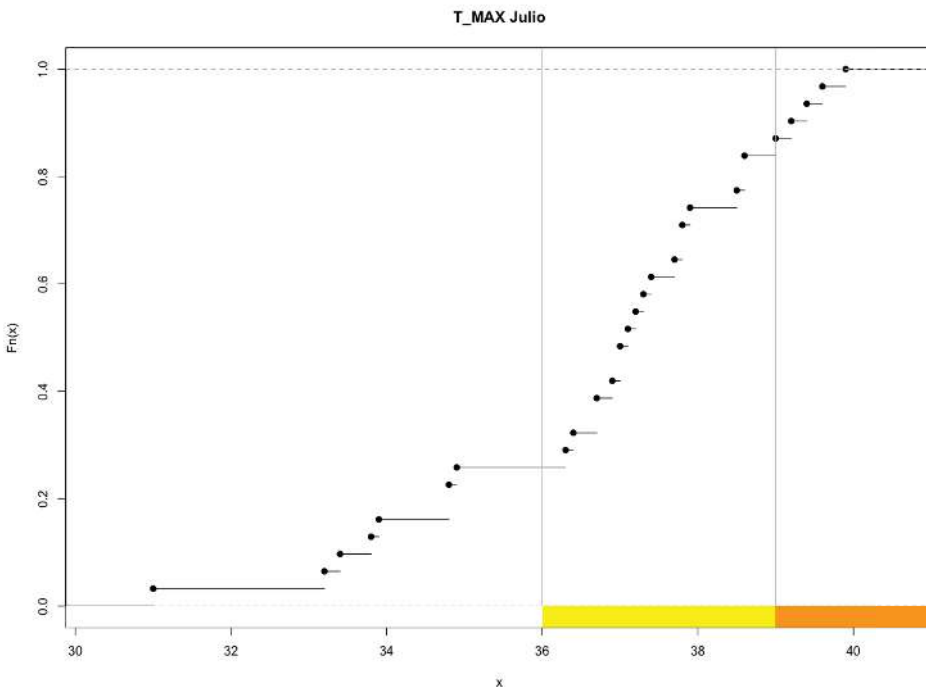


Fig. 3.8. Discretización de la temperatura máxima.

Pero otras particiones son posibles utilizando diferentes criterios, por ejemplo:

```
(classIntervals(T_MAX_JUL, style = "fisher", intervalClosure = "left"))
(classIntervals(T_MAX_JUL, n = 4, style = "quantile", intervalClosure = "left"))
```

style: fisher
 one of 53,130 possible partitions of this variable into 6 classes

[31,32.1)	[32.1,34.35)	[34.35,35.6)	[35.6,37.55)	[37.55,38.8)	[38.8,39.9]
1	4	3	11	7	5

style: quantile
 one of 2,300 possible partitions of this variable into 4 classes

[31,35.6)	[35.6,37.1)	[37.1,38.2)	[38.2,39.9]
8	7	8	8

Se aprecia que la partición de Fisher refleja cierta estructura de los datos. Se aplica el algoritmo de Fisher-Jenks, que agrupa un conjunto de datos en clases tratando de minimizar la varianza dentro de cada clase maximizando la varianza entre ellas.

Los análisis estadísticos realizados para la temperatura máxima tendrían que aplicarse para el resto de variables con el fin de tener una vista general de la información que contienen los datos y las posibles relaciones que existen entre ellos.

3.2.1 Variables categóricas

El conjunto de datos de calidad del aire no tiene variables nominales. El mes y día de la semana son variables ordinales, ya que representan una secuencia temporal ordenada. Se puede establecer una clasificación del tipo de variables en función del valor que almacena. En el primer nivel se distinguen las que toman valores numéricos (cuantitativas) frente a las que sus valores son etiquetas (cualitativas). Tal como se muestra en la figura siguiente:

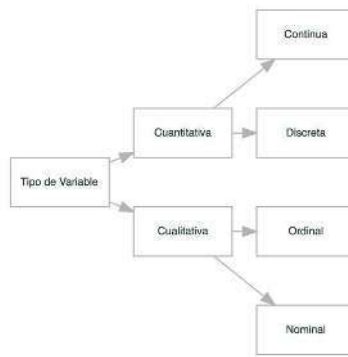


Fig. 3.9. Taxonomía de tipos de variables.

Las variables nominales o categóricas son aquellas que toman un valor de un conjunto finito, mutuamente excluyentes y exhaustivas. Se van a crear nuevas variables nominales para ilustrar algunas estadísticas específicas de este tipo de variables. Se crea una nueva variable, *Lluvia_SN*, que considera si ese día llovió o no, es decir, tiene un valor binario de “Sí” o “No”. Se van a añadir dos variables que consideran si el umbral medio diario de partículas en suspensión supera el límite establecido. Finalmente, para la temperatura máxima diaria se va a utilizar el código AEMET de avisos por colores.

```

Lluvia_SN <- ifelse(dfMedidas$Lluvia > 0, "Si", "No") #Si las precipitaciones son mayor que 0->Si
Lluvia_SN <- as.factor(Lluvia_SN) #Se convierte a factor
dfMedidas <- as.data.frame(cbind(dfMedidas[,1:5], Lluvia_SN, dfMedidas[,6:33]))

PM2.5_UMBRAL <- as.factor(ifelse(dfMedidas$PM2.5 >= 25, "Si", "No")) #Si PM2.5 supera umbral ->Si
dfMedidas <- as.data.frame(cbind(dfMedidas[,1:15], PM2.5_UMBRAL, dfMedidas[,16:34]))

PM10_UMBRAL <- as.factor(ifelse(dfMedidas$PM10 >= 50, "Si", "No")) #Si PM10 supera umbral ->Si
dfMedidas <- as.data.frame(cbind(dfMedidas[,1:17], PM10_UMBRAL, dfMedidas[,18:35]))

T_MAX_ALERTA <- as.factor(ifelse(T_MAX >= 42, "R", ifelse(T_MAX >=39, "N", ifelse(T_MAX >=36, "A", "V"))))
dfMedidas <- as.data.frame(cbind(dfMedidas[,1:7], T_MAX_ALERTA, dfMedidas[,8:36]))
  
```

Para las variables cualitativas se pueden calcular las frecuencias de ocurrencia de las diferentes alternativas, de forma independiente o condicionada a otras variables, es decir, mediante tablas de contingencia.

```

table(PM2.5_UMBRAL)
table(PM2.5_UMBRAL, PM10_UMBRAL)
xtabs(~ Mes + T_MAX_ALERTA, data = dfMedidas) # más flexible que table, usando fórmula
round(100*prop.table(table(Mes, T_MAX_ALERTA),1),1)

```

```

PM2.5_UMBRAL
  No  Si
347 18

```

```

          PM10_UMBRAL
PM2.5_UMBRAL  No  Si
          No 341   6
          Si  15   3

```

```

      T_MAX_ALERTA

```

```

Mes  A  N  V

```

```

ABR  0  0 30

```

```

AGO  3  0 28

```

```

DIC  0  0 31

```

```

ENE  0  0 31

```

```

FEB  0  0 28

```

```

JUL 18  5  8

```

```

JUN  2  2 26

```

```

MAR  0  0 31

```

```

MAY  0  0 31

```

```

NOV  0  0 30

```

```

OCT  0  0 31

```

```

SEP  0  0 30

```

```

T_MAX_ALERTA

```

```

Mes    A      N      V

```

```

ENE    0.0    0.0 100.0

```

```

FEB    0.0    0.0 100.0

```

```

MAR    0.0    0.0 100.0

```

```

ABR    0.0    0.0 100.0

```

```

MAY    0.0    0.0 100.0

```

```

JUN    6.7    6.7  86.7

```

```

JUL   58.1   16.1  25.8

```

```

AGO    9.7    0.0  90.3

```

```

SEP    0.0    0.0 100.0

```

```

OCT    0.0    0.0 100.0

```

```

NOV    0.0    0.0 100.0

```

```

DIC    0.0    0.0 100.0

```

Se pueden representar gráficamente las tablas de contingencia con:

```
mosaicplot(table(PM2.5_UMBRAL, PM10_UMBRAL), color=c("red", "green"))
```

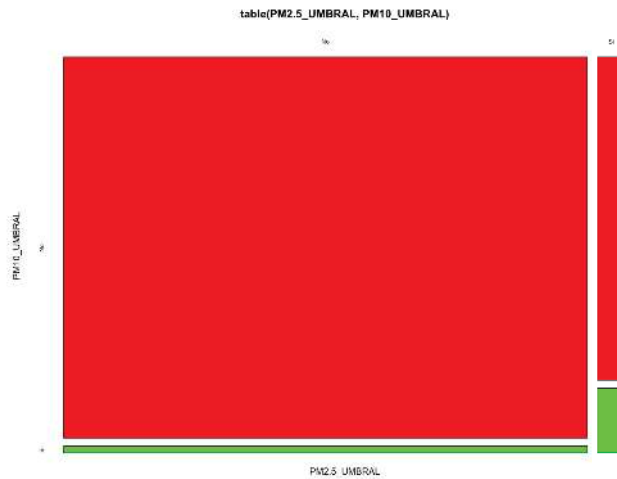


Fig. 3.10. Gráfico mosaico de la tabla de frecuencias.

```
barplot(table(T_MAX_ALERTA, Mes), xlab = "Mes", ylab = "Frecuencia", legend.
text=c("36-39", "39-42", "<36"), beside=TRUE,
col = c("yellow", "orange", "green"))
```

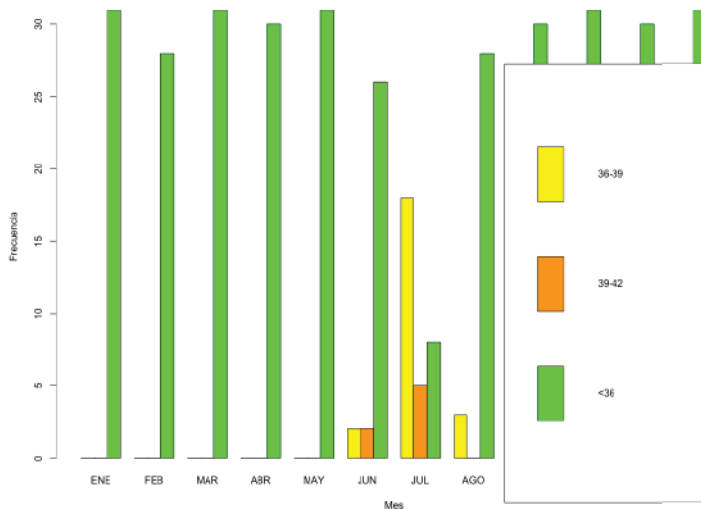


Fig. 3.11. Histograma de la frecuencia del nivel de alerta de la temperatura máxima.

3.2.2 Correlación

Realizados los estudios estadísticos de las variables de forma independiente, el siguiente paso consiste en encontrar las posibles relaciones que existen entre ellas.

3.2.2.1 Visualización

La primera inspección acerca de posibles relaciones se establece en las representaciones gráficas de unas variables frente a otras, los conocidos *diagramas de dispersión* (*scatter plot*). En la gráfica siguiente se muestra en el eje de abscisas la temperatura máxima y en el de ordenadas el nivel de monóxido de carbono y de ozono. En rojo se superpone la recta de ajuste lineal.

```
par(mfrow=c(2,2))
plot(T_MAX, CO, main="Diagrama de Dispersión", xlab="T max °C",
     ylab="CO", pch=19)
abline(lm(CO ~ T_MAX), col = "red") # regresión (y~x)
smoothScatter(T_MAX, CO, cex= 2,colramp = colorRampPalette(c("white",
"blue"), space = "Lab"))

plot(T_MAX, O3, main="Diagrama de Dispersión", xlab="T max °C",
     ylab="O3", pch=19)
abline(lm(O3 ~ T_MAX), col = "red") # regresión (y~x)
smoothScatter(T_MAX, O3, cex= 2,colramp = colorRampPalette(c("white",
"blue"), space = "Lab"))
```

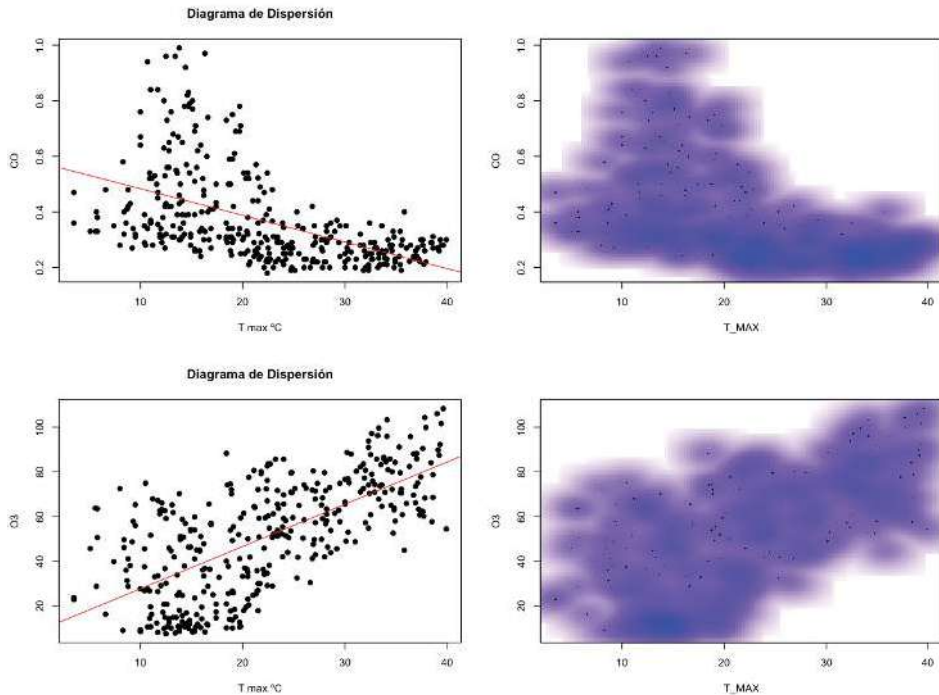


Fig. 3.12. Gráficos de dispersión con tendencia y densidad.

A simple vista se aprecia que existe mayor relación entre la temperatura máxima y el ozono que con el monóxido de carbono, posteriormente se realizarán los cálculos que permitirán cuantificar esta relación para realizar una comparación exhaustiva.

En las gráficas siguientes se representa la relación del nivel de ozono con la temperatura máxima utilizando los diagramas de dispersión que provee la librería *car*. Se representan la regresión lineal, un mapa de densidad mediante elipses y una gráfica (en rojo) de regresión local, conocida como *LOESS*.

```
library(car)

scatterplot(O3 ~ T_MAX, data = dfMedidas, main = "Diagrama de Dispersión", xlab = "T
max °C", ylab = "O3", ellipse = TRUE, lwd = 2, lty = 2)
```

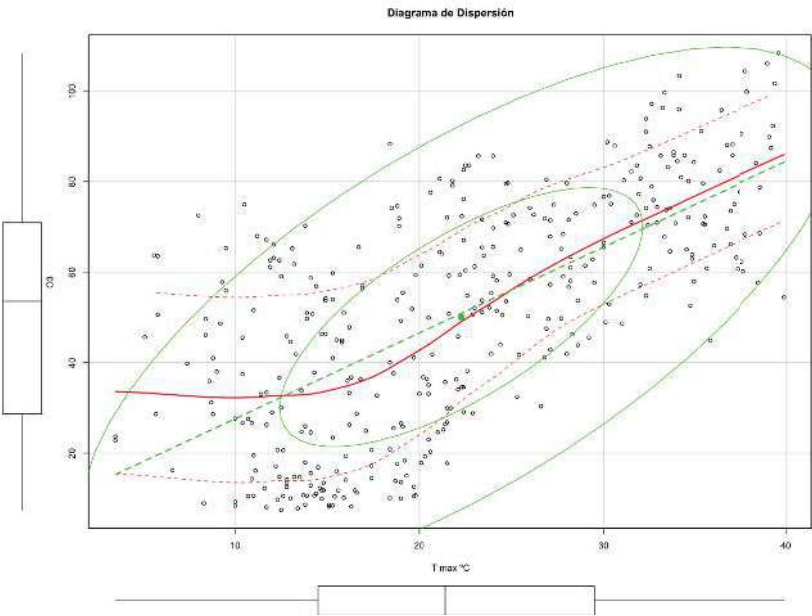


Fig. 3.13. Gráfico de dispersión de la temperatura y ozono con regresión lineal local.

```
scatter.hist(T_MAX, O3, main = "Diagrama de Dispersión", xlab = "T max °C", ylab = "O3")  
#Igual que la anterior, pero con histogramas y coeficiente de regresión
```

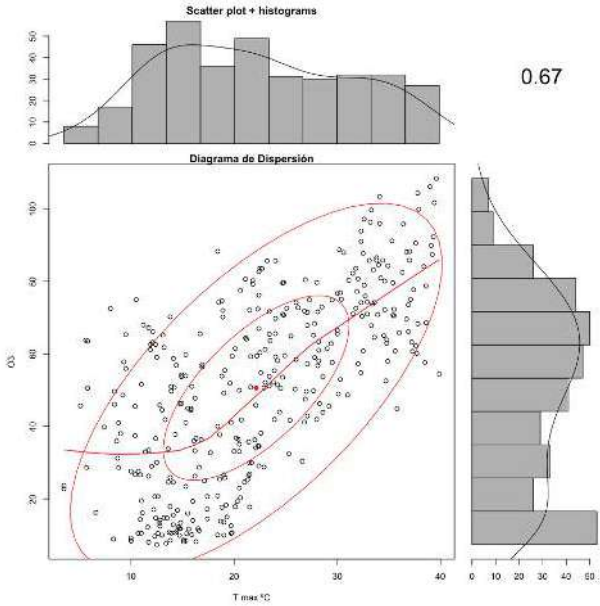


Fig. 3.14. Gráfico de dispersión con histogramas.

En la gráfica siguiente se muestra un resultado interesante relacionado con la relación del nivel de ozono y la temperatura máxima. Se ha añadido una dimensión adicional, si llovió o no. Los puntos circulares negros indican que ese día no llovió y la recta es el ajuste lineal entre la temperatura máxima y el nivel de ozono. En rojo y con triángulos están los días con lluvia y su correspondiente recta de regresión lineal. Se aprecia que la pendiente de la línea negra es mayor que la roja. Las rectas se cortan en torno a 30°C, para temperaturas mayores a ésta, la lluvia favorece la disminución del ozono, mientras que para las inferiores lo aumenta. Con estos datos se puede decir que a temperaturas máximas más bajas en días de lluvia se produce más ozono que sin lluvia, pero ocurre a la inversa para temperaturas máximas superiores a 30°C, que es cuando los niveles de ozono también son mayores y, por tanto, peligrosos, la lluvia parece atenuar el aumento del nivel de ozono.

```
scatterplot(O3 ~ T_MAX | Lluvia_SN, data = dfMedidas,
main = "Diagrama de Dispersión", xlab = "T max °C",
ylab = "O3", smoother = TRUE)
```

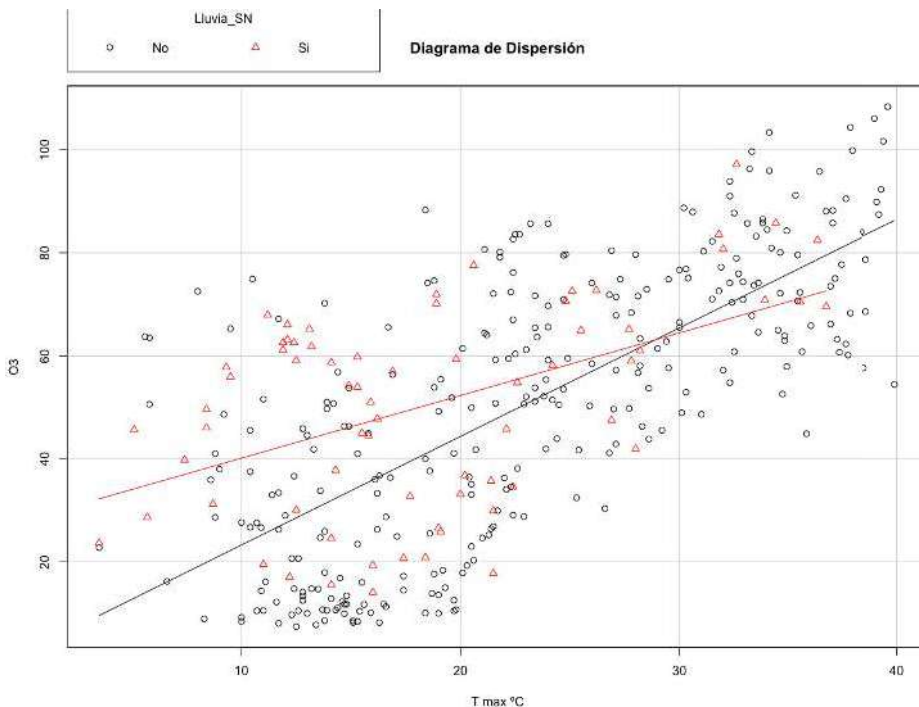


Fig. 3.15. Diagrama de dispersión con una dimensión adicional para indicar si fue o no un día de lluvia.

Se puede realizar una matriz de diagramas de dispersión para mostrar simultáneamente la relación entre varias variables.

```
scatterplotMatrix(~ O3 + SO2 + NO + Viento_MED + T_MAX, data = dfMedidas)
```

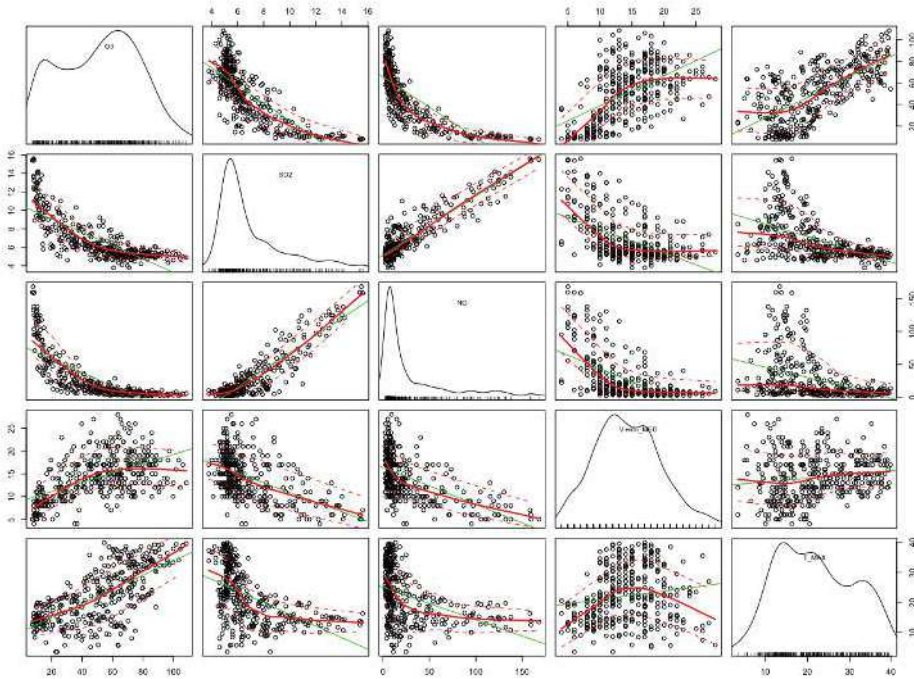


Fig. 3.16. Matriz de diagramas de dispersión para mostrar la relación entre varios atributos.

Se aprecia que entre el nivel de NO y de SO2 hay una clara relación lineal directa, los dos crecen y decrecen simultáneamente y una relación no lineal inversa entre el nivel de O3 y de NO.

La medida más simple para cuantificar la relación entre dos variables es la covarianza.

```
cov(T_MIN, CO)
cov(T_MAX, CO)
cov(T_MIN, O3)
cov(T_MAX, O3)
```

```
[1] -0.6855642
[1] -0.7795665
[1] 112.9891
[1] 154.0466
```

Se observa que la covarianza entre la temperatura máxima y el ozono tiene un valor muy grande positivo, mayor que con la temperatura mínima, de modo que puede inferirse una mayor dependencia del ozono con temperaturas altas. Sin embargo, el monóxido de carbono tiene un valor de covarianza con la temperatura negativo pero muy próximo a cero, lo cual indica que no hay relación entre ambas.

Supongamos que la temperatura que está en grados Celsius se desea tener en grados Fahrenheit y entonces se mide de nuevo el valor de covarianza.

```
cov((T_MIN * 9/5)+32, CO)
cov((T_MAX * 9/5)+32, CO)
cov((T_MIN * 9/5)+32, O3)
cov((T_MAX * 9/5)+32, O3)
```

```
[1] -1.234016
[1] -1.40322
[1] 203.3803
[1] 277.2839
```

Se observa que han cambiado los valores, aumentando, ¿puede estar más relacionado el ozono con la temperatura en grados Fahrenheit? Obviamente, no. Este ejemplo ilustra cómo el valor de covarianza cambia con la escala de las magnitudes, por tanto, hay que interpretar sus resultados siempre con suma precaución.

El coeficiente de correlación de Pearson ofrece el tipo de medida independiente de la escala que permite establecer cuantitativamente el grado de relación entre variables numéricas. Cuando se indica la correlación entre variables, se hace referencia a la medida de correlación de Pearson.

```
print("En °C")
cor(T_MIN, CO)
cor(T_MAX, CO)
cor(T_MIN, O3)
cor(T_MAX, O3)
print("En °F")
cor((T_MIN * 9/5)+32, CO)
cor((T_MAX * 9/5)+32, CO)
cor((T_MIN * 9/5)+32, O3)
cor((T_MAX * 9/5)+32, O3)
```

```
[1] "En °C"
[1] -0.5962738
[1] -0.5162716
[1] 0.6463094
[1] 0.6709398
[1] "En °F"
[1] -0.5962738
[1] -0.5162716
[1] 0.6463094
[1] 0.6709398
```

Se aprecia que se obtiene el mismo valor para la temperatura en °C y en °F. El coeficiente de correlación de Pearson está acotado en el intervalo $[-1,1]$, es adimensional y tiene el signo de la covarianza. La cuestión siguiente es acerca del umbral para el cual puede considerarse relevante la correlación. Desafortunadamente no hay una respuesta cerrada, depende del tamaño de las muestras, del tipo de variables, no tienen el mismo tratamiento las medidas de las ciencias físicas que las sociales para poder establecer relaciones entre ellas. Para una muestra mayor de 150 elementos, un valor inferior a 0.2 para la correlación indicaría una relación muy baja, baja si está entre 0.2 y 0.4, moderada para 0.4 y 0.6, alta si está entre 0.6 y 0.8 y muy alta para valores superiores a 0.8. Ése sería el caso al medir la correlación entre el monóxido de carbono y dióxido de azufre.

```
cor(CO, SO2)
```

```
[1] 0.9346811
```

La interpretación que puede hacerse es que o uno es causa del otro, es decir, el aumento en la concentración de una sustancia provoca el aumento en la otra, o que ambos tienen una causa común que los produce, o (y nunca debe desdeñarse) es una relación espúrea. Es muy conocida la ilusión de causalidad, *cum hoc ergo propter hoc*. Se tiene la falsa percepción de que los sucesos que ocurren simultáneamente están relacionados causalmente. Es decir, se considera que hay simetría en el enunciado; si existe una relación causal entre dos sucesos entonces estarán correlados. Por tanto, la inspección no termina y hay que buscar el mecanismo, la explicación que permite mostrar una relación causal donde se ha encontrado correlación lineal.

De las gráficas de dispersión mostradas anteriormente se apreciaba que para algunas variables la relación existente entre ellas podría ser no lineal. El coeficiente de correlación de Spearman permite inspeccionar relaciones más allá de la linealidad. Se aplica la función *cor* y *cor.test* de la librería *stats* que presenta más datos acerca del coeficiente de correlación.

```
cor(O3,T_MAX, method = "spearman")
cor.test(O3,T_MIN, method = "spearman")

cor(CO,T_MAX, method = "spearman")
cor.test(CO,T_MIN, method = "spearman")
```

```
[1] 0.6656529
      Spearman's rank correlation rho
data:  O3 and T_MIN
S = 2909800, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6409665

[1] -0.6266669
      Spearman's rank correlation rho
data:  CO and T_MIN
S = 13620000, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
-0.6804987
```

Comparando con el valor de correlación de Pearson se pone de manifiesto una posible relación no lineal inversa entre el monóxido de carbono y la temperatura, todavía mayor con la temperatura mínima.

Se ha calculado el grado de correlación entre variables numéricas. Si se quiere encontrar la correlación del monóxido de carbono y el ozono con la lluvia, con la variable dicotomizada, entonces se utiliza el coeficiente de correlación biserial, una modificación sobre el coeficiente de correlación de Pearson.

```
library(ltm)
biserial.cor(CO, Lluvia_SN)
biserial.cor(O3, Lluvia_SN)
```

```
[1] 0.1943241
[1] -0.001176286
```

En la tabla siguiente se hace un resumen de algunos de los coeficientes de correlación que se pueden aplicar en función del tipo de las variables.

	Cuantitativa	Ordinal	Nominal
Cuantitativa	Pearson		
Ordinal	Biserial	Spearman, Tetracórica, Policórica	
Nominal	Punto Biserial	Biserial por rangos	Phi, L, C, Lambda

Afortunadamente, hay funciones que abstraen al analista de aplicar el tipo de correlación correspondiente al tipo de dato. Por ejemplo, la función *hetcor* automáticamente realiza esa tarea.

```
hetcor(T_MAX,T_MAX_ALERTA, Lluvia_SN,O3,SO2,PM10_UMBRAL)

the correlation matrix has been adjusted to make it positive-definite
Two-Step Estimates

Correlations/Type of Correlation:
T_MAX T_MAX_ALERTA Lluvia_SN O3 SO2 PM10_UMBRAL
T_MAX 1 Polyserial Polyserial Pearson Pearson Polyserial
T_MAX_ALERTA -0.9488 1 Polychoric Polyserial Polyserial Polychoric
Lluvia_SN -0.3064 0.2621 1 Polyserial Polyserial Polychoric
O3 0.6602 -0.6853 0.02306 1 Pearson Polyserial
SO2 -0.5367 0.7083 -0.2993 -0.7543 1 Polyserial
PM10_UMBRAL 0.4185 -0.4297 -0.9457 0.0249 0.06774 1

Standard Errors:
T_MAX T_MAX_ALERTA Lluvia_SN O3 SO2
T_MAX
T_MAX_ALERTA 0.01057
Lluvia_SN 0.07179 0.1457
O3 0.02887 0.06577 0.07793
SO2 0.03837 0.05713 0.09035 0.02146
PM10_UMBRAL 0.1213 0.1773 Inf 0.141 0.1276

n = 365

P-values for Tests of Bivariate Normality:
T_MAX T_MAX_ALERTA Lluvia_SN O3 SO2
T_MAX
```

```

T_MAX_ALERTA 8.253e-05
Lluvia_SN      0.005087      0.233
O3             1.152e-07      1.516e-05      0.000816
SO2            6.064e-24      1.427e-14      6.323e-15      5.644e-19
PM10_UMBRALE  0.0001857      0.4444      <NA>      0.003571      1.209e-11

```

3.2.3 Test de hipótesis

Cuando se tratan atributos numéricos, la medida de correlación permite establecer una primera aproximación para encontrar relaciones causales entre atributos. Pero si los atributos son nominales entonces no puede establecerse un valor de correlación entre ambos. Hay medidas alternativas a la correlación que permiten encontrar relaciones entre atributos nominales. Las tablas de contingencia permiten establecer una primera aproximación para extraer relaciones entre atributos nominales.

```

library(descr)
freq(O3_Nivel, plot=FALSE) #frecuencias marginales
round(prop.table(table(T_MAX_ALERTA, O3_Nivel), 2), 2) #tabla de contingencia
round(prop.table(table(Lluvia_SN, O3_Nivel), 2), 2) #tabla de contingencia

```

```

O3_Nivel
      Frequency Percent
V           208    56.986
A           131    35.890
R            26     7.123
Total        365   100.000

      O3_Nivel
T_MAX_ALERTA  V    A    R
      A 0.02 0.10 0.23
      N 0.00 0.00 0.23
      V 0.98 0.90 0.54

      O3_Nivel
Lluvia_SN    V    A    R
      No 0.82 0.77 0.92
      Si 0.18 0.23 0.08

```

De los valores de las tablas de contingencia, observando su variación por columnas, se aprecia que las frecuencias de los niveles de ozono son diferentes dependiendo del nivel

de la temperatura máxima y si llueve o no. Pero la cuestión es si es significativamente diferente, de forma que pueda concluirse si existe o no dependencia entre el nivel de ozono con la temperatura y la lluvia. Se aplica la prueba χ^2 para medir el nivel de significancia estadística. La prueba χ^2 compara la tabla de contingencia de las frecuencias con la hipotética tabla de contingencia si no existe relación entre los atributos.

```
round(chisq.test(table(Lluvia_SN, O3_Nivel))$expected, 0) #esperada
table(Lluvia_SN, O3_Nivel) #real

round(chisq.test(table(T_MAX_ALERTA, O3_Nivel))$expected, 0) #esperada
table(T_MAX_ALERTA, O3_Nivel) #real
```

```
      O3_Nivel
Lluvia_SN V   A   R
      No 168 106 21
      Si  40  25  5

      O3_Nivel
Lluvia_SN V   A   R
      No 170 101 24
      Si  38  30  2

      O3_Nivel
T_MAX_ALERTA V   A   R
      A   13   8   2
      N    4   3   0
      V 191 120 24

      O3_Nivel
T_MAX_ALERTA V   A   R
      A    4  13   6
      N    1   0   6
      V 203 118 14
```

Los valores esperados son similares de los reales, más para el caso de la lluvia, pero ¿son suficientemente semejantes o las diferencias son significativas?

```
chisq.test(table(Lluvia_SN, O3_Nivel))
chisq.test(table(T_MAX_ALERTA, O3_Nivel))
```

```
Pearson's Chi-squared test

data:  table(Lluvia_SN, O3_Nivel)
X-squared = 3.495, df = 2, p-value = 0.1742

Warning message:
In native_encode(res, to = encoding) :
  some characters may not work under the current locale

Pearson's Chi-squared test

data:  table(T_MAX_ALERTA, O3_Nivel)
X-squared = 91, df = 4, p-value < 2.2e-16
```

De los valores de *p-value* se puede concluir que al nivel de significancia de 0.05, para el nivel de lluvia, no se puede descartar la hipótesis de nula, $p\text{-value}=0.1741$, de forma que no hay diferencia significativa entre los valores previstos y obtenidos en la relación entre la lluvia y el ozono. Por contra, para la temperatura, el valor de *p-value* es mucho menor que 0.05, de modo que se puede rechazar la hipótesis nula y, por tanto, existe una relación significativa entre la temperatura máxima y el ozono.

Una forma rápida de realizar las medidas anteriores es usar la función *assoc* de la librería *vcd*.

```
assoc(table(Lluvia_SN, O3_Nivel), shade=TRUE)
assoc(table(T_MAX_ALERTA, O3_Nivel), shade=TRUE)
```

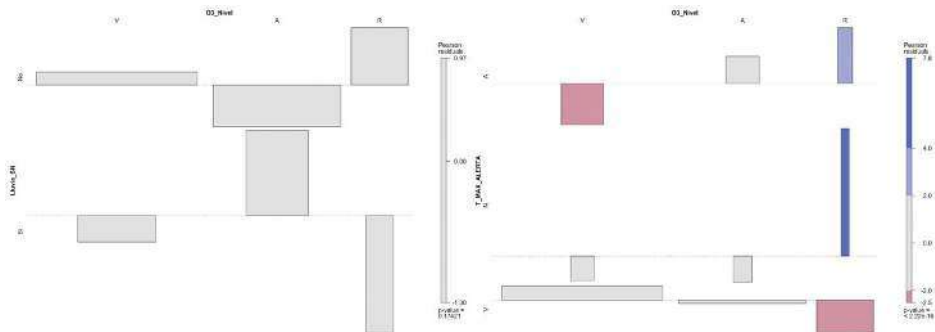


Fig. 3.17. Prueba χ^2 entre lluvia, temperatura y su relación con el ozono.

Para realizar la representación gráfica de la figura 3.17, para cada valor de la tabla de contingencia se calcula la significancia estadística respecto del valor esperado. El área del cuadrado es proporcional a la diferencia entre la frecuencia esperada y la observada, y el color es proporcional a la correlación entre los valores, más roja para una relación inversa y azul para la directa. Para la lluvia, las diferencias no son significativas, mientras que para la temperatura se pueden establecer las reglas de que, en nivel verde de temperatura máxima, el nivel del ozono no será rojo, en nivel amarillo de temperatura, el nivel del ozono no será verde y posiblemente será rojo, y finalmente, para nivel naranja de temperatura, lo más probable para el ozono es que estará en nivel rojo.

3.2.4 Representación de datos

Otra forma de poner de manifiesto relaciones entre los datos es realizando representaciones gráficas que, de forma cualitativa, permitan relacionar unas distribuciones de datos con otras.

En la siguiente gráfica se muestra el histograma de valores de O₃ máximo.

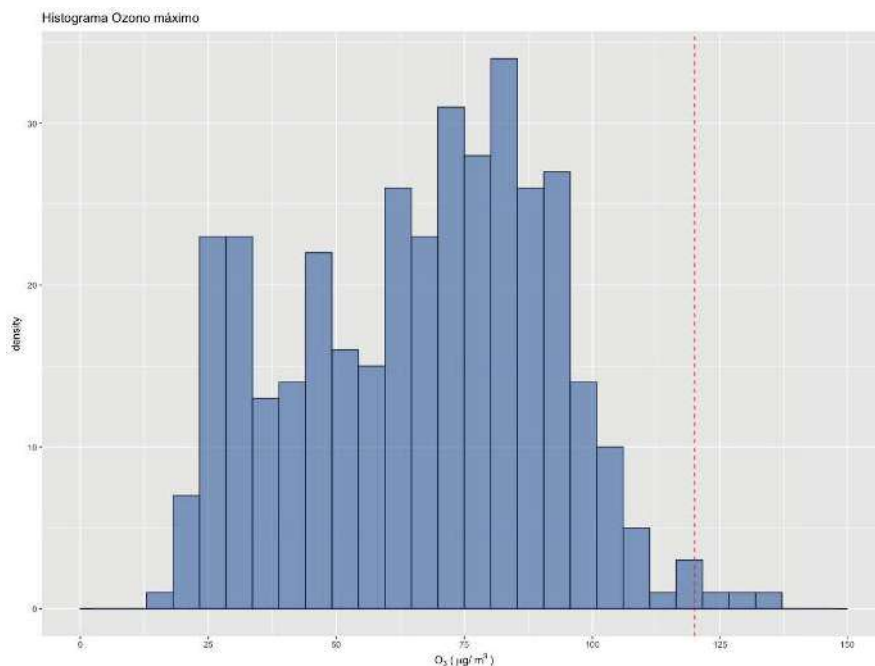


Fig. 3.18. Histograma de concentración de ozono máximo.

```
qplot(dfMedidas$O3_MAX, data=dfMedidas, geom="density", alpha=I(.5)) +
  xlab(bquote(O[3]~"(" ~mu*"g/"*~m^3~")")) +
  geom_histogram(fill = "#4271AE", colour = "#1F3552", alpha = 0.6) +
  scale_x_continuous(breaks = seq(0, 150, 25), limits=c(0, 150)) +
  geom_vline(xintercept=120, linetype="dashed", color = "red") +
  ggtitle("Histograma Ozono máximo")
```

En la línea roja se muestra la circunstancia en la que el ozono alcanza un nivel peligroso, como se observa, ocurre con poca frecuencia. Una forma más conveniente es representar la función “distribución de densidad”.

```
dfMedidas$Mes2 <- factor(dfMedidas$Mes, unique(dfMedidas$Mes))

qplot(O3_MAX, data = dfMedidas, geom = "density", alpha = I(.5)) +
  labs(x = bquote(O[3]~"(" ~mu*"g/"*~m^3~")")) +
  geom_density(fill = "#4271AE", colour = "#1F3552", alpha = 0.6) +
  scale_x_continuous(breaks = seq(0, 150, 25), limits = c(0, 150)) +
  geom_vline(xintercept = 120, linetype="dashed", color = "red") +
  ggtitle("Distribución de Densidad Máxima de Ozono")
```

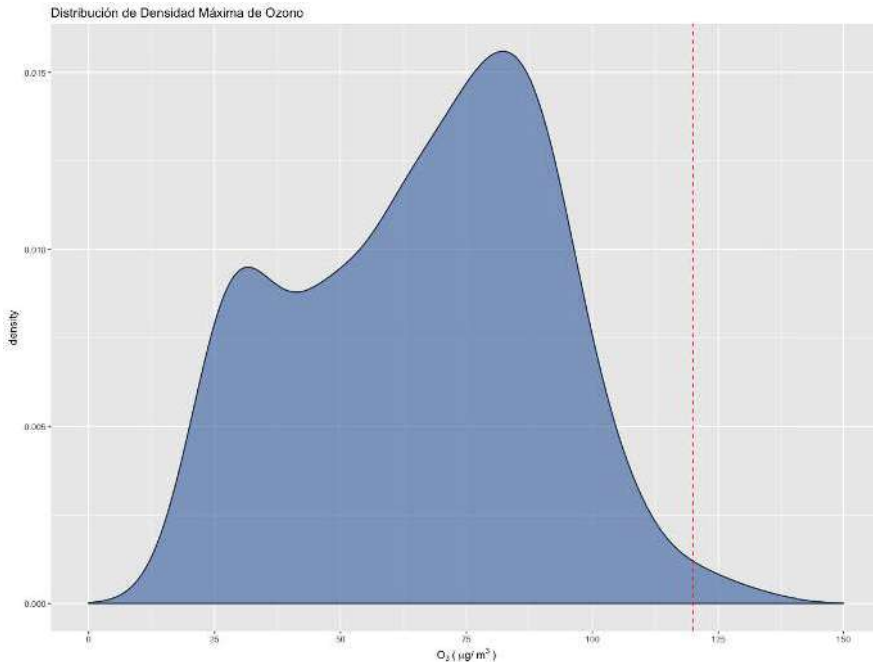


Fig. 3.19. Distribución de densidad de ozono.

Se puede comparar con otras medidas de calidad del aire para comprobar si existen semejanzas entre las distribuciones. Pero es más razonable comparar las distribuciones sobre los valores estandarizados. Para estandarizar un atributo se restan los valores por la media y se divide por su desviación estándar. De esta forma, todos los atributos tendrán una distribución de media 0 y desviación igual a 1. La estandarización hace que magnitudes diferentes sean comparables, y es un proceso, junto con la normalización, necesario para poder posteriormente aplicar las técnicas de análisis de datos. En caso contrario, se introducen sesgos, atributos cuyos rangos de valores están en valores grandes, serán pesados como más importantes, frente a otros atributos que tienen sus valores en rangos de valores pequeños.

```
h1 <- ggplot(data=dfMedidas, aes((O3_MAX - mean(O3_MAX)) / sd(O3_MAX))) +
  geom_histogram(breaks=seq(-3, 3, by = 0.5), col = "gray", fill = "green", alpha =
0.2) +
  labs(title = "Histograma") +
  labs(x = bquote(O[3]~" ("~mu*"g/"*~m^3~" )"), y = "Cantidad")

h2 <- ggplot(data=dfMedidas, aes((dfMedidas$CO - mean(dfMedidas$CO)) / sd(dfMedidas$-
CO))) +
  geom_histogram(breaks=seq(-3, 3, by = 0.5), col="gray", fill="green", alpha = 0.2) +
  labs(title="Histograma") +
  labs(x = bquote(CO~" ("~mu*"g/"*~m^3~" )"), y="Cantidad")

h3 <- ggplot(data=dfMedidas, aes((dfMedidas$NO2 - mean(dfMedidas$NO2)) / sd(dfMedi-
das$NO2))) +
  geom_histogram(breaks=seq(-3, 3, by = 0.5), col="gray", fill="green", alpha = 0.2) +
  labs(title="Histograma") +
  labs(x = bquote(NO[2]~" ("~mu*"g/"*~m^3~" )"), y="Cantidad")

library(gridExtra)
grid.arrange(h1, h2, h3, ncol = 1, nrow = 3)
```

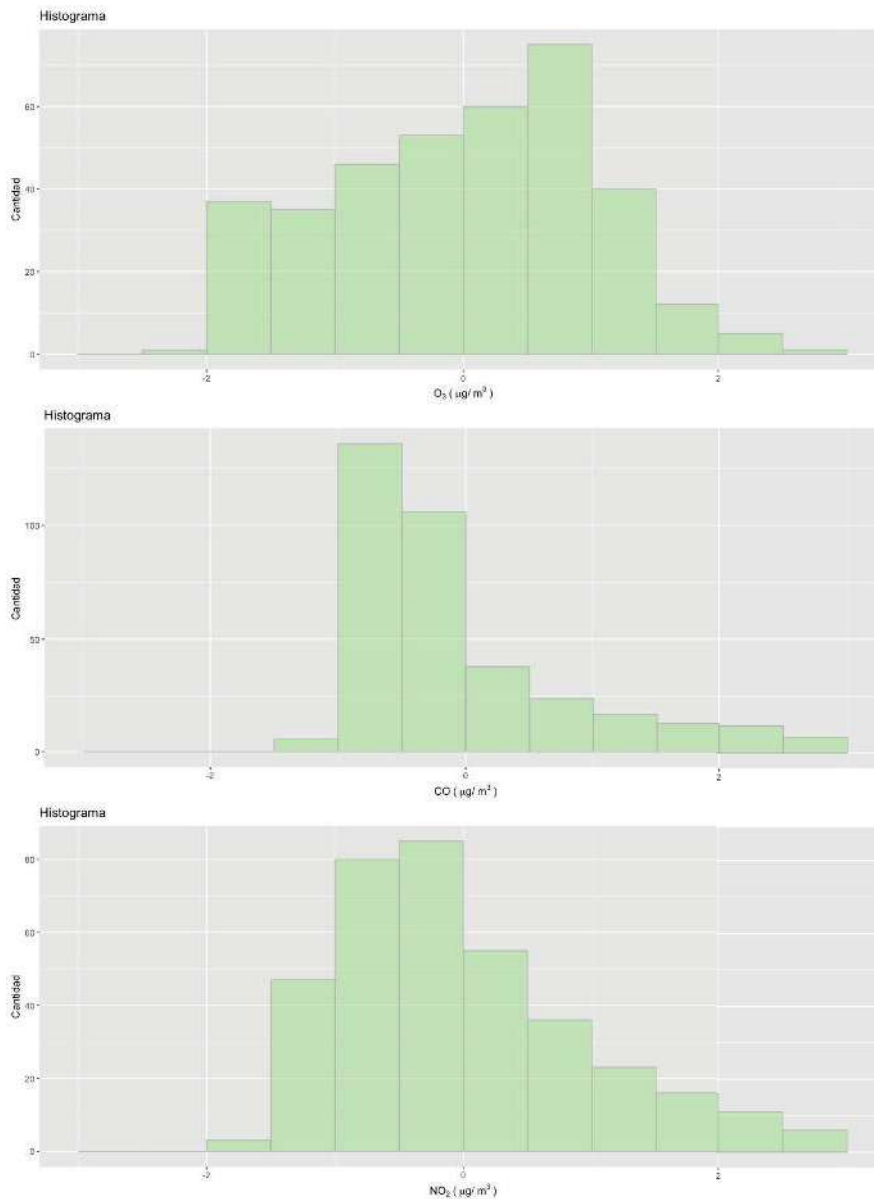


Fig. 3.20. Histograma de los valores estandarizados.

Se aprecia en la figura anterior que la distribución de valores de las concentraciones de NO_2 y CO es más parecida entre sí que con la concentración de ozono.

Otras formas de poner de relieve información adicional que permite encontrar relaciones entre datos consiste en realizar representaciones gráficas que introduzcan información de dimensiones alternativas. Por ejemplo, cambiando el tamaño, forma, color, transparencia se pueden añadir hasta cuatro dimensiones adicionales a un diagrama de dispersión. Seguramente utilizar tantas dimensiones adicionales no sea una buena idea, ya que nuestra percepción no está preparada para manejar relaciones entre seis atributos simultáneamente. La elección de la representación queda a la eficacia de la comunicación, que no debe ser confundida con la cantidad.

Haciendo uso de la magnífica librería para representación gráfica, *ggplot2*, se muestran en el ejemplo siguiente cuatro atributos simultáneamente.

```
ggplot(dfMedidas, aes(x = T_MAX, y = O3_MAX, size=T_MIN, fill = Lluvia_SN)) +
  ylab(bquote(O[3]~" ("~mu*"g/"*m^3~")")) +
  xlab("T máxima (°C)") +
  geom_point(shape = 21)+
  ggtitle("Concentración Máxima de Ozono") +
  scale_size(range = c(1, 5))+
  geom_hline(yintercept=120, linetype="dashed", color = "red")
```

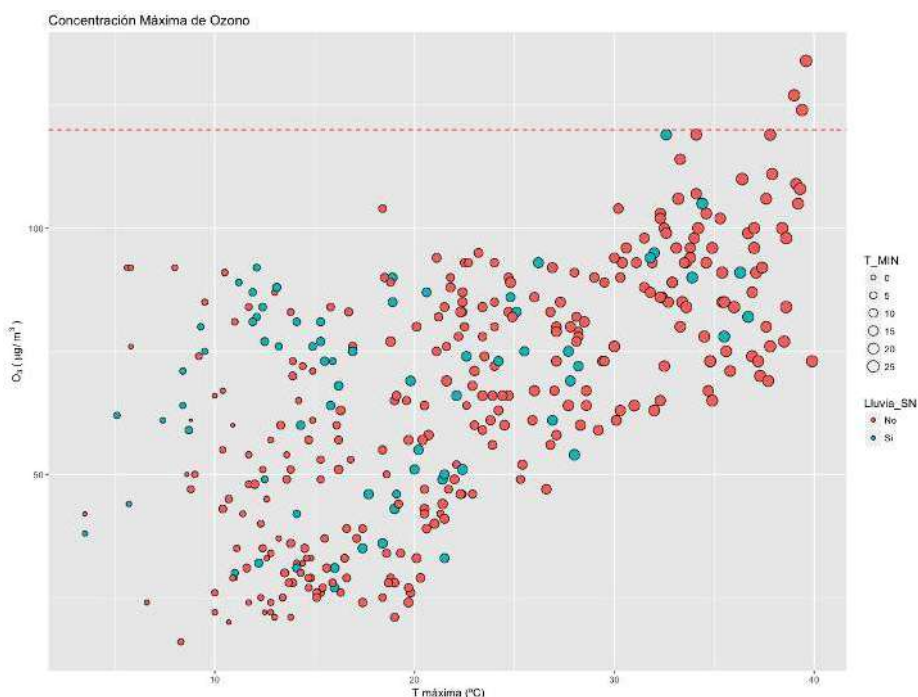


Fig. 3.21. Diagrama de dispersión del ozono y la temperatura máxima junto con temperatura mínima y día con lluvia.

El tamaño del círculo, en la figura anterior, es proporcional a la temperatura mínima y el color indica si ese día llovió o no. Se observa que la concentración de ozono es mayor cuando tanto la temperatura máxima como la mínima son mayores, es decir, más a la derecha y círculos más grandes, pero en apariencia independiente de la lluvia. Si ahora representamos la concentración de SO₂:

```
ggplot(dfMedidas, aes(x = T_MAX, y = SO2, size=T_MIN, fill = Lluvia_SN)) +
  ylab(bquote(SO[2]~" ("~mu*g/"*~m^3~")" )) +
  xlab("T máxima (°C)") +
  geom_point(shape = 21)+
  ggtitle("Concentración de Dióxido de Azufre")
```

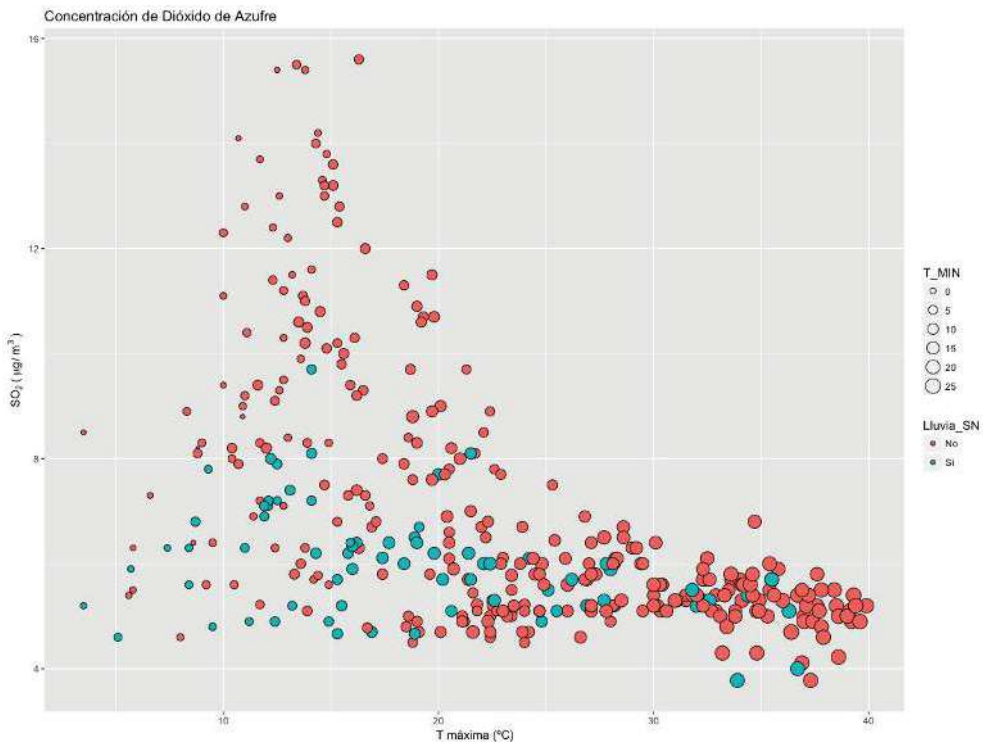


Fig. 3.22. Diagrama de dispersión del dióxido de azufre y la temperatura máxima junto con temperatura mínima y día con lluvia.

Se aprecia una diferencia notable con respecto a la relación entre la temperatura máxima y mínima y la concentración de SO₂ y el ozono. Ahora la concentración máxima de SO₂ ocurre cuando la temperatura mínima es más pequeña y para temperaturas máximas y mínimas superiores su concentración decrece de forma clara.

En un diagrama de dispersión de la temperatura máxima y mínima se puede apreciar aún más la dependencia de las concentraciones de ozono y SO2 con la temperatura.

```
ggplot(data = dfMedidas, aes(x = T_MAX, y = T_MIN)) +
  geom_point(aes(color = O3_MAX, size = SO2)) +
  xlab("T Máxima (°C)") +
  ylab("T Mínima (°C)") +
  scale_colour_gradient2(low = "#006000", mid = "#FFFF00", high = "#FF0000", midpoint = 75) +
  labs(size = bquote(SO[2]), colour = bquote(O[3]~"máx")) +
  geom_smooth(method = "lm")
```

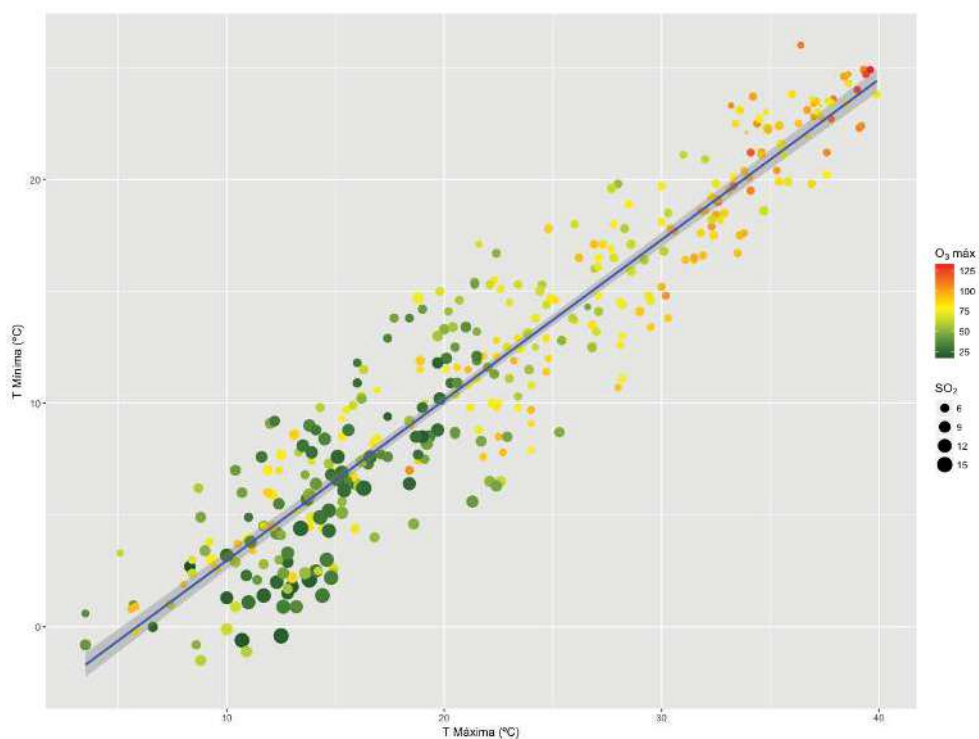


Fig. 3.23. Gráfica de dispersión de ozono y dióxido de azufre.

En código de color se representa la concentración de ozono y mediante el tamaño del punto la concentración de dióxido de azufre. Se aprecia que para temperaturas mínimas y máximas más altas los colores son más naranjas-rojos, y verdes en el caso contrario. El tamaño del punto tiene el comportamiento inverso. En azul se representa la regresión lineal entre las temperaturas máximas y mínimas.

Otra comparación adecuada es mediante el *boxplot* de las concentraciones.

```
h1 <- ggplot(dfMedidas, aes(x=Mes2,y=SO2,fill=Mes)) +
geom_boxplot(outlier.size=0.5) +
  labs(x="Mes",y= bquote(SO[2]~"(" ~mu*"g/"*~m^3~")")) +
  guides(fill=FALSE) +
  theme(axis.text.x=element_text(size=5), axis.title.x=element_text(size=10,-
face="bold")) +
ggtitle(bquote("Emisiones mensuales medias"~SO[2]))

h2 <- ggplot(dfMedidas, aes(x=Mes2,y=O3,fill=Mes)) +
geom_boxplot(outlier.size=0.5) +
  labs(x="Mes",y= bquote(O[3]~"(" ~mu*"g/"*~m^3~")")) +
  guides(fill=FALSE) +
  theme(axis.text.x=element_text(size=5), axis.title.x=element_text(size=10,-
face="bold")) +
ggtitle(bquote("Emisiones mensuales medias"~O[3]))

h3 <- ggplot(dfMedidas, aes(x=Mes2,y=T_MAX,fill=Mes)) +
geom_boxplot(outlier.size=0.5)+guides(fill=FALSE) +
labs(x="Mes",y="Temperatura") +
theme(axis.text.x=element_text(size=5), axis.title.x=element_text(size=10,face="bold"))
+
ggtitle("Temperatura máxima por mes")

grid.arrange(h1, h2, h3, ncol = 1, nrow = 3)
```

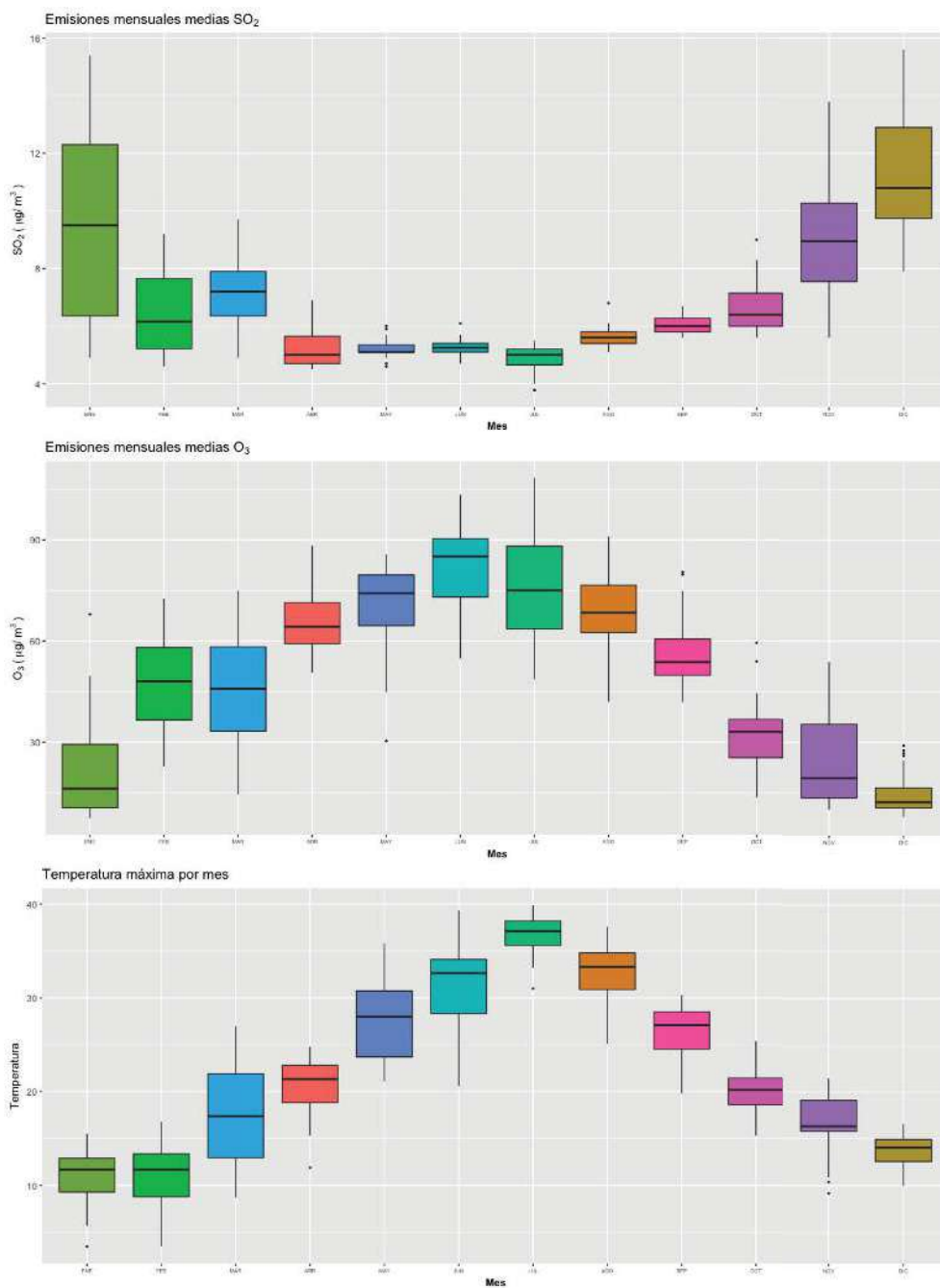


Fig. 3.24. *Boxplot* de las concentraciones de ozono, dióxido de azufre por mes y temperatura máxima por mes.

Estas gráficas con los *boxplots* ponen también de manifiesto la variación de las concentraciones con la temperatura. Se quiere mostrar que la misma información puede visualizarse de diferentes maneras.

Una forma muy habitual de trabajar con R es utilizando librerías específicas desarrolladas para áreas concretas. Por ejemplo, la librería *openair* está creada para facilitar la representación gráfica de valores de calidad del aire.

Requiere tener los datos en una serie temporal con el atributo de la fecha con el nombre *date*.

```
library(openair)
library(dplyr)

dfMedidas_t <- dfMedidas

dfMedidas_t <- cbind(seq(c(ISOdate(2015,1,1)), by = "day", length.out = 365), dfMedidas)
colnames(dfMedidas_t)[1] <- "date"

summaryPlot(select(dfMedidas_t, c(date,T_MAX,T_MIN,O3_MAX,SO2_MAX)))
```

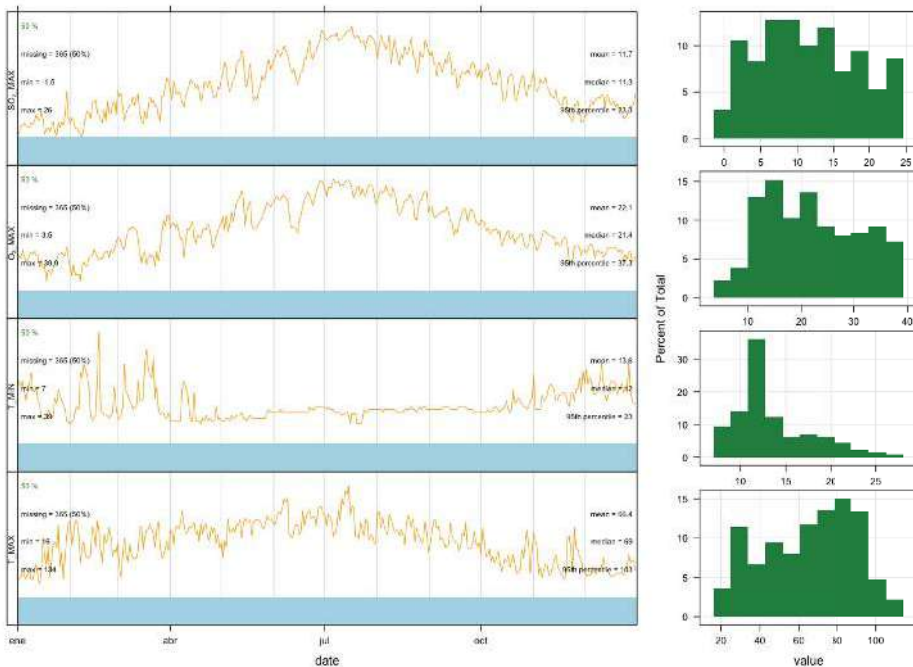


Fig. 3.25. Gráficos resumen de magnitudes temporales.

En los gráficos de sumario de las magnitudes representadas se muestra mucha información estadística; evolución temporal, histograma, percentiles, valores extremos.

Se puede representar sólo la evolución temporal anual o promediándola mensualmente.

```
timePlot(dfMedidas_t,pollutant = c("T_MAX","T_MIN","O3_MAX","SO2_MAX"))
monthplot(dfMedidas_t$O3_MAX, main=expression(paste("2015 Gráfico Mensual O"[3])),
ylab="Conc (ug/m3)")
```

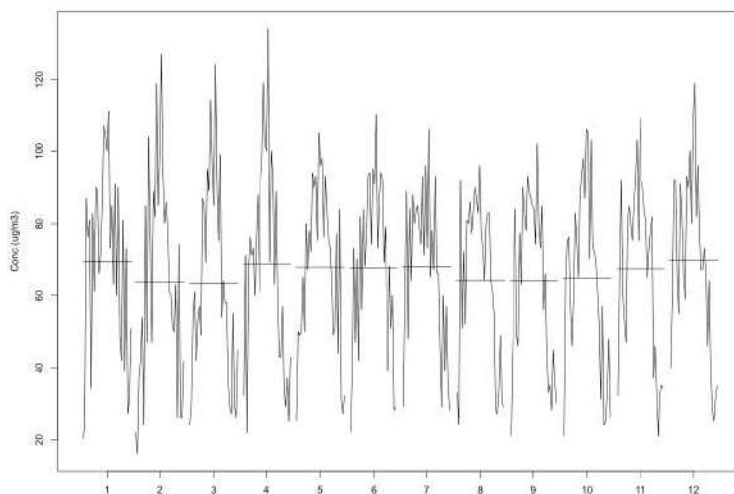
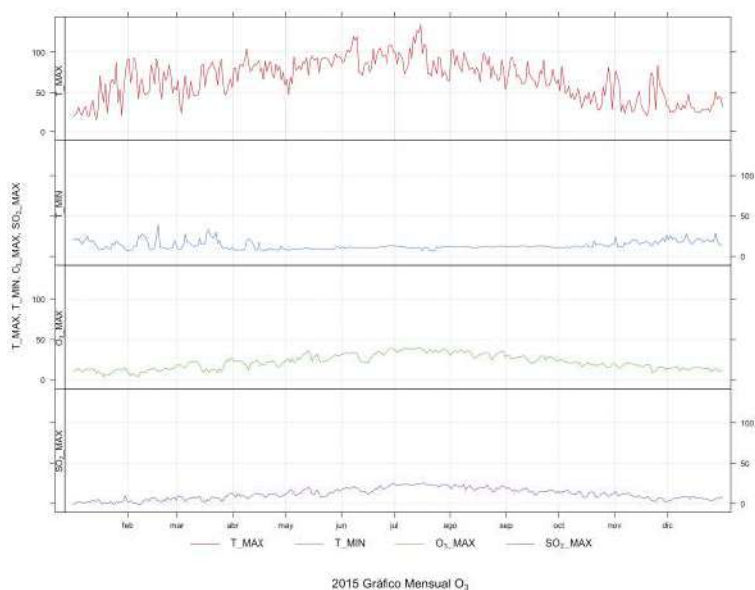


Fig. 3.26. Gráficos de evolución temporal.

Se pueden comparar las variaciones temporales en diferentes rangos; horas, días, semanas, meses. Dado que los datos se tienen sólo con periodicidad diaria y la representación que permite la función *timeVariation* puede mostrar las variaciones horarias, en la gráfica siguiente, las correspondientes imágenes por hora del día y por hora y día de la semana carecen de sentido y aparecen con líneas verticales.

```
timeVariation(dfMedidas_t, pollutant = "O3")
```

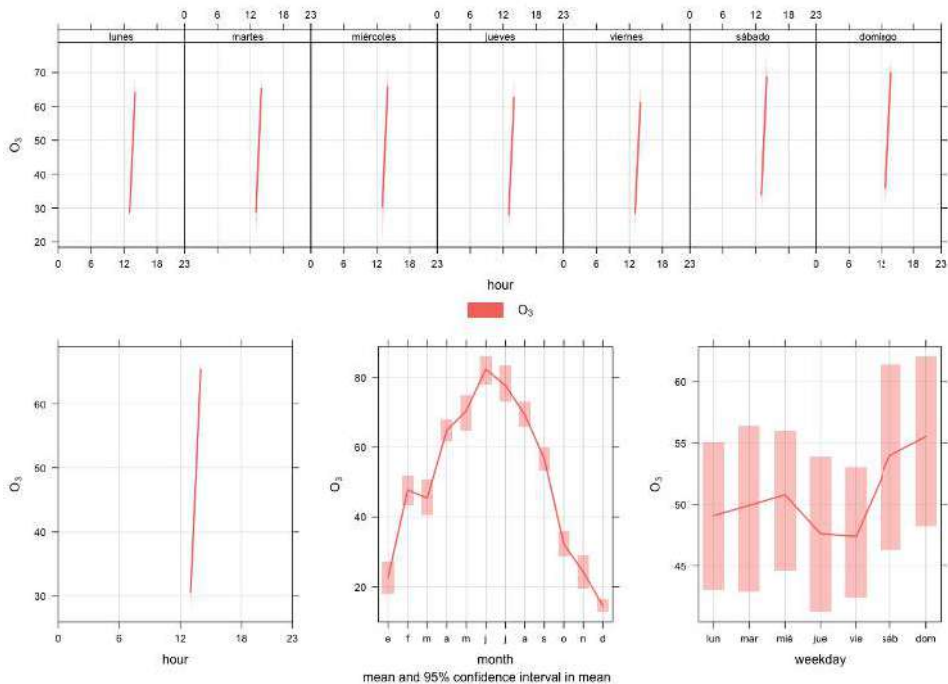


Fig. 3.27. Diferentes variaciones temporales para la concentración de ozono.

Mapas de calor especificando la temporalidad de los ejes:

```
trendLevel(dfMedidas_t, pollutant = "SO2")
trendLevel(dfMedidas_t, pollutant = "SO2", x= "month", y="weekday")
```

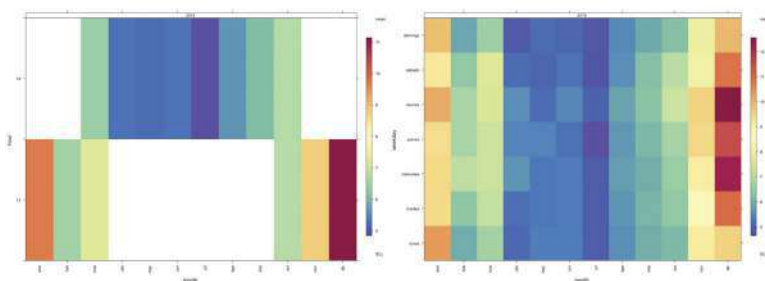


Fig. 3.28. Mapas de calor para el dióxido de azufre.

O modelos predictivos a nivel de percentil:

```
smoothTrend(dfMedidas_t, pollutant = "SO2", deseason = TRUE, statistic = "percentile",
percentile = c(25, 50, 75, 95))
```

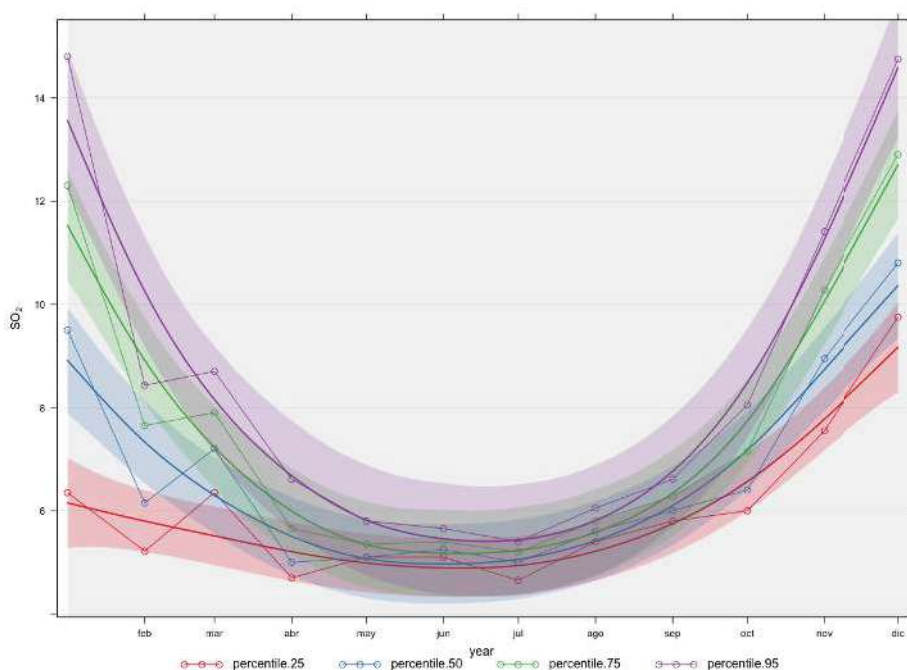


Fig. 3.29. Modelo predictivo de la evolución de los percentiles de la concentración de dióxido de azufre.

Tiene un gráfico específico para mostrar los valores en un calendario, asociando niveles a valores de concentración:

```
calendarPlot(dfMedidas_t,pollutant = "O3_MAX", w.shift = 2, breaks = c(0,50,100,120,140),
labels = c("Bajo","Medio","Alto","Peligro"), cols = c("darkgreen","green","yellow","red"))
```



Fig. 3.30. Gráfico de calendario de la concentración máxima de ozono.

Hay áreas, como las finanzas, genética, donde se disponen de gráficos y representaciones propias adecuadas al tipo de dato e información de estos dominios. Para finalizar esta sección dedicada a la representación de datos, se van a mostrar algunos ejemplos con gráficos interactivos. Este tipo de gráfico está pensado para interactuar, incluido en una página web, y permite que el usuario muestre información contextual al pinchar con el ratón o seleccionar una parte del mismo, así como salvar datos en diferentes formatos, etc.

```
library(rAmCharts)

ZoomButton <- data.frame(Unit = c("MM", "MM", "MAX"), multiple = c(1, 3, 1),
  label = c("Mes", "3 Meses", "Año"))

amTimeSeries(dfMedidas_t, "date", c("O3", "O3_MAX"), bullet = "round",
  color = c("blue", "red"),
  ZoomButton = ZoomButton, main = "Ozono 2015",
  export = TRUE)
```

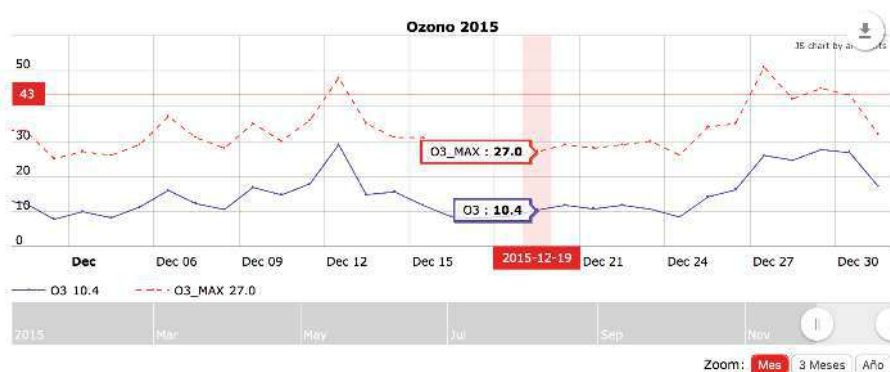


Fig. 3.31. Gráfico interactivo de la evolución temporal del ozono.

Una forma muy flexible y potente de utilizar gráficos interactivos es usando la librería *shiny*, que permite la creación de una página web con controles y salidas de datos interactivos, ya sean en tablas o gráficos. Con poco esfuerzo pueden desarrollarse cuadros de mandos interactivos. Con *shiny* pueden usarse los gráficos generados con el resto de librerías base, como *ggplot*, y otros de carácter interactivo como *plotly*.

```
library(plotly)

p <- ggplot(data = dfMedidas, aes(x = Fecha, y = O3_MAX)) +
  geom_point(aes(colour = T_MAX, text = paste("O3:", O3)), size = 1) +
  geom_smooth()

ggplotly(p)
```

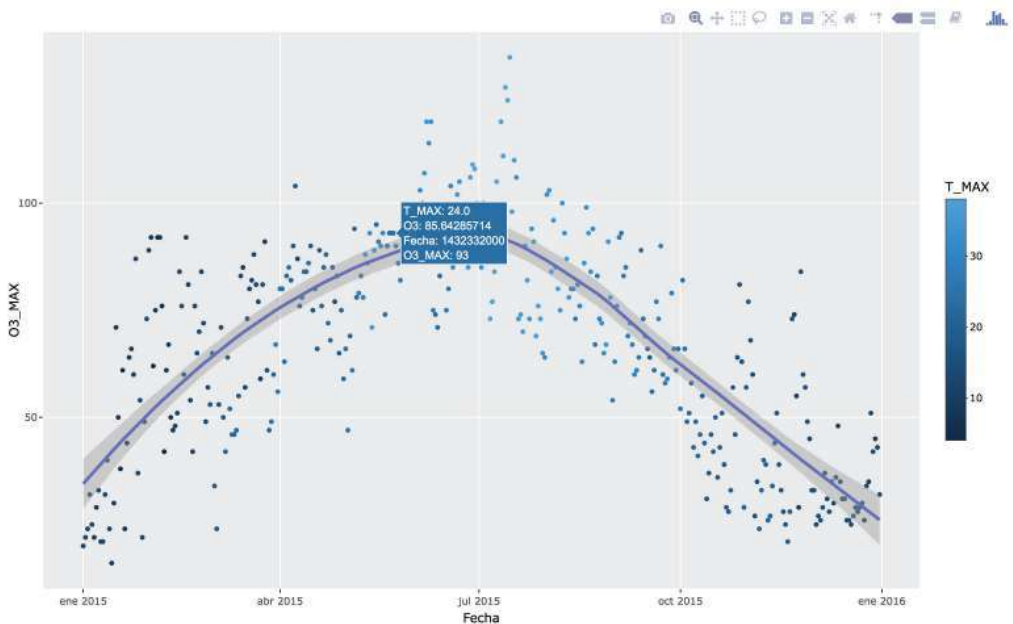


Fig. 3.32. Gráfico interactivo con *plotly*.

Al situar el ratón sobre el punto, aparece un cuadro de texto mostrando el valor.

Con esta panorámica finaliza esta sección donde se realiza una exploración de los datos. Podría decirse que todas las técnicas vistas hasta el momento permiten analizar y explicar el pasado, en las secciones siguientes se explorarán los datos para poder predecir el futuro.

Predicción y clasificación con técnicas numéricas

CAPÍTULO 4

4.1 Técnicas numéricas de predicción

La predicción numérica es el proceso que intenta determinar los valores de una o varias variables a partir de un conjunto de datos. La predicción de valores continuos puede planificarse por las técnicas estadísticas de regresión [JAM85, DEV95, AGR96], por ejemplo, para predecir el sueldo de un graduado de la universidad con diez años de experiencia de trabajo o las ventas potenciales de un nuevo producto dado su precio. Se pueden resolver muchos problemas por medio de la regresión lineal, y puede conseguirse todavía más aplicando las transformaciones a las variables para que un problema no lineal pueda convertirse en uno lineal. A continuación, se presenta una introducción intuitiva de las ideas de regresión lineal, múltiple y no lineal, así como la generalización a los modelos lineales.

Más adelante, dentro de la clasificación, se estudiarán técnicas de predicción que pueden servir para la clasificación, destacando la regresión logística y a continuación la clasificación bayesiana, primero para variables numéricas y después nominales.

4.1.1 Regresión lineal

La regresión numérica es uno de los procedimientos que más se ha utilizado para el análisis predictivo. La idea es partir de una hipótesis de modelo de relación entre las variables de entrada y salida, que a continuación es aplicado a los datos disponibles para generar los parámetros que mejor ajustan el modelo, y es validado con algún test apropiado que nos dé la confianza que permita aplicar este modelo para predecir los

valores con datos futuros. Los modelos más aplicados son los que asumen linealidad en las relaciones, lo que facilita la aplicación de métodos muy generales y eficientes, como es la regresión por mínimos cuadrados (MC), que tiene una extensión directa cuando existe un modelo gaussiano de la incertidumbre de cada variable de entrada, el método mínimos cuadrados ponderados (MCP). Cuando no puede suponerse esta relación lineal directa, la primera estrategia posible es la transformación de los datos a un espacio en el que puedan aplicarse los modelos lineales, lo que constituye la base del método conocido como GLM (*Generalized Linear Model*).

4.1.1.1 Regresión lineal simple

La regresión lineal [DOB90] es la forma más simple de regresión, ya que en ella se modelan los datos usando una línea recta. Se caracteriza, por tanto, por la utilización de dos variables, una aleatoria, y (llamada *variable respuesta*), que es función lineal de otra variable aleatoria, x (llamada *variable independiente*), formándose la ecuación:

$$y=a+bx \quad \text{Ec. 4.1}$$

En esta ecuación la variación de y se asume que es constante, y a y b son los coeficientes de regresión que especifican la intersección con el eje de ordenadas y la pendiente de la recta, respectivamente. Estos coeficientes se calculan utilizando el método de los mínimos cuadrados [PTVF96] que minimizan el error entre los datos reales y la estimación de la línea. Dados s ejemplos de datos en forma de puntos $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, entonces los coeficientes de la regresión pueden estimarse según el método de los mínimos cuadrados con las ecuaciones:

$$b = \frac{S_{xy}}{S_x^2} \quad \text{Ec. 4.2}$$

$$a=y-bx \quad \text{Ec. 4.3}$$

En la ecuación 4.2, S_{xy} es la covarianza de x e y , y S_x^2 la varianza de x .

Para comprobar la validez de la recta de regresión construida, se emplea el coeficiente de regresión (ecuación 4.4), que es una medida del ajuste de la muestra.

$$R^2 = \frac{S_{xy}^2}{S_x^2 S_y^2} \quad \text{Ec. 4.4}$$

El valor de R^2 debe estar entre 0 y 1. Si se acerca a 0 la recta de regresión no tiene un buen ajuste, mientras que si se acerca a 1 el ajuste es “perfecto”. Los coeficientes a y b a menudo proporcionan buenas aproximaciones a otras ecuaciones de regresión complicadas. Como ejemplo, a continuación se muestra un conjunto de treinta y cinco marcas de cerveza, se estudia la relación entre el grado de alcohol de las cervezas y su contenido calórico, y se representa un pequeño conjunto de datos.

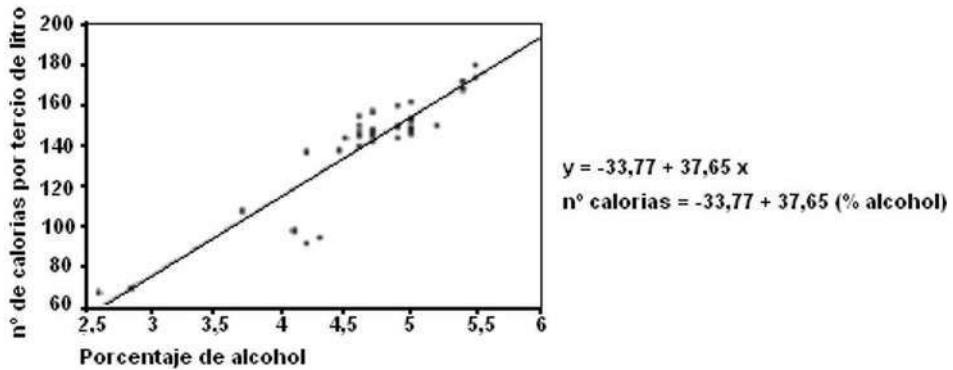


Fig. 4.1. Regresión lineal simple.

El eje vertical muestra el número de calorías (por cada tercio de litro) y el horizontal el contenido de alcohol (expresado en porcentaje). La nube de puntos es la representación de los datos de la muestra, y la recta es el resultado de la regresión lineal aplicando el ajuste de los mínimos cuadrados. En los siguientes apartados se mostrarán dos tipos de regresiones que amplían la regresión lineal simple.

4.1.1.2 Regresión lineal múltiple

La regresión lineal múltiple [PTVF96] es una extensión de regresión lineal que involucra más de una variable independiente, y permite que la variable respuesta y sea planteada como una función lineal de un vector multidimensional. El modelo de regresión múltiple para I variables predictoras sería como el que se muestra en la ecuación 4.5.

$$y = a_0 + a_1 x_1 + a_2 x_2 + \dots + a_I x_I \quad \text{Ec. 4.5}$$

Por tanto, se consideran relaciones de una variable de salida (dependiente) con múltiples variables de entrada (independientes). Este problema se puede representar de la siguiente manera:

Dada la muestra de datos $\{(\vec{X}_1, y_1), (\vec{X}_2, y_2), \dots, (\vec{X}_n, y_n)\}$, donde \vec{X} son vectores de I dimensiones, se busca estimar la función que mejor “explique” dichos datos:

$$g(.): R^I \rightarrow R$$

$$\bar{X} \rightarrow \hat{y} = g(\bar{X})$$

Ec. 4.6

El procedimiento de resolución para estimar dicha función es el procedimiento de mínimos cuadrados, que estima el vector de coeficientes que minimiza el error:

$$\hat{y}_i = g(\bar{X}^i) = a_0 + a_1 x_1^i + \dots + a_I x_I^i = a_0 + \sum_{p=1}^I a_p x_p^i = (\bar{X}^i)^t \bar{a}$$

Ec. 4.7

Siendo:

$$\bar{a} = [a_0 \quad a_1 \quad \dots \quad a_I]^t; \quad \bar{X}^i = [1 \quad x_1^i \quad \dots \quad x_I^i]^t$$

Y siendo \hat{y}_i el estimador calculado para la variable y_i .

Con este modelo se supone que la variable y puede predecirse a partir de las variables independientes X_i con un error que se asume independiente y únicamente en la variable dependiente:

$$y_i = (\bar{X}^i)^t \bar{a} + \varepsilon_i$$

Ec. 4.8

El objetivo es que dadas N muestras, el procedimiento debe determinar coeficientes que minimicen el error de predicción global.

$$E = \sum_{i=1}^n \varepsilon_i = \sum_{i=1}^n (g(\bar{X}^i) - y_i)^2$$

Ec. 4.9

Éste es un problema clásico de minimización de función cuadrática cuya solución global es única. La formulación genérica matricial del problema se puede expresar así:

$$\bar{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^n \end{bmatrix}; \quad \hat{g} = \begin{bmatrix} \hat{y}^1 \\ \vdots \\ \hat{y}^n \end{bmatrix} = \begin{bmatrix} g(\bar{X}^1) \\ \vdots \\ g(\bar{X}^n) \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \dots & x_I^1 \\ 1 & x_1^2 & \dots & x_I^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^n & \dots & x_I^n \end{bmatrix} \bar{a} = H \bar{a}$$

Ec. 4.10

Con esta formulación, la solución al problema de estimación de mínimos cuadrados (MC) viene dada por la fórmula:

$$E = \sum_{i=1}^n \varepsilon_i = \sum_{i=1}^n [g(\vec{X}^i) - y_i]^2 \quad \text{Ec. 4.11}$$

$$\vec{a} = [H^t H]^{-1} H^t \vec{y} \quad \text{Ec. 4.12}$$

Puede observarse que se mantiene la dimensionalidad:

- \vec{a} : vector de coeficientes de tamaño 1×1 .
- \vec{y} : vector de datos de salida de tamaño $n \times 1$.
- H : matriz de datos de entrada de tamaño $n \times 1$.

Por tanto, la dimensión de salida es $(1 \times n) \times (n \times 1) \times (1 \times n) \times (n \times 1) = 1 \times 1$.

Esta solución es óptima cuando puede suponerse que los errores ε_i son independientes y de distribución gaussiana, y que no existe error en las variables \vec{X}^i (o que es despreciable frente al error en y_i).

4.1.1.3 Regresión lineal ponderada localmente

Una generalización de la regresión lineal es la regresión lineal ponderada localmente (*locally weighted linear regression*). Con este método se generan modelos locales durante el proceso de predicción dando diferente peso a los ejemplares de entrenamiento en función de algún criterio determinado. Una primera posibilidad teórica sería aquella en la que conocemos la varianza de error de cada dato de entrenamiento.

Esta extensión permite ponderar los coeficientes lineales teniendo en cuenta la precisión relativa de las variables de entrada. Éste es el modelo WLS (*Weighted Least Squares*), que obtiene los coeficientes de mínimo error ponderado.

Error ponderado de predicción:

$$\text{Error} = \sum_{j=1}^n w_i (\hat{y}_j - y_j)^2 = \sum_{j=1}^n w_i (\varepsilon_j)^2 \quad \text{Ec. 4.13}$$

Si ya no puede cumplirse la hipótesis de que los datos y_i tengan varianza constante s_y , pero tenemos información de cuál sería la varianza de cada dato, s_{y_i} , en ese caso, el estimador óptimo de mínima varianza sería el que aplica un peso inversamente proporcional a la varianza de cada dato:

$$w_i = \frac{1}{\sigma_{y_i}^2}$$

De este modo, la importancia de cada dato en el cálculo del error es inversamente proporcional a la varianza de su error.

Por tanto, se puede formular el problema aplicando una matriz diagonal de pesos a los datos de la variable dependiente:

$$Error = (y - \hat{y})^t W (y - \hat{y}) \quad \text{Ec. 4.14}$$

Empleando el planteamiento anterior, llegamos a la ecuación:

$$W \vec{y} = W H \vec{A}$$

$$y = \begin{bmatrix} y^1 \\ \vdots \\ y^n \end{bmatrix}; \quad W = \begin{bmatrix} \frac{1}{\sigma_{y1}^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_{y2}^2} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma_{yn}^2} \end{bmatrix} \quad \text{Ec. 4.15}$$

Y, en este caso, el estimador óptimo de mínimo error ponderado es la expresión:

$$\bar{a} = [H^t W H]^{-1} H^t W \vec{y} \quad \text{Ec. 4.16}$$

Que constituye el estimador LWS (*Least Weighted Squares*). Cuando no se tiene el conocimiento de la varianza de cada dato de entrada, hay que decidir el peso de cada dato, una posibilidad es pesar más a los más cercanos al que hay que predecir. Dicho de otro modo, la construcción del algoritmo de predicción consiste en el almacenamiento de los ejemplos de entrenamiento, mientras que el proceso de validación o de clasificación de un ejemplo de test consiste en la generación de una regresión lineal específica, esto es, una regresión lineal en la que se da más peso a aquellos ejemplos de entrenamiento cercanos al ejemplo a clasificar. De esta forma, este tipo de regresión está íntimamente relacionado con los algoritmos basados en ejemplares. Para utilizar este tipo de regresión es necesario decidir un esquema de ponderación para los ejemplos de entrenamiento, esto es, decidir cuánto peso se le va a dar a cada ejemplo de entrenamiento para la clasificación de un ejemplo de test. Una medida usual es ponderar el ejemplo de entrenamiento con la inversa de la distancia euclídea entre dicho ejemplo y el de test, tal y como se muestra en la ecuación 4.17.

$$\omega_i = \frac{1}{1 + d_{ij}} \quad \text{Ec. 4.17}$$

En esta ecuación, ω_i es el peso que se le otorgará al ejemplo de entrenamiento i para predecir el valor del ejemplo j , y d_{ij} será la distancia euclídea de i con respecto a j .

Más crítico que la elección del método para ponderar es el “parámetro de suavizado” que se utilizará para escalar la función de distancia, esto es, la distancia será multiplicada por la inversa de este parámetro. Si este parámetro es muy pequeño, sólo los ejemplos muy cercanos recibirán un gran peso, mientras que si es demasiado grande los ejemplos muy lejanos podrían tener peso. Un modo de asignar un valor a este parámetro es dándole el valor de la distancia del k -ésimo vecino más cercano al ejemplo a predecir. El valor de k dependerá del ruido de los datos. Cuanto más ruido, más grande deberá ser k . Una ventaja de este método de estimación consiste en que es capaz de aproximar funciones no lineales. Además, se puede actualizar el clasificador (modelo incremental), dado que únicamente sería necesario añadirlo al conjunto de entrenamiento. Sin embargo, como el resto de algoritmos basados en ejemplares, es lento.

4.1.1.4 Atributos nominales

Cuando se dispone de atributos nominales, a veces conviene transformarlos para utilizarlos en la regresión, debido a que pueden contener información útil para construir la aproximación. Por ejemplo, el día de la semana, mes del año, etc. en algunos casos puede ser relevante.

Al incorporar atributos nominales con valores discretos, ahora cada atributo nominal A_i tendría n_i valores posibles:

- $A_k = \{V_1, \dots, V_{n_k}\}$: atributo con n_k valores posibles.

Una primera estrategia puede ser construir cada atributo nominal con k valores en k atributos binarios de valor $\{0,1\}$, e incorporar estas variables binarias en la regresión como columnas adicionales de las variables de entrada.

Por ejemplo, en el caso del mes, la transformación resultaría en doce atributos binarios para los valores posibles: (MES=ENERO?, MES=FEBRERO?..., MES=DICIEMBRE?), cada uno tomando valores binarios. De esta manera, su incorporación en el modelo de regresión incorporaría estas nuevas doce variables y el resultado calcularía los pesos más altos para aquellos valores más influyentes en la variable dependiente.

Otra alternativa un poco más elaborada para tratar datos nominales consiste en organizarlos en función de su relación con la variable objetivo de la predicción, también a través de atributos binarios. Para ello, en primer lugar, se calcula el promedio de la clase en los ejemplos de entrenamiento para cada posible valor del atributo nominal, y se ordenan dichos valores de acuerdo a este promedio. Cada atributo nominal con

k posibles valores se transforma en $k-1$ atributos binarios. El i -ésimo atributo binario tendrá, para un ejemplo dado, un 0 si el valor del atributo nominal es uno de los primeros i valores del orden establecido y un 1 en caso contrario. Con este proceso se logra tratar los atributos nominales como numéricos.

4.1.2 Evaluación del modelo de regresión

En la evaluación del modelo de regresión caben dos enfoques. El más directo es cuantificar el error de predicción y el factor de correlación entre la variable estimada y los datos, midiendo su “parecido”. Un enfoque más sistemático consiste en validar estadísticamente la significatividad del modelo construido a través del análisis de la varianza de los errores.

Para determinar si el modelo de regresión lineal múltiple está bien ajustado, el método más directo es el mismo que en el caso de la regresión lineal simple: el coeficiente de correlación. La evaluación del modelo realiza el análisis de validez del modelo asumido, es decir, se van a calcular una serie de medidas de “parecido” entre la variable de salida estimada mediante la función y los valores de la variable de salida real, y de esta manera analizaremos la influencia de las variables de entrada en el cálculo de la variable de salida (si existe o no una relación lineal entre las variables de entrada que permita determinar la variable de salida). Estas medidas son: el factor de correlación (que muestra si existe la relación lineal), el error de predicción (diferencia entre la predicha y la real) y el error en coeficientes.

El factor de correlación entre predicciones y variables se evalúa como:

$$\begin{aligned} \text{Corr}(\hat{y}, y) &= \frac{1}{\sqrt{S_{\hat{y}}S_y}} \sum_{j=1}^n (\hat{y}_j - \bar{\hat{y}})(y_j - \bar{y}) = \frac{\text{Cov}(\hat{y}, y)}{\sqrt{\text{Var}(\hat{y})\text{Var}(y)}} \\ S_{\hat{y}} &= \sum_{j=1}^n (\hat{y}_j - \bar{\hat{y}})^2; \quad S_y = \sum_{j=1}^n (y_j - \bar{y})^2; \\ \bar{\hat{y}} &= \frac{1}{N} \sum_{j=1}^N \hat{y}_j, \quad \bar{y} = \frac{1}{N} \sum_{j=1}^N y_j \end{aligned} \quad \text{Ec. 4.18}$$

La validación mediante la correlación existente entre las variables de entrada con la salida se realizará si este valor $\text{Corr}(\hat{y}, y)$ está próximo a 1.

En cuanto al error de predicción, se evalúa como el error cuadrático (MSE , *Mean Squared Error*):

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \frac{1}{n} \sum_{i=1}^n (\varepsilon_i)^2 ; \quad RMSE = \sqrt{MSE} \quad \text{Ec. 4.19}$$

A veces es más útil determinar el error relativo, normalizado por la propia varianza de los datos:

$$RSE: \frac{(\hat{y}_1 - y_1)^2 + \dots + (\hat{y}_n - y_n)^2}{(y_1 - \bar{y})^2 + \dots + (y_n - \bar{y})^2}; \quad \bar{y} = \frac{y_1 + \dots + y_n}{n};$$

$$RRSE = \sqrt{RSE} \quad \text{Ec. 4.20}$$

Bajo la hipótesis de que los datos y_i tengan una varianza constante, σ_y^2 , sean independientes y que el modelo lineal sea adecuado, el error relativo es:

$$RSE: \frac{(\hat{y}_1 - y_1)^2 + \dots + (\hat{y}_n - y_n)^2}{(y_1 - \bar{y})^2 + \dots + (y_n - \bar{y})^2} = \frac{MSE}{(n-1)\sigma_y^2} \quad \text{Ec. 4.21}$$

En cuanto a la validación del modelo mediante análisis de varianza, se puede aplicar tanto para validar los parámetros como la relación completa:

- Hipótesis de significatividad de parámetros: distribución t-Student.
- Hipótesis de significatividad de relación total: F de Fisher-Snedecor (análisis de varianza particular).

Para analizar la validez del modelo, un procedimiento general consiste en llevar a cabo un análisis de significatividad. Se puede analizar la varianza, lo que permite rechazar o no la hipótesis de que no existe relación entre variables (relación debida al azar, correlación nula). Para ello, a partir del valor:

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad \text{Ec. 4.22}$$

Puede plantearse un análisis de varianza particularizado a la relación entre las variables y los estimadores como se indica a continuación, siendo n el número de observaciones e l el número de parámetros de estimación:

- **Suma de cuadrados del modelo:**

$$SSM = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 \quad \text{Ec. 4.23}$$

- **Suma de cuadrados residual** (también llamada *suma de cuadrados de residuos*):

$$SSR = \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad \text{Ec. 4.24}$$

- **Suma de cuadrados total:**

$$TSE = nMSE = SSM + SSR = \sum_{i=1}^n (y_i - \bar{y})^2 \approx (n-1)\sigma_y^2 \pi \quad \text{Ec. 4.25}$$

Este término es la varianza muestral multiplicada por $n-1$.

- **Grados de libertad del modelo:** DFM = 1
- **Grados de libertad del residuo:** DFR = $n-1-1$
- **Grados de libertad total:** DFT = $n-1$

Como en el análisis de varianza general, se pretende que la variación debida al modelo (SM) sea significativamente superior a la variación residual (SE). Por tanto, si calculamos el estadístico,

$$F = \frac{\frac{SSM}{DFM}}{\frac{SSR}{DFR}} \quad \text{Ec. 4.26}$$

en el caso de hipótesis nula (ninguna relación con el modelo, no diferencia de una aproximación por el término medio), se sigue una distribución: F de Snedecor, $F(n_1, n_2)$, donde los grados de libertad son: 1, $n-1-1$.

4.1.2.1 Error de regresión y selección de variables

Además del proceso anterior para la generación de la regresión lineal y validación del modelo, se puede realizar un procedimiento adicional que permite validar las variables predictoras, ya que no todas tienen la misma importancia, y al reducir su número hará que computacionalmente mejore el tiempo de respuesta del modelo.

El planteamiento más directo es un test de significatividad a partir del error propagado hacia los coeficientes. El error en coeficientes se evalúa a partir de la expresión anterior:

$$\bar{a} = [H^t H]^{-1} H^t \bar{y}; \quad \bar{\varepsilon}_{\bar{a}} = [H^t H]^{-1} H^t \bar{\varepsilon}_{\bar{y}} \quad \text{Ec. 4.27}$$

La relación entre los errores en predicción y en coeficientes estimados se evalúa:

$$C_{\bar{a}} = \begin{bmatrix} \sigma_{a_0}^2 & \dots & \dots \\ \vdots & \sigma_{a_1}^2 & \vdots \\ \vdots & \dots & \ddots \\ \vdots & \dots & \dots & \sigma_{a_I}^2 \end{bmatrix} = \sigma_y^2 [H^t H]^{-1} \quad \text{Ec. 4.28}$$

De modo que el error en los coeficientes depende del error en la variable dependiente y , con varianza σ_y^2 , y del recorrido de datos independientes X , es decir, de la matriz H .

(Esta aproximación es válida siempre que se mantenga la hipótesis de varianza constante en las observaciones de la variable dependiente y , y que el error en las variables predictoras sea despreciable frente al error en y , de forma que se pueda tratar H como una matriz constante).

Ejemplos:

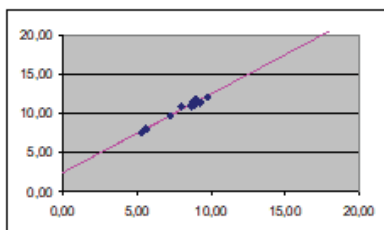
x	y
5,33	8,15
5,65	7,84
7,27	9,33
8,05	10,07
8,66	11,60
8,80	11,48
8,89	11,89
8,98	11,12
9,35	12,01
9,82	12,01

Rango: [5,10]

$\sigma_y=1$

$\sigma_{A0}=0.6$

$\sigma_{A1}=0.07$



x	y
1,32	3,67
1,68	4,66
4,69	7,57
4,99	7,48
6,98	9,66
8,80	11,51
10,01	12,02
15,01	17,47
17,10	19,82
19,67	21,94

Rango: [0,20]

$\sigma_y=1$

$\sigma_{A0}=0.16$

$\sigma_{A1}=0.02$

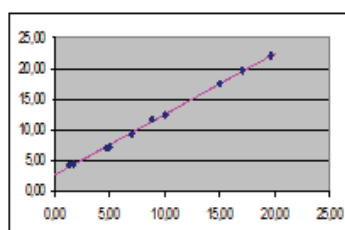


Fig. 4.2. Efecto de la dispersión de los datos en la precisión del estimador LS.

Puede verse en este ejemplo como la precisión de los coeficientes aumenta drásticamente al incrementar el rango de los datos, confirmando lo que intuitivamente puede inferirse a partir de las figuras.

Otro uso directo del conocimiento del error en los coeficientes es el cálculo del error de las propias predicciones. Efectivamente, el modelo de predicción para el i -ésimo dato es:

$$\hat{y}_i = (\bar{X}^i)^t \bar{a} \quad \text{Ec. 4.29}$$

El error de predicción puede obtenerse directamente a partir del error en los coeficientes:

$$\Delta \hat{y}_i = (\bar{X}^i)^t \Delta \bar{a} \quad \text{Ec. 4.30}$$

Por tanto, puede obtenerse la varianza de esta predicción directamente a partir de la expresión anterior:

$$\sigma_{\hat{y}_i}^2 = \sigma_y^2 (\bar{X}^i)^t [H^t H]^{-1} (\bar{X}^i) \quad \text{Ec. 4.31}$$

De esa forma, el error de la predicción depende de cada dato X^i . Este resultado es lógico, porque la hipótesis de partida de un error uniforme en las observaciones de los datos (varianza constante, σ_y^2) no implica que el error de predicción sea constante, sino que depende de cada dato predicho. Intuitivamente puede verse en el caso de una sola dimensión (recta en 2D): al aproximar con un tramo recto el conjunto de datos, el error de predicción será máximo en los extremos y mínimo en el punto central de los datos de entrada. A modo de ejemplo, en la siguiente figura se muestra una simulación de estimación mediante regresión obtenida con 500 puntos generando ruido gaussiano con varianza 100:

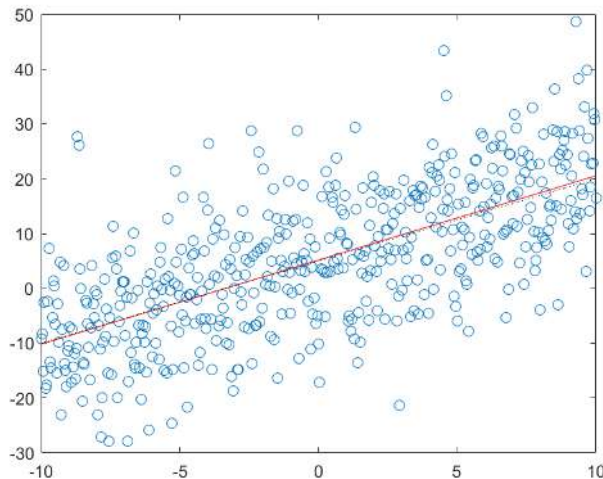


Fig. 4.3. Ejemplo de regresión lineal con una variable.

El error de estimación se muestra en la figura siguiente, y el valor de error (1 sigma) obtenido con la expresión anterior se muestra a continuación:

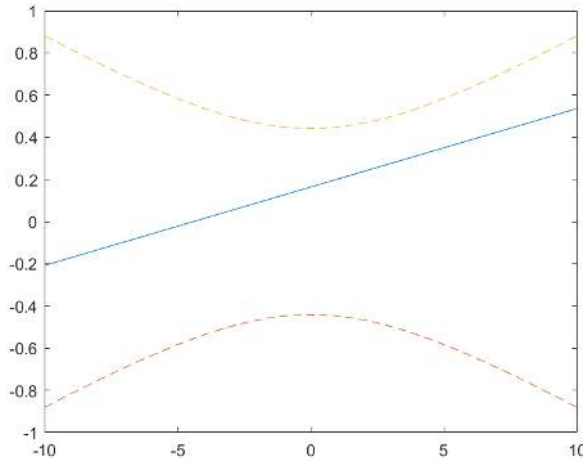


Fig. 4.4. Error de estimación de regresión lineal con una variable (error simulado y 2 sigma).

Por otro lado, los valores de varianza de los coeficientes permiten analizar la “calidad” del modelo mediante los test de hipótesis: las hipótesis de significatividad de parámetros (gaussiana o t-Student) y la hipótesis de ausencia de relación (F de Fisher-Snedecor).

Para evaluar la significatividad de parámetros, partimos de varianzas de parámetros $\{\sigma_{a0}, \sigma_{a1}, \dots, \sigma_{aI}\}$ y de los propios valores estimados, y nos preguntamos si son significativos los parámetros $\frac{a_0}{\sigma_{a0}}, \frac{a_1}{\sigma_{a1}}, \dots, \frac{a_I}{\sigma_{aI}}$. Este test puede resolverse mediante una gaussiana estándar si tenemos gran cantidad de datos, o bien si hay pocos datos: en vez de estadística normal, una t-Student con n-F-1 grados de libertad. También podemos extender el modelo y analizarlo. Ejemplo: dependencia cuadrática, ver si son significativos nuevos términos.

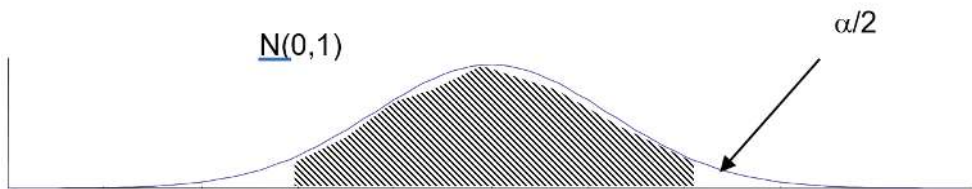


Fig. 4.5. Significatividad de parámetros mediante test normal.

En general, se pueden abordar otros procesos para la selección de variables predictoras, son básicamente dos: **eliminación hacia atrás** (*backward elimination*), consistente en obtener la regresión lineal para todos los parámetros e ir eliminando uno a uno los me-

nos importantes; y **selección hacia delante** (*forward selection*), que consiste en generar una regresión lineal simple (con el mejor parámetro, esto es, el más correlacionado con la variable a predecir) e ir añadiendo parámetros al modelo.

Hay un gran número de estadísticos que permiten seleccionar los parámetros, y a modo de ejemplo se comentará el basado en el **criterio de información Akaike** [AKA73], que se basa en la teoría de la información y cuya formulación se muestra en la ecuación 2.21.

$$AIC = -2 \times \log(L) + 2p \quad \text{Ec. 4.32}$$

En esta ecuación, L es la **verosimilitud** (*likelihood*) y p el número de variables predictorias. Aplicado a la regresión, el resultado sería el que se muestra en las ecuaciones 2.22 y 2.23.

$$AIC = m \times \log(MSE) + 2p \quad \text{Ec. 4.33}$$

$$MSE = \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{m} \quad \text{Ec. 4.34}$$

En la ecuación 4.34, m es el número de ejemplos disponibles, y MSE es el error cuadrático medio del modelo. En esta ecuación, y_i es el valor de la clase para el ejemplo i , e \hat{y}_i el valor que la regresión lineal da al ejemplo i . En la práctica, algunas herramientas no utilizan exactamente las ecuaciones anteriores, sino aproximaciones de dichas ecuaciones.

4.1.3 Regresión no lineal

En muchas ocasiones, los datos no muestran una dependencia lineal [FRI91]. Esto es lo que sucede si, por ejemplo, la variable “respuesta” depende de las variables independientes según una función polinómica, dando lugar a una regresión polinómica que puede planearse agregando las condiciones polinómicas al modelo lineal básico. De esta forma y aplicando ciertas transformaciones a las variables, se puede convertir el modelo no lineal en uno lineal que puede resolverse entonces por el método de mínimos cuadrados. Por ejemplo, considérese una relación polinómica cúbica dada por:

$$y = a + b_1x + b_2x^2 + b_3x^3. \quad \text{Ec. 4.35}$$

Para convertir esta ecuación a la forma lineal, se definen las nuevas variables:

$$x_1 = x \quad x_2 = x^2 \quad x_3 = x^3 \quad \text{Ec. 4.36}$$

De forma que la ecuación anterior puede convertirse entonces a la forma lineal aplicando los cambios de variables, y resultando la ecuación 4.37, que es resoluble por el método de mínimos cuadrados.

$$y = a + b_1 x_1 + b_2 x_2 + b_3 x_3 \quad \text{Ec. 4.37}$$

No obstante, algunos modelos son especialmente no lineales como, por ejemplo, la suma de términos exponenciales y no pueden convertirse a un modelo lineal. Para estos casos, puede ser posible obtener las estimaciones del mínimo cuadrado a través de cálculos extensos en fórmulas más complejas.

Los modelos lineales generalizados representan el fundamento teórico en que la regresión lineal puede aplicarse para modelar las categorías de las variables dependientes. En los modelos lineales generalizados, la variación de la variable y es una función del valor medio de y , distinto a la regresión lineal, donde la variación de y es constante. Los tipos comunes de modelos lineales generalizados incluyen regresión logística y regresión de Poisson. La regresión logística modela la probabilidad de algún evento que ocurre como una función lineal de un conjunto de variables independientes. Frecuentemente los datos exhiben una distribución de Poisson y se modelan normalmente usando la regresión de Poisson.

Los modelos lineales logarítmicos [PEA88] aproximan las distribuciones de probabilidad multidimensionales discretas, y pueden usarse para estimar el valor de probabilidad asociado con los datos de las células cúbicas. Por ejemplo, supongamos que se tienen los datos para los atributos ciudad, artículo, año y ventas. En el método logarítmico lineal, todos los atributos deben ser categorías; de modo que los atributos estimados continuos (como las ventas) deben ser previamente discretizados.

4.1.3.1 Transformaciones sencillas

Los modelos lineales generalizados representan el fundamento teórico en que la regresión lineal puede aplicarse para modelar las categorías de las variables dependientes. En los modelos lineales generalizados, la variación de la variable y es una función del valor medio de y , distinto a la regresión lineal donde la variación de y es constante.

Las extensiones a la regresión lineal constituyen los llamados *modelos lineales generalizados* (GLM), que son de dos tipos básicos:

- Relación no lineal entre variables numéricas.
- Transformaciones de las variables independientes a un nuevo espacio y construcción de un modelo lineal sobre las nuevas variables transformadas.

Los tipos comunes de modelos lineales generalizados (GLM) incluyen regresión polinómica, logarítmica, exponencial y potencial. Además, otros métodos generalizados son la regresión logística y regresión de Poisson, como ya hemos comentado más arriba.

En muchas ocasiones los datos no muestran una dependencia lineal [FRI91]. Esto es lo que sucede si, por ejemplo, la variable “respuesta” depende de las variables independientes según una función polinómica, logarítmica o exponencial. En este caso es posible aplicar la regresión agregando los términos polinómicos al modelo lineal básico. De esta forma y aplicando ciertas transformaciones a las variables, se puede convertir el modelo no lineal en uno lineal que puede resolverse entonces por el método de mínimos cuadrados, como la transformación que se indicó en las ecuaciones 4.35-4.37.

Este procedimiento puede aplicarse en general a otras relaciones no lineales como, por ejemplo, la suma de términos exponenciales. A continuación se muestran tres familias de regresión no lineal muy frecuentes:

- **Regresión exponencial:**

$$\text{Log}(y) = a_0 + a_1 x_1 + \dots + a_I x_I \quad \text{Ec. 4.38}$$

$$y = \exp(a_0 + a_1 x_1 + \dots + a_I x_I)$$

Por tanto, puede aplicarse la metodología general reemplazando los valores de y_i por $\log(y_i)$.

- **Regresión logarítmica:**

$$y = a_0 + a_1 \log x_1 + a_2 \log x_2 \dots + a_I \log x_I \quad \text{Ec. 4.39}$$

Por tanto, puede aplicarse la metodología general reemplazando los valores de x_i por $\log(x_i)$.

- **Regresión potencial:**

$$\log y = a_0 + a_1 \log x_1 + a_2 \log x_2 + \dots + a_I \log x_I \quad \text{Ec. 4.40}$$

Que utilizando propiedades de logaritmos puede transformarse en:

$$y = c_0 (x_1)^{c_1} (x_2)^{c_2} \dots (x_I)^{c_I} \quad \text{Ec. 4.41}$$

Por tanto, puede aplicarse la metodología general reemplazando los valores de x_j^i e y_i por $\log(x_j^i)$, $\log(y_i)$.

4.1.3.2 Otras transformaciones

La regresión logística se utiliza para aproximar la probabilidad de pertenencia de los datos a las clases, por tanto, es un problema de clasificación que se detalla en el capítulo siguiente, y la regresión de Poisson parte de un modelo de variable que representa número de eventos a partir de un parámetro que representa la frecuencia de ocurrencia media, permitiendo estimar la probabilidad de que ocurra un determinado número de eventos durante cierto periodo de tiempo. Este modelo de regresión utiliza la regresión lineal para aproximar la media de esta distribución, es decir:

$$E(Y|x) = \text{Log}(a_0 + a_1 x_1 + \dots + a_n x_n) \quad \text{Ec. 4.42}$$

De este modo, el parámetro de la media de la distribución sigue una relación exponencial y puede aplicarse el mismo procedimiento de transformar los datos sustituyendo los valores de x_j^i por $\log(x_j^i)$.

Por tanto tenemos que la regresión logística, empleada para resolver problemas de clasificación, sería un caso particular de model lineal logarítmico [PEA88].

4.1.4 Ejemplos de regresión lineal

Ejemplo: regresión lineal de una variable.

Año	Renta	Consumo	consumo E
1970	1959,75	1751,87	1683,473374
1971	2239,09	1986,35	1942,43325
1972	2623,84	2327,9	2299,11261
1973	3176,06	2600,1	2811,043671
1974	3921,6	3550,7	3502,190468
1975	4624,7	4101,7	4153,993607
1976	5566,02	5012,6	5026,63666
1977	6977,84	6360,2	6335,452914
1978	8542,51	7990,13	7785,967518
1979	9949,9	9053,5	9090,676976
1980	11447,5	10695,4	10479,01488
1981	13123,04	12093,8	12032,31062
1982	15069,5	12906,27	13836,76054
1983	16801,6	15720,1	15442,48976
1984	18523,5	17309,7	17038,76316

Estimación Lineal	
a1	a0
0,927041871	-133,296932

$$\text{ConsumoE} = a0 + a1 * \text{Renta}$$

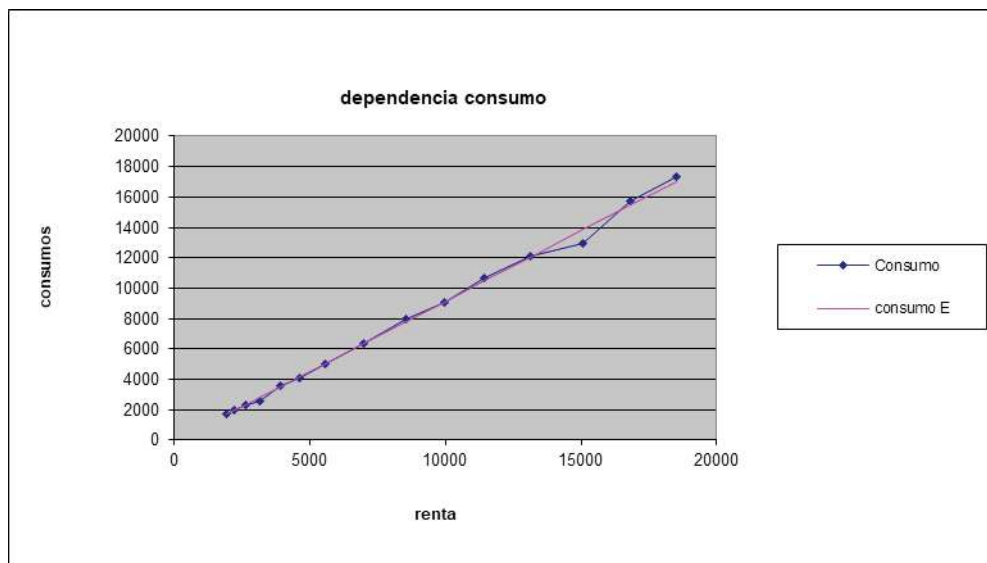


Fig. 4.6. Ejemplo de regresión lineal con una variable.

Ejemplo: regresión lineal de dos variables.

x1	x2	y	Valor
Superficie	Antigüedad	Valor	predicho
310	20	106.287 Euros	109.180 Euros
333	12	107.784 Euros	112.283 Euros
356	33	113.024 Euros	108.993 Euros
379	43	112.275 Euros	108.128 Euros
402	53	104.042 Euros	107.262 Euros
425	23	126.497 Euros	115.215 Euros
448	99	94.311 Euros	99.800 Euros
471	34	106.961 Euros	115.469 Euros
494	23	122.006 Euros	119.233 Euros
517	55	126.497 Euros	113.518 Euros
540	22	111.527 Euros	122.132 Euros

Estimación Lineal		
a2	a1	a0
-220,444829	58,2271936	95538,7217

$$\text{Valor} = a_0 + a_1 * \text{Superficie} + a_2 * \text{Antigüedad}$$

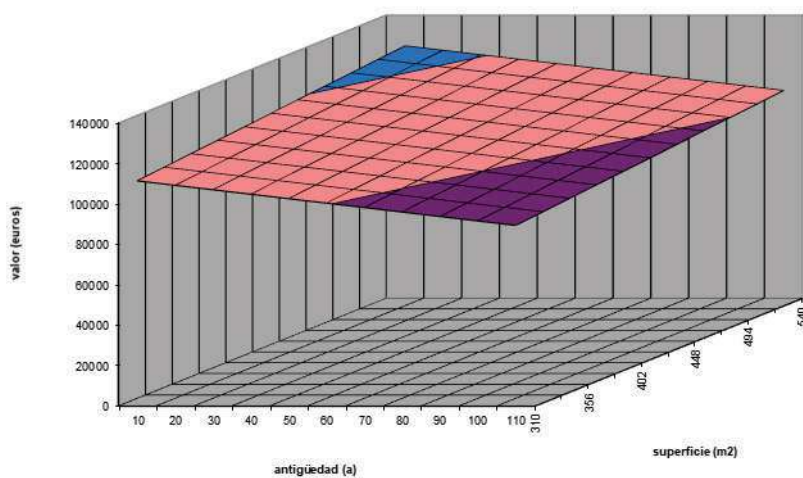
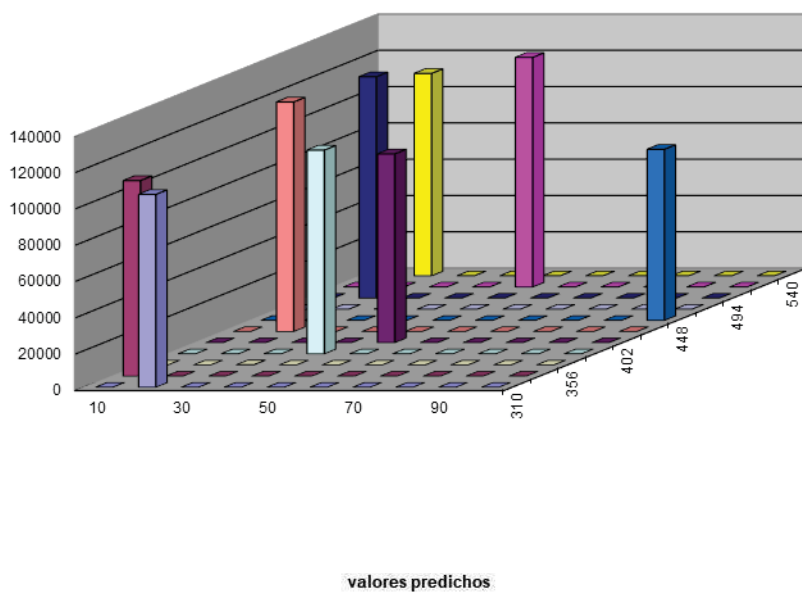


Fig. 4.7. Ejemplo de regresión lineal con dos variables.

4.2 Técnicas numéricas de clasificación

Uno de los métodos más directos de construcción de clasificadores está basado en la aplicación de las técnicas de predicción para construir modelos de clasificación. El modelado de datos con atributos numéricos para su aplicación a clasificación puede representarse del modo siguiente:

- Clases: $\{C_1, \dots, C_M\}$.
- Para cada clase, C_i , hay n_i patrones, cada uno con i atributos.

$$\{\vec{X}_1^{(i)}, \dots, \vec{X}_{n_i}^{(i)}\}$$

$$\vec{X}_j^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_i^{(i)} \end{bmatrix}; \quad j = 1, \dots, n_i$$

- Muestras totales: $E = \{\vec{X}_1^{(1)}, \dots, \vec{X}_{n_1}^{(1)}, \vec{X}_1^{(2)}, \dots, \vec{X}_{n_2}^{(2)}, \vec{X}_1^{(M)}, \dots, \vec{X}_{n_M}^{(M)}\}$
- Tamaño total: $n = \sum_{j=1}^M n_j$

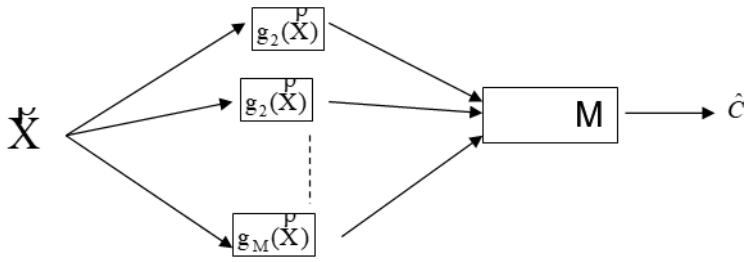
$$n = \sum_{j=1}^M n_j$$

Por tanto, el procedimiento consiste en construir la función discriminante de cada clase:

$$g(.): R^I \rightarrow C = \{C_1, \dots, C_M\}$$

$$\vec{X} \rightarrow \hat{C} = g(\vec{X})$$

Ec. 4.43



La propiedad deseable para el diseño de cada función discriminante $g_i(\cdot)$, dado el conjunto de entrenamiento, consiste en hacer que cada patrón de su clase C_i tenga un valor máximo con respecto al conjunto de discriminantes $g_k(\cdot)$:

$$g_i(\vec{X}_j^{(i)}) = \max_{k=1, \dots, M} \{g_k(\vec{X}_j^{(i)})\}, \forall j = 1, \dots, n_i \quad \text{Ec. 4.44}$$

La obtención de esta función discriminante es equivalente al cálculo de las fronteras de decisión, que determinan los límites de cada clase. A continuación, se muestran ejemplos de fronteras en un problema de clasificación con dos atributos numéricos:

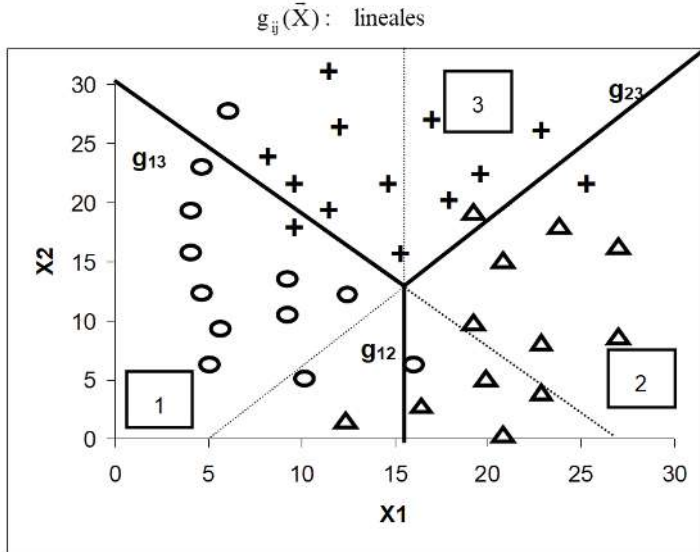


Fig. 4.8. Fronteras de decisión lineales.

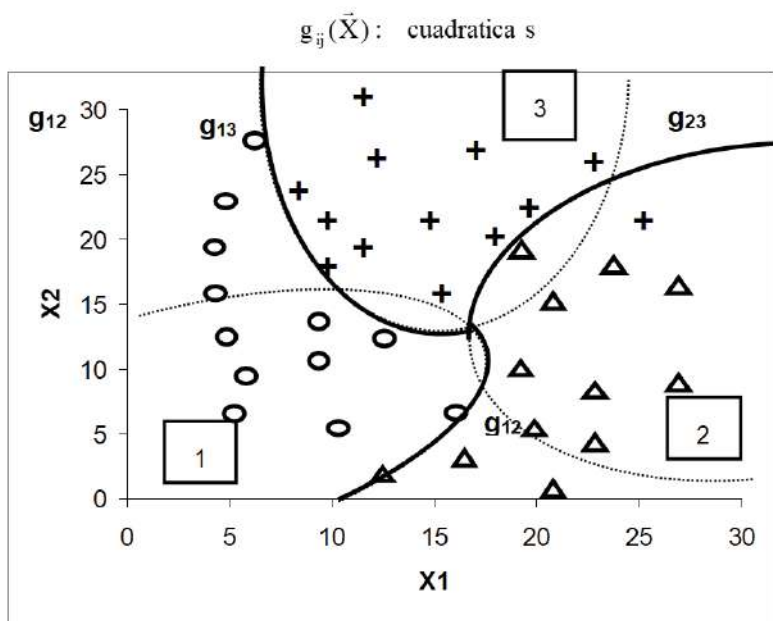


Fig. 4.9. Fronteras de decisión cuadráticas.

4.2.1 Clasificación mediante regresión lineal

El método más directo es la formulación de una función que asigne 1 a los datos de la clase y 0 al resto, con dos variantes posibles:

- **Clasificación multiclase**, donde se obtiene la frontera de decisión de cada clase con el resto.
- **Clasificación por pares**, donde se genera una frontera de decisión para cada par.

Clasificación con regresión lineal: 1

- Para cada clase se define la función de pertenencia g_i :

$$g_i(\vec{X}) = \begin{cases} 1; & \vec{X} \in C_i \\ 0; & \vec{X} \notin C_i \end{cases} \quad \text{Ec. 4.45}$$

- Se construye una función lineal que “aproxime” g_i :

$$\vec{y}_i = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \quad H_i = \begin{bmatrix} 1 & (\vec{X}_1^{(i)})^t \\ \vdots & \vdots \\ 1 & (\vec{X}_{n_i}^{(i)})^t \\ 1 & (\vec{X}_1^{(1)})^t \\ \vdots & \vdots \\ 1 & (\vec{X}_{n_l}^{(l)})^t \end{bmatrix}; \quad \vec{A}_i = [H_i^t H_i]^{-1} H_i^t \vec{y}_i \quad \text{Ec. 4.46}$$

- Hay que “aprender” M funciones g_i .

Alternativamente, otra opción sería calcular la regresión que permita separar cada par de clases, función frontera g_{ij} :

$$g_{ij}(\vec{X}) = \begin{cases} +1; & \vec{X} \in C_i \\ -1; & \vec{X} \in C_j \end{cases} \quad \text{Ec. 4.47}$$

- Funciones lineales para todos los pares:

$$\vec{y}_{ij} = \begin{bmatrix} +1 \\ \vdots \\ +1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}; \quad H_{ij} = \begin{bmatrix} 1 & (\vec{X}_1^{(i)})^t \\ \vdots & \vdots \\ 1 & (\vec{X}_{n_i}^{(i)})^t \\ 1 & (\vec{X}_1^{(j)})^t \\ \vdots & \vdots \\ 1 & (\vec{X}_{n_j}^{(j)})^t \end{bmatrix}; \quad \vec{A}_{ij} = [H_{ij}^t H_{ij}]^{-1} H_{ij}^t \vec{y}_{ij} \quad \text{Ec. 4.48}$$

En este caso habría que calcular las $M(M-1)/2$ fronteras posibles para todos los pares g_{ij} .

4.2.2 Clasificación mediante regresión logística

El principal problema del método anterior de clasificación mediante regresión directa es la poca discriminación al asignar a todos los datos el mismo peso (1 si son de la clase, 0 si no), y además no se puede imponer la restricción a la regresión, de modo que ocasiona errores arbitrariamente grandes con respecto a los valores $\{0,1\}$, lo que pueden distorsionar el cálculo de la frontera de decisión.

Una técnica más apropiada para resolver este problema es la regresión logística, que puede asignar un valor arbitrariamente grande o pequeño en función de la distancia a la frontera de decisión, permitiendo mucha mayor precisión en la obtención del resultado.

Puede verse como un caso particular de regresión no lineal, en el que se hace una transformación del valor objetivo (probabilidad de pertenencia o no a la clase de cada dato), a través de la transformación logística:

$$\log \frac{\pi_i(\vec{X})}{1 - \pi_i(\vec{X})}; \quad \pi_i(\vec{X}) = \begin{cases} 1; & \vec{X} \in C_i \\ 0; & \vec{X} \notin C_i \end{cases} \quad \text{Ec. 4.49}$$

La ventaja de esta transformación es que permite la generación de valores arbitrariamente grandes en función de su separación a la frontera de decisión.

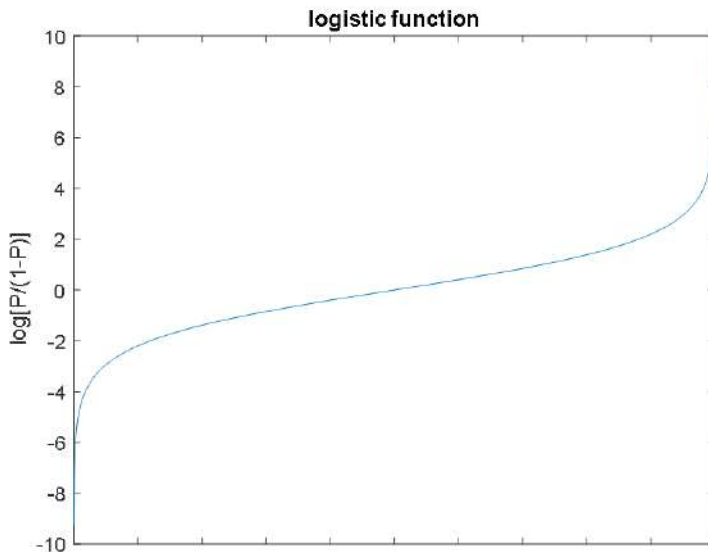


Fig. 4.10. Función logística.

De esta manera, la frontera de decisión viene dada por la función logística, quedando:

$$g_i(\vec{X}) = \log \frac{\pi_i(\vec{X})}{1 - \pi_i(\vec{X})} = a_0 + a_1 x_1 + \dots + a_I x_I \quad \text{Ec. 4.50}$$

$$\pi_i(\vec{X}) = \frac{1}{1 + e^{-(a_0 + a_1 x_1 + \dots + a_I x_I)}}$$

Se trata de un caso de regresión lineal generalizada (GLM), aplicada para construir fronteras de decisión lineales que “aproximen” la función transformada g_i . La ventaja de este método es que ahora puede ponderar la pertenencia a cada clase con una probabilidad, y la regresión se aplica sobre el cociente de probabilidades, pudiendo tener un

valor arbitrariamente alto o bajo, y, por tanto, evita el planteamiento inicial de generar un hiperplano con el mismo valor para todos los datos de la clase. Una puntualización es que no se puede aplicar “directamente” haciendo una transformación, porque tendríamos valores singulares en los datos, pues la función logística es singular para los extremos 0 y 1 de probabilidad, de modo que se utilizan métodos numéricos iterativos, destacando el Newton-Raphson para generar los coeficientes óptimos, que maximizan el parecido, *likelihood*, entre datos de entrenamiento y valores de probabilidad.

Finalmente, para la regresión logística caben las dos aproximaciones mencionadas anteriormente de clasificación multiclase y clasificación por pares.

4.2.3 Clasificación bayesiana

Este segundo método consiste en la aplicación de modelos estadísticos con una distribución conocida (gaussiana) para determinar las fronteras de decisión. En primer lugar, lo aplicaremos sobre atributos numéricos como en el caso de la regresión lineal, y posteriormente sobre su extensión para incorporar atributos simbólicos.

La clasificación bayesiana parte de un modelo de estructura probabilística conocida. Se dispone de los siguientes elementos:

- Clases $\{C_1, \dots, C_M\}$. Se conocen *a priori*, con probabilidades de clase: $P(C_i)$.
- Distribuciones de probabilidad condicionadas (parámetros constantes).

$$F_{\vec{X}}(x_1, \dots, x_I | C_i) = P(X_1 \leq x_1, \dots, X_I \leq x_I | C_i) = \frac{P(X_1 \leq x_1, \dots, X_I \leq x_I, C_i)}{P(C_i)}$$

$$\text{Densidad: } f_{\vec{X}}(x_1, \dots, x_I | C_i) = \frac{\partial F_{\vec{X}}(x_1, \dots, x_I | C_i)}{\partial x_1 \dots \partial x_I}$$

4.2.3.1 Clasificación bayesiana de atributos numéricos

Por ejemplo, lo habitual es partir de una distribución normal multivariada, que tiene como parámetros el vector de medias y la matriz covarianzas $\{\vec{\mu}, S\}$.

$$f(\vec{x}) = \frac{1}{(2\pi)^{n/2} \sqrt{|S|}} \exp \left[-\frac{1}{2} (\vec{x} - \vec{\mu})^t S^{-1} (\vec{x} - \vec{\mu}) \right]$$

$$\vec{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}; \quad S = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1 x_2} & \dots & \sigma_{x_1 x_F} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{x_F x_1} & \sigma_{x_F x_2} & \dots & \sigma_{x_F}^2 \end{bmatrix}$$

Ec. 4.51

El clasificador bayesiano aplica el teorema de Bayes para calcular la probabilidad de pertenencia a cada clase para un dato de entrada, \vec{X} , partiendo de estos elementos:

- Probabilidad *a priori*: $P(C_i)$ es la probabilidad total de cada clase.
- Verosimilitud: $f(\vec{X}|C_i)$ es la distribución de probabilidad asociada a cada clase C_i .
- El teorema de Bayes nos permite obtener el valor de la probabilidad *a posteriori*; es la probabilidad de que el patrón tenga clase C_i : $P(C_i|\vec{X})$.

$$P(C_i|\vec{X}) = \frac{f(\vec{X}|C_i)p(C_i)}{f(\vec{X})} \quad \text{Ec. 4.52}$$

Con esto, podemos generar la decisión de clasificación aplicando un criterio máximo *a posteriori* (MAP):

$$\text{Clase}(\vec{X}) = \max_i \{P(C_i|\vec{X})\} = \max_i \{f(\vec{X}|C_i)p(C_i)\} \quad \text{Ec. 4.53}$$

Por tanto, volviendo a la formulación general, la función discriminante asociada a la clase C_i sería el valor de probabilidad *a posteriori*:

$$g_i(\vec{X}) = f(\vec{X}|C_i)p(C_i) \quad \text{Ec. 4.54}$$

De esta forma, la clase generada es aquella que maximiza el discriminante $g_i(\cdot)$

Para llevar a cabo la clasificación bayesiana con distribución normal, en la fase de entrenamiento se precisa obtener las distribuciones condicionales a cada clase, para ello se trata de obtener los parámetros de distribución condicionada a cada clase $\{\vec{\mu}_i, S_i\}$, $i=1, \dots, M$.

$$g_i(\vec{x}) = \log(P(C_i)f(\vec{x}|C_i)) = \log \frac{P(C_i)}{(2\pi)^{n/2} \sigma_{1i} \sigma_{2i} \dots \sigma_{Fi}} - \frac{1}{2} ((\vec{x} - \vec{\mu}_i)^t (S_i)^{-1} (\vec{x} - \vec{\mu}_i)) \quad \text{Ec. 4.55}$$

En el caso particular de que las covarianzas sean diagonales (independencia entre las variables independientes del vector x), la función discriminante se simplifica a:

$$g_i(\vec{x}) = \log(P(C_i)f(\vec{x}|C_i)) = \log \frac{P(C_i)}{(2\pi)^{n/2} \sigma_{1i} \sigma_{2i} \dots \sigma_{Ii}} - \frac{\frac{1}{2} \sum_{j=1}^I (x_j - \mu_{ij})^2}{\sigma_{ij}^2} \quad \text{Ec. 4.56}$$

En este caso tenemos una matriz de covarianza diagonal:

$$S = \begin{bmatrix} \sigma_{i1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{i2}^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{x_{iF}}^2 \end{bmatrix} \quad \text{Ec. 4.57}$$

Regiones de decisión:

- En este caso las regiones de decisión son funciones cuadráticas (hipérbolas) dadas por diferencias:

$$g_{ij}(\vec{x}) = g_i(\vec{x}) - g_j(\vec{x})$$

A continuación se muestra un ejemplo con distribución normal:

$$P_1 = 0.3; \quad C_1 = [-30 \quad 5]^t; \quad R_1 = \begin{bmatrix} 21 & -6 \\ -6 & 21 \end{bmatrix}$$

$$P_2 = 0.2; \quad C_2 = [35 \quad 5]^t; \quad R_2 = \begin{bmatrix} 16 & 6 \\ 6 & 16 \end{bmatrix}$$

$$P_3 = 0.5; \quad C_3 = [10 \quad -10]^t; \quad R_3 = \begin{bmatrix} 16 & 2 \\ 2 & 4 \end{bmatrix}$$

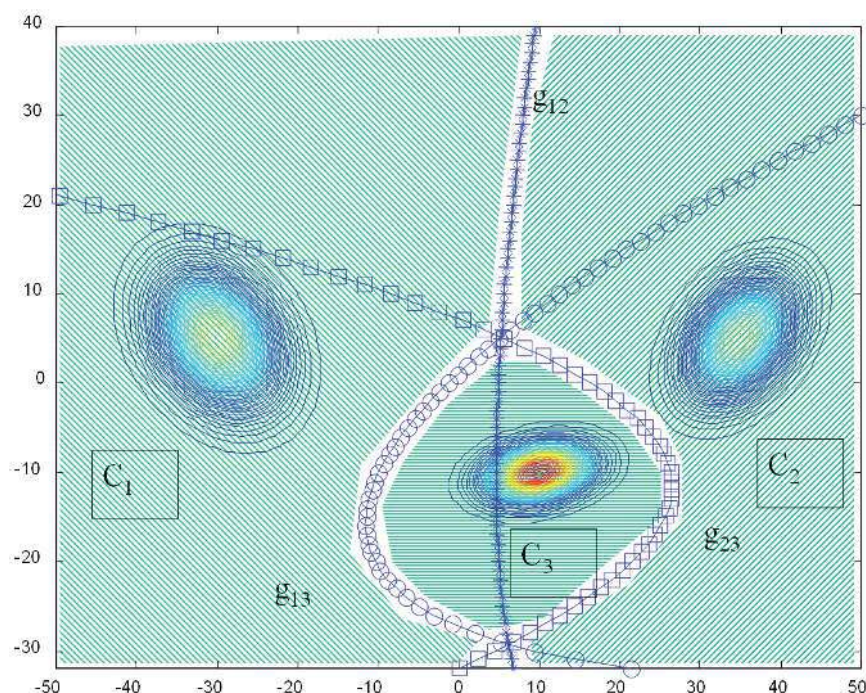


Fig. 4.11. Ejemplo de fronteras de decisión con distribución gaussiana 2D.

Puede verse que la forma general de las fronteras de decisión es hiperbólica, dada por las diferencias ponderadas por las matrices de covarianzas invertidas. El clasificador bayesiano tendrá fronteras lineales únicamente en el caso particular en el que la ecuación discriminante tenga covarianzas diagonales y del mismo valor. A continuación se muestra un ejemplo:

$$P_1 = 0.3; \quad C_1 = [-30 \quad 5]^t; \quad R_1 = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix}$$

$$P_2 = 0.2; \quad C_2 = [35 \quad 5]^t; \quad R_2 = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix}$$

$$P_3 = 0.5; \quad C_3 = [10 \quad -10]^t; \quad R_3 = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix}$$

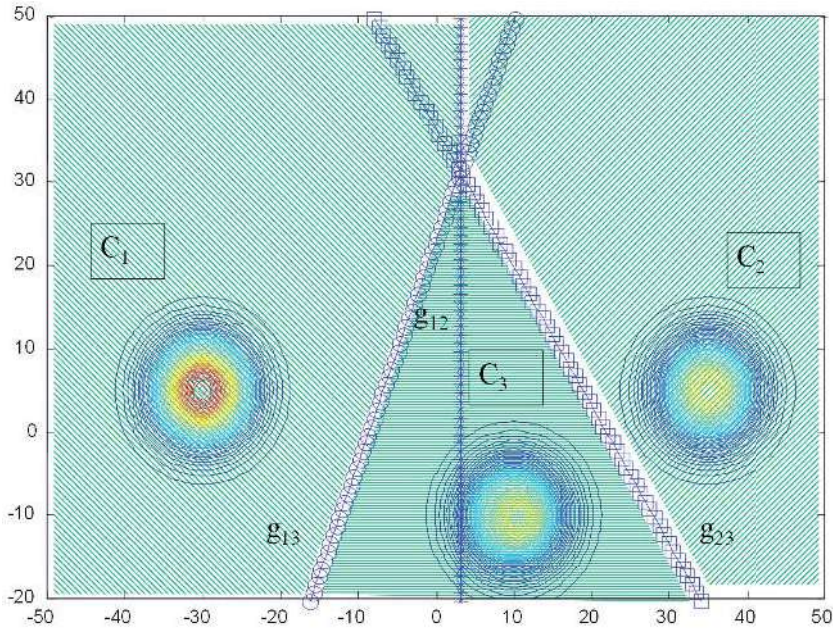


Fig. 4.12. Ejemplo de fronteras de decisión con distribución gaussiana de covarianza circular.

Resumen clasificador bayesiano numérico. El clasificador bayesiano puede resumirse en el algoritmo siguiente:

1. Estimar parámetros de cada clase C_i (entrenamiento).

$$C_i: \{\vec{x}_1^{(i)}, \dots, \vec{x}_{n_i}^{(i)}\} \rightarrow \vec{\mu}_i, \quad C_i \quad \text{Ec. 4.58}$$

$$\vec{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \vec{x}_j^{(i)}$$

$$C_i = \frac{1}{n_i} \sum_{j=1}^{n_i} (\vec{x}_j - \vec{\mu}_i)^2$$

2. Estimar probabilidad de cada clase.

$$\hat{P}(C_i) = \frac{n_i}{N}; \quad n = \sum_{i=1}^M n_i \quad \text{Ec. 4.59}$$

3. Obtener regiones de decisión: $g_{ij}(\cdot)$.

$$g_i(\vec{x}) = \log(P(C_i)f(\vec{x}|C_i)) = \log \frac{P(C_i)}{(2\pi)^{n/2} \sigma_{1i} \sigma_{2i} \dots \sigma_{Fi}} - \frac{1}{2} (\bar{x}(S_i))^{-1} (\bar{x} - \bar{\mu}_i) \quad \text{Ec. 4.60}$$

4. Clasificar como un nuevo ejemplo.

$$\text{Clase}(\vec{X}) = \max_i \{P(C_i|\vec{X})\} = \max_i \{f(\vec{X}|C_i)p(C_i)\} \quad \text{Ec. 4.61}$$

4.2.3.2 Clasificación bayesiana con atributos nominales

Al incorporar atributos nominales con valores discretos, ahora cada atributo nominal A_i tendría n_i valores posibles.

- $A_k = \{V_1, \dots, V_{n_k}\}$: atributo con n_k valores posibles.

En la formulación probabilística del clasificador bayesiano ahora podemos utilizar probabilidades directamente en lugar de densidades de probabilidades como en los atributos numéricos. Con la formulación bayesiana, podemos calcular la probabilidad *a priori* de cada valor del atributo, condicionado a cada clase C_i , $p(A_k = V_j | C_i)$, contando el número de casos en el conjunto de entrenamiento:

$$p(A_k = V_j | C_i) = \frac{\text{nº de ejemplos de clase } C_i \text{ con } A_k = V_j}{\text{nº de ejemplos de clase } C_i} \quad \text{Ec. 4.62}$$

Al igual que en los atributos numéricos, una simplificación importante aparece si podemos considerar la independencia condicional de atributos (*naive bayesiano*), de manera que la probabilidad de un dato específico que tiene una combinación determinada de los atributos nominales se correspondería con el producto de las probabilidades de cada atributo:

$$X = (A_1 = V_1, A_2 = V_2, \dots, A_I = V_I) \\ p(X|C_i) = p(A_1 = V_1|C_i) * p(A_2 = V_2|C_i) * \dots * p(A_I = V_I|C_i) \quad \text{Ec. 4.63}$$

Con esta simplificación, el procedimiento de clasificación queda como sigue:

$$p(C_i|X) = \frac{p(X|C_i) * p(C_i)}{p(X)} = \frac{p(A_1 = V_1|C_i) * p(A_2 = V_2|C_i) * ... * p(A_I = V_I|C_i) * p(C_i)}{p(X)} \quad \text{Ec. 4.64}$$

4.2.4 Ejemplos de clasificación bayesiana

- Ejemplo con atributos nominales:

SALARIO	CLIENTE	EDAD	HIJOS	CRÉDITO
Poco	Si	Joven	Uno	NO
Mucho	Si	Joven	Uno	SI
Mucho	Si	Joven	Uno	SI
Poco	Si	Joven	Uno	NO
Mucho	Si	Joven	Dos	SI
Poco	Si	Joven	Dos	NO
Mucho	Si	Adulto	Dos	SI
Mucho	Si	Adulto	Dos	SI
Poco	No	Adulto	Dos	NO
Mucho	Si	Adulto	Dos	SI
Medio	No	Adulto	Tres	NO
Mucho	Si	Adulto	Dos	SI
Medio	Si	Adulto	Dos	SI
Medio	No	Adulto	Tres	NO
Medio	No	Adulto	Dos	SI
Mucho	No	Mayor	Tres	NO
Poco	No	Mayor	Tres	SI
Poco	No	Mayor	Tres	SI
Mucho	No	Mayor	Tres	NO
Mucho	No	Mayor	Tres	SI

$$p(SI) = 12/20$$

$$p(NO) = 8/20$$

	Crédito	No	Si
Salario			
Poco	4/8	2/12	
Mucho	2/8	8/12	
Medio	2/8	2/12	
Cliente	Crédito	No	Si
Si	3/8	8/12	
No	5/8	4/12	
Edad	Crédito	No	Si
Joven	3/8	3/12	
Adulto	3/8	6/12	
Mayor	2/8	3/12	
Hijos	Crédito	No	Si
Uno	2/8	2/12	
Dos	2/8	7/12	
Tres	4/8	3/12	

Fig. 4.13. Ejemplo de clasificación bayesiana discreta.

(salario=poco, cliente=sí, edad=adulto, hijos=tres)

$$p(SI|X) =$$

$$p(s = poco | SI) * p(c = si | SI) * p(e = adulto | SI) * p(h = tres | SI) * p(SI) / p(X) =$$

$$2/12 * 8/12 * 6/12 * 3/12 * 12/20 / p(Xi) = 0.0083 / p(X)$$

$$p(NO|X) =$$

$$p(s = poco | NO) * p(c = si | NO) * p(e = adulto | NO) * p(h = tres | NO) * p(NO) / p(X) =$$

$$4/8 * 3/8 * 3/8 * 4/8 * 8/20 / p(Xi) = 0.0141 / p(X)$$

$$p(SI|X) =$$

$$p(s = poco|SI) * p(c = si|SI) * p(e = adulto|SI) * p(h = tres|SI) * \frac{p(SI)}{p(X)} =$$

$$\frac{2}{12} * \frac{8}{12} * \frac{6}{12} * \frac{3}{12} * \frac{\frac{12}{20}}{p(Xi)} = \frac{0.0083}{p(X)}$$

$$p(NO|X) =$$

$$p(s = poco|NO) * p(c = si|NO) * p(e = adult|NO) * p(h = tres|NO) * \frac{p(NO)}{p(X)} =$$

$$\frac{4}{8} * \frac{3}{8} * \frac{3}{8} * \frac{4}{8} * \frac{\frac{8}{20}}{p(Xi)} = \frac{0.0141}{p(X)}$$

- **Atributos sin valores:**

Si el ejemplo a clasificar no tiene un atributo, simplemente se omite.

(salario=poco, cliente=sí, edad=?, hijos=3)

$$p(SI|X) =$$

$$p(s = poco|SI) * p(c = si|SI) * p(h = tres|SI) * \frac{p(SI)}{p(X)} =$$

$$\frac{2}{12} * \frac{8}{12} * \frac{3}{12} * \frac{\frac{12}{20}}{p(Xi)} = \frac{0.0167}{p(X)}$$

$$p(NO|X) =$$

$$p(s = poco|NO) * p(c = si|NO) * p(h = tres|NO) * \frac{p(NO)}{p(X)} =$$

$$\frac{4}{8} * \frac{3}{8} * \frac{4}{8} * \frac{\frac{8}{20}}{p(Xi)} = \frac{0.0375}{p(X)}$$

- Si hay faltas en la muestra de entrenamiento, no cuentan en la estimación de probabilidades de ese atributo.

Faltas en atributo *edad*:

SALARIO	CLIENTE	EDAD	HIJOS	CRÉDITO
Poco	Si	Joven	Uno	NO
Mucho	Si	Joven	Uno	SI
Mucho	Si	Joven	Uno	SI
Poco	Si	?	Uno	NO
Mucho	Si	?	Dos	SI
Poco	Si	?	Dos	NO
Mucho	Si	?	Dos	SI
Mucho	Si	Adulto	Dos	SI
Poco	No	Adulto	Dos	NO
Mucho	Si	Adulto	Dos	SI
Medio	No	Adulto	Tres	NO
Mucho	Si	Adulto	Dos	SI
Medio	Si	Adulto	Dos	SI
Medio	No	Adulto	Tres	NO
Medio	No	Adulto	Dos	SI
Mucho	No	Mayor	Tres	NO
Poco	No	Mayor	Tres	SI
Poco	No	Mayor	Tres	SI
Mucho	No	Mayor	Tres	NO
Mucho	No	Mayor	Tres	SI

Salario	Crédito	
	No	Si
Poco	4/8	2/12
Mucho	2/8	8/12
Medio	2/8	2/12

Cliente	Crédito	
	No	Si
Si	3/8	8/12
No	5/8	4/12

Edad	Crédito	
	No	Si
Joven	1/6	2/10
Adulto	3/6	5/10
Mayor	2/6	3/10

Hijos	Crédito	
	No	Si
Uno	2/8	2/12
Dos	2/8	7/12
Tres	4/8	3/12

$$p(SI) = 12/20$$

$$p(NO) = 8/20$$

Fig. 4.14. Ejemplo de clasificación bayesiana con faltas.

• Atributos no representados. Ley μ .

- Problema: con una muestra poco representativa, puede ocurrir que en alguna clase un valor de atributo no aparezca: $p(A_i=V_j|C_k)=0$.
- Cualquier ejemplo X con $A_i=V_j$ generará $P(C_k|X)=0$, independientemente de los otros atributos.
- Se suele modificar la estimación de las probabilidades *a priori* con un factor que elimina los ceros.
- Ejemplo: $P(\text{Edad}|\text{Crédito}=\text{NO}) = \left\{ \text{Joven: } \frac{3}{8}, \text{ Adulto: } \frac{3}{8}, \text{ Mayor: } \frac{2}{8} \right\}$
- Ley m : $\left\{ \text{Joven: } \frac{3+\mu}{8+\mu}, \text{ Adulto: } \frac{3+\mu}{8+\mu}, \text{ Mayor: } \frac{2+\mu}{8+\mu} \right\}$
- A veces simplemente se inicializan las cuentas a 1 en vez de 0:

$$\left\{ \text{Joven: } \frac{3+1}{8+3}, \text{ Adulto: } \frac{3+1}{8+3}, \text{ Mayor: } \frac{2+1}{8+3} \right\}$$

• Atributos mixtos:

- Independencia de atributos (*naive bayesiano*).
- $p(X_i|C_k) = p(A_1 = V_1|C_k) * p(A_2 = V_2|C_k) * \dots * p(A_F = V_F|C_k)$
- Atributos discretos: probabilidades *a priori* con cada clase C_k .

$$p(A_k = V_j | C_i) = \frac{n^{\circ} \text{ de ejemplos de clase } C_i \text{ con } A_k = V_j}{n^{\circ} \text{ de ejemplos de clase } C_i}$$

- Atributos continuos: densidades de clase C_i : normales de parámetros μ , σ .

$$p(A_k = V_j | C_i) \rightarrow f_{A_k}(V_j | C_i) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp \left[-\frac{1}{2} \frac{(V_j - \mu_{ij})^2}{\sigma_{ij}^2} \right]$$

Ejemplo con atributos mixtos:

SALARIO	CLIENTE	EDAD	HIJOS	CRÉDITO
525	Si	Joven	1	NO
2000	Si	Joven	1	SI
2500	Si	Joven	1	SI
470	Si	Joven	1	NO
3000	Si	Joven	2	SI
510	Si	Joven	2	NO
2800	Si	Adulto	2	SI
2700	Si	Adulto	2	SI
550	No	Adulto	2	NO
2600	Si	Adulto	2	SI
1100	No	Adulto	3	NO
2300	Si	Adulto	2	SI
1200	Si	Adulto	2	SI
900	No	Adulto	3	NO
800	No	Adulto	2	SI
800	No	Mayor	3	NO
1300	No	Mayor	3	SI
1100	No	Mayor	3	SI
1000	No	Mayor	3	NO
4000	No	Mayor	3	SI

$$p(\text{SI}) = 12/20$$

$$p(\text{NO}) = 8/20$$

Salario \ Crédito	No	Si
Media	732	2192
Desv Estándar	249	942

Cliente \ Crédito	No	Si
Si	3/8	8/12
No	5/8	4/12

Edad \ Crédito	No	Si
Joven	3/8	3/12
Adulto	3/8	6/12
Mayor	2/8	3/12

Hijos \ Crédito	No	Si
Media	2.25	2.08
Desv Estándar	0.89	0.67

Fig. 4.15. Ejemplo de clasificador bayesiano con atributos mixtos.

- (salario=700, cliente=sí, edad=adulto, hijos=3)

$$p(SI|X) =$$

$$f_S(s = 700|SI) * p(c = si|SI) * p(e = adulto|SI) * f_H(h = 3|SI) * \frac{p(SI)}{p(X)} =$$

$$\frac{1}{\sqrt{2\pi}942} \exp \left[-\frac{1}{2} \frac{(700 - 2192)^2}{942^2} \right] * \frac{8}{12} * \frac{6}{12} * \frac{1}{\sqrt{2\pi}0.67} \exp \left[-\frac{1}{2} \frac{(3 - 2.08)^2}{0.67^2} \right] * \frac{12}{20} * \frac{1}{P(X)} =$$

$$= 5.61e - \frac{6}{p(X)}$$

$$p(NO|X) =$$

$$\begin{aligned} & f_S(s = 700|NO) * p(c = si|NO) * p(e = adulto|NO) * f_H(h = 3|NO) * \frac{p(NO)}{p(X)} = \\ & \frac{1}{\sqrt{2\pi}249} \exp\left[-\frac{1}{2} \frac{(700 - 732)^2}{249^2}\right] * \frac{3}{8} * \frac{3}{8} * \\ & \frac{1}{\sqrt{2\pi}0.89} \exp\left[-\frac{1}{2} \frac{(3 - 2.25)^2}{0.89^2}\right] * \frac{8}{20} * \frac{1}{P(X)} = \\ & = 2.81e - \frac{5}{p(X)} \end{aligned}$$

- **Clasificación con costes:**

- El método visto hasta ahora, *Maxim a Posteriori* (MAP), proporciona la clasificación con mínima probabilidad de error. Sin embargo, con frecuencia los costes son asimétricos, y unos errores son más graves que otros, lo que queda reflejado en la matriz de costes. Un ejemplo para clasificación con tres categorías sería la matriz siguiente:

$$\begin{array}{c} \text{Clasificado como} \\ \begin{pmatrix} 0 & c_{12} & c_{13} \\ c_{21} & 0 & c_{23} \\ c_{31} & c_{32} & 0 \end{pmatrix} \\ \text{Clase real} \end{array}$$

Costes de cada decisión. Criterio de mínimo coste medio: dada una decisión, el promedio de los costes de cada equivocación y su coste.

$$\text{coste}(D_1|\vec{X}) = c_{21}p(c_2|\vec{X}) + c_{31}p(c_3|\vec{X})$$

$$\text{coste}(D_2|\vec{X}) = c_{12}p(c_1|\vec{X}) + c_{32}p(c_3|\vec{X})$$

$$\text{coste}(D_3|\vec{X}) = c_{13}p(c_1|\vec{X}) + c_{23}p(c_2|\vec{X})$$

Ejemplo de clasificación con costes.

- Clasificación de setas con dos atributos (X, Y) y tres categorías: *venenosa*, *mal sabor*, *comestible*: {V, MS, C}.

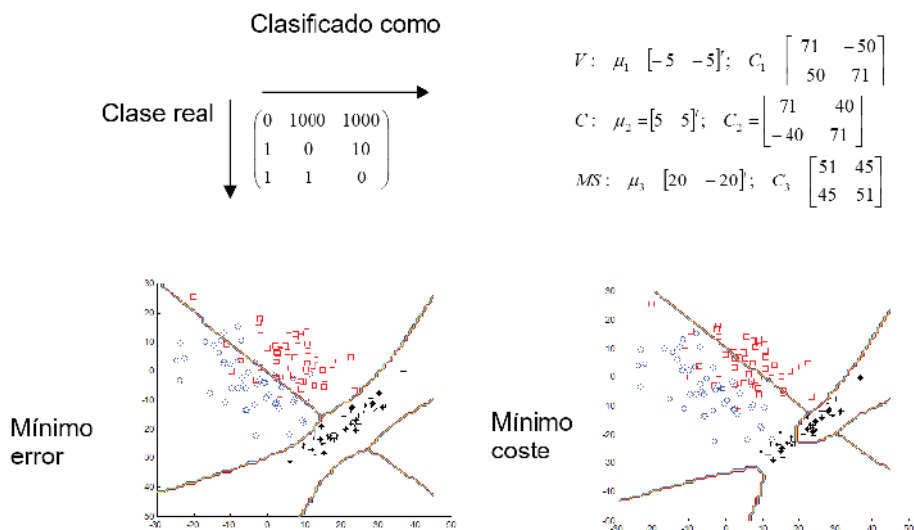


Fig. 4.16. Ejemplo de fronteras de decisión con clasificador bayesiano con costes.

Como puede verse, incorporar costes de clasificación implica un desplazamiento de las fronteras de decisión, en este caso para evitar errores de clasificación con los ejemplos de la clase más costosa (venenosa).

Predicción y clasificación con R

CAPÍTULO 5

5.1 Regresión

5.1.1 Regresión lineal

Se ha visto anteriormente medidas de correlación y representación en gráficos de dispersión del ajuste lineal entre los diferentes atributos. Un valor alto en el coeficiente de correlación es un indicio de que se puede utilizar el modelo de regresión lineal como predicción de valores futuros.

El objetivo consiste en predecir la concentración de NO₂ a partir del resto de atributos. Para poder valorar la bondad de la predicción se va a reservar un conjunto de datos que es usado para realizar el modelo. Esta forma de proceder es la más simple, dividiendo los datos en dos conjuntos disjuntos: uno de **entrenamiento** y otro de **validación**. Hay otras alternativas, como utilizar un conjunto de entrenamiento, uno de validación y un tercero de test. Ésta es muy común para las técnicas que ajustan sus parámetros a partir del proceso de entrenamiento y detectan situaciones de sobreaprendizaje observando la evolución del error de validación. Un tercer procedimiento, muy utilizado en las técnicas de aprendizaje automático, es el de la **validación cruzada**. Consiste en fijar un porcentaje para el conjunto de validación seleccionado, obtener el modelo de predicción y repetir el proceso cambiando los datos que conformaron el conjunto de validación por otros diferentes.

Otro problema adicional es la elección de los datos concretos que formarán los conjuntos de entrenamiento y validación. La información contenida en los datos guía el proceso de ajuste del modelo de predicción, si los datos están sesgados, también lo estará el modelo. Por tanto, es un punto clave la elección adecuada de los datos que se presentan al modelo.

```
dfMedidas <- as.data.frame(lapply(dfMedidas, function (x) if (!is.numeric(x)) fac-
tor(x) else x))

h <- hetcor(dfMedidas)

heatmap(h$correlations, Rowv = NA, Colv = NA, col = heat.colors(256), na.rm=TRUE)

library(gplots)

mi_color <- colorRampPalette(c("red", "black", "red"))(n = 299)

heatmap.2(h$correlations,col=mi_color, Rowv = FALSE,
notecol="black", density.info="none", trace="none",
margins =c(8,8), dendrogram = "none",Colv="NA")
```

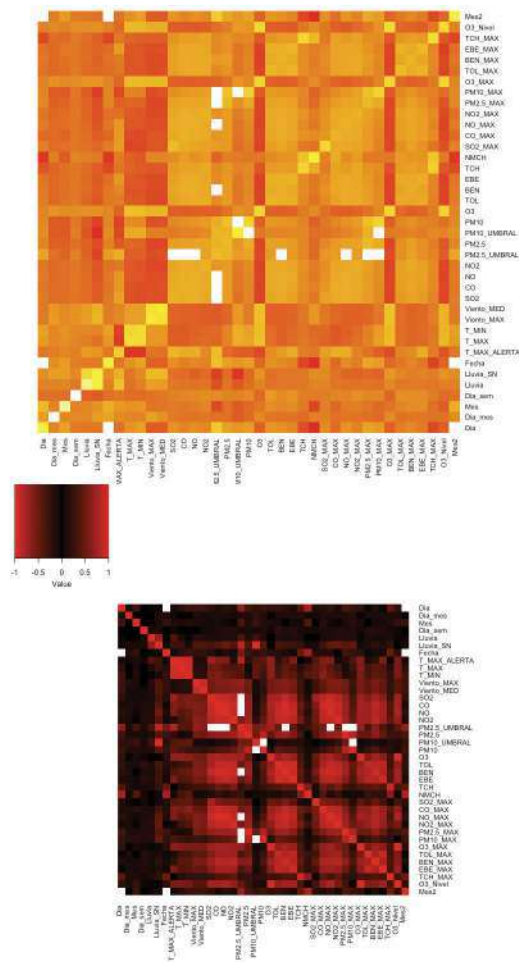


Fig. 5.1. Mapas de calor de las correlaciones entre los atributos.

En el mapa de calor se pueden ver las variables más correladas; dado que el objetivo es la predicción de NO2, se van a mostrar sólo los valores de correlación ordenados para este atributo.

```
h$correlations[order(abs(h$correlations[, "NO2"]), decreasing = TRUE), "NO2"]
```

NO2	NO2_MAX	CO	BEN	NO	NO_MAX	TOL
1.00	0.97	0.93	0.91	0.90	0.89	0.88
BEN_MAX	CO_MAX	EBE	SO2	O3	EBE_MAX	
0.87	0.86	0.85	0.84	-0.76	0.74	0.74
PM2.5_MAX	O3_MAX	PM2.5	O3_Nivel	Viento_MAX	Viento_MED	PM2.5_UMBRAL
0.74	-0.72	0.72	-0.67	-0.66	-0.63	0.59
TCH	SO2_MAX	PM10_MAX	TCH_MAX	PM10	T_MIN	Lluvia_SN
0.56	0.50	0.49	0.46	0.46	-0.44	-0.39
T_MAX	NMCH	PM10_UMBRAL	T_MAX_ALERTA	Dia_sem	Dia_sem_num	Lluvia
-0.35	0.24	0.21	0.15	0.14	0.12	-0.12
Dia	Dia_mes	Fecha	Mes			
0.12	-0.08	0.06	0.02			

De los resultados se aprecia que el atributo más correlado con la concentración de NO2, obviamente, es NO2_MAX. Los siguientes atributos que están más correlados (CO, BEN, NO) no lo están porque haya una relación causal entre ellos, si no porque todos son causa del mismo efecto, la quema de combustibles fósiles. Por tanto, no son adecuados para la predicción de NO2 en términos de relación causa-efecto.

```
library("ggpubr")
ggscatter(dfMedidas, x = "CO", y = "NO2",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Concentración CO", ylab = "Concentración NO2")
```

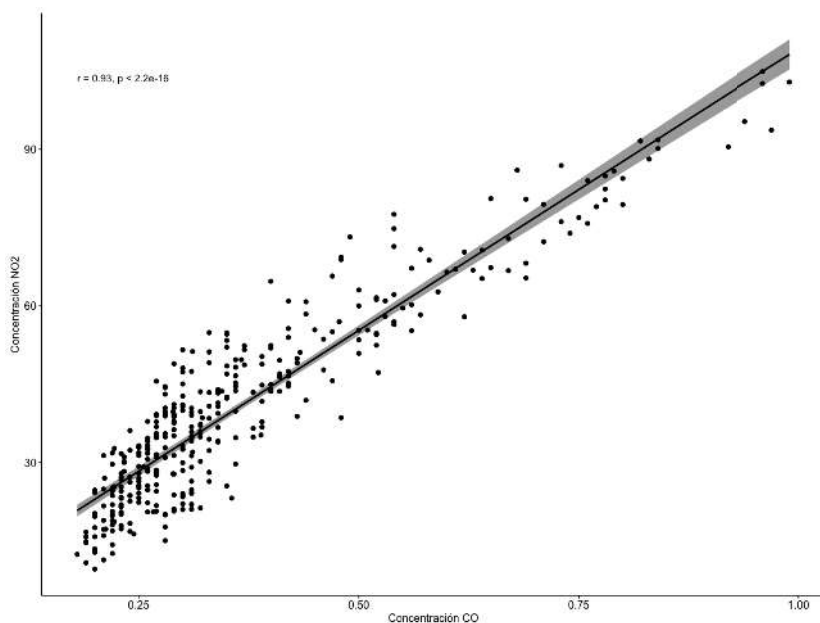


Fig. 5.2. Regresión lineal entre la concentración de CO y de NO2.

Una forma de obtener la regresión junto con los residuos y significancia estadística es utilizando explícitamente la función *lm* (*linear model*).

```
l1 <-lm(NO2 ~ CO, data = dfMedidas)
summary(l1)

Call:
lm(formula = NO2 ~ CO, data = dfMedidas)

Residuals:
    Min       1Q   Median       3Q      Max
-16.5468  -5.2064  -0.4276   4.8146  20.1863

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3812     0.9196   1.502   0.134
CO           107.8521     2.2799  47.305 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 7.286 on 363 degrees of freedom
Multiple R-squared:  0.8604,    Adjusted R-squared:  0.86
F-statistic: 2238 on 1 and 363 DF,  p-value: < 2.2e-16
```

Se obtiene el coeficiente y el punto de corte con el eje. Se pueden mostrar gráficas que determinan la estructura del error, si es gaussiano o no (gráfica superior derecha), o en la gráfica inferior derecha la influencia de cada valor de CO2 en la obtención del modelo. Los puntos muy alejados (alto valor de Leverage) en esta última gráfica serían candidatos a ser eliminados del conjunto de datos con los que se realiza el modelo, ya que son candidatos a ser *outliers*.

```
par(mfrow=c(2,2))
plot(l1)
```

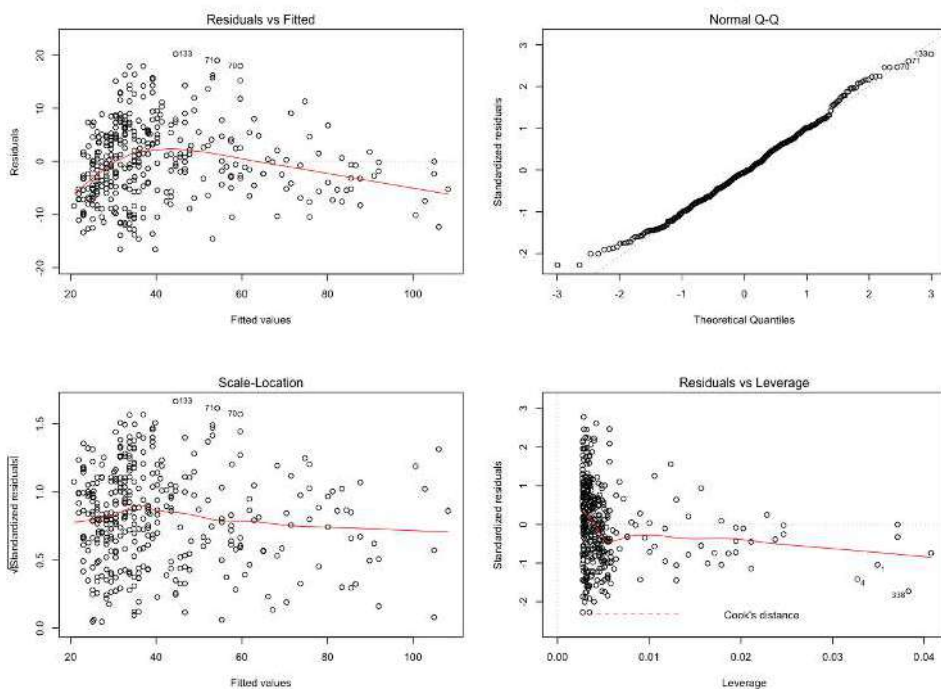


Fig. 5.3. Gráficas que muestran la estructura del error de regresión.

Realizemos ahora la regresión de la concentración de CO2 con los atributos meteorológicos.

Se van a crear tres modelos, uno sólo con el atributo de velocidad del viento (*fit1*), un segundo que añade la temperatura máxima (*fit2*) y un tercero que incluye además la

cantidad de lluvia (fit3).

```
fit1 <- lm(NO2 ~ Viento_MAX , data = dfMedidas)
fit2 <- lm(NO2 ~ Viento_MAX + T_MAX, data = dfMedidas)
fit3 <- lm(NO2 ~ Viento_MAX + T_MAX + Lluvia, data = dfMedidas)
fit1
fit2
fit3
```

```
Call:
lm(formula = NO2 ~ Viento_MAX, data = dfMedidas)

Coefficients:
(Intercept)    Viento_MAX
      75.859         -1.107

Call:
lm(formula = NO2 ~ Viento_MAX + T_MAX, data = dfMedidas)

Coefficients:
(Intercept)    Viento_MAX         T_MAX
  86.1814      -1.0428      -0.5581

Call:
lm(formula = NO2 ~ Viento_MAX + T_MAX + Lluvia, data = dfMedidas)

Coefficients:
(Intercept)    Viento_MAX         T_MAX         Lluvia
  86.2859      -1.0276      -0.5740      -0.3159
```

Se pueden obtener los estadísticos de cada modelo en su conjunto con *summary* o utilizando funciones para extraer algunos concretos.

```

coefficients(fit1) # coeficientes del modelo
confint(fit1, level=0.95) # coeficientes con intervalos de confianza
head(fitted(fit1),10) # valores predichos
head(residuals(fit1),10) # residuos
anova(fit1) # tabla anova
vcov(fit1) # matriz de covarianza
    
```

```

(Intercept)  Viento_MAX
    75.859254   -1.107232
           2.5 %    97.5 %
(Intercept)  71.515432  80.2030764
Viento_MAX   -1.236548 -0.9779164
9           1          2          3          4          5          6          7          8
9          10         60.35800  62.57247  54.82184  64.78693  59.25077  54.82184  59.25077
60.35800  63.67970  60.35800  62.57247  54.82184  64.78693  59.25077  54.82184  59.25077
62.57247
9           1          2          3          4          5          6          7          8
9          10         34.93367  38.90363  29.76700  27.88587  29.59483  11.00474  32.54090  32.05316  43.62423
21.42753
Analysis of Variance Table

Response: NO2

           Df Sum Sq Mean Sq F value    Pr(>F)
Viento_MAX   1  60549   60549   283.51 < 2.2e-16 ***
Residuals  363  77526     214
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Intercept)  Viento_MAX
(Intercept)    4.879182 -0.136266043
Viento_MAX    -0.136266  0.004324214
    
```

La comparación de los tres modelos se puede realizar mediante un análisis de varianza, ANOVA.

```

anova(fit1, fit2, fit3)
Analysis of Variance Table
    
```

```

Model 1: NO2 ~ Viento_MAX
Model 2: NO2 ~ Viento_MAX + T_MAX
Model 3: NO2 ~ Viento_MAX + T_MAX + Lluvia

Res.Df    RSS Df Sum of Sq      F      Pr(>F)
1       363 77526
2       362 68518   1    9007.4 47.6843 2.266e-11 ***
3       361 68192   1     326.6  1.7292   0.1894
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Se aprecia que el modelo 2 mejora la suma de los cuadrados de los residuos (RSS) en 9007 sobre el modelo 1, y el modelo 3 sobre el modelo 2 en 326. La cuestión es si esta mejora es significativa. Para dar respuesta hay que observar los valores de F y de $\text{Pr}(>F)$. El valor de este último para el modelo 2 es mucho menor que 0.05, de modo que es significativa la mejora del modelo 2 sobre el 1, pero esta condición no ocurre para la diferencia entre el modelo 3 y 2, de manera que añadir la cantidad de lluvia no hace que mejore significativamente el modelo predictivo.

```

aux <- data.frame(Dia, NO2, fitted(fit1), fitted(fit2), fitted(fit3) )
names(aux) <- c("Dia", "Real", "Modelo1", "Modelo2", "Modelo3")
aux <- aux[order(aux$Real),]
aux$Dia <- seq(1:nrow(aux))

ggplot(aux) +
  geom_jitter(aes(Dia, Real), colour="red", size = 0.5) + geom_smooth(aes(Dia, Real),
    method = lm, se=FALSE, colour="red", size = 0.5) +
  geom_jitter(aes(Dia, Modelo1), colour="black", size = 0.5) + geom_smooth(aes(Dia,
    Modelo1),
    method = lm, se=FALSE, colour="black", size = 0.5) +
  geom_jitter(aes(Dia, Modelo2), colour="blue", size = 0.5) + geom_smooth(aes(Dia,
    Modelo2),
    method = lm, se=FALSE, colour="blue", size = 0.5) +
  geom_jitter(aes(Dia, Modelo3), colour="green", size = 0.5) + geom_smooth(aes(Dia,
    Modelo3),
    method = lm, se=FALSE, colour="green", size = 0.5) +
  annotate("text", x=96, y=100, label= "\U2022 Real", colour = "red") +
  annotate("text", x=100, y=95, label= "\U2022 Modelo 1", colour = "black") +
  annotate("text", x=100, y=90, label= "\U2022 Modelo 2", colour = "blue") +
  annotate("text", x=100, y=85, label= "\U2022 Modelo 3", colour = "green")

```

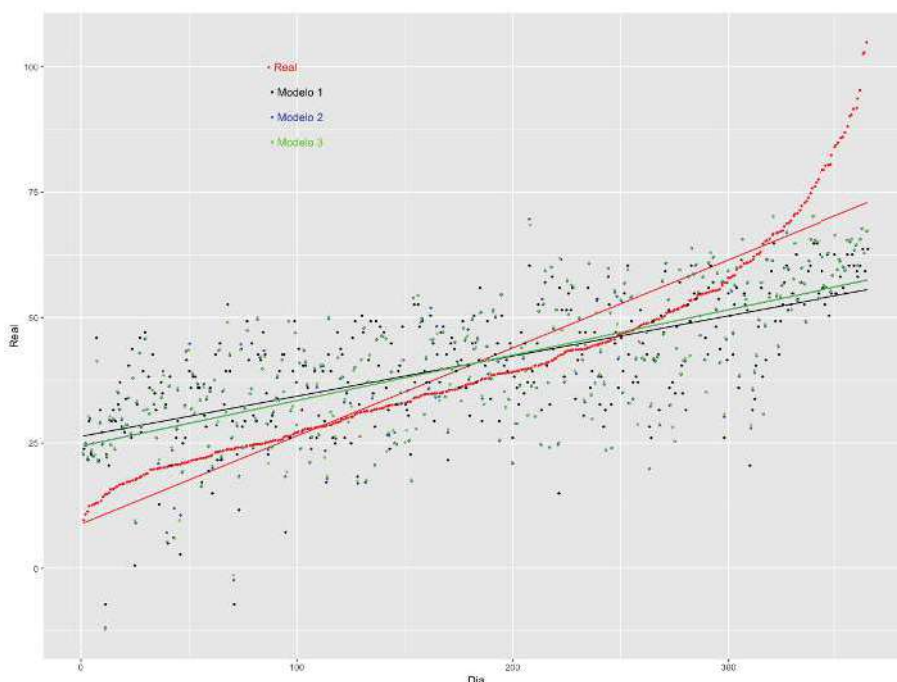


Fig. 5.4. Comparación de diferentes modelos de regresión.

En el gráfico anterior, en rojo, están ordenados los valores de NO₂ reales y con otros colores los valores obtenidos por los modelos de regresión. Se aprecia la mejora que suponen los modelos 2 y 3 frente al 1. Si bien, entre el modelo 2 y 3 el resultado es prácticamente el mismo.

Esta forma de proceder, partiendo de un modelo simple e ir añadiendo atributos, uno a uno (en estricto sentido habría que añadir el más influyente), observando si el modelo mejora o empeora, es una técnica de regresión múltiple llamada *introducción progresiva* (*Forward Stepwise Regression*). El contrario, partir de un modelo de regresión con todos los atributos, seleccionar el menos influyente y determinar si mejora o empeora, se denomina *eliminación progresiva* (*Backward Stepwise Regression*). Existe una alternativa intermedia, regresión paso a paso (*Stepwise Regression*). Este tipo de proceso se puede realizar en R utilizando *step*.

```
step(lm(NO2 ~ Viento_MAX + T_MAX + Lluvia, data = dfMedidas), direction = "both")
```

```
Start:  AIC=1917.02
```

```
NO2 ~ Viento_MAX + T_MAX + Lluvia
```

	Df	Sum of Sq	RSS	AIC
- Lluvia	1		327	68518 1916.8
<none>				68192 1917.0
- T_MAX	1		9319	77511 1961.8
- Viento_MAX	1		49312	117504 2113.6

```
Step:  AIC=1916.76
```

```
NO2 ~ Viento_MAX + T_MAX
```

	Df	Sum of Sq	RSS	AIC
<none>				68518 1916.8
+ Lluvia	1		327	68192 1917.0
- T_MAX	1		9007	77526 1959.8
- Viento_MAX	1		52517	121035 2122.4

```
Call:
```

```
lm(formula = NO2 ~ Viento_MAX + T_MAX, data = dfMedidas)
```

```
Coefficients:
```

(Intercept)	Viento_MAX	T_MAX
86.1814	-1.0428	-0.5581

Como resultado, se obtiene que el mejor modelo es el que utiliza Viento_MAX y T_MAX.

Hay una cuestión que hemos aplazado hasta este punto. Los atributos que se han utilizado para realizar el modelo de regresión multineal han sido numéricos continuos. ¿Pueden utilizarse otro tipo de atributos? La respuesta es sí. Efectivamente, los atributos con valores numéricos discretos o los factores también pueden ser variables en los modelos de regresión lineal.

Si añadimos un modelo lineal donde sustituimos la cantidad de lluvia por el atributo donde sólo indica si ha llovido o no, Lluvia_SN, se obtiene:

```
fit4 <- lm(NO2 ~ Viento_MAX + T_MAX + Lluvia_SN, data = dfMedidas)
anova(fit1, fit2, fit4)
```

Analysis of Variance Table

Model 1: NO2 ~ Viento_MAX

Model 2: NO2 ~ Viento_MAX + T_MAX

Model 3: NO2 ~ Viento_MAX + T_MAX + Lluvia_SN

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	363	77526			
2	362	68518	1	9007.4	48.382 1.655e-11 ***
3	361	67208	1	1310.5	7.039 0.008327 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

El resultado es que la mejora con ese atributo es significativa. Lo podemos comprobar aplicando una regresión progresiva.

```
step(lm(NO2 ~ Viento_MAX + T_MAX + Lluvia_SN, data = dfMedidas), direction = "both")
```

Expand/Collapse Output

Start: AIC=1911.71

NO2 ~ Viento_MAX + T_MAX + Lluvia_SN

	Df	Sum of Sq	RSS	AIC
<none>			67208	1911.7
- Lluvia_SN	1	1310	68518	1916.8
- T_MAX	1	10175	77383	1961.2
- Viento_MAX	1	42023	109231	2087.0

Call:

```
lm(formula = NO2 ~ Viento_MAX + T_MAX + Lluvia_SN, data = dfMedidas)
```

Coefficients:

(Intercept)	Viento_MAX	T_MAX	Lluvia_SNSi
86.5788	-0.9863	-0.6117	-5.1813

Ahora, el modelo mejor es el que tiene todos los atributos. Se aprecia que, en los coeficientes, para Lluvia_SN está asociado -5.18 al valor “Sí”, cuando este atributo toma valor “No” el coeficiente es 0. Esta forma de proceder con los atributos categóricos es muy habitual. Consiste en crear nuevas variables *dummy* para cada una de las alternativas, tomando únicamente valores binarios. Por ejemplo, para el día de la semana se crearían siete nuevos atributos, uno para el lunes, etc. Una de las nuevas variables *dummy* tomaría el valor 1 y el resto sería 0. Lo que tiene lugar es la obtención de dos modelos de regresión, uno para cada valor de la variable *dummy*.

Representando la función en 3D para el modelo con el viento y la temperatura máxima, obviamente ésta definirá un plano, ya que se utilizan modelos lineales para los atributos.

```
V_max <- seq(min(dfMedidas$Viento_MAX),max(dfMedidas$Viento_MAX),length.out=20)
T_max <- seq(min(dfMedidas$T_MAX),max(dfMedidas$T_MAX),length.out=20)
z <- function (x, y) {
  return (x*(-1.0428) + y*(-0.5581) + (86.1814 ))
}
NO2_pred <- outer(V_max, T_max, z)
color <- gray((0:32)/32)
nrz <- nrow(NO2_pred)
ncz <- ncol(NO2_pred)
zfacet <- NO2_pred[-1, -1] + NO2_pred[-1, -ncz] + NO2_pred[-nrz, -1] + NO2_pred[-nrz, -ncz]
facetcol <- cut(zfacet, 32)

p <- persp(V_max, T_max, NO2_pred, col=color[facetcol], theta = 30, phi = 30, tick-
type = "detailed")
obs <- trans3d(dfMedidas$Viento_MAX, dfMedidas$T_MAX, dfMedidas$NO2, p)
points(obs, col = "red", pch = 16)
pred <- trans3d(dfMedidas$Viento_MAX, dfMedidas$T_MAX, fit2$fitted.values, p)
segments(obs$x, obs$y, pred$x, pred$y, col = "gray")
```

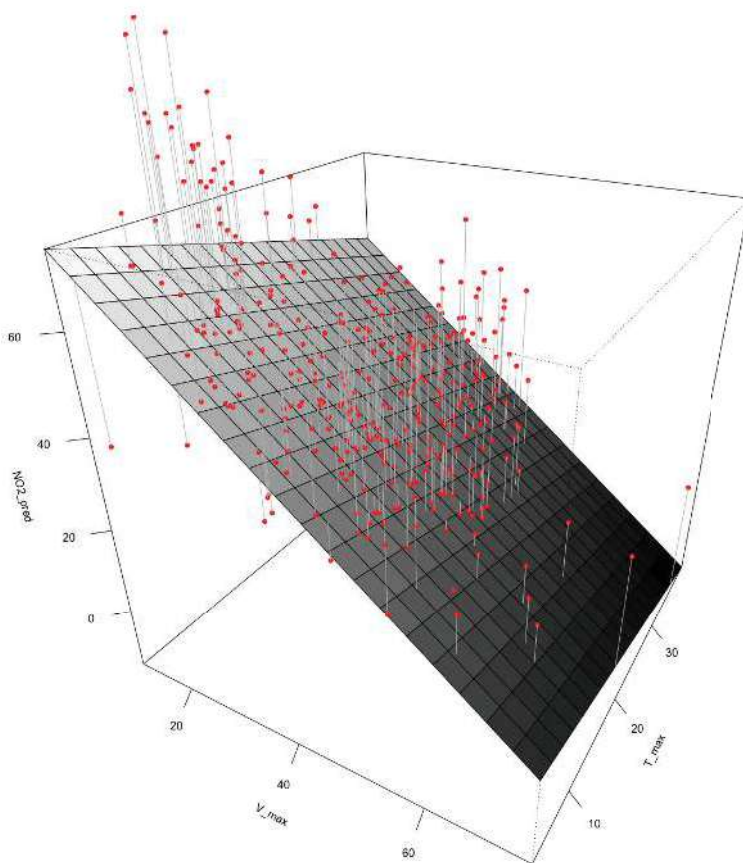


Fig. 5.5. Representación del plano de regresión sobre tres atributos.

Los puntos rojos muestran los valores reales y la proyección sobre el plano serían los valores predichos por el modelo.

5.1.2 Selección de atributos

La selección de atributos que optimizan la capacidad de predicción de un modelo es uno de los puntos más importantes del proceso de análisis. En el mejor de los casos, incluir atributos que no son relevantes puede resultar inocuo y el único impacto que tiene es hacer el modelo innecesariamente más grande o aumentar el tiempo de obtención de éste. Pero lo normal es que estos atributos introduzcan ruido al modelo, pudiéndolo sesgar y empeorando así su capacidad predictiva.

Hay bastantes técnicas para realizar este proceso, se irán mostrando algunas de ellas en función del contexto del proceso de análisis que se esté realizando.

Para determinar los atributos relevantes en una regresión lineal se ha visto en la sección anterior el análisis de la varianza y la regresión progresiva. En R hay más funciones, implementadas en otras librerías para aplicar esta técnica. Por ejemplo, la librería MASS realiza la regresión progresiva aplicando el criterio de información de Akaike.

```
library(leaps)

fit_leaps <- regsubsets(NO2 ~ Viento_MAX + T_MAX + Lluvia + Lluvia_SNi, data = dfMe-
didas, nbest = 16)

summary(fit_leaps)

par(mfrow=c(1,2))

plot(fit_leaps, scale = "r2")

library(car)

subsets(fit_leaps, statistic = "rss")
```

```
Lluvia          FALSE      FALSE
Lluvia_SNi      FALSE      FALSE
16 subsets of each size up to 4
Selection Algorithm: exhaustive
```

		Viento_MAX	T_MAX	Lluvia	Lluvia_SNi
1	(1)	"*"	" "	" "	" "
1	(2)	" "	"*"	" "	" "
1	(3)	" "	" "	" "	"*"
1	(4)	" "	" "	"*"	" "
2	(1)	"*"	"*"	" "	" "
2	(2)	"*"	" "	" "	"*"
2	(3)	"*"	" "	"*"	" "
2	(4)	" "	"*"	" "	"*"
2	(5)	" "	"*"	"*"	" "
2	(6)	" "	" "	"*"	"*"
3	(1)	"*"	"*"	" "	"*"
3	(2)	"*"	"*"	"*"	" "
3	(3)	"*"	" "	"*"	"*"
3	(4)	" "	"*"	"*"	"*"
4	(1)	"*"	"*"	"*"	"*"

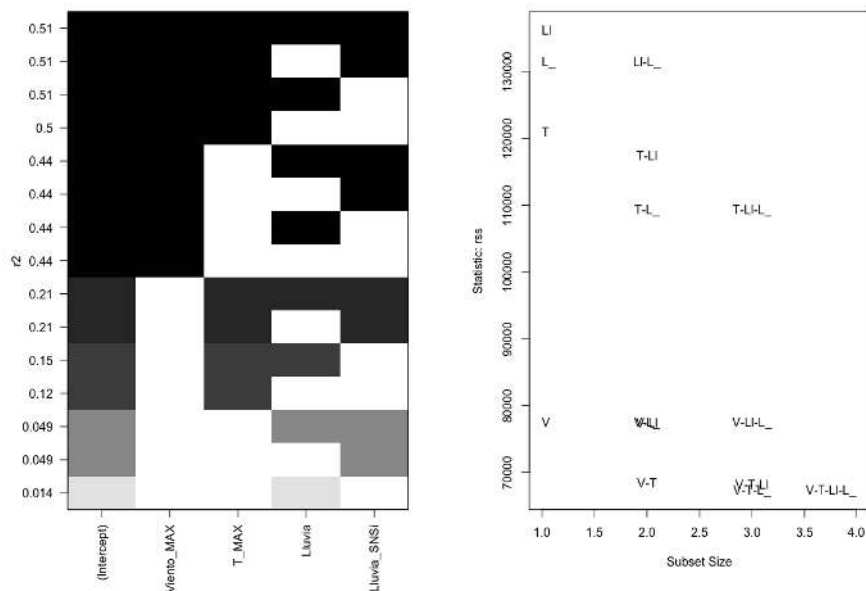


Fig. 5.6. Importancia de los atributos en los modelos de regresión.

Es interesante observar los resultados en los gráficos. Se ha realizado una búsqueda exhaustiva y se han llevado a cabo modelos de regresión para todas las combinaciones de los atributos. En el primer gráfico se muestra el resultado de las diferentes combinaciones de atributos según el coeficiente de determinación. En el siguiente gráfico aparecen en columnas los atributos por la cantidad que se usan para realizar la regresión, en el eje de ordenadas el valor de rss . El mejor atributo es el viento y la mejor combinación de dos atributos el viento con la temperatura, etc.

Para calcular la importancia relativa entre los atributos pueden utilizarse algunas de las funciones de la librería *relaimpo*.

```
library(relaimpo)

importancia <- calc.relimp(lm(NO2 ~ Viento_MAX + T_MAX + Lluvia + Lluvia_SN, data = dfMedidas), type = c("lmg", "last", "first", "betasq", "pratt", "genizi", "car"), rela = TRUE)

importancia
plot(importancia)
```

Response variable: NO2

Total response variance: 379.3271

Analysis based on 365 observations

4 Regressors:

Viento_MAX T_MAX Lluvia Lluvia_SN

Proportion of variance explained by model: 51.33%

Metrics are normalized to sum to 100% (rela=TRUE).

Relative importance metrics:

	lmg	last	first	betasq	pratt	genizi
Viento_MAX	0.72662219	7.898152e-01	0.70239573	7.930996e-01	0.760816357	0.72510237
T_MAX	0.20354657	1.915307e-01	0.19767032	1.830475e-01	0.193899960	0.20163108
Lluvia	0.01126820	6.715681e-05	0.02171285	7.778132e-05	0.001324711	0.01229679
Lluvia_SN	0.05856305	1.858688e-02	0.07822109	2.377503e-02	0.043958972	0.06096976
car						
Viento_MAX	0.746743256					
T_MAX	0.199685706					
Lluvia	0.005948197					
Lluvia_SN	0.047622841					

Average coefficients for different model sizes:

	1X	2Xs	3Xs	4Xs
Viento_MAX	-1.1072324	-1.0794624	-1.0350862	-0.98599878
T_MAX	-0.7590041	-0.7463954	-0.6891339	-0.61209666
Lluvia	-0.7381084	-0.3847131	-0.1461728	-0.03702236
Lluvia_SN	-10.9172223	-9.0091956	-7.0274839	-5.04400410

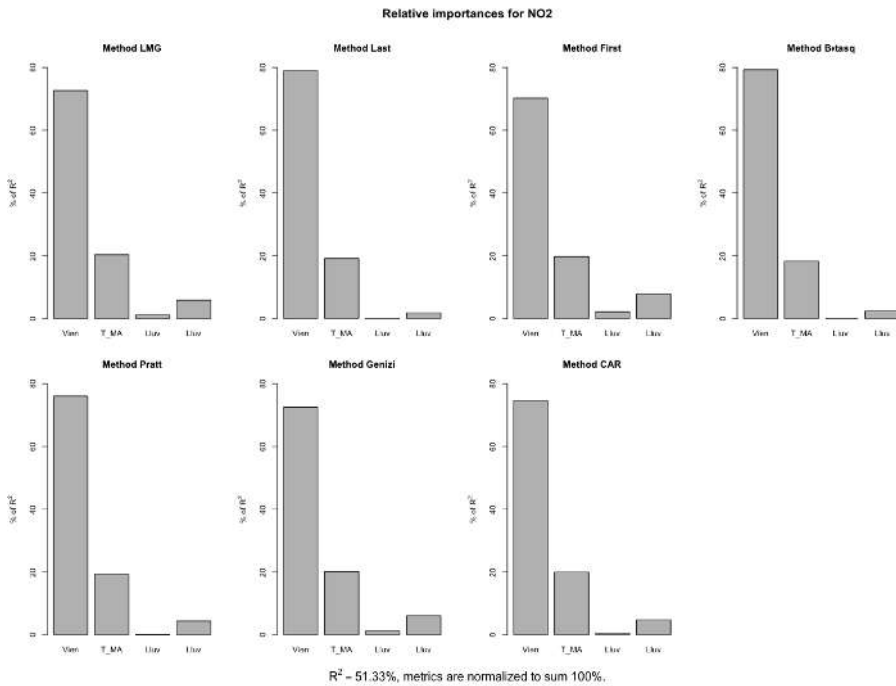


Fig. 5.7. Importancia relativa de los atributos para predecir la concentración de NO2.

Todas las figuras de mérito coinciden en que el viento tiene una importancia superior al 70 % y la temperatura está en torno al 20 %.

5.1.3 Regresión no lineal

El siguiente paso natural es considerar que la relación entre los atributos es más compleja que la que proporciona una relación lineal.

Representando los atributos en un gráfico de dispersión, se aprecia que una posible relación sería un polinomio de orden 2 o una exponencial decreciente.

```
ggplot(dfMedidas, aes(x = Viento_MAX, y = NO2)) + geom_point(size=0.5)
```

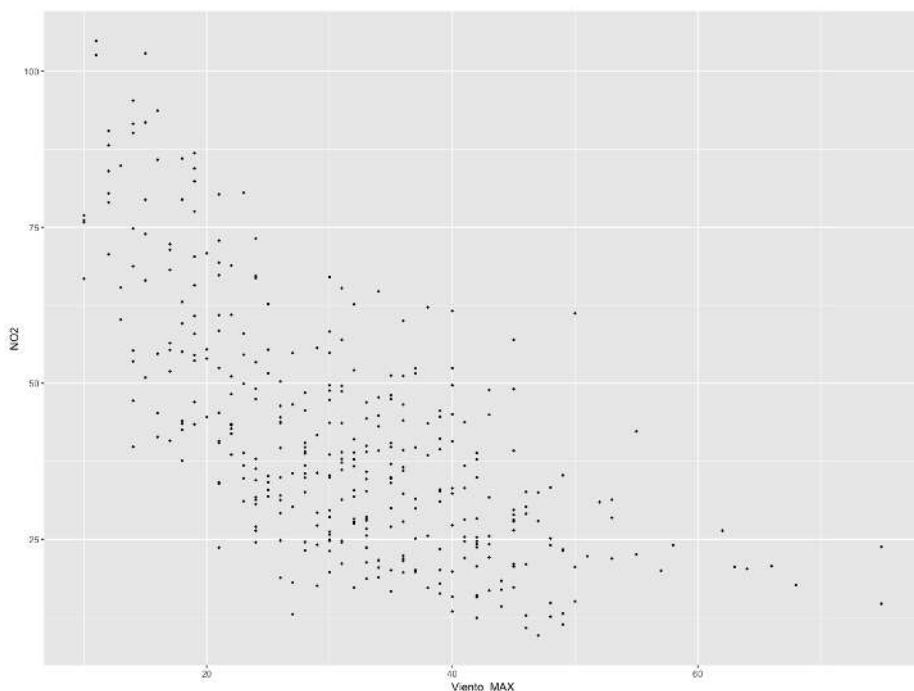


Fig. 5.8. Diagrama de dispersión entre el viento máximo y la concentración de NO2.

El procedimiento de regresión no lineal consiste en presentar una expresión posible que liga los atributos independientes con el dependiente con parámetros que deben ser determinados. Siguiendo con el ejemplo de encontrar un modelo para la concentración de NO2 en función de atributos meteorológicos, se va a proponer un ajuste polinómico y un ajuste exponencial.

```
fit_nls1 <- nls(NO2 ~ a*exp(-b * Viento_MAX) + c, dfMedidas, start = list(a = 100, b
= 0.01, c = 0.01))
fit_nls1

fit_nls2 <- nls(NO2 ~ a * Viento_MAX^2 + b * Viento_MAX + c, dfMedidas, start = list(a
= 0.01, b = 0.01, c = 0.01))
fit_nls2
```

```

Nonlinear regression model

model: NO2 ~ a * exp(-b * Viento_MAX) + c
data: dfMedidas
      a          b          c
137.61589   0.07417  22.60333
residual sum-of-squares: 62362

Number of iterations to convergence: 9
Achieved convergence tolerance: 5.942e-06

Nonlinear regression model

model: NO2 ~ a * Viento_MAX^2 + b * Viento_MAX + c
data: dfMedidas
      a          b          c
 0.02842  -3.07359 105.75524
residual sum-of-squares: 65355

Number of iterations to convergence: 2
Achieved convergence tolerance: 6.192e-09

```

Observando la suma del cuadrado de los residuos de los ajustes no lineales con el lineal obtenido anteriormente, se aprecia una notable mejora. En la siguiente gráfica se representan tanto el ajuste lineal como los ajustes no lineales.

```

ggplot(dfMedidas, aes(x = Viento_MAX, y = NO2)) + geom_point(size=0.5) +
  geom_line(aes(x = Viento_MAX, y = fit1$fitted.values), colour = "red") +
  geom_line(aes(x = Viento_MAX, y = predict(fit_nls1, Viento_MAX)), colour = "blue") +
  geom_line(aes(x = Viento_MAX, y = predict(fit_nls2, Viento_MAX)), colour = "green") +
  annotate("text", x=50, y=95, label= "\U2022 Lineal", colour = "red") +
  annotate("text", x=50, y=90, label= "\U2022 Exponencial", colour = "blue") +
  annotate("text", x=50, y=85, label= "\U2022 Polinómico", colour = "green")

```

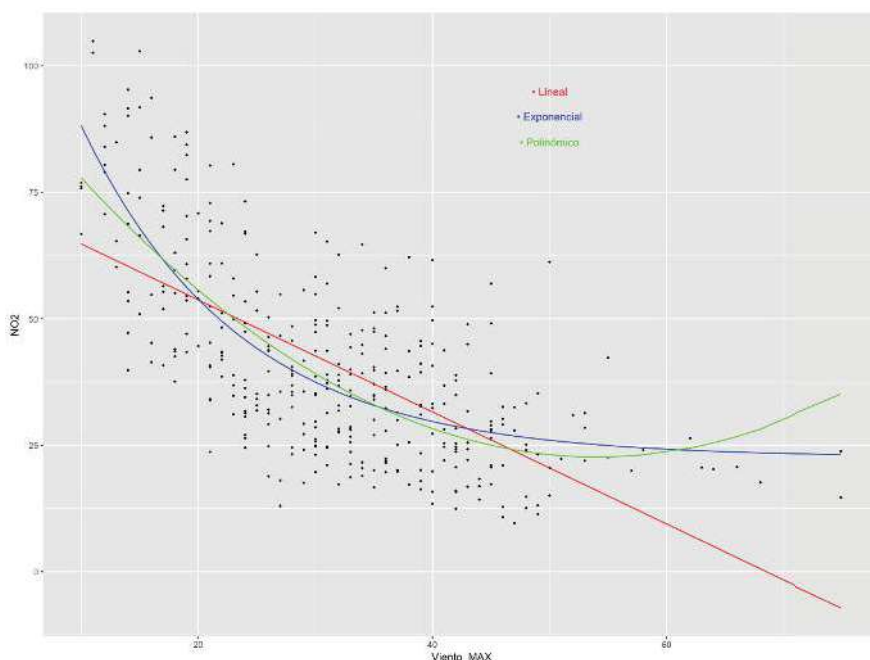


Fig. 5.9. Comparación de regresión lineal con exponencial y polinómica.

5.1.4 Regresión de atributos no continuos

Se ha realizado la regresión de un atributo dependiente continuo con atributos independientes que podían ser continuos o discretos. En estricto sentido, la regresión lineal sólo puede realizarse cuando se cumplen un conjunto de condiciones:

1. Los parámetros del modelo son numéricos y lineales.
2. La media de los residuos es cero (o muy cercana a cero).
3. Homocedasticidad. La varianza del error sobre las predicciones permanece constante.
4. Independencia de los errores (no existe autocorrelación de los errores).

5. No están correlados los residuos y los atributos independientes.
6. Multicolinealidad. No debe existir una correlación grande entre los atributos independientes.
7. La variabilidad de los atributos independientes debe ser positiva.
8. Normalidad de los residuos.

Veamos para el caso del modelo de regresión lineal de la concentración de NO₂ en función del Viento_MAX y la temperatura máxima, T_MAX, si se cumplen todas las condiciones:

1. Se cumple. En este caso, los dos atributos independientes son numéricos, cuando se amplió el modelo al atributo que consideraba si había presencia o no de lluvia, éste tenía que transformarse a variables “_dummy_”, como se mencionó anteriormente.
2. Se cumple, la media es prácticamente 0.

```
mean(fit2$residuals)
```

```
[1] -1.689471e-15
```

3. Observando los gráficos de la primera columna, el superior, cuanto más cercana esté la línea roja al eje de ordenadas más cumplirá la condición de homocedasticidad. En este caso, la línea roja se curva y se aprecia que hay diferencias entre los residuos de los valores ajustados. Es indicación de presencia de heterocedasticidad. Para confirmarlo o desecharlo hay varios test que permiten cuantificar la presencia de heterocedasticidad. Como los valores de p son inferiores al valor de significancia, 0.05, se confirma que existe heterocedasticidad. No se cumple la tercera condición.

```
par(mfrow=c(2,2))
plot(fit2)
library(lmtest)
library(car)
bptest(fit2) #test Breush-Pagan
ncvTest(fit2) #test NCV
```

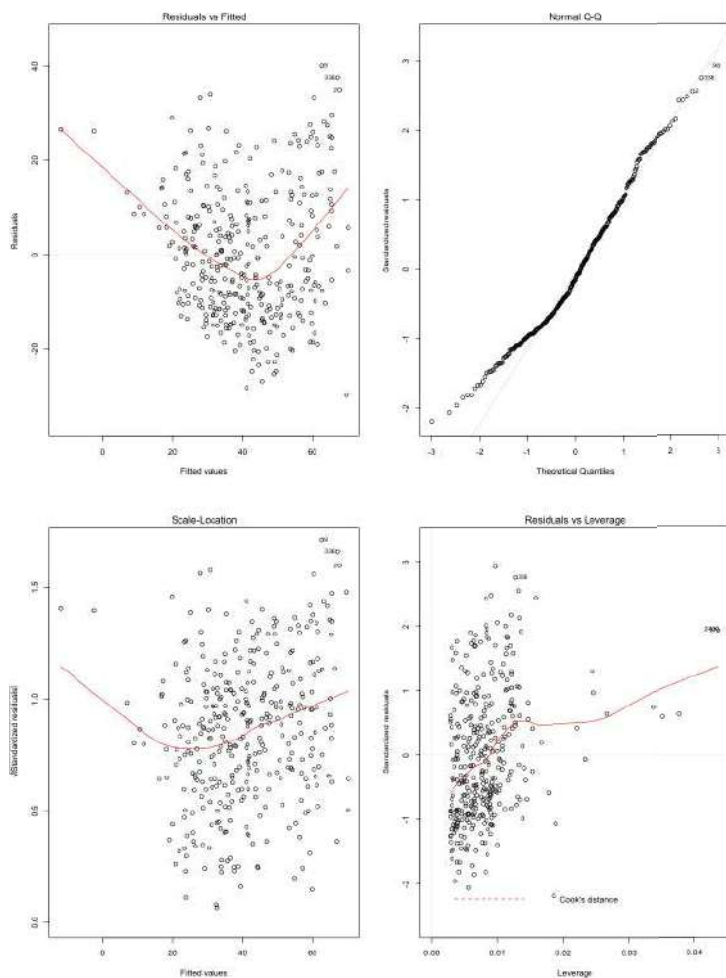


Fig. 5.10. Estructura del error del modelo de regresión lineal.

```
studentized Breusch-Pagan test
```

```
data: fit2
```

```
BP = 15.523, df = 2, p-value = 0.0004258
```

```
Non-constant Variance Score Test
```

```
Variance formula: ~ fitted.values
```

```
Chisquare = 12.45883    Df = 1    p = 0.0004160211
```

4. Hay diferentes formas de comprobar esta condición, gráficamente o aplicando test que midan la aleatoriedad de los residuos. Tanto el test de Durbin-Watson, donde p es prácticamente 0, como el gráfico muestran que sí hay autocorrelación. El gráfico muestra un patrón de dependencia del valor del residuo con valores previos. Cuando no existe correlación de un valor al siguiente, el segmento cae abruptamente a 0. Esto significa que el valor de la concentración de NO₂ de un día depende de los valores de días anteriores y el modelo lineal propuesto no contempla esta circunstancia. No cumple el cuarto criterio.

```
dwtest(fit2) #test Durbin-Watson
acf(fit2$residuals)
```

Durbin-Watson test

data: fit2

DW = 1.0089, p-value < 2.2e-16

alternative hypothesis: true autocorrelation is greater than 0

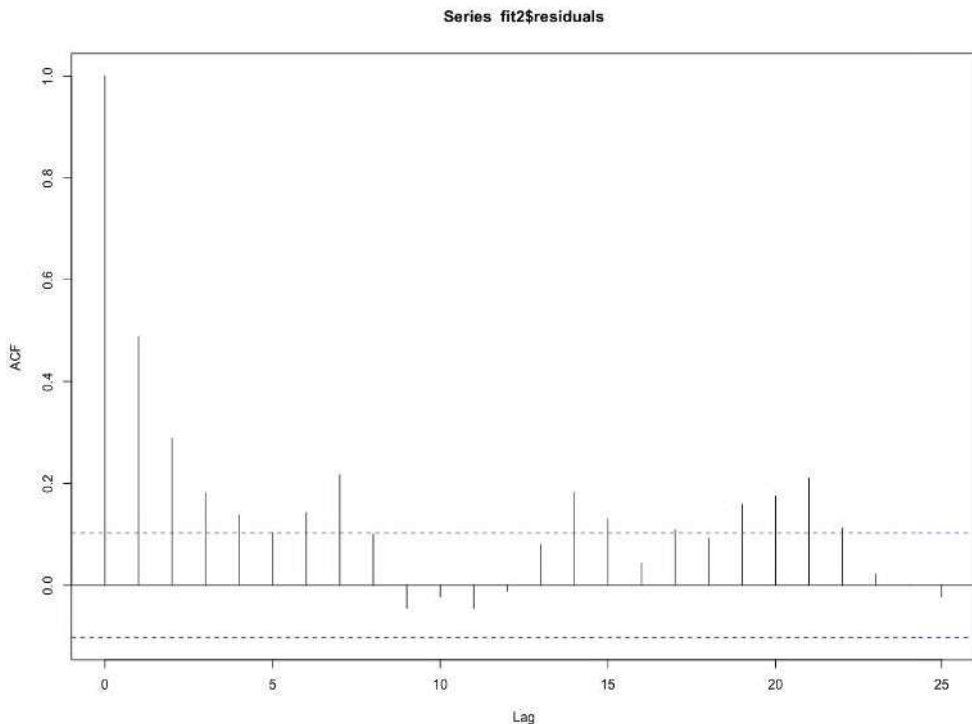


Fig. 5.11. Gráfica de la autocorrelación de los errores.

5. Se puede medir la correlación entre los atributos independientes y los residuos. Se aprecia que no existe correlación. Se cumple.

```
hetcor(Viento_MAX, T_MAX, fit2$residuals)
```

Two-Step Estimates

Correlations/Type of Correlation:

	Viento_MAX	T_MAX	fit2.residuals
Viento_MAX	1	Pearson	Pearson
T_MAX	0.1491	1	Pearson
fit2.residuals	-8.994e-16	1.015e-16	1

Standard Errors:

	Viento_MAX	T_MAX
Viento_MAX		
T_MAX	0.05124	
fit2.residuals	0.0524	0.0524

n = 365

P-values for Tests of Bivariate Normality:

	Viento_MAX	T_MAX
Viento_MAX		
T_MAX	3.992e-11	
fit2.residuals	0.003812	6.069e-07

6. Se puede aplicar el test del factor de inflación de la varianza (VIF). Se puede tomar como referencia que cuando VIF es igual a 1 no están correladas en absoluto, mayor de 1 y menor de 6 se considera una correlación moderada y mayor de 5 es una alta correlación. Los valores obtenidos están muy próximos a 1, puede asumirse que no existe multicolinealidad.

```
vif(fit2)
```

Viento_MAX	T_MAX
1.022739	1.022739

7. La variabilidad de los atributos independientes es mucho mayor que cero. Se cumple.

```
var(Viento_MAX)
var(T_MAX)
```

```
[1] 135.6846
[1] 81.26049
```

8. Volviendo a los gráficos del punto 3, en el superior derecho se puede observar el grado de normalidad de los residuos. Cuanto más distribuidos sobre la línea punteada, mayor es el grado de normalidad, se permite que los valores de los extremos se desvíen de la línea. También se pueden aplicar los test de normalidad vistos anteriormente. En este caso se cumple.

```
ad.test(fit2$residuals)
cvm.test(fit2$residuals)
lillie.test(fit2$residuals)
```

```
Anderson-Darling normality test
data:  fit2$residuals
A = 2.9235, p-value = 2.331e-07

Cramer-von Mises normality test
data:  fit2$residuals
W = 0.47821, p-value = 4.357e-06

Lilliefors (Kolmogorov-Smirnov) normality test
data:  fit2$residuals
D = 0.084399, p-value = 1.465e-06
```

En R es muy habitual que cuando hay que aplicar un conjunto de test y métodos de forma repetitiva se realicen funciones que en un solo paso los integran. En este caso, la comprobación de las condiciones para la realización de un modelo de regresión lineal puede aplicarse en un paso con la función *gvmlm* de la librería de igual nombre.

```
library(gvlma)
gvlma(fit2)
```

```
Call:
lm(formula = NO2 ~ Viento_MAX + T_MAX, data = dfMedidas)
```

Coefficients:

(Intercept)	Viento_MAX	T_MAX
86.1814	-1.0428	-0.5581

```
ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance = 0.05
```

```
Call:
gvlma(x = fit2)
```

	Value	p-value	Decision
Global Stat	65.815	1.733e-13	Assumptions NOT satisfied!
Skewness	13.010	3.098e-04	Assumptions NOT satisfied!
Kurtosis	2.191	1.388e-01	Assumptions acceptable.
Link Function	47.955	4.360e-12	Assumptions NOT satisfied!
Heteroscedasticity	2.658	1.030e-01	Assumptions acceptable.

Se presenta lo que anteriormente se había calculado paso a paso, apreciándose que hay condiciones que no se cumplen. Se pueden emprender modificaciones sobre los datos, eliminación de *outliers*, transformación de Box-Cox, etc. que podrían llevar al cumplimiento de las condiciones y darían validez completa al modelo lineal.

Hay otros tipos de modelos de regresión aparte del lineal, como la utilización de *splines*, que son polinomios lineales o de orden superior definidos en intervalos, o la aplicación de diferentes métodos de regularización, como *ridge regression*, Lasso, *elastic net*, etc. No van a ser tratados aquí, pero R dispone de librerías; siguiendo el mismo tipo de proceder que con los modelos lineales, pueden ser fácilmente aplicados.

Falta abordar una cuestión importante. ¿Se puede realizar un modelo de regresión de atributos dependientes que no sean continuos?

5.1.5 Modelos lineales generalizados

La respuesta a la pregunta anterior es afirmativa. La regresión lineal se incluye en un marco conceptual llamado *modelo lineal generalizado*, GLM por sus siglas en inglés, que permite la obtención de modelos con atributos dependientes que tienen distribuciones de errores diferentes de la normal. Se pueden aplicar tanto para atributos con valores binarios en una regresión logística como para discretos con una regresión de Poisson. Se recupera la regresión lineal cuando el modelo de error es gaussiano.

```
fit_glm <- glm(NO2 ~ Viento_MAX + T_MAX, data = dfMedidas, family = gaussian)
fit_glm
```

```
Call:  glm(formula = NO2 ~ Viento_MAX + T_MAX, family = gaussian, data = dfMedidas)

Coefficients:
(Intercept)  Viento_MAX      T_MAX
      86.1814      -1.0428     -0.5581

Degrees of Freedom: 364 Total (i.e. Null);  362 Residual
Null Deviance:      138100
Residual Deviance: 68520 AIC: 2955
```

También se puede utilizar la regresión progresiva con *glm*.

```
step(fit_glm)
```

```
Start:  AIC=2954.59
NO2 ~ Viento_MAX + T_MAX
```

		Df	Deviance	AIC
<none>			68518	2954.6
- T_MAX	1		77526	2997.7
- Viento_MAX	1	121035	3160.3	

```
Call:  glm(formula = NO2 ~ Viento_MAX + T_MAX, family = gaussian, data = dfMedidas)

Coefficients:
(Intercept)  Viento_MAX      T_MAX
      86.1814      -1.0428     -0.5581

Degrees of Freedom: 364 Total (i.e. Null);  362 Residual
Null Deviance:      138100
Residual Deviance: 68520 AIC: 2955
```

Se pueden introducir transformaciones al atributo dependiente a través del parámetro *link*.

```
fit_glm2 <- glm(NO2 ~ Viento_MAX + T_MAX, data = dfMedidas, family = gaussian(link = "log"))
fit_glm2
```

```
Call:  glm(formula = NO2 ~ Viento_MAX + T_MAX, family = gaussian(link = "log"),
  data = dfMedidas)

Coefficients:
(Intercept)  Viento_MAX      T_MAX
  4.82665      -0.02883      -0.01248

Degrees of Freedom: 364 Total (i.e. Null);  362 Residual
Null Deviance:      138100
Residual Deviance: 60380 AIC: 2908
```

Que sería equivalente a calcular $\log(NO2) = a \text{ Viento_MAX} + b \text{ T_MAX}$.

Trataremos ahora de predecir cuándo superan un umbral peligroso los valores de la concentración de partículas en suspensión, PM2.5_UMBRAI. El atributo se modela con la función logística.

```
fit_log <- glm(PM2.5_UMBRAI ~ Viento_MAX + T_MAX, data = dfMedidas, family = binomial)
summary(fit_log)
```

```
Call:
glm(formula = PM2.5_UMBRAI ~ Viento_MAX + T_MAX, family = binomial,
  data = dfMedidas)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1514  -0.2462  -0.1076  -0.0495   4.0249

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.35042    0.96777   2.429  0.0152 *
Viento_MAX  -0.18655    0.04754  -3.924  8.7e-05 ***
T_MAX       -0.05463    0.04888  -1.118  0.2637
---

```

```

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 143.44  on 364  degrees of freedom
Residual deviance: 100.20  on 362  degrees of freedom
AIC: 106.2

Number of Fisher Scoring iterations: 8

```

Los términos en los que se trata la regresión logística son los mismos que los de un problema de clasificación y una de las formas de evaluar el resultado es mediante la matriz de confusión. Se establece el punto de corte en función de si se desea mejorar la predicción de una clase u otra o un término medio. En este caso se prima la predicción de que sea superado el umbral.

```

library(InformationValue)

corteOptimo <- optimalCutoff(as.numeric(PM2.5_UMBRALE)-1, fit_log$fitted.values, optimizeFor = "Ones")[1]

confusionMatrix(as.numeric(PM2.5_UMBRALE)-1, fit_log$fitted.values, threshold = corteOptimo)

```

0	1	
0	295	1
1	52	17

```

Call:

```

En este caso, la clasificación obtenida muestra en la matriz de confusión que sólo hay un caso en el que se superó el umbral, que es incorrectamente clasificado. Con una aceptable tasa de falsos positivos.

Una forma muy conveniente para determinar la bondad de un clasificador es utilizar la curva ROC. Es una gráfica en la que se representa la sensibilidad frente a la especificidad en función del umbral de discriminación. Un clasificador perfecto es el que tiene una especificidad y sensibilidad del 100 %, valor (0,1). Un clasificador aleatorio, el peor resultado posible, estaría representado por la recta que une los puntos (0,0) a (1,1). Se pueden equiparar los resultados de diferentes modelos predictivos binarios comparando sus curvas ROC, el clasificador cuya área bajo la curva ROC sea mayor será el mejor clasificador. Se puede tomar como referencia que un valor para la AUROC (área bajo la curva ROC) en el intervalo [0.5, 0.6) es malo, [0.6, 0.75) regular, [0.75, 0.9) bueno, [0.9, 0.97) muy bueno y [0.97, 1) excelente.

```
plotROC(as.numeric(PM2.5_UMBRAL)-1, fit_log$fitted.values)
```

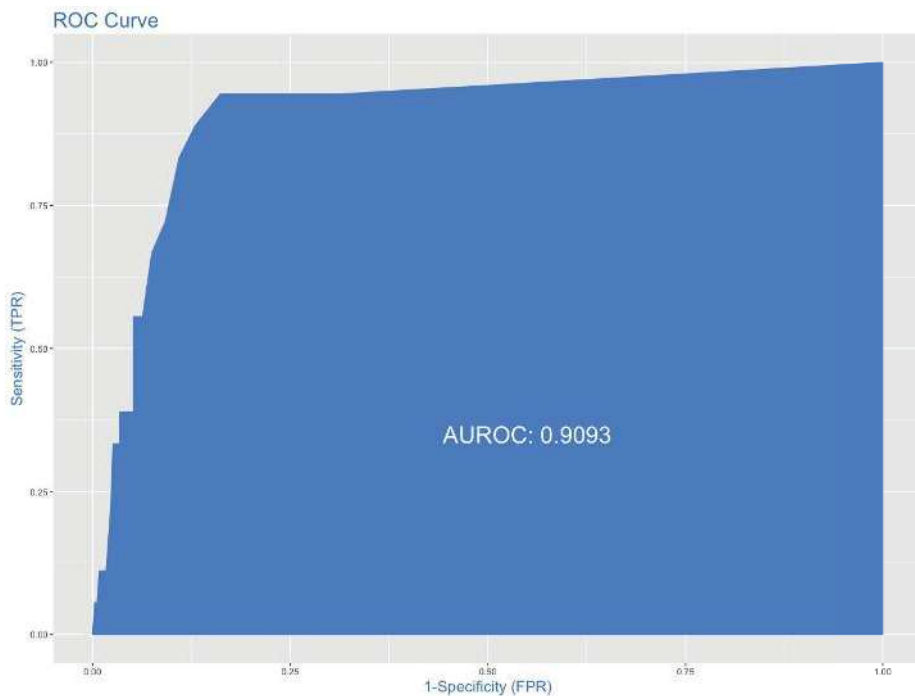


Fig. 5.12. Curva ROC.

En este caso, el clasificador que se ha obtenido es muy bueno. Otras medidas de bondad que pueden obtenerse son:

```
Concordance(as.numeric(PM2.5_UMBRAL)-1, fit_log$fitted.values) #Concordancia
precision(as.numeric(PM2.5_UMBRAL)-1, fit_log$fitted.values, threshold = corteOptimo)
#Precisión
npv(as.numeric(PM2.5_UMBRAL)-1, fit_log$fitted.values, threshold = corteOptimo) #
sensitivity(as.numeric(PM2.5_UMBRAL)-1, fit_log$fitted.values, threshold = corteOptimo)
#Sensitividad
specificity(as.numeric(PM2.5_UMBRAL)-1, fit_log$fitted.values, threshold = corteOptimo)
#Especificidad
```

```

$Concordance
[1] 0.8940122

$Discordance
[1] 0.1059878

$Tied
[1] 2.775558e-17

$Pairs
[1] 6246

[1] 0.2463768
[1] 0.9966216
[1] 0.9444444
[1] 0.8501441

```

Cuando el atributo dependiente es de tipo contable, la distribución asociada es la de Poisson. Se va a utilizar para predecir la concentración máxima de partículas en suspensión PM2.5_MAX.

```

fit_poi <- glm(PM2.5_MAX ~ Viento_MAX + T_MAX + Lluvia_SN, data = dfMedidas, family = poisson)
summary(fit_poi)

```

```

Call:
glm(formula = PM2.5_MAX ~ Viento_MAX + T_MAX + Lluvia_SN, family = poisson,
    data = dfMedidas)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3666  -1.0777  -0.3078   0.6477   5.4052

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   3.352995   0.046869  71.540  <2e-16 ***
Viento_MAX    -0.024629   0.001422 -17.326  <2e-16 ***
T_MAX          0.001673   0.001687   0.992   0.3211
Lluvia_SNSi   -0.102114   0.041619  -2.454   0.0141 *
---

```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for poisson family taken to be 1)
```

```
Null deviance: 1158.06 on 364 degrees of freedom
```

```
Residual deviance: 756.12 on 361 degrees of freedom
```

```
AIC: 2361.1
```

```
Number of Fisher Scoring iterations: 4
```

Podemos proponer mejorar el modelo determinando si la eliminación de un atributo independiente mejora la predicción. Para hacer esto utilizamos *drop1*.

```
drop1(fit_poi, test = "F")
```

```
Single term deletions
```

```
Model:
```

```
PM2.5_MAX ~ Viento_MAX + T_MAX + Lluvia_SN
```

	Df	Deviance	AIC	F value	Pr(>F)
<none>		756.12	2361.1		
Viento_MAX	1	1077.57	2680.5	153.4711	<2e-16 ***
T_MAX	1	757.10	2360.1	0.4698	0.4935
Lluvia_SN	1	762.26	2365.2	2.9301	0.0878 .

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Observando la columna de AIC, se aprecia que el modelo (con AIC 2361.1) mejoraría su AIC eliminando el atributo T_MAX, pasaría a ser 2360.1, para el resto de atributos empeoraría el modelo si se eliminan.

Por tanto, el nuevo modelo sin T_MAX:

```
fit_poi2 <- glm(PM2.5_MAX ~ Viento_MAX + Lluvia_SN, data = dfMedidas, family = poisson)
summary(fit_poi2)
```

```

Call:
glm(formula = PM2.5_MAX ~ Viento_MAX + Lluvia_SN, family = poisson,
     data = dfMedidas)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3574  -1.0695  -0.2847   0.6417   5.4496

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   3.377773    0.039549   85.408 < 2e-16 ***
Viento_MAX    -0.024170    0.001339  -18.054 < 2e-16 ***
Lluvia_SNSi   -0.111715    0.040486   -2.759  0.00579 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1158.1  on 364  degrees of freedom
Residual deviance:  757.1  on 362  degrees of freedom
AIC: 2360.1

Number of Fisher Scoring iterations: 4

```

En los ejemplos que se han desarrollado en esta sección, se han utilizado todos los datos para encontrar los modelos de regresión. Esto es válido para la regresión lineal, pero no para el resto. La forma de proceder adecuada, como ya ha sido mencionado, habría sido la de dividir el conjunto de datos en entrenamiento y validación y comprobar la bondad del ajuste sobre estos últimos, repitiendo el proceso para diferentes conjuntos de entrenamiento y validación y eligiendo el mejor ajuste. Ésta es la manera adecuada de tratar problemas de aprendizaje y se aplicará en la siguiente sección.

5.2 Algoritmos de clasificación

Los algoritmos de regresión que ya han sido vistos en las anteriores secciones pueden entenderse como técnicas de clasificación supervisada. Hay un atributo que debe ser predicho, atributo dependiente, a partir de otros atributos independientes. En los algoritmos de regresión, el modelo debe ser propuesto y se obtienen automáticamente los coeficientes del ajuste. Las técnicas de clasificación que se van a presentar en esta

sección construyen modelos sin asumir modelos previos, con muy pocos parámetros de ajuste.

Se ha mencionado en la sección dedicada a la regresión logística que el análisis de sus resultados es el mismo que se realiza para los problemas de clasificación. Esto es, un problema de clasificación trata de asignar una categoría a un conjunto de datos a partir de la información que puede extraerse de las categorías asignadas previamente. En el caso de la regresión logística, la salida es binaria y, por tanto, puede entenderse que son dos clases. Para problemas en los que hay más de dos clases se extiende el modelo binomial a la regresión logística multinomial.

Se va a crear un nuevo atributo nominal, `O3_Nivel`, a partir de la concentración máxima de `O3` que divide ésta en tres niveles de peligrosidad: verde, amarillo y rojo. Estos niveles se han tomado arbitrariamente y no tienen rigor en el sentido de las ciencias ambientales, únicamente tienen interés didáctico.

```
dfMedidas$O3_Nivel <- cut(dfMedidas$O3_MAX, c(min(dfMedidas$O3_MAX), 75, 100,
max(dfMedidas$O3_MAX)), include.lowest = TRUE, right = FALSE, labels = c("V", "A", "R"))
summary(dfMedidas$O3_Nivel)
```

```
V    A    R
208 131  26
```

El objetivo va a ser encontrar un modelo que permita etiquetar de forma correcta el nivel de alerta de `O3_MAX` para nuevas situaciones futuras. El primer paso va a consistir en eliminar del conjunto de datos aquellos que pueden considerarse anomalías, valores atípicos, fruto de errores de medida o singularidades estadísticas que desvirtúan el comportamiento general, los llamados *outliers*, valores atípicos o anómalos.

5.2.1 Detección de valores atípicos

El primer y más simple criterio es aportar la aproximación univariada. Consiste en tratar aquellos valores que están fuera una vez y media del rango intercuartil (diferencia entre el tercer y segundo cuartil) como atípicos leves. Se representan en los *boxplots* como puntos. Los atípicos severos son los que tienen valores por debajo del primer cuartil o por encima del tercero, tres veces el rango intercuartil.

Si nos fijamos en la concentración de NO y de O3_MAX:

```
boxplot(NO, O3_MAX, names = c("NO", expression(O[3])))
```

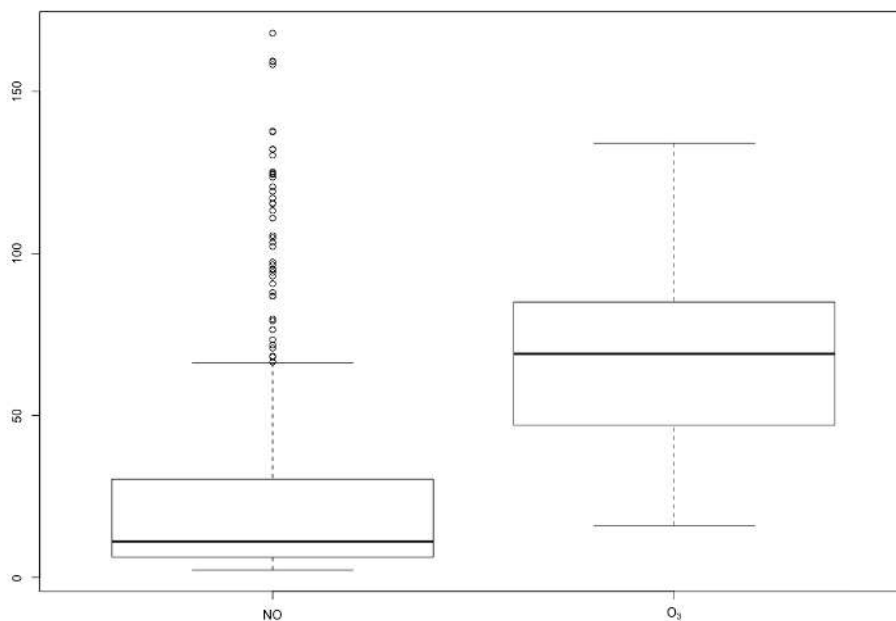


Fig. 5.13. Diagrama de caja para las concentraciones de NO y O3.

Se aprecia que la concentración de O3_MAX no tiene valores atípicos leves, sin embargo, la concentración de NO tiene bastantes (aproximadamente, valores superiores a 70). Extendiendo tres veces el rango intercuartil se muestran los valores atípicos extremos (superiores a 100).

```
boxplot(NO, xlab = "NO", range = 3)
```

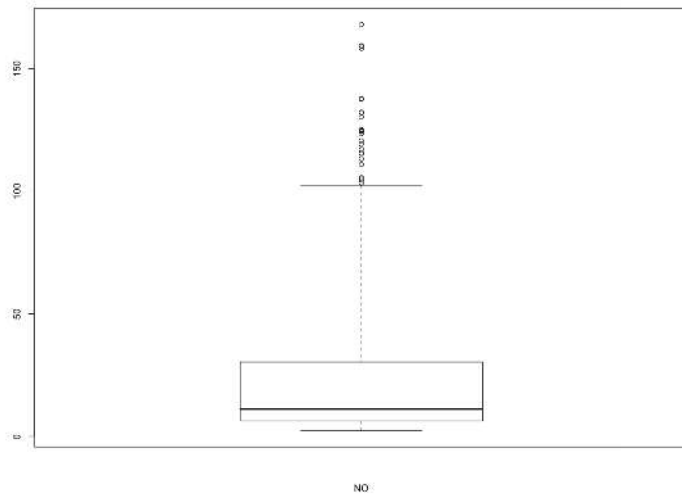


Fig. 5.14. Diagrama de caja para la concentración de NO con el rango intercuartil extendido.

Esta caracterización (test de Tukey) considera anómalos aquellos valores muy alejados de las frecuencias centrales y debe ser tomada con mucha precaución. Si observamos los valores atípicos de la precipitación de lluvia:

```
boxplot(Lluvia, xlab = "Lluvia")
```

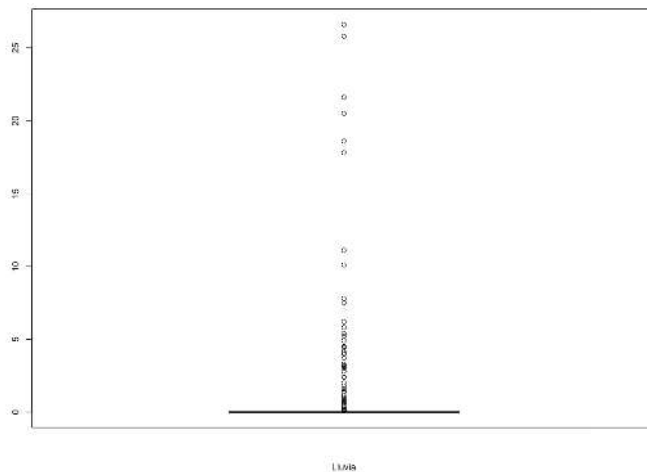


Fig. 5.15. Diagrama de caja para la cantidad de lluvia.

¡Todos los valores distintos de 0 son atípicos! Claramente sería un error descartar los días de lluvia.

Una forma indirecta de determinar puntos atípicos es encontrar puntos influyentes o puntos de apalancamiento (*leverage points*). Un punto influyente es aquel que modifica sensiblemente un modelo de regresión. Los puntos influyentes no tienen que ser necesariamente atípicos, lo que se indica es que deben ser analizados por si lo fueran. Hay diferentes estadísticos para encontrar estos puntos: distancia de Cook, medida de Hadi, DFFITS y DFBETAS (estandarizadas y sin estandarizar), residuos estandarizados y estudentizados, etc.

La medida de distancia de Cook calcula la importancia de cada valor sobre una regresión lineal. Por tanto, ya no se considera la distribución aislada de un atributo, si no la relación que tiene con otros en la relación lineal.

```
fit_lm <- lm(O3_MAX ~ Viento_MAX + T_MAX, data = dfMedidas)
cook_lm <- cooks.distance(fit_lm)

plot(cook_lm, pch = "o", cex = 0.5, main = "Distancia de Cook")
abline(h = 4*mean(cook_lm), col = "red") # línea de corte
text(x = 1 : length(cook_lm) + 1, y = cook_lm - 0.005, cex = 0.75,
      labels = ifelse(cook_lm > 4 * mean(cook_lm), paste0(names(cook_
lm), "|", round(dfMedidas$O3_MAX[as.numeric(names(cook_lm))], 0)), " "), col="red")
```

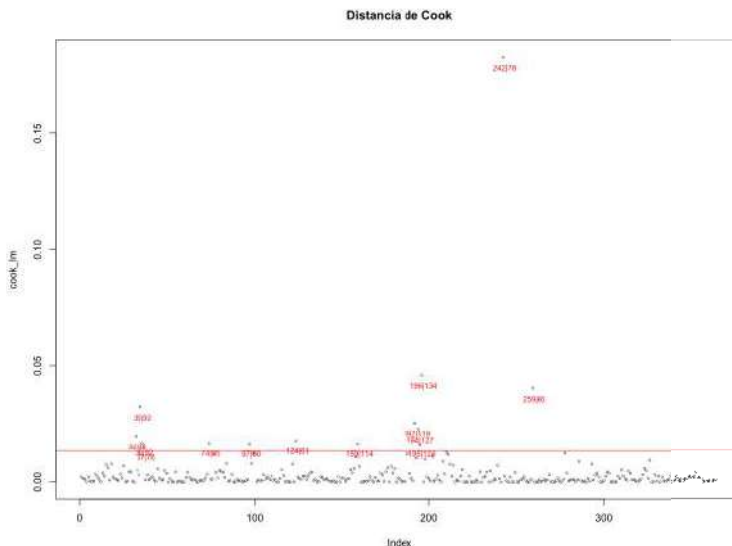


Fig. 5.16. Distancia de Cook para la determinación de datos atípicos.

En el gráfico anterior se muestran, con una etiqueta roja, el índice de valor y la concentración de O3_MAX que están cuatro veces el valor medio de la distancia de Cook. Es muy destacable que el valor con 78 no tiene un valor extremo de concentración, pero, sin embargo, en distancia de Cook destaca muy apreciablemente. Es un claro candidato, junto con los que tienen valor 134 y 66, para ser valores atípicos, que deberían ser eliminados del conjunto de datos con el que se va a entrenar el clasificador.

La librería *olsrr* incluye la medida Cook y la representación en un gráfico en una sola función. En el gráfico, se muestra en las etiquetas el índice del valor.

```
library(olsrr)
out <- ols_cooksd_barplot(fit_lm)
```

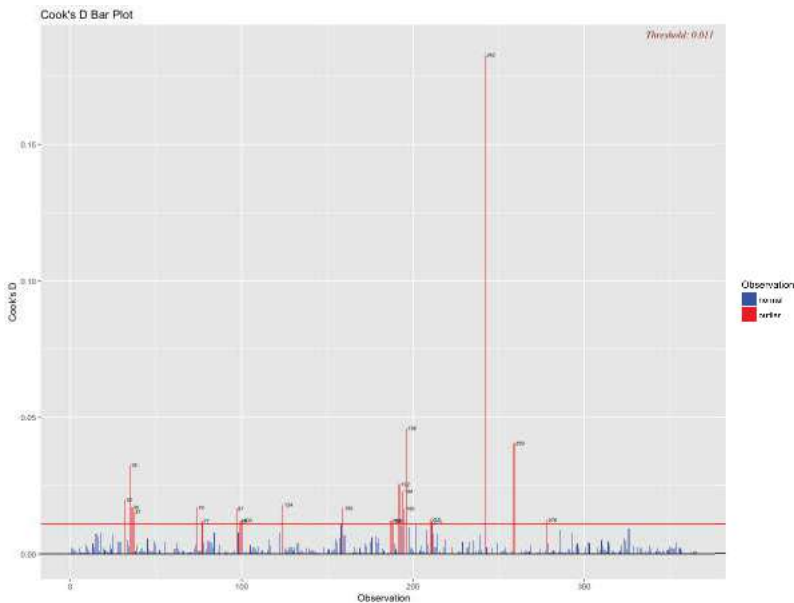


Fig. 5.17. Distancia de Cook con etiquetas para los datos atípicos.

Además del gráfico, en el atributo `out$outliers`, se encuentra una lista con el índice del elemento propuesto como atípico y el valor de la distancia Cook.

Otras medidas sobre la influencia que realiza cada valor sobre un ajuste lineal son DFBETA y DFFIT. DFBETA mide, para cada valor, su impacto en los coeficientes de la regresión lineal. En el gráfico siguiente se representa el valor de DFBETA para todos los atributos de la regresión lineal y se aprecia que algunos valores son atípicos en todos los coeficientes, pero otros lo son sólo en uno de ellos y no para el resto.

```
ols_dfbetas_panel(fit_lm)
```

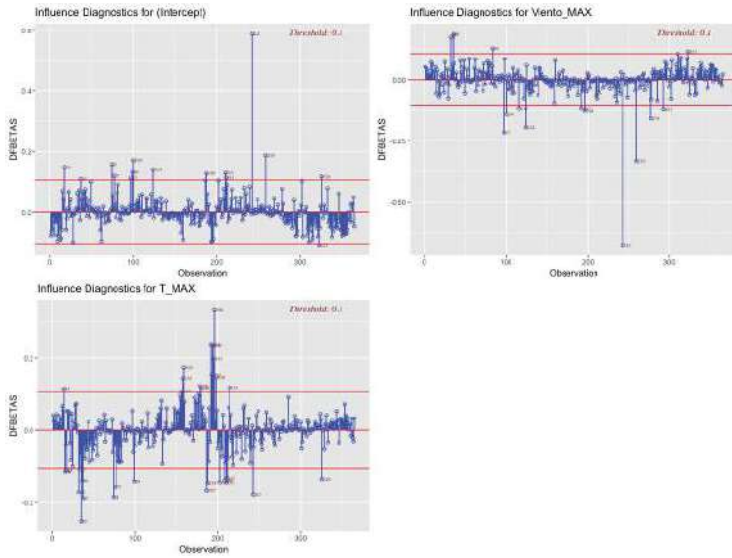


Fig. 5.18. Medidas de DFBETAS para determinar datos atípicos.

DFFIT mide cuánto cambia el ajuste sobre la variable dependiente, con la presencia o ausencia de un valor.

```
ols_dffits_plot(fit_lm)
```

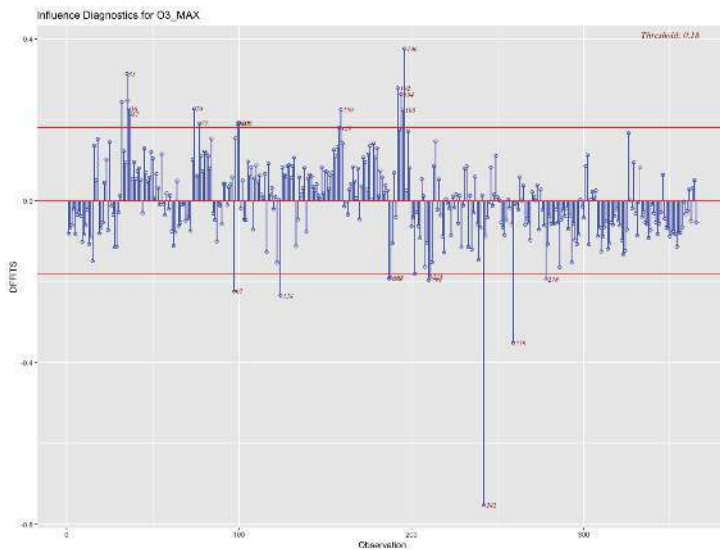


Fig. 5.19. Medidas de DFFITS para determinar datos atípicos.

El gráfico con los residuos eliminados estudentizados mide la diferencia entre los residuos cuando está o no presente un dato en la regresión. Este tipo de gráfico es muy eficiente para determinar valores atípicos.

```
ols_srstd_plot(fit_lm)
```

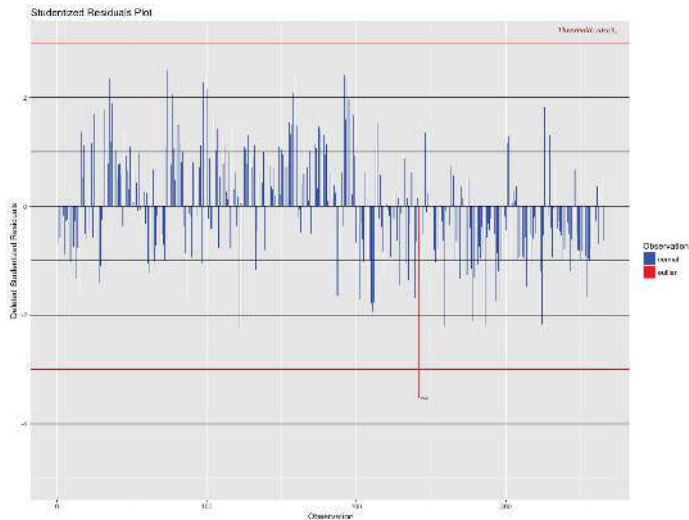


Fig. 5.20. Gráfico con los residuos eliminados estudentizados para determinar datos atípicos.

Se muestra claramente que el valor con índice 242 es candidato a ser atípico. Se recomienda usar el valor estudentizado en lugar del estandarizado.

```
ols_srstd_chart(fit_lm)
```

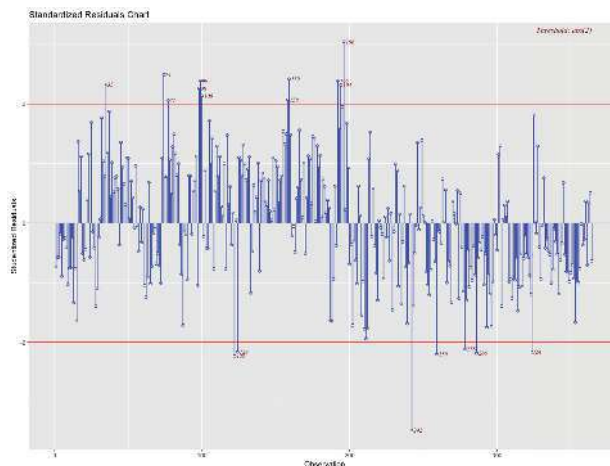


Fig. 5.21. Gráfico con los residuos estudentizados para determinar datos atípicos.

Una representación que permite aclarar si determinados datos son puntos de apalancamiento, atípicos o ambos, es el gráfico que muestra el nivel de apalancamiento frente a los residuos estudentizados.

```
ols_rsdlev_plot(fit_lm)
```

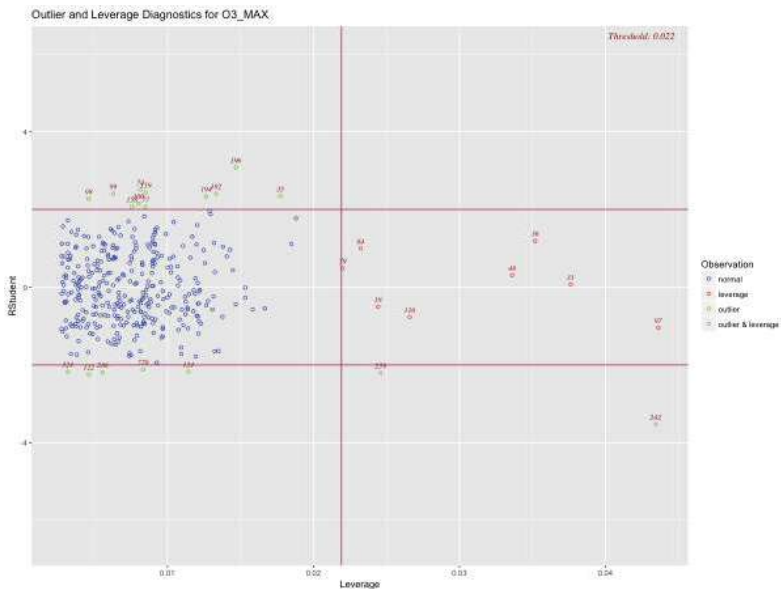


Fig. 5.22. Gráfico con el nivel de apalancamiento para determinar datos atípicos.

En verde, se representan potenciales valores atípicos pero que no tienen un gran impacto en el resultado de la regresión. En rojo, puntos de apalancamiento, tienen gran efecto sobre el resultado de la regresión, pero son considerados dentro de la normalidad. Finalmente en magenta, puntos atípicos pero que sí tienen gran impacto sobre el resultado de la regresión. Estos últimos valores deben ser examinados con detenimiento para tomar la decisión de si deben ser o no eliminados.

```
ols_dsrvsp_plot(fit_lm)
ols_hadi_plot(fit_lm)
ols_potrsd_plot(fit_lm)
```

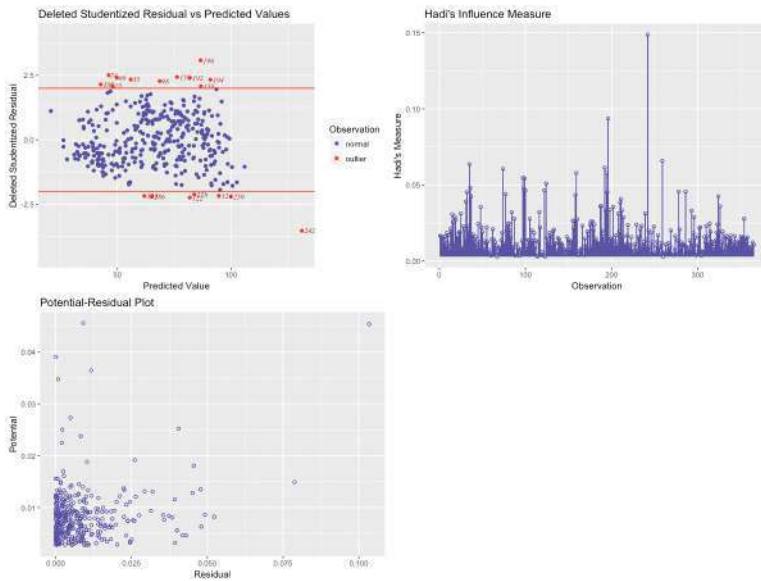


Fig. 5.23. Otras medidas para detectar datos atípicos.

A la vista de todos estos gráficos, el valor con índice 242 es un valor con gran influencia en el modelo de regresión lineal y que, además, su residuo difiere significativamente del resto. Observemos sus valores:

```
dfMedidas[242,c("Viento_MAX", "T_MAX", "O3_MAX")]
fit_lm$fitted.values[242]
fit_lm$residuals[242]
```

```
Viento_MAX T_MAX O3_MAX
242      75    35.5    78
242
131.2359
242
-53.23595
```

El dato con índice 242 corresponde al 30 de agosto, la temperatura máxima fue muy alta y con rachas de viento también altas, el nivel máximo de ozono fue aceptable. ¿Por qué este valor resulta atípico? La predicción de concentración de ozono máximo que arroja el modelo de correlación lineal es de 131, es decir, muy alto, hay una diferencia de 53 sobre el valor real. El motivo es que si vemos el dato con el valor de ozono máximo, el correspondiente al 15 de julio, se aprecia que la temperatura fue muy alta,

mayor que la del 30 de agosto. Esto es coherente con la correlación positiva que existe entre la temperatura máxima y el ozono. Mirando la racha de viento máxima, la de julio fue bastante menor que la de agosto, y también la correlación entre viento y ozono es positiva, de modo que el modelo predice un valor alto para el ozono. Es decir, no parece coherente un valor bajo de concentración con esa temperatura y ese viento. La lógica del modelo sabemos que es errónea, porque, por el conocimiento que tenemos del problema, el viento fuerte dispersaría el ozono. Por tanto, no se debe eliminar ese dato que el análisis nos dice que es atípico.

Con este ejemplo se quiere mostrar el proceso habitual que hay que realizar con los datos. Es un error tratar de aplicar las técnicas como una caja negra. Hay que hacer modelos, obtener estadísticos, analizar los resultados con conocimiento del dominio del problema y volver a empezar. Cuando se realiza una fase, nunca se debe pensar que es una etapa cubierta, el análisis de datos es un proceso iterativo que finaliza cuando las hipótesis de partida, los resultados, el conocimiento experto y las observaciones del mundo real alcanzan un equilibrio.

5.2.2 LDA, *Linear Discriminant Analysis*

El primer paso consistirá en dividir el conjunto de datos en uno para entrenamiento y otro para validación. Este paso es común para todos los algoritmos de clasificación que se van a utilizar a continuación. Se va a usar la función `createDataPartition`, de la librería `caret`, con el fin de asegurar que el conjunto de entrenamiento y el de validación tienen las mismas propiedades estadísticas.

```
library(caret)
set.seed(1234)

entrena      <- createDataPartition(dfMedidas$O3_Nivel, p=0.6, list=FALSE)
setEntrena   <- dfMedidas[entrena,] #el conjunto de entrenamiento
setValida    <- dfMedidas[-entrena,] #el conjunto de validación
```

El análisis discriminante lineal es bastante similar a PCA. PCA puede reducir el número de atributos creando otros nuevos como combinación lineal de los atributos independientes. Pero, a diferencia de éste, que no necesitaba definir un atributo a predecir, LDA sí requiere el atributo dependiente. En la implementación de LDA de la librería MASS, se contempla la posibilidad de definir los índices de las instancias que se usarán para entrenamiento. Por mantener una estructura similar para las llamadas a los algoritmos de clasificación no se va a utilizar y se entrenará con el conjunto de entrenamiento.

Comenzamos con la aplicación de LDA para la reducción de la dimensionalidad.

```
library(MASS)

res.lda <- lda(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED, data = setEntrena)

plot(res.lda,
      panel = function(x, y, ...) {
        points(x, y, cex = 2, ...)
        #text(x+0.2,y+0.2,labels=seq_along(x),...)
      },
      col = as.integer(df$O3_Nivel), pch = 20)
```

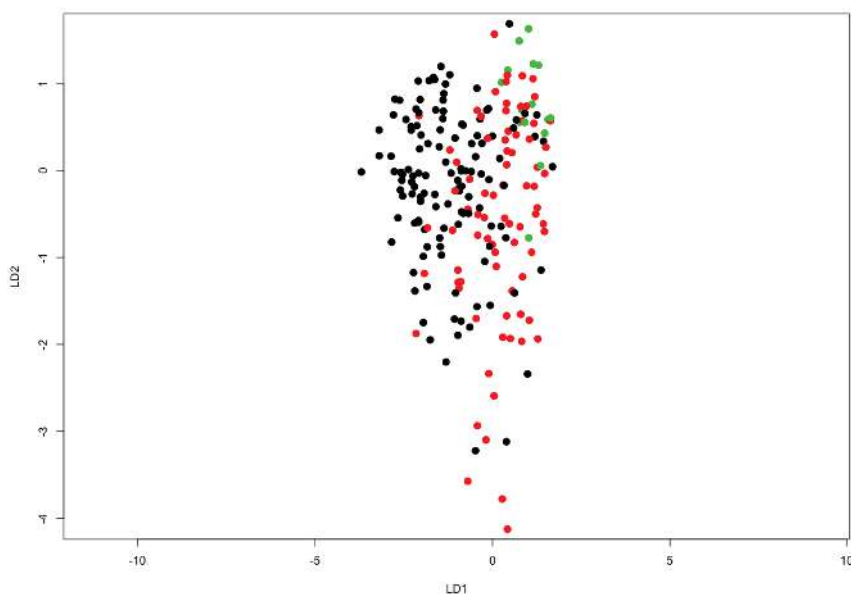


Fig. 5.24. Agrupamiento de las clases aplicando un LDA.

Se aprecia que las clases están bastante mezcladas, no puede establecerse una frontera clara entre ellas.

```
plot(T_MIN, Viento_MAX, pch = 20, col = as.integer(O3_Nivel))
points(res.lda$means[,c(4,5)], pch = 4, lwd = 2, cex = 4, col = 142)
```

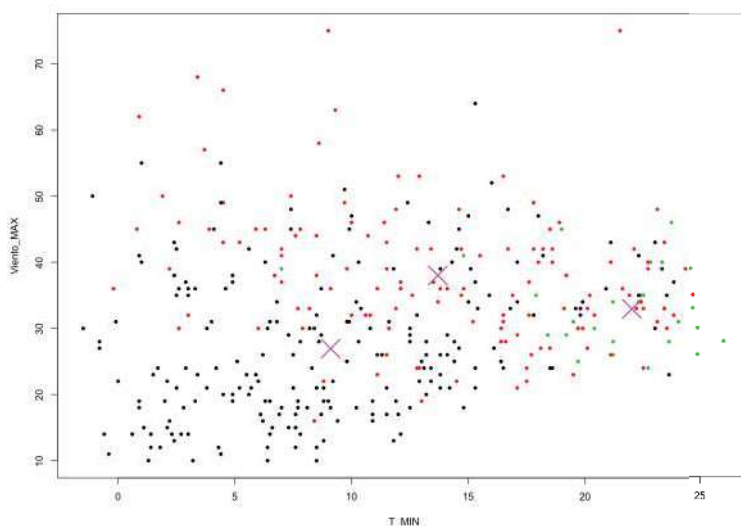


Fig. 5.25. Diagrama de dispersión con las clases en código de color y el centroide de cada una de ellas representado con un aspa.

En este gráfico se muestra, para dos atributos, los valores con las clases en código de color y mediante aspas el valor central de cada una de las tres clases.

El modelo LDA, como los modelos de regresión, es una combinación lineal de los atributos.

```
res.lda$scaling
```

	LD1	LD2
Lluvia_SNSi	0.136473469	-0.23171096
Lluvia	-0.003835861	-0.02755869
T_MAX	0.146891399	-0.03802910
T_MIN	-0.055576858	0.15111071
Viento_MAX	0.033611096	-0.06908118
Viento_MED	0.056233388	-0.00853981

En la gráfica siguiente se muestra la predicción con la primera función discriminante para los datos. Se aprecia que no hay una separación lineal entre ellas.

```
predice.lda <- predict(res.lda, dfMedidas)
plot(predice.lda$x[,1], col = as.integer(O3_Nivel), pch = 20)
```

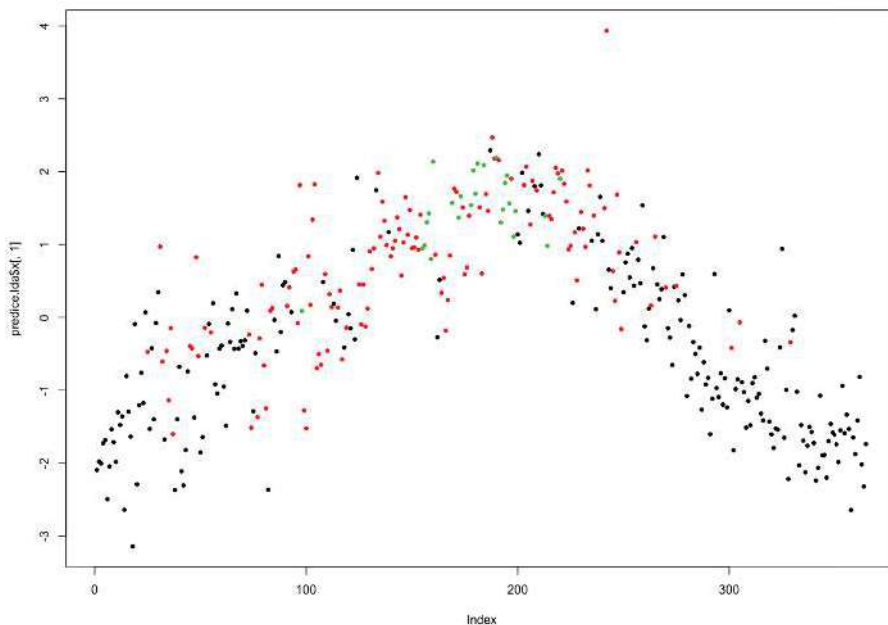


Fig. 5.26. Predicción de valores con LDA.

Abordemos ahora el problema de clasificación con LDA, para ello, haremos uso de los conjuntos de entrenamiento y validación.

```
modelo.lda <- lda(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED, data = setEntrena)
```

Calculemos ahora la predicción del modelo sobre el conjunto de validación y obtengamos la matriz de confusión y las medidas de mérito.

```
predice.modelo.lda <- predict(modelo.lda, setValida)
c <- confusionMatrix(predice.modelo.lda$class, setValida$O3_Nivel)
print(c)

compara_ACC <- data.frame(c$overall[1], row.names = "LDA")
colnames(compara_ACC) <- "Accuracy"
```

Confusion Matrix and Statistics

	Reference		
Prediction	V	A	R
V	61	28	0
A	22	21	7
R	0	3	3

Overall Statistics

Accuracy : 0.5862
 95% CI : (0.5015, 0.6673)
 No Information Rate : 0.5724
 P-Value [Acc > NIR] : 0.4021

Kappa : 0.2075
 McNemar's Test P-Value : NA

Statistics by Class:

	Class: V	Class: A	Class: R
Sensitivity	0.7349	0.4038	0.30000
Specificity	0.5484	0.6882	0.97778
Pos Pred Value	0.6854	0.4200	0.50000
Neg Pred Value	0.6071	0.6737	0.94964
Prevalence	0.5724	0.3586	0.06897
Detection Rate	0.4207	0.1448	0.02069
Detection Prevalence	0.6138	0.3448	0.04138
Balanced Accuracy	0.6417	0.5460	0.63889

Se aprecia que la precisión balanceada es mayor para las clases V y R (el 64 %) y menor para la clase A (55 %).

Este análisis mostrando la matriz de confusión hay que realizarlo para cada algoritmo de clasificación que se aplique. La técnica es la misma, predecir las clases en el conjunto de validación y obtener la matriz de confusión. No se van a mostrar en adelante, por una cuestión de espacio, los valores de la matriz de confusión y se deja al lector la obtención de estos resultados. Se guardará el valor de *Accuracy* con el fin de comparar los resultados de los clasificadores.

Con la librería *klaR* puede representarse el resultado de la clasificación tomando los atributos dos a dos y representando la partición de las clases.

```
library(klaR)

partimat(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED,
data = setEntrena, method = "lda")
```

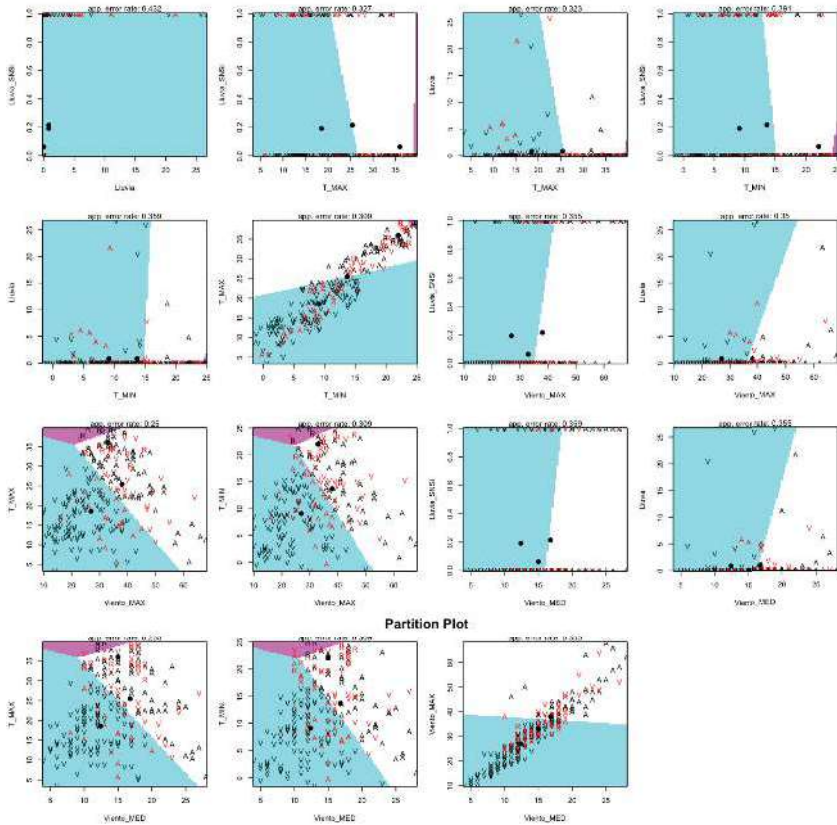


Fig. 5.27. Matriz de diagramas de dispersión con la anotación de las clases.

Para poder aplicar LDA se requiere que exista igual varianza-covarianza entre los atributos que van a predecir la clase. Esta condición se puede comprobar con el test M de Box, en caso de no verificarse podría aplicarse la extensión QDA (*Quadratic Discriminant Analysis*).

5.2.3 Clasificadores probabilísticos

Bajo esta denominación se agrupan un conjunto de técnicas que aplican la probabilidad de los sucesos para la obtención de modelos de predicción, usando el teorema de

Bayes o modelos de Markov. Se van a aplicar dos de los clasificadores probabilísticos más utilizados: *naive* bayesiano y redes bayesianas.

5.2.3.1 *Naive* bayesiano

El adjetivo “ingenuo” para este tipo de clasificador proviene del hecho de considerar la premisa de independencia entre los atributos y las clases a predecir. Con esta premisa, no es necesario considerar la matriz de covarianzas y basta con estimar la media y la varianza de los atributos con las clases.

Hay en R muchísimas implementaciones de este tipo de clasificador, se va a utilizar la librería *naivebayes*, creada específicamente para este tipo de clasificación.

```
library(naivebayes)

modelo.nb <- naive_bayes(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX +
+ Viento_MED, data = setEntrena)

plot(modelo.nb)
```

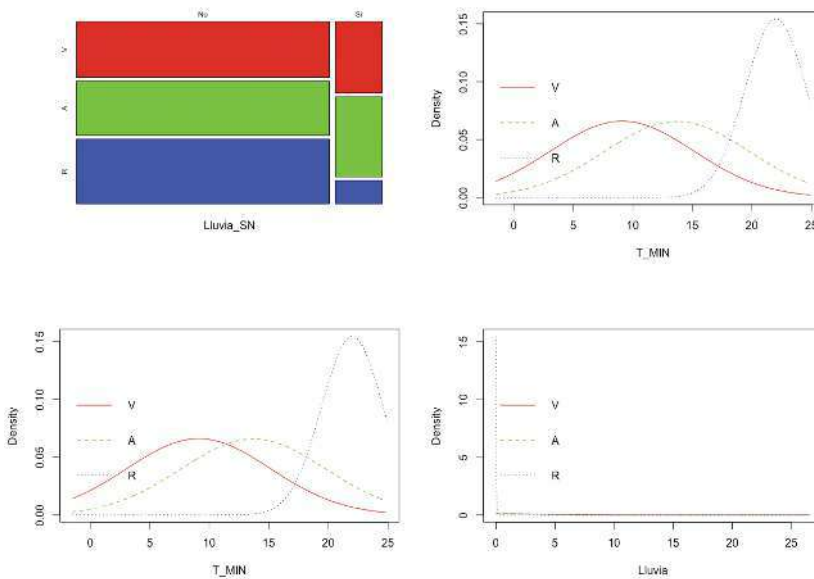


Fig. 5.28. Representación del modelo *naive* bayesiano obtenido.

El modelo *naive* bayesiano se representa en los gráficos anteriores. Como se aprecia, son un conjunto de distribuciones de probabilidad para cada clase relacionada con cada atributo, definidas mediante su media y varianza. Para el caso del atributo nominal *Lluvia_SN*, se tiene la probabilidad de ocurrencia (que llueva o no) para cada clase.

Para obtener las predicciones se calcula con:

```
predict(modelo.nb, setValida[1,], type = "prob")

predice.modelo.nb <- predict(modelo.nb, newdata = setValida)
levels(predice.modelo.nb) <- c("V", "A", "R") #hay que reasignar las clases, predict
las ha cambiado

c <- confusionMatrix(predice.modelo.nb, setValida$O3_Nivel)

compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "Naive Bayes"
```

	A	R	V
[1,]	0.9989109	0.001089076	8.96178e-34

Se obtiene la probabilidad de pertenencia a cada una de las clases. En este caso, el resultado muestra claramente que la mayor probabilidad de pertenencia corresponde a la clase A.

5.2.3.2 Redes bayesianas

En la red bayesiana se representa en un grafo dirigido acíclico las dependencias entre los atributos que se usan para predecir el atributo de la clase. Si hay atributos que no están conectados, entonces significa que son independientes. La forma de proceder consiste primero en determinar la red, dada por construcción con el conocimiento de los expertos en el problema o mediante algoritmos que la estiman. Y en una segunda fase, construida la red, determinar sus parámetros: las distribuciones de probabilidad que ligan los nodos hijos con sus nodos padres.

De nuevo, hay diferentes librerías que permiten aplicar redes bayesianas como clasificadores. Se va a ilustrar un ejemplo de cálculo utilizando la librería *bnlearn*. Esta librería dispone de diferentes algoritmos, tanto para el cálculo de la red como de sus parámetros.

```
library(bnlearn)

setEntrena$Viento_MAX <- as.numeric(setEntrena$Viento_MAX) #no pueden usarse atributos
enteros

setEntrena$Viento_MED <- as.numeric(setEntrena$Viento_MED)

red.bn <- hc(setEntrena[,c("O3_Nivel", "Lluvia_SN", "Lluvia", "T_MAX", "T_MIN", "Viento_MAX",
"Viento_MED")], score = "aic-cg", optimized = TRUE) #obtener la red

modelo.bn <- bn.fit(red.bn, setEntrena[,c("O3_Nivel", "Lluvia_SN", "Lluvia", "T_MAX",
"T_MIN", "Viento_MAX", "Viento_MED")]) #obtener parámetros

plot(red.bn)
```

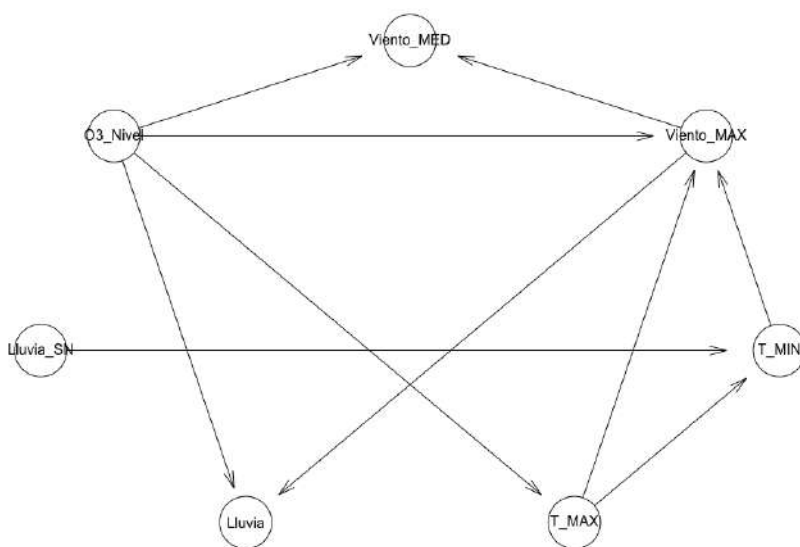


Fig. 5.29. Red bayesiana obtenida representada con un grafo acíclico dirigido.

Se aprecia en la red que no hay relación directa entre O3_Nivel con T_MIN y Lluvia_SN, por tanto, significa que son independientes.

Para realizar las inferencias sobre el modelo, conocidas algunas evidencias, se utiliza:

```
cpquery(modelo.bn, event = (O3_Nivel == "A"), evidence = ( T_MAX < 25 & Viento_MED > 5 ) )
```

En este caso se pregunta por la probabilidad de que el O3_Nivel sea A sabiendo que la T_MAX es menor de 25 y que Viento_MED es superior a 5.

```
[1] 0.2842843
```

También se puede predecir sobre el conjunto de entrenamiento.

```
setValida$Viento_MAX <- as.numeric(setValida$Viento_MAX) #no pueden usarse atributos enteros
setValida$Viento_MED <- as.numeric(setValida$Viento_MED)

predice.modelo.bn <- predict(modelo.bn, node = "O3_Nivel", data = setValida[,c("O3_Nivel", "Lluvia_SN", "Lluvia", "T_MAX", "T_MIN", "Viento_MAX", "Viento_MED")])

c <- confusionMatrix(predice.modelo.bn, setValida$O3_Nivel)

compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "Red Bayes"
```

6.1 Técnicas de minería de datos

Como ya se ha comentado, las técnicas de minería de datos (una etapa dentro del proceso completo de KDD [FAYY96]) intentan obtener patrones o modelos a partir de los datos recopilados. Decidir si los modelos obtenidos son útiles o no suele requerir una valoración subjetiva por parte del usuario. Las técnicas de minería de datos se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas [WI98].

Técnicas	No supervisadas	Clustering	Numérico Conceptual Probabilístico
		Asociación	A priori
	Supervisadas	Predicción	Regresión Árboles de predicción Estimador de núcleos
		Clasificación	Tabla de decisión Árboles de decisión Inducción de reglas Bayesiana Basado en ejemplares Redes de neuronas Máquinas de soporte vectorial Técnicas de IA (lógica borrosa, algoritmos genéticos, etc.)

Fig. 6.1. Técnicas de minería de datos.

Una técnica constituye el enfoque conceptual para extraer la información de los datos, y, en general, es implementada por varios algoritmos. Cada algoritmo representa, en la práctica, la manera de desarrollar una determinada técnica paso a paso, de forma que es preciso un entendimiento de alto nivel de los algoritmos para saber cuál es la técnica más apropiada para cada problema. Así mismo, es preciso entender los parámetros y las características de los algoritmos para preparar los datos a analizar.

Las predicciones se utilizan para prever el comportamiento futuro de algún tipo de entidad, mientras que una descripción puede ayudar a su comprensión. De hecho, los modelos predictivos pueden ser descriptivos (hasta donde sean comprensibles por personas) y los modelos descriptivos pueden emplearse para realizar predicciones. De esta forma, hay algoritmos o técnicas que pueden servir para distintos propósitos, de modo que la figura anterior únicamente representa para qué propósito son más utilizadas las técnicas. Por ejemplo, las redes de neuronas pueden servir para predicción, clasificación e incluso para aprendizaje no supervisado.

El aprendizaje inductivo no supervisado estudia el aprendizaje sin la ayuda del maestro; es decir, se aborda el aprendizaje sin supervisión, que trata de ordenar los ejemplos en una jerarquía según las regularidades en la distribución de los pares atributo-valor sin la guía del atributo especial clase. Éste es el proceder de los sistemas que realizan *clustering* conceptual y de los que se dice también que adquieren nuevos conceptos. Otra posibilidad contemplada para estos sistemas es la de sintetizar conocimiento cualitativo o cuantitativo, objetivo de los sistemas que llevan a cabo tareas de descubrimiento.

En el aprendizaje inductivo supervisado existe un atributo especial, normalmente denominado *clase*, presente en todos los ejemplos, que especifica si el ejemplo pertenece o no a un cierto concepto, que será el objetivo del aprendizaje. El atributo “clase” normalmente toma los valores + y -, que significan la pertenencia o no del ejemplo al concepto que se trata de aprender; es decir, que el ejemplo ejemplifica positivamente al concepto (pertenece al concepto) o bien lo ejemplifica negativamente (que no pertenece al concepto). Mediante una generalización del papel del atributo “clase”, cualquier atributo puede desempeñar ese papel, convirtiéndose la clasificación de los ejemplos según los valores del atributo en cuestión en el objeto del aprendizaje. Expresado en una forma breve, el objetivo del aprendizaje supervisado es: a partir de un conjunto de ejemplos, denominados *de entrenamiento*, de un cierto dominio D de ellos, construir criterios para determinar el valor del atributo clase en un ejemplo cualquiera del dominio. Esos criterios están basados en los valores de uno o varios de los otros pares (atributo; valor) que intervienen en la definición de los ejemplos. Es sencillo transmitir esa idea al caso en el que el atributo que juega el papel de la clase sea uno cualquiera o con más de dos valores. Dentro de este tipo de aprendizaje se pueden distinguir dos

grandes grupos de técnicas: la predicción y la clasificación [WK91]. A continuación se presentan las principales técnicas (supervisadas y no supervisadas) de minería de datos.

6.2 Clustering

También llamadas *técnicas de agrupamiento* o *técnicas de segmentación*, permiten la identificación de tipologías o grupos donde los elementos guardan gran similitud entre sí y muchas diferencias con los de otros grupos. Así se puede segmentar el colectivo de clientes, el conjunto de valores e índices financieros, el espectro de observaciones astronómicas, el conjunto de zonas forestales, el conjunto de empleados y de sucursales u oficinas, etc. La segmentación está teniendo mucho interés desde hace ya tiempo dadas las importantes ventajas que aporta al permitir el tratamiento de grandes colectivos de forma pseudoparticularizada, en el más idóneo punto de equilibrio entre el tratamiento individualizado y aquel totalmente masificado.

Las herramientas de *clustering* pueden estar basadas en múltiples tipos de técnicas, como estadística, empleo de algoritmos matemáticos o generación de reglas o de redes neuronales para el tratamiento de registros. Para otro tipo de elementos a agrupar o segmentar, como texto y documentos, se usan técnicas de reconocimiento de conceptos. Esta técnica suele servir de punto de partida para después hacer un análisis de clasificación sobre los clústeres.

La principal característica de esta familia de técnicas es la utilización de una medida de similaridad que, en general, está basada en los atributos que describen a los objetos, y se define usualmente por proximidad en un espacio multidimensional. Para datos numéricos, suele ser preciso preparar los datos antes de realizar *data mining* sobre ellos, de manera que en primer lugar se someten a un proceso de estandarización. Una de las técnicas empleadas para conseguir la normalización de los datos es utilizar la medida z (*z-score*) que elimina las unidades de los datos. Esta medida, z , es la que se muestra en la ecuación 6.1, donde μ_f es la media de la variable f y σ_f la desviación típica de la misma.

$$z_{if} = \frac{x_{if} - \mu_f}{\sigma_f} \quad \text{Ec. 6.1}$$

Entre las medidas de similaridad destaca la distancia euclídea, ecuación 6.2.

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^n (x_{il} - x_{jl})^2} \quad \text{Ec. 6.2}$$

$$d(\bar{x}_i, \bar{x}_j) = \sqrt{\sum_{m=1}^F (x_{im} - x_{jm})^2}$$

Cuando los atributos tienen rangos muy diferentes, para evitar la distorsión por diferente peso en la distancia, se utilizan métodos de normalización, el más directo sería normalizar por los rangos:

$$a_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)} \quad \text{Ec. 6.3}$$

Una opción más recomendable es normalizar con la medida de dispersión, considerando las varianzas de cada atributo.

$$d(\bar{x}_i, \bar{x}_j) = \sqrt{\sum_{l=1}^F \frac{(x_{il} - x_{jl})^2}{\sigma_l^2}} \quad \text{Ec. 6.4}$$

Incluso si se conocen las covarianzas entre atributos puede utilizarse la distancia estadística.

$$d(\bar{x}_i, \bar{x}_j) = \sqrt{(\vec{X}_i - \vec{X}_j)^t S^{-1} (\vec{X}_i - \vec{X}_j)} \quad \text{Ec. 6.5}$$

Con los atributos nominales, el problema es obtener una distancia numérica, la solución más común es la distancia binaria:

$$d(x_i, x_j) = \begin{cases} 1, & \text{si } x_i \neq x_j \\ 0, & \text{si } x_i = x_j \end{cases} \quad \text{Ec. 6.6}$$

En el caso de tratarse de atributos nominales ordinales, en los que el orden importa (por ejemplo, EDAD={NIÑO, JOVEN, ADULTO, ANCIANO}), esta distancia puede calcularse teniendo en cuenta el número de escalones de la diferencia:

$$d(x_i, x_j) = \frac{r_{ij} - 1}{M - 1} \quad \text{Ec. 6.7}$$

Hay varios algoritmos de *clustering*. A continuación, se exponen los más conocidos.

6.2.1 *Clustering* numérico (k-medias)

Uno de los algoritmos más utilizados para hacer *clustering* es el k-medias (*k-means*) [MAC67], que se caracteriza por su sencillez. En primer lugar, se debe especificar por adelantado cuántos clústeres se van a crear, éste es el parámetro k , para lo cual se seleccionan k elementos aleatoriamente, que representarán el centro o media de cada clúster. A continuación, cada una de las instancias, ejemplos, es asignada al centro del clúster más cercano de acuerdo con la distancia euclídea que le separa de él. Para cada uno de los clústeres así contruidos se calcula el centroide de todas sus instancias. Estos centroides son tomados como los nuevos centros de sus respectivos clústeres. Finalmente se repite el proceso completo con los nuevos centros de los clústeres. La iteración continúa hasta que se repite la asignación de los mismos ejemplos a los mismos clústeres, ya que los puntos centrales de éstos se han estabilizado y permanecerán invariables después de cada iteración. El algoritmo de k-medias es el siguiente:

1. Elegir k ejemplos que actúan como semillas (k número de clústeres).
2. Para cada ejemplo, añadir ejemplo a la clase más similar.
3. Calcular el centroide de cada clase, que pasan a ser las nuevas semillas.
4. Si no se llega a un criterio de convergencia (por ejemplo, dos iteraciones no cambian las clasificaciones de los ejemplos), volver a 2.

Fig. 6.2. Pseudocódigo del algoritmo de k-medias.

Para obtener los centroides, se calcula la media (*mean*) o la moda (*mode*) según se trate de atributos numéricos o simbólicos. A continuación, en la figura 6.3, se muestra un ejemplo de *clustering* con el algoritmo k-medias.

En este caso se parte de un total de nueve ejemplos o instancias, se configura el algoritmo para que obtenga tres clústeres y se inician aleatoriamente los centroides de los clústeres a un ejemplo determinado. Una vez inicializados los datos, se comienza el bucle del algoritmo. En cada una de las gráficas inferiores se muestra un paso por el algoritmo. Cada uno de los ejemplos se representa con un tono de color diferente que

indica la pertenencia del ejemplo a un clúster determinado, mientras que los centroides siguen mostrándose como círculos de mayor tamaño y sin relleno. Por último, el proceso de *clustering* finaliza en el paso 3, ya que en la siguiente pasada del algoritmo (realmente haría cuatro pasadas, si se configurara así) ningún ejemplo cambiaría de clúster.

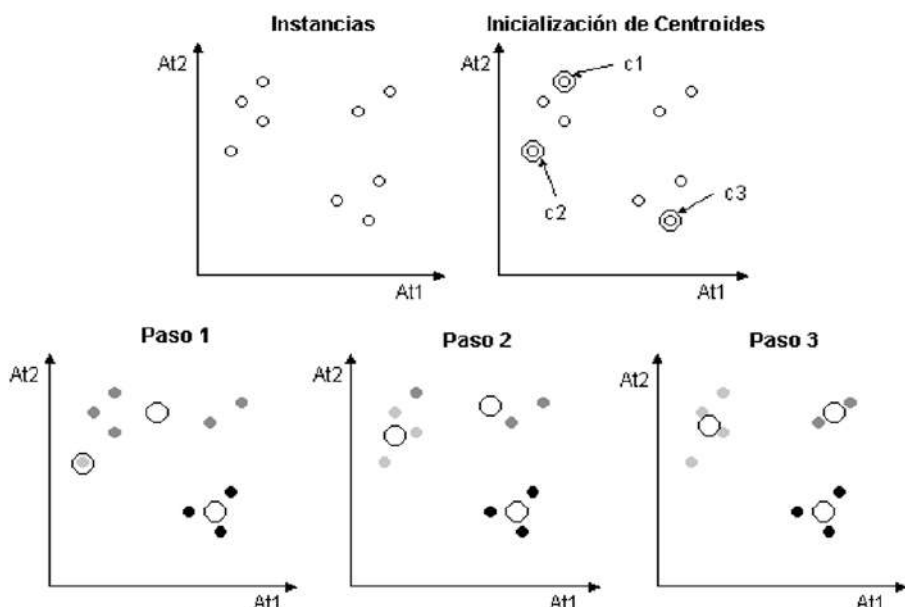


Fig. 6.3. Ejemplo de *clustering* con k-medias.

6.2.2 Clustering conceptual (COBWEB)

El algoritmo de k-medias se encuentra con un problema cuando los atributos no son numéricos, ya que en ese caso la distancia entre ejemplares no está tan clara. Para resolver este problema, Michalski [MS83] presenta la noción de *clustering* conceptual, que utiliza para justificar la necesidad de un *clustering* cualitativo frente al *clustering* cuantitativo, basado en la vecindad entre los elementos de la población. En este tipo de *clustering* una partición de los datos es buena si cada clase tiene una buena interpretación conceptual (modelo cognitivo de jerarquías). Una de las principales motivaciones de la categorización de un conjunto de ejemplos, que básicamente supone la formación de conceptos, es la predicción de características de las categorías que heredarán sus

subcategorías. Esta conjetura es la base de COBWEB [FIS87]. A semejanza de los humanos, COBWEB forma los conceptos por agrupación de ejemplos con atributos similares. Representa los clústeres como una distribución de probabilidad sobre el espacio de los valores de los atributos, generando un árbol de clasificación jerárquica en el que los nodos intermedios definen subconceptos. El objetivo de COBWEB es hallar un conjunto de clases o clústeres (subconjuntos de ejemplos) que maximice la utilidad de la categoría (partición del conjunto de ejemplos cuyos miembros son clases). La descripción probabilística se basa en dos conceptos:

- **Predicibilidad.** Probabilidad condicional de que un suceso tenga un cierto atributo dada la clase, $P(A_i=V_{ij}/C_k)$. El mayor de estos valores corresponde al valor del atributo más predecible y es el de los miembros de la clase (alta similaridad entre los elementos de la clase).
- **Previsibilidad.** Probabilidad condicional de que un ejemplo sea una instancia de una cierta clase, dado el valor de un atributo particular, $P(C_k/A_i=V_{ij})$. Un valor alto indica que pocos ejemplos de las otras clases comparten este valor del atributo, y el valor del atributo de mayor probabilidad es el de los miembros de la clase (baja similaridad interclase).

Estas dos medidas, combinadas mediante el teorema de Bayes, proporcionan una función que evalúa la utilidad de una categoría (CU), que se muestra en la ecuación 6.8.

$$CU = \frac{P(C_k) \left[\sum_i \sum_j P(A_i = V_{ij} | C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2 \right]}{n} \quad \text{Ec. 6.8}$$

En esta ecuación, n es el número de clases y las sumas se extienden a todos los atributos A_i y sus valores V_{ij} en cada una de las n clases C_k . La división por n sirve para incentivar tener clústeres con más de un elemento. La utilidad de la categoría mide el valor esperado de valores de atributos que pueden ser adivinados a partir de la partición sobre los valores que se pueden adivinar sin esa partición. Si la partición no ayuda en esto, entonces no es una buena partición. El árbol resultante de este algoritmo cabe denominarse *organización probabilística o jerárquica de conceptos*. En la figura 6.4 se muestra un ejemplo de árbol que se podría generar mediante COBWEB. En la construcción del árbol, incrementalmente se incorpora cada ejemplo al mismo, donde cada nodo es un concepto probabilístico que representa una clase de objetos. COBWEB desciende por el árbol buscando el mejor lugar o nodo para cada ejemplo. Esto se basa en medir en cuál se tiene la mayor ganancia de utilidad de categoría.

Datos de Entrada

Hombre	Recubierto de	Cavidades corazón	temperatura del cuerpo	Fertilización
Mamífero	Pelo	Cuatro	Regulada	Interna
Pez	Escamas	Dos	Sin regulación	Externa
Anfibio	Piel húmeda	Tres	Sin regulación	Externa
Ave	Plumas	Cuatro	Regulada	Interna
Reptil	Piel dura	Cuatro Imperfectas	Sin regulación	Interna

Árbol Generado

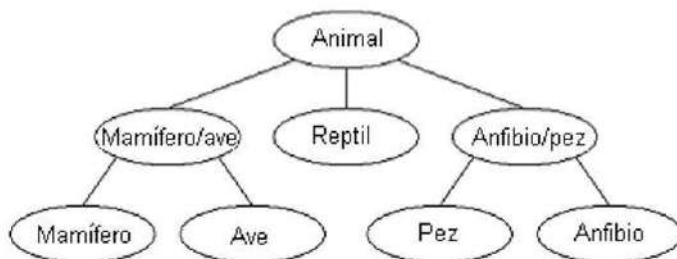


Fig. 6.4. Ejemplo de árbol generado por COBWEB.

Sin embargo, no se puede garantizar que se genere este árbol, dado que el algoritmo es sensible al orden en que se introduzcan los ejemplos. En cuanto a las etiquetas de los nodos, éstas fueron puestas *a posteriori*, coherentes con los valores de los atributos que determinan el nodo. Cuando COBWEB incorpora un nuevo ejemplo en el nodo de clasificación, desciende a lo largo del camino apropiado, actualizando las cuentas de cada nodo, y llevando a cabo por medio de los diferentes operadores una de las siguientes acciones:

- **Incorporación.** Añadir un nuevo ejemplo a un nodo ya existente.
- **Creación de una nueva disyunción.** Crear una nueva clase.
- **Unión.** Combinar dos clases en una sola.
- **División.** Dividir una clase existente en varias clases.

La búsqueda, que se realiza en el espacio de conceptos, es por medio de un heurístico basado en el método de escalada gracias a los operadores de unión y división. En la figura 6.5 se muestra el resultado de aplicar cada una de estas operaciones.

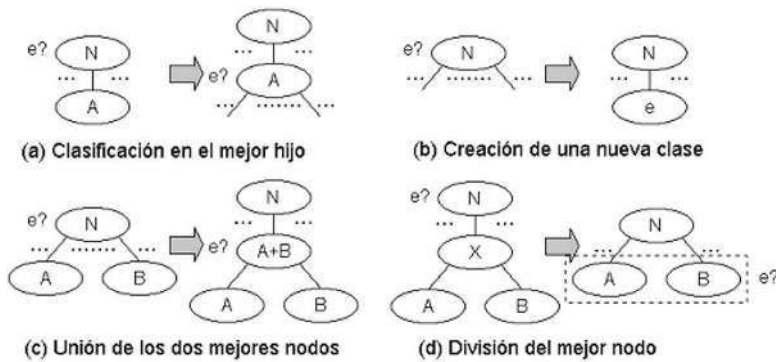


Fig. 6.5. Operaciones de COBWEB.

1. Nuevo ejemplo: lee un ejemplo e. Si no hay más ejemplos, terminar.
2. Actualiza raíz. Actualiza el cálculo de la raíz.
3. Si la raíz es hoja, entonces: expandir en dos nodos hijos y acomodar en cada uno de ellos un ejemplo; volver a 1.
4. Avanzar hasta el siguiente nivel: aplicar la función de evaluación a varias opciones para determinar, mediante la fórmula de utilidad de una categoría, el mejor (máxima CU) lugar donde incorporar el ejemplo en el nivel siguiente de la jerarquía. En las opciones que se evaluarán se considerarán únicamente el nodo actual y sus hijos y se elegirá la mejor opción de las siguientes:
 - a. Añadir e a un nodo que existe (al mejor hijo) y, si esta opción resulta ganadora, comenzar de nuevo el proceso de avance hacia el siguiente nivel en ese nodo hijo.
 - b. Crear un nuevo nodo conteniendo únicamente a e y, si esta opción resulta ganadora, volver a 1.
 - c. Juntar los dos mejores nodos hijos con e incorporado al nuevo nodo combinado y, si esta opción resulta ganadora, comenzar el nuevo proceso de avanzar hacia el siguiente nivel en ese nuevo nodo.
 - d. Dividir el mejor nodo, reemplazando este nodo con sus hijos y, si esta opción resulta ganadora, aplicar la función de evaluación para incorporar e en los nodos originados por la división.

Fig. 6.6. Algoritmo de COBWEB.

El algoritmo se puede extender a valores numéricos usando distribuciones gaussianas, ecuación 6.9. De esta forma, el sumatorio de probabilidades es ahora como se muestra en la ecuación 6.10.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{Ec. 6.9}$$

$$\sum_j P(A_i = V_{ij})^2 \leftrightarrow \int_{-\infty}^{+\infty} f(x_i)^2 dx_i = \frac{1}{2\sqrt{\pi}\sigma_i} \quad \text{Ec. 6.10}$$

De modo que la ecuación de la utilidad de la categoría quedaría como se muestra en la ecuación 6.11.

$$CU = \frac{1}{k} \sum_{k=1}^n P(C_k) \frac{1}{2\sqrt{\pi}} \sum_i \left(\frac{1}{\sigma_{ik}} - \frac{1}{\sigma_i} \right) \quad \text{Ec. 6.11}$$

6.2.3 Clustering probabilístico (EM)

Los algoritmos de *clustering* estudiados hasta el momento presentan ciertos defectos entre los que destacan la dependencia que tiene el resultado del orden de los ejemplos y la tendencia de estos algoritmos al sobreajuste (*overfitting*). Una aproximación estadística al asunto del *clustering* resuelve estos problemas. Desde este punto de vista, lo que se busca es el grupo de clústeres más probables dados los datos. Ahora los ejemplos tienen ciertas probabilidades de pertenecer a un clúster. La base de este tipo de *clustering* se encuentra en un modelo estadístico llamado *mezcla de distribuciones* (*finite mixtures*). Cada distribución representa la probabilidad de que un objeto tenga un conjunto particular de pares atributo-valor, si se supiera que es miembro de ese clúster. Se tienen k distribuciones de probabilidad que representan los k clústeres. La mezcla más sencilla se tiene cuando los atributos son numéricos con distribuciones gaussianas. Cada distribución (normal) se caracteriza por dos parámetros: la media (μ) y la varianza (σ^2). Además, cada distribución tendrá cierta probabilidad de aparición p , que vendrá determinada por la proporción de ejemplos que pertenecen a dicho clúster respecto del número total de ejemplos. En ese caso, si hay k clústeres, habrá que calcular un total de $3k-1$ parámetros: las k medias, k varianzas y $k-1$ probabilidades de la distribución, dado que la suma de probabilidades debe ser 1, con lo que conocidas $k-1$ se puede determinar la k -ésima.

Si se conociera el clúster al que pertenece, en un principio, cada uno de los ejemplos de entrenamiento sería muy sencillo obtener los $3k-1$ parámetros necesarios para definir totalmente las distribuciones de dichos clústeres, ya que simplemente se aplicarían las ecuaciones de la media y de la varianza para cada uno de los clústeres. Además, para

calcular la probabilidad de cada una de las distribuciones únicamente se dividiría el número de ejemplos de entrenamiento que pertenecen al clúster en cuestión entre el número total de ejemplos de entrenamiento. Una vez obtenidos estos parámetros, si se deseara calcular la probabilidad de pertenencia de un determinado ejemplo de test a cada clúster, simplemente se aplicaría el teorema de Bayes a cada problema concreto, con lo que quedaría la ecuación 6.12.

$$P(A|x) = \frac{P(x|A)P(A)}{P(x)} = \frac{f(x; \mu_A, \sigma_A)p_A}{P(x)} \quad \text{Ec. 6.12}$$

En esta ecuación, A es un clúster del sistema, x el ejemplo de test, p_A la probabilidad del clúster A y $f(x; \mu_A, \sigma_A)$ la función de la distribución normal del clúster A , que se expresa con la ecuación 6.12. Sin embargo, el problema es que no se sabe de qué distribución viene cada dato y se desconocen los parámetros de las distribuciones. Por ello se adopta el procedimiento empleado por el algoritmo de *clustering* k-medias, y se itera.

El algoritmo EM (*Expectation Maximization*) empieza adivinando los parámetros de las distribuciones (dicho de otro modo, se empieza adivinando las probabilidades de que un objeto pertenezca a una clase) y, a continuación, los utiliza para calcular las probabilidades de que cada objeto pertenezca a un clúster y usándolas para reestimar los parámetros de dichas probabilidades, hasta converger. Este algoritmo recibe su nombre de los dos pasos en los que se basa cada iteración: el cálculo de las probabilidades de los grupos o los valores esperados de los grupos, mediante la ecuación 6.12, denominado *expectation*; y el cálculo de los valores de los parámetros de las distribuciones, denominado *maximization*, en el que se maximiza la verosimilitud de las distribuciones dados los datos.

Para estimar los parámetros de las distribuciones se tiene que considerar que se conocen únicamente las probabilidades de pertenencia a cada clúster, y no los clústeres en sí. Estas probabilidades actúan como pesos, con lo que el cálculo de la media y la varianza se realiza con las ecuaciones 6.13 y 6.14 respectivamente.

$$\mu_A = \frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i} \quad \text{Ec. 6.13}$$

$$\sigma_A^2 = \frac{\sum_{i=1}^N w_i (x_i - \sigma_i)^2}{\sum_{i=1}^N w_i} \quad \text{Ec. 6.14}$$

Donde N es el número total de ejemplos del conjunto de entrenamiento y w_i es la probabilidad de que el ejemplo i pertenezca al clúster A . La cuestión es determinar cuándo se finaliza el procedimiento, es decir, en qué momento se dejan de realizar iteraciones. En el algoritmo k-medias se finalizaba cuando ningún ejemplo de entrenamiento cam-

biaba de clúster en una iteración, alcanzándose así un punto fijo (*fixed point*). En el algoritmo EM es un poco más complicado, dado que éste tiende a converger pero nunca se llega a ningún punto fijo. Sin embargo, se puede ver cuánto se acerca calculando la verosimilitud (*likelihood*) general de los datos con esos parámetros, multiplicando las probabilidades de los ejemplos, tal y como se muestra en la ecuación 6.15.

$$\prod_{i=1}^N \left(\sum_j^{clusters} p_j P(x_i | j) \right) \quad \text{Ec. 6.15}$$

En esta ecuación, j representa cada uno de los clústeres del sistema, y p_j la probabilidad de dicho clúster. La verosimilitud es una medida de lo “bueno” que es el *clustering*, y se incrementa con cada iteración del algoritmo EM. Se seguirá iterando hasta que dicha medida se incremente un valor despreciable.

Aunque EM garantiza la convergencia, ésta puede ser a un máximo local, de modo que se recomienda repetir el proceso varias veces, con diferentes parámetros iniciales para las distribuciones. Tras estas repeticiones, se pueden comparar las medidas de verosimilitud obtenidas y escoger la mayor de todas ellas. En la figura 6.7 se muestra un ejemplo de *clustering* probabilístico con el algoritmo EM.

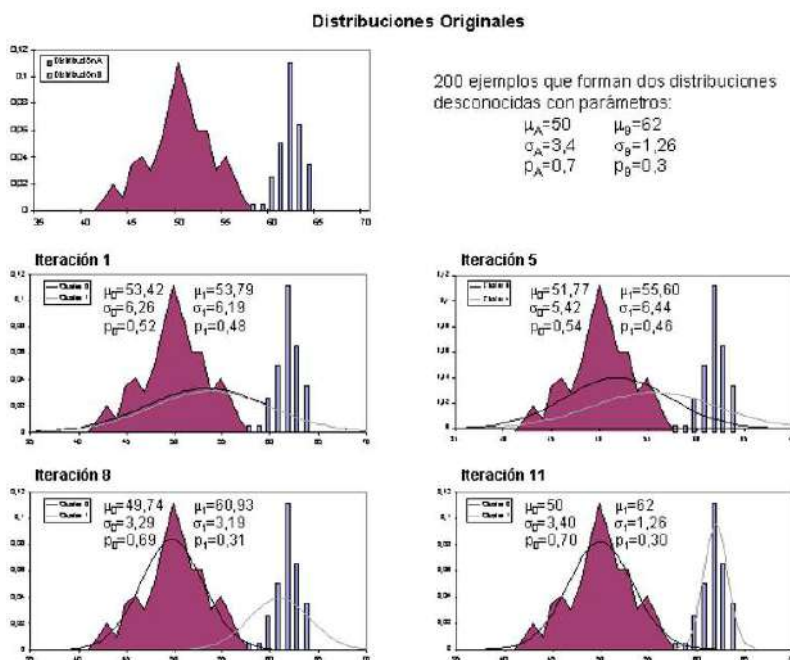


Fig. 6.7. Ejemplo de *clustering* con EM.

En este experimento se introducen un total de doscientos ejemplos que constituyen dos distribuciones desconocidas para el algoritmo. Lo único que conoce el algoritmo es que hay dos clústeres, dado que este dato se introduce como parámetro de entrada. En la iteración 0 se inicializan los parámetros de los clústeres a 0 (media, desviación típica y probabilidad). En las siguientes iteraciones, estos parámetros van tomando forma hasta finalizar en la iteración 11, iteración en la que finaliza el proceso, por el incremento de la medida de verosimilitud, tan sólo del orden de 10^{-4} .

Extensiones al algoritmo EM

El modelo puede extenderse desde un atributo numérico, como se ha visto hasta el momento, hasta múltiples atributos, asumiendo independencia entre atributos. Las probabilidades de cada atributo se multiplican entre sí para obtener una probabilidad conjunta para la instancia, tal y como se hace en el algoritmo *naïve* bayesiano. También puede haber atributos correlacionados, en cuyo caso se puede modelar con una distribución normal bivariable, en donde se utiliza una matriz de covarianza. En este caso, el número de parámetros crece según el cuadrado del número de atributos que se consideren correlacionados entre sí, ya que se debe construir una matriz de covarianza. Esta escalabilidad en el número de parámetros tiene serias consecuencias de sobreajuste.

En el caso de un atributo nominal con v posibles valores, se caracteriza mediante v valores numéricos que representan la probabilidad de cada valor. Se necesitarán otros kv valores numéricos, que serán las probabilidades condicionadas de cada posible valor del atributo con respecto a cada clúster. En cuanto a los valores desconocidos, se puede optar por varias soluciones: ignorarlo en el productorio de probabilidades; añadir un nuevo valor a los posibles, sólo en el caso de atributos nominales; o tomar la media o la moda del atributo, según se trate de atributos numéricos o nominales. Por último, aunque se puede especificar el número de clústeres, también es posible dejar que sea el algoritmo el que determine automáticamente cuál es el número de clústeres mediante validación cruzada.

6.3 Reglas de asociación

Este tipo de técnicas se emplea para establecer las posibles relaciones o correlaciones entre distintas acciones o sucesos aparentemente independientes, pudiendo reconocer como la ocurrencia de un suceso o acción puede inducir o generar la aparición de otros [AIS93b]. Son utilizadas cuando el objetivo es realizar **análisis exploratorios**, buscando relaciones dentro del conjunto de datos. Las asociaciones identificadas pueden usarse para predecir comportamientos, y permiten descubrir correlaciones y coocurrencias de eventos [AS94, AS94a, AS94b]. Debido a sus características, estas técnicas tienen una gran aplicación práctica en muchos campos como, por ejemplo, el comercial, ya que son especialmente interesantes a la hora de comprender los hábitos de compra

de los clientes y constituyen un pilar básico en la concepción de las ofertas y ventas cruzadas, así como del *merchandising* [RMS98]. En otros entornos como el sanitario, estas herramientas se emplean para identificar factores de riesgo en la aparición o complicación de enfermedades. Para su utilización es necesario disponer de información de cada uno de los sucesos llevados a cabo por un mismo individuo o cliente en un determinado periodo temporal. Por lo general, esta forma de extracción de conocimiento se fundamenta en técnicas estadísticas [CHY96], como los análisis de correlación y de variación [BMS97]. Uno de los algoritmos más utilizados es el algoritmo *a priori*, que se presenta a continuación.

Algoritmo *a priori*

La generación de reglas de asociación se logra basándose en un procedimiento de *covering*. Las reglas de asociación son parecidas, en su forma, a las reglas de clasificación, si bien en su lado derecho puede aparecer cualquier par o pares atributo-valor. De manera que para encontrar ese tipo de reglas es preciso considerar cada posible combinación de pares atributo-valor del lado derecho. Para evaluar las reglas se emplean la medida del soporte (*support*), ecuación 6.16, que indica el número de casos, ejemplos, que cubre la regla, y la confianza (*confidence*), ecuación 6.17, que indica el número de casos que predice la regla correctamente, y que viene expresado como el cociente entre el número de casos en que se cumple la regla y el número de casos en que se aplica, ya que se cumplen las premisas.

$$\text{soporte}(A \Rightarrow B) = P(A \cap B) \quad \text{Ec. 6.16}$$

$$\text{confianza}(A \Rightarrow B) = P(B|A) = \frac{P(A \cap B)}{P(A)} \quad \text{Ec. 6.17}$$

Las reglas que interesan son únicamente aquellas que tienen su valor de soporte muy alto, de forma que se buscan, independientemente de en qué lado aparezcan, pares atributo-valor que cubran una gran cantidad de ejemplos. A cada par atributo-valor se le denomina *ítem*, mientras que a un conjunto de ítems se les denomina *ítem-sets*. Por supuesto, para la formación de *ítem-sets* no se pueden unir ítems referidos al mismo atributo pero con distinto valor, dado que eso nunca se podría producir en un ejemplo. Se buscan *ítem-sets* con un máximo soporte, para lo que se comienza con *ítem-sets* con un único ítem. Se eliminan aquéllos cuyo valor de soporte sea inferior al mínimo establecido, y se combinan el resto formando *ítem-sets* con dos ítems. A su vez se eliminan aquellos nuevos que no cumplan con la condición del soporte, y al resto se le añadirá un nuevo ítem, formando *ítem-sets* con tres ítems. El proceso continuará hasta que ya no se puedan formar *ítem-sets* con un ítem más. Además, para generar

los ítem-sets de un determinado nivel, sólo es necesario emplear los del nivel inferior (con $n-1$ coincidencias, siendo n el número de ítems del nivel). Una vez se han obtenido todos, se pasará a la generación de reglas. Se tomará cada ítem-set y se formarán reglas que cumplan con la condición de confianza. Debe tenerse en cuenta que un ítem-set puede dar lugar a más de una regla de asociación, al igual que también puede no dar lugar a ninguna regla.

Un ejemplo típico de reglas de asociación es el análisis de la cesta de la compra (*market-basket analysis*). Básicamente consiste en encontrar asociaciones entre los productos que habitualmente compran los clientes, para utilizarlas en el desarrollo de las estrategias mercadotécnicas. En la figura 6.8 se muestra un ejemplo sencillo de obtención de reglas de asociación aplicado a este campo.

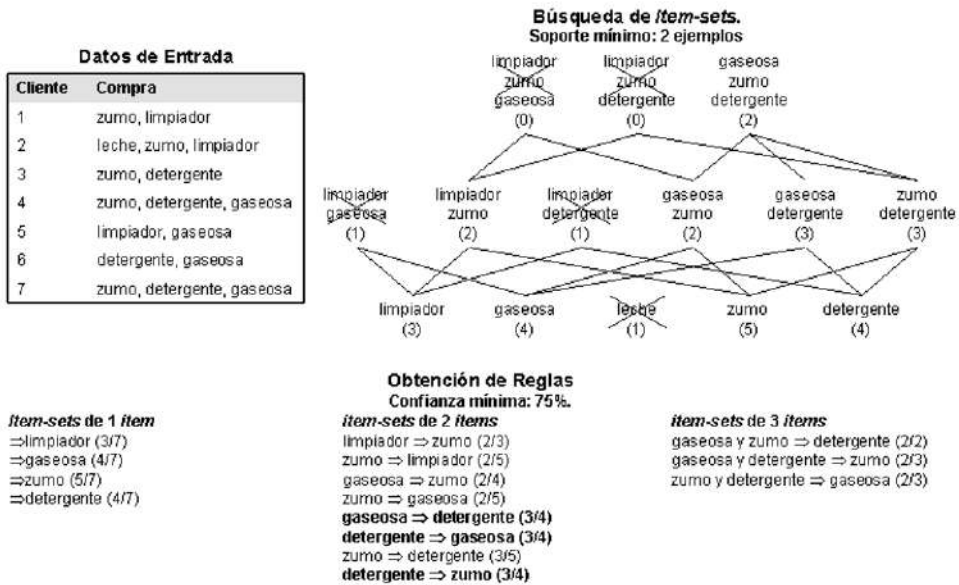


Fig. 6.8. Ejemplo de obtención de reglas de asociación *a priori*.

En esta imagen se muestra cómo se forman los ítem-sets a partir de los del nivel inferior, y cómo posteriormente se obtienen las reglas de asociación a partir de los ítem-sets seleccionados. Las reglas en negrita son las que se obtendrían, dado que cumplen con la confianza mínima requerida. El proceso de obtención de las reglas de asociación que se comentó anteriormente se basa en el algoritmo que se muestra en la figura 6.9 (*A priori*, Agrawal *et al.* 94).

1. Genera todos los ítem-sets con un elemento. Usa éstos para generar los de dos elementos y así sucesivamente. Se toman todos los posibles pares que cumplen con las medidas mínimas del soporte. Esto permite ir eliminando posibles combinaciones ya que no todas se tienen que considerar.
2. Genera las reglas revisando que cumplan con el criterio mínimo de confianza.

Fig. 6.9. Algoritmo de obtención de reglas de asociación *a priori*.

Una observación interesante es que si una conjunción de consecuentes de una regla cumple con los niveles mínimos de soporte y confianza, sus subconjuntos (consecuentes) también los cumplen. Por el contrario, si algún ítem no los cumple, no tiene caso considerar sus superconjuntos. Esto da una forma de ir construyendo reglas, con un solo consecuente, y a partir de ellas construir de dos consecuentes y así sucesivamente.

6.4 Predicción numérica

En el capítulo anterior se introdujo la regresión lineal como método básico de predicción numérica en el que el modelo consistía en un conjunto de coeficientes de una función lineal, posiblemente extensible a problemas no lineales.

En este capítulo estamos viendo métodos basados en el aprendizaje automático para la predicción, que evitan la restricción de un único modelo para todos los datos, y se permite la búsqueda de modelos más complejos como árboles o núcleos de distribuciones multimodales.

6.4.1 Predicción no lineal con árboles de regresión

Los árboles de predicción numérica son similares a los árboles de decisión, que se estudiarán más adelante, excepto en que la clase a predecir es continua. En este caso, cada nodo hoja almacena un valor de clase consistente en la media de las instancias que se clasifican con esa hoja, en cuyo caso estamos hablando de un **árbol de regresión**, o bien un modelo lineal que predice el valor de la clase, y en ese caso se habla de **árbol de modelos**. En el caso del algoritmo M5 [WF00], se trata de obtener un árbol de modelos, si bien se puede utilizar para obtener un árbol de regresión, por ser éste un caso específico de árbol de modelos.

Mientras que en el caso de los árboles de decisión se emplea la entropía de clases para definir el atributo con el que dividir, en el caso de la predicción numérica se emplea la varianza del error en cada hoja. Una vez construido el árbol que clasifica las instancias se realiza la poda del mismo, tras lo cual se obtiene para cada nodo hoja una constante,

en el caso de los árboles de regresión, o un plano de regresión, en el caso de árboles de modelos. En este último caso, los atributos que formarán parte de la regresión serán aquellos que participaban en el subárbol que ha sido podado.

Para construir el árbol se emplea como heurística el minimizar la variación interna de los valores de la clase dentro de cada subconjunto. Se trata de seleccionar aquel atributo que maximice la reducción de la desviación estándar de error (SDR, *Standard Deviation Reduction*) con la fórmula que se muestra en la ecuación 6.18.

$$SDR = SD(E) - \sum_i \frac{|E_i|}{|E|} SD(E_i) \quad \text{Ec. 6.18}$$

En esta ecuación E es el conjunto de ejemplos en el nodo a dividir, E_i es cada uno de los conjuntos de ejemplos que resultan en la división en el nodo según el atributo considerado, $|E|$ es el número de ejemplos del conjunto E y $SD(E)$ la desviación típica de los valores de la clase en E . El proceso de división puede finalizar porque la desviación típica es una pequeña fracción (por ejemplo, el 5 %) de la desviación típica del conjunto original de instancias o porque hay pocas instancias (por ejemplo, 2).

En la figura 6.10 se muestra un ejemplo de generación del árbol de predicción con el algoritmo M5. Para ello se muestra en primer lugar los ejemplos de entrenamiento, en los que se trata de predecir los puntos que un jugador de baloncesto anotaría en un partido.

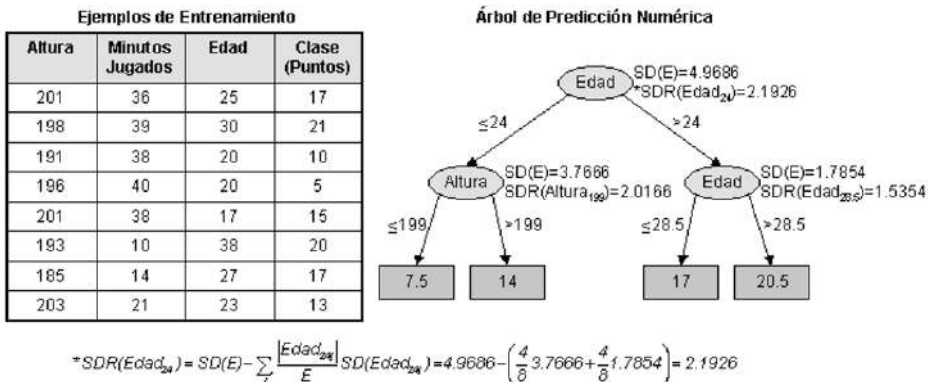


Fig. 6.10. Ejemplo de generación del árbol de predicción con M5.

En cada nodo del árbol se muestra la desviación típica de los ejemplos de entrenamiento que inciden en el nodo ($SD(E)$) y la desviación estándar del error para el atributo y el punto de corte que lo maximiza, de modo que es el seleccionado. Para obtener el atributo y el punto de corte se debe calcular la desviación estándar del error para cada posible punto de corte. En este caso, la finalización de la construcción del árbol

ocurre porque no se puede seguir subdividiendo, ya que en cada hoja hay dos ejemplos (número mínimo permitido). Por último, tras generar el árbol, en cada hoja se añade la media de los valores de la clase de los ejemplos que se clasifican a través de dicha hoja. Una vez se ha construido el árbol se va definiendo para cada nodo interior (no para las hojas con el fin de emplear el proceso de suavizado) un modelo lineal, concretamente una regresión lineal múltiple, tal y como se mostró anteriormente. Únicamente se emplean para realizar esta regresión aquellos atributos que se utilizan en el subárbol del nodo en cuestión.

A continuación, se pasa al proceso de poda, en el que se estima, para cada nodo, el error esperado en el conjunto de test. Para ello, lo primero que se hace es calcular la desviación de las predicciones del nodo con los valores reales de la clase para los ejemplos de entrenamiento que se clasifican por el mismo nodo. Sin embargo, dado que el árbol se ha construido con estos ejemplos, el error puede infravalorarse, con lo que se compensa con el factor $(n + v)/(n - v)$, donde n es el número de ejemplos de entrenamiento que se clasifican por el nodo actual y v es el número de parámetros del modelo lineal. De esta forma, la estimación del error en un conjunto I de ejemplos se realizaría con la ecuación 6.19.

$$e(I) = \frac{n + v}{n - v} \times \text{MAE} = \frac{n + v}{n - v} \times \frac{\sum_{i \in I} |y_i - \hat{y}_i|}{n} \quad \text{Ec. 6.19}$$

En esta ecuación, MAE es el error medio absoluto (*mean absolute error*) del modelo, donde y_i es el valor de la clase para el ejemplo i y \hat{y}_i la predicción del modelo para el mismo ejemplo. Para podar el árbol, se comienza por las hojas del mismo y se va comparando el error estimado para el nodo con el error estimado para los hijos del mismo, para lo cual se emplea la ecuación 6.20.

$$e(\text{subárbol}) = \frac{e(i)|i| + e(d)|d|}{n} \quad \text{Ec. 6.20}$$

En la ecuación 6.20, $e(i)$ y $e(d)$ son los errores estimados para los nodos hijo izquierdo y derecho, $|x|$ el número de ejemplos que se clasifica por el nodo x y n el número de ejemplos que se clasifica por el nodo padre. Comparando el error estimado para el nodo con el error estimado para el subárbol, se decide podar si no es menor el error para el subárbol.

El proceso explicado hasta el momento sirve para el caso de que los atributos sean numéricos pero, si los atributos son nominales, será preciso modificar el proceso: en primer lugar, se calcula el promedio de la clase en los ejemplos de entrenamiento para cada posible valor del atributo nominal, y se ordenan dichos valores de acuerdo a este promedio. Entonces, un atributo nominal con k posibles valores se transforma en $k-1$ atributos binarios. El i -ésimo atributo binario tendrá, para un ejemplo dado, un 0 si

el valor del atributo nominal es uno de los primeros i valores del orden establecido y un 1 en caso contrario. Con este proceso se logra tratar los atributos nominales como numéricos.

Al construir un árbol de modelos y definir para cada hoja un modelo lineal con los atributos del subárbol podado, suele ser beneficioso, sobre todo cuando se tiene un pequeño conjunto de entrenamiento, realizar un **proceso de suavizado** (*smoothing*) que compense las discontinuidades que ocurren entre modelos lineales adyacentes. Para ello, cuando se predice el valor de una instancia de test con el modelo lineal del nodo hoja correspondiente, este valor obtenido se filtra hacia atrás hasta el nodo hoja, suavizando dicho valor al combinarlo con el modelo lineal de cada nodo interior por el que pasa. Un modelo que se suele utilizar es el que se muestra en la ecuación 6.21.

$$p' = \frac{np + kq}{n + k} \quad \text{Ec. 6.21}$$

En esta ecuación, p es la predicción que llega al nodo (desde abajo), p' es la predicción filtrada hacia el nivel superior, q el valor obtenido por el modelo lineal de este nodo, n es el número de ejemplos que alcanzan el nodo inferior y k el factor de suavizado.

Finalmente, es conveniente determinar cómo se actuará frente a los atributos para los que faltan valores. En este caso, se modifica ligeramente la ecuación 6.18 para llegar hasta la ecuación 6.22.

$$\text{SDR} = \frac{c}{|E|} \left[\text{SD}(E) - \sum_i \frac{|E_i|}{|E|} \text{SD}(E_i) \right] \quad \text{Ec. 6.22}$$

En esta ecuación, c es el número de ejemplos con el atributo conocido. Una vez explicadas las características de los árboles de predicción numérica, se pasa a mostrar el algoritmo M5, cuyo pseudocódigo se recoge en la figura 6.11.

```
M5 (ejemplos) {
    SD = sd(ejemplos)
    Para cada atributo nominal con k-valores
        convertir en k-1 atributos binarios
    raíz = nuevo nodo
    raíz.ejemplos = ejemplos
    Dividir(raíz)
    Podar(raíz)
    Dibujar(raíz)
}
```

```

Dividir(nodo) {
    Si tamaño(nodo.ejemplos)<4 O sd(nodo.ejemplos)<=0.05*SD Entonces
        nodo.tipo = HOJA
    Si no
        nodo.tipo = INTERIOR
        Para cada atributo
            Para cada posible punto de división del atributo
                calcular el SDR del atributo
            nodo.atributo = atributo con mayor SDR
            Dividir(nodo.izquierda)
            Dividir(nodo.derecha)
}

Podar(nodo) {
    Si nodo = INTERIOR
        Podar(nodo.hijoizquierdo)
        Podar(nodo.hijoderecho)
    nodo.modelo = RegresionLineal(nodo)
    Si ErrorSubarbol(nodo) > Error(nodo) Entonces
        nodo.tipo = HOJA
}

ErrorSubarbol(nodo) {
    l = nodo.izquierda
    r = nodo.derecha
    Si nodo = INTERIOR Entonces
        ErrorSubarbol = (tamaño(l.ejemplos)*ErrorSubarbol(l) +
tamaño(r.ejemplos)*ErrorSubarbol(r))/tamaño(nodo.ejemplos)
    Si no
        ErrorSubarbol = error(nodo)
}

```

Fig. 6.11. Pseudocódigo del algoritmo M5.

La función “regresión lineal” generará la regresión correspondiente al nodo en el que nos encontramos. La función “error” evaluará el error del nodo mediante la ecuación 6.19.

6.4.2 Estimador de núcleos

Los estimadores de densidad de núcleo (*kernel density*) son estimadores no paramétricos, puesto que no parten de modelos de distribuciones de probabilidad conocidas para los datos. Destaca el método conocido como *de histograma*, por ser uno de los más utilizados, que tiene ciertas deficiencias relacionadas con la continuidad que llevaron a desarrollar otras técnicas. El estimador de núcleos fue propuesto por Rosenblatt en 1956 y Parzen en 1962 [DFL96]. La idea en la que se basan los estimadores de densidad de núcleo es la siguiente: si X es una variable aleatoria con función de distribución F y densidad f , entonces en cada punto de continuidad x de f se confirma la ecuación 6.23.

$$f(x) = \lim_{h \rightarrow 0} \frac{1}{2h} (F(x+h) - F(x-h)) \quad \text{Ec. 6.23}$$

Dada una muestra X_1, \dots, X_n proveniente de la distribución F , para cada h fijo, $F(x+h) - F(x-h)$ se puede estimar por la proporción de observaciones que están dentro del intervalo $(x-h, x+h)$. Por lo tanto, tomando h pequeño, lo que se muestra en la ecuación 6.24 es un estimador natural de la densidad, donde $\#A$ es el número de elementos del conjunto A .

$$\hat{f}_{n,h}(x) = \frac{1}{2hn} \{X_i: X_i \in (x-h, x+h)\} \quad \text{Ec. 6.24}$$

Otra manera de expresar este estimador es considerando la función de peso w definida como se muestra en la ecuación siguiente, de manera que el estimador de la densidad f en el punto x se puede expresar como se expresa en dicha ecuación.

$$w(x) = \begin{cases} 1/2 & \text{si } |x| < 1 \\ 0 & \text{en c. c.} \end{cases} \quad \text{Ec. 6.25}$$

$$\hat{f}_{n,h}(x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{h} w\left(\frac{x - X_i}{h}\right) \quad \text{Ec. 6.26}$$

Pero este estimador no es una función continua, ya que tiene saltos en los puntos $X_i \pm h$ y su derivada es 0 en todos los otros puntos. Por ello se ha sugerido reemplazar a la función w por funciones más suaves K , llamadas *núcleos*, lo que da origen a los estimadores de núcleos. El estimador de núcleos de una función de densidad f calculado a partir de una muestra aleatoria X_1, \dots, X_n de dicha densidad se define según la ecuación 6.27.

$$\hat{f}_{n,h}(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - X_i}{h}\right) \quad \text{Ec. 6.27}$$

En esta ecuación, la función K se elige generalmente entre las funciones de densidad conocidas, por ejemplo gaussiana, que se muestra en la ecuación siguiente, donde σ es la desviación típica de la distribución y μ la media.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{Ec. 6.28}$$

El otro parámetro de la ecuación 6.27 es h , llamado *ventana*, parámetro de suavizado o ancho de banda, el cual determina las propiedades estadísticas del estimador: el sesgo crece y la varianza decrece con h [HAL94]. Es decir, que si h es grande, los estimadores están sobresuavizados y son sesgados, y si h es pequeño, los estimadores resultantes están subsuavizados, lo que equivale a decir que su varianza es grande.

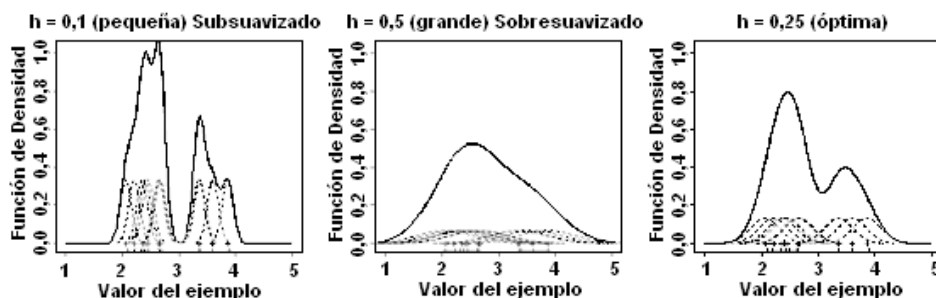


Fig. 6.12. Importancia del parámetro “tamaño de ventana” en el estimador de núcleos.

A pesar de que la elección del núcleo K determina la forma de la densidad estimada, la literatura sugiere que esta elección no es crítica, al menos entre las alternativas usuales [DEA97]. Más importante es la elección del tamaño de ventana. En la figura 6.12 se muestra cómo un valor pequeño para este factor hace que la función de distribución generada esté subsuavizada. Mientras, al emplear un h demasiado grande provoca el sobresuavizado de la función de distribución. Por último, empleando el h óptimo se obtiene la función de distribución adecuada.

Para determinar un ancho de banda con el cual comenzar, una alternativa es calcular el ancho de banda óptimo si se supone que la densidad tiene una forma específica. La ventana óptima en el sentido de minimizar el error medio cuadrático integrado, definido

como la esperanza de la integral del error cuadrático sobre toda la densidad, fue calculada por Bowman [BOW85] y Silverman [SIL86] y depende de la verdadera densidad f y del núcleo K . Al suponer que ambos, la densidad y el núcleo, son normales, la ventana óptima resulta ser la que se muestra en la ecuación 6.29.

$$h^* = 1.06 \sigma n^{-1/5} \quad \text{Ec. 6.29}$$

En la ecuación 6.29, σ es la desviación típica de la densidad. La utilización de esta h será adecuada si la población se asemeja en su distribución a la de la normal; sin embargo, si trabajamos con poblaciones multimodales se producirá una sobresuavización de la estimación. Por ello, el mismo autor sugiere utilizar medidas robustas de dispersión en lugar de σ , con lo cual el ancho de banda óptimo se obtiene como se muestra en la ecuación 6.30.

$$h^* = 1.06 \min(\sigma, 0.75 \text{ IQR}) n^{-1/5} \quad \text{Ec. 6.30}$$

En esta ecuación, IQR es el rango intercuartílico, esto es, la diferencia entre los percentiles 75 y 25 [DEA97].

Una vez definidos todos los parámetros a tener en cuenta para emplear un estimador de núcleos, hay que definir cómo se obtiene, a partir del mismo, el valor de la variable a predecir, y , en función del valor de la variable dependiente, x . Esto se realiza mediante el estimador de Nadaraya-Watson, que se muestra en la ecuación 6.31.

$$\hat{m}(x) = E(Y|X = x) = \frac{\sum_{i=1}^n K\left(\frac{x-X_i}{h}\right) Y_i}{\sum_{r=1}^n K\left(\frac{x-X_r}{h}\right)} \quad \text{Ec. 6.31}$$

En la ecuación 6.31 es el valor del atributo dependiente a partir del cual se debe obtener el valor de la variable independiente y ; Y_i es el valor del atributo independiente para el ejemplo de entrenamiento i .

Una vez completada la explicación de cómo aplicar los estimadores de núcleos para predecir el valor de una clase numérica, se muestra, en la figura 6.13, un ejemplo de su utilización tomando la variable “temperatura” como predictora y la variable “humedad” como dependiente o a predecir.

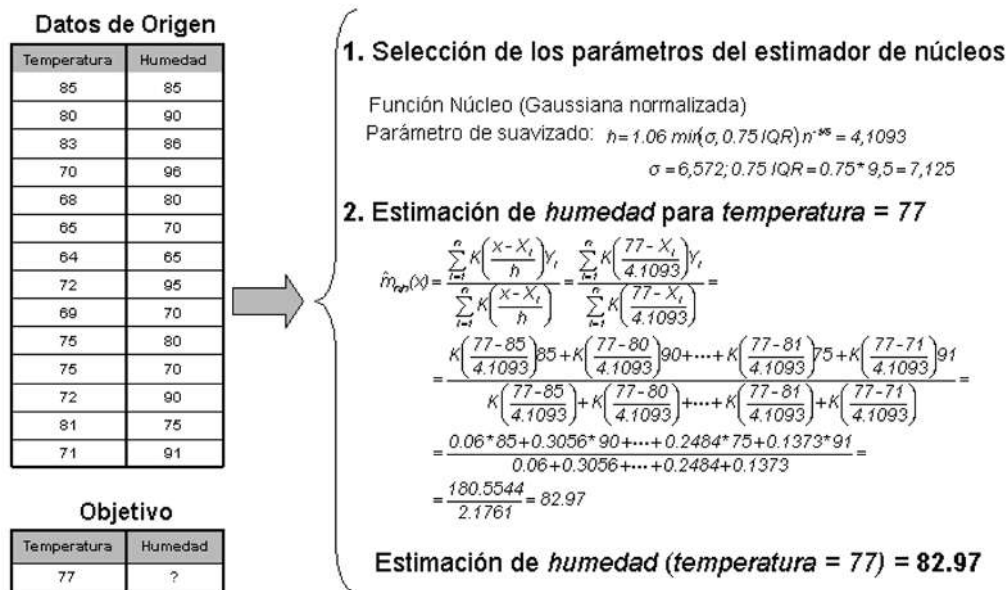


Fig. 6.13. Ejemplo de predicción con un estimador de núcleos.

En primer lugar se definen los parámetros que se van a emplear para el estimador de núcleos: la función núcleo y el parámetro de suavizado. Posteriormente se puede realizar la predicción, que en este caso consiste en predecir el valor del atributo “humedad” sabiendo que la temperatura es igual a 77. Después de completar el proceso se determina que el valor de la humedad es igual a 82.97.

6.4.2.1 Aplicación a problemas multivariantes

Hasta el momento se han explicado las bases sobre las que se sustentan los estimadores de núcleos, pero en los problemas reales no es una única variable la que debe tratarse, sino que han de tenerse en cuenta un número indeterminado de variables. Por ello, es necesario ampliar el modelo explicado para permitir la introducción de d variables. Así, supongamos n ejemplos X_i , siendo X_i un vector d -dimensional. El estimador de núcleos de la función de densidad f calculado a partir de la muestra aleatoria X_1, \dots, X_n de dicha densidad se define como se muestra en la ecuación 6.32.

$$\hat{f}_{n,H}(x) = \frac{1}{n|H|} \sum_{i=1}^n K(H^{-1}(x - X_i)) \quad \text{Ec. 6.32}$$

Tal y como puede verse, la ecuación 2.42 es una mera ampliación de la ecuación 2.37: en este caso, H no es ya un único valor numérico, sino una matriz simétrica y definida positiva de orden $d \times d$, denominada *matriz de anchos de ventana*. Por su parte, K es generalmente una función de densidad multivariante. Por ejemplo, la función gaussiana normalizada en este caso pasaría a ser la que se muestra en la ecuación 6.33.

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{\mathbf{x}^T \mathbf{x}}{2}} \quad \text{Ec. 6.33}$$

De nuevo, es más importante definir una correcta matriz H que la función núcleo elegida. También el estimador de Nadaraya-Watson, que se muestra en la ecuación 6.34, como una ampliación del caso anterior.

$$\hat{m}(\mathbf{x}) = E(Y|X = \mathbf{x}) = \frac{\sum_{i=1}^n K(H^{-1}(\mathbf{x} - \mathbf{X}_i)) Y_i}{\sum_{i=1}^n K(H^{-1}(\mathbf{x} - \mathbf{X}_i))} \quad \text{Ec. 6.34}$$

Tal y como se ve en la ecuación 6.34, el cambio radica en que se tiene una matriz de anchos de ventana en lugar de un único valor de ancho de ventana.

6.4.2.2 Aplicación a problemas de clasificación

Si bien los estimadores de núcleo son diseñados para la predicción numérica, también pueden utilizarse para la clasificación. En este caso, se dispone de un conjunto de c clases a las que puede pertenecer un ejemplo determinado. Y estos ejemplos se componen de d variables o atributos. Se puede estimar la densidad de la clase j mediante la ecuación 6.35, en la que n_j es el número de ejemplos de entrenamiento que pertenecen a la clase j , Y_i^j será 1 si el ejemplo i pertenece a la clase j y 0 en otro caso, K vuelve a ser la función núcleo y h el ancho de ventana. En este caso se ha realizado la simplificación del modelo multivariante, empleando, en lugar de una matriz de anchos de ventana, un único valor escalar porque es el modelo que se utiliza en la implementación que realiza Weka de los estimadores de núcleo.

$$\hat{f}_{n,H}(\mathbf{x}) = \frac{1}{n_j} \sum_{i=1}^n Y_i^j h^{-d} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) \quad \text{Ec. 6.35}$$

La probabilidad *a priori* de que un ejemplo pertenezca a la clase j es igual a $P_j = n_j/n$. Se puede estimar la probabilidad *a posteriori*, definida mediante $q_j(\mathbf{x})$, de que el ejemplo pertenezca a j , tal y como se muestra en la ecuación 6.36.

$$q_j(x) = \frac{P_j \hat{f}_j(x)}{\hat{f}(x)} \approx \frac{P_j \hat{f}_j(x)}{\sum_{k=1}^c P_k \hat{f}_k(x)} = \frac{\sum_{i=1}^n Y_i^j h^{-d} K\left(\frac{x-X_i}{h}\right)}{\sum_{r=1}^n h^{-d} K\left(\frac{x-X_r}{h}\right)} = \hat{q}_j(x) \quad \text{Ec. 6.36}$$

De esta forma, el estimador aquí es idéntico al estimador de Nadayara-Watson presentado anteriormente. Por último, se muestra un ejemplo de la aplicación de un estimador de núcleos a un problema de clasificación: se trata del problema planteado en la tabla siguiente, y más concretamente se trata de predecir el valor de la clase “jugar” a partir únicamente del atributo numérico “temperatura”. Este ejemplo se muestra en la figura siguiente.

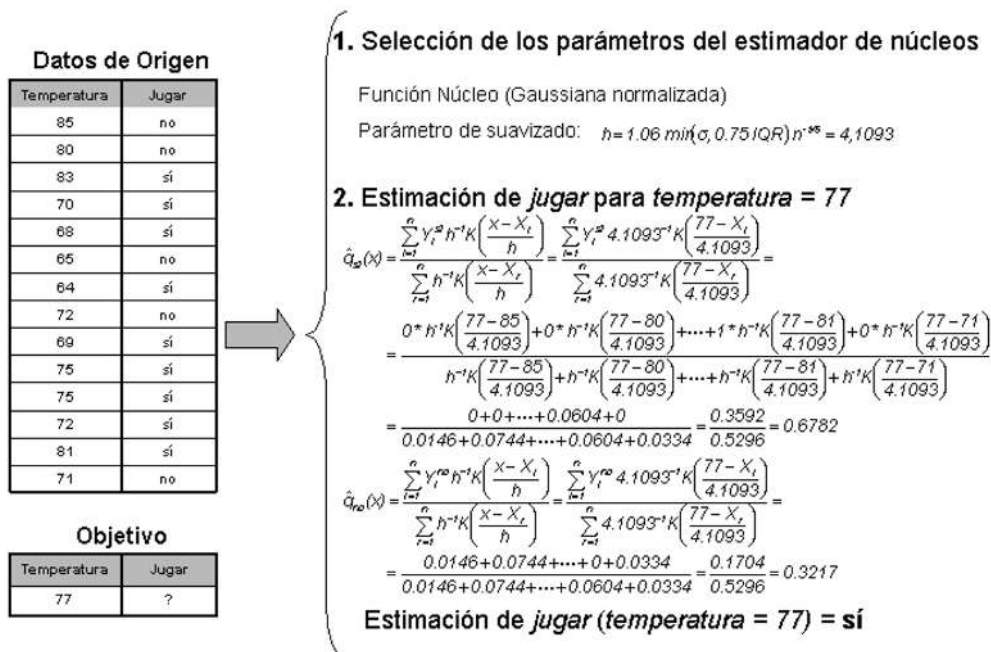


Fig. 6.14. Ejemplo de clasificación mediante un estimador de núcleos.

Al igual que para el problema de predicción, en primer lugar se definen los parámetros del estimador de núcleos para, posteriormente, estimar la clase a la que pertenece el ejemplo de test. En este caso se trata de predecir si se puede jugar o no al tenis teniendo en cuenta que la temperatura es igual a 77. Y la conclusión a la que se llega utilizando el estimador de núcleos es que sí se puede jugar.

6.5 Clasificación

La clasificación es el proceso de dividir un conjunto de datos en grupos mutuamente excluyentes [WK91, LAN96, MIT97], de tal forma que cada miembro de un grupo esté lo más cerca posible de otros y grupos diferentes estén lo más lejos posible de otros, donde la distancia se mide con respecto a las variables especificadas que se quieren predecir.

Ejemplo	Vista	Temperatura	Humedad	Viento	Jugar
1	Soleado	Alta (85)	Alta (85)	No	No
2	Soleado	Alta (80)	Alta (90)	Sí	No
3	Nublado	Alta (83)	Alta (86)	No	Sí
4	Lluvioso	Media (70)	Alta (96)	No	Sí
5	Lluvioso	Baja (68)	Normal (80)	No	Sí
6	Lluvioso	Baja (65)	Normal (70)	Sí	No
7	Nublado	Baja (64)	Normal (65)	Sí	Sí
8	Soleado	Media (72)	Alta (95)	No	No
9	Soleado	Baja (69)	Normal (70)	No	Sí
10	Lluvioso	Media (75)	Normal (80)	No	Sí
11	Soleado	Media (75)	Normal (70)	Sí	Sí
12	Nublado	Media (72)	Alta (90)	Sí	Sí
13	Nublado	Alta (81)	Normal (75)	No	Sí
14	Lluvioso	Media (71)	Alta (91)	Sí	No

Fig. 6.15. Ejemplo de problema de clasificación.

El ejemplo empleado tiene dos atributos, temperatura y humedad, que pueden emplearse como simbólicos o numéricos. Entre paréntesis se presentan sus valores numéricos.

En los siguientes apartados se presentan y explican las principales técnicas de clasificación. Además, se mostrarán ejemplos que permiten observar el funcionamiento del algoritmo, para lo que se utilizará la tabla de la figura 6.15, que presenta un sencillo problema de clasificación consistente en, a partir de los atributos que modelan el tiempo (vista, temperatura, humedad y viento), determinar si se puede o no jugar al tenis.

6.5.1 Tabla de decisión

La tabla de decisión constituye la forma más simple y rudimentaria de representar la salida de un algoritmo de aprendizaje, que es justamente representarlo como la entrada.

Estos algoritmos consisten en seleccionar subconjuntos de atributos y calcular su precisión (*accuracy*) para predecir o clasificar los ejemplos. Una vez seleccionado el mejor de los subconjuntos, la tabla de decisión estará formada por los atributos seleccionados (más la clase) en la que se insertarán todos los ejemplos de entrenamiento únicamente con el subconjunto de atributos elegido. Si hay dos ejemplos con exactamente los mismos pares atributo-valor para todos los atributos del subconjunto, la clase que se

elija será la media de los ejemplos (en el caso de una clase numérica) o la que mayor probabilidad de aparición tenga (en el caso de una clase simbólica).

La precisión de un subconjunto S de atributos para todos los ejemplos de entrenamientos se calculará mediante la ecuación 6.37, para el caso de que la clase sea simbólica, o mediante la ecuación 6.38, en el caso de que la clase sea numérica:

$$\text{precisión}(S) = \frac{\text{ejemplos bien clasificados}}{\text{ejemplos totales}} \quad \text{Ec. 6.37}$$

$$\text{precisión}(S) = -\text{RMSE} = -\sqrt{\frac{\sum_{i \in I} (y_i - \hat{y}_i)^2}{n}} \quad \text{Ec. 6.38}$$

Donde RMSE es la raíz cuadrada del error cuadrático medio (*root mean square error*), n es el número de ejemplos totales, y_i el valor de la clase para el ejemplo i e \hat{y}_i el valor predicho por el modelo para el ejemplo i . Como ejemplo de tabla de decisión, simplemente se puede utilizar la propia tabla de la figura 6.15, dado que si se comenzasen a combinar atributos y a probar la precisión de dicha combinación, se obtendría como resultado que los cuatro atributos deben emplearse, con lo que la tabla de salida sería la misma. Esto no tiene por qué ser así, ya que en otros problemas no serán necesarios todos los atributos para generar la tabla de decisión, como ocurre en el ejemplo de la tabla de la figura 6.16, donde se dispone de un conjunto de entrenamiento en el que aparecen los atributos “sexo” y “tipo” (tipo de profesor) y la clase a determinar es si el tipo de contrato es o no fijo.

Ejemplo Nº	Atributos		Clase
	Sexo	Tipo	Fijo
1	Hombre	Asociado	No
2	Mujer	Catedrático	Sí
3	Hombre	Titular	Sí
4	Mujer	Asociado	No
5	Hombre	Catedrático	Sí
6	Mujer	Asociado	No
7	Hombre	Ayudante	No
8	Mujer	Titular	Sí
9	Hombre	Asociado	No
10	Mujer	Ayudante	No
11	Hombre	Asociado	No

Fig. 6.16. Determinación del tipo de contrato.

Si se toma como primer subconjunto el formado por el atributo “sexo” y se eliminan las repeticiones resulta la tabla de la figura 6.17.

Ejemplo Nº	Sexo	Fijo
1	Hombre	No
2	Mujer	Sí
3	Hombre	Sí
4	Mujer	No

Fig. 6.17. Subconjunto 1.

Con lo que se pone de manifiesto que la probabilidad de clasificar bien es del 50 %. Si por el contrario se elimina el atributo “sexo”, quedará la tabla de la figura 6.18.

Ejemplo Nº	Tipo	Fijo
1	Asociado	No
2	Catedrático	Sí
3	Titular	Sí
7	Ayudante	No

Fig. 6.18. Subconjunto 2.

Que tiene una precisión de aciertos del 100 %, de modo que se deduce que esta última tabla es la que se debe tomar como tabla de decisión. El resultado es lógico ya que el atributo “sexo” es irrelevante a la hora de determinar si el contrato es o no fijo.

6.5.2 Árboles de decisión

El aprendizaje de árboles de decisión está englobado como una metodología del aprendizaje supervisado. La representación que se utiliza para las descripciones del concepto adquirido es el árbol de decisión, que consiste en una representación del conocimiento relativamente simple y que es una de las causas por la que los procedimientos utilizados en su aprendizaje son más sencillos que los de sistemas que utilizan lenguajes de representación más potentes, como redes semánticas, representaciones en lógica de primer orden, etc. No obstante, la potencia expresiva de los árboles de decisión es también menor que la de esos otros sistemas. El aprendizaje de árboles de decisión suele ser más robusto frente al ruido y conceptualmente sencillo, aunque los sistemas que han resultado del perfeccionamiento y de la evolución de los más antiguos se complican con los procesos que incorporan para ganar fiabilidad. La mayoría de los sistemas de aprendizaje de árboles suelen ser no incrementales, pero existe alguna excepción [UTG88].

El primer sistema que construía árboles de decisión fue CLS de Hunt, desarrollado en 1959 y depurado a lo largo de los años sesenta. CLS es un sistema desarrollado por psicólogos como un modelo del proceso cognitivo de formación de conceptos

sencillos. Su contribución fundamental fue la propia metodología, pero no resultaba computacionalmente eficiente debido al método que empleaba en la extensión de los nodos. Se guiaba por una estrategia similar al *minimax* con una función que integraba diferentes costes.

En 1979, Quinlan desarrolla el sistema ID3 [QUIN79], que él denominaría simplemente *herramienta* porque lo consideraba experimental. Conceptualmente es fiel a la metodología de CLS, pero le aventaja en el método de expansión de los nodos, basado en una función que utiliza la medida de la información de Shannon. La versión definitiva, presentada por su autor, Quinlan, como un sistema de aprendizaje, es el sistema C4.5 que expone con cierto detalle en la obra *C4.5: Programs for Machine Learning* [QUIN93]. La evolución (comercial) de ese sistema es otro denominado C5 del mismo autor, del que se puede obtener una versión de demostración restringida en cuanto a capacidades; por ejemplo, el número máximo de ejemplos de entrenamiento.

Representación de un árbol de decisión

Un árbol de decisión [MUR98] puede interpretarse esencialmente como una serie de reglas compactadas para su representación en forma de árbol. Dado un conjunto de ejemplos, estructurados como vectores de pares ordenados atributo-valor, de acuerdo con el formato general en el aprendizaje inductivo a partir de ejemplos, el concepto que estos sistemas adquieren durante el proceso de aprendizaje consiste en un árbol. Cada eje está etiquetado con un par atributo-valor y las hojas con una clase, de forma que la trayectoria que determinan desde la raíz los pares de un ejemplo de entrenamiento alcanza una hoja etiquetada (normalmente) con la clase del ejemplo. La clasificación de un ejemplo nuevo del que se desconoce su clase se hace con la misma técnica, solamente que en ese caso al atributo clase, cuyo valor se desconoce, se le asigna de acuerdo con la etiqueta de la hoja a la que se accede con ese ejemplo.

Problemas apropiados para este tipo de aprendizaje

Las características de los problemas apropiados para resolver mediante este aprendizaje dependen del sistema de aprendizaje específico utilizado, pero hay una serie de ellas generales y comunes a la mayoría y que se describen a continuación:

- Que la representación de los ejemplos sea mediante vectores de pares atributo-valor, especialmente cuando los valores son disjuntos y en un número pequeño. Los sistemas actuales están preparados para tratar atributos con valores continuos, valores desconocidos e incluso valores con una distribución de probabilidad.
- Que el atributo que hace el papel de la clase sea de tipo discreto y con un número pequeño de valores, sin embargo, existen sistemas que adquieren como concepto aprendido funciones con valores continuos.
- Que las descripciones del concepto adquirido deban ser expresadas en forma normal disyuntiva.

- Que posiblemente existan errores de clasificación en el conjunto de ejemplos de entrenamiento, así como valores desconocidos en algunos de los atributos en algunos ejemplos. Estos sistemas, por lo general, son robustos frente a los errores del tipo mencionado.

A continuación se presentan tres algoritmos de árboles de decisión, los dos primeros diseñados por Quinlan [QUIN86, QUIN93], los sistemas ID3 y C4.5; y el tercero un árbol de decisión muy sencillo, con un único nivel de decisión.

El método ID3

El método ID3 [QUIN86] es un algoritmo simple y, sin embargo, potente, cuya misión es la elaboración de un árbol de decisión. El procedimiento para generar un árbol de decisión consiste, como se comentó anteriormente, en seleccionar un atributo como raíz del árbol y crear una rama con cada uno de los posibles valores de dicho atributo. Con cada rama resultante (nuevo nodo del árbol), se realiza el mismo proceso, esto es, se selecciona otro atributo y se genera una nueva rama para cada posible valor del atributo. Este procedimiento continúa hasta que los ejemplos se clasifiquen a través de uno de los caminos del árbol. El nodo final de cada camino será un nodo hoja, al que se le asignará la clase correspondiente. Así, el objetivo de los árboles de decisión es obtener reglas o relaciones que permitan clasificar a partir de los atributos.

En cada nodo del árbol de decisión se debe seleccionar un atributo para seguir dividiendo, y el criterio que se toma para elegirlo es: se selecciona el atributo que mejor separe (ordene) los ejemplos de acuerdo a las clases. Para ello se emplea la entropía, que es una medida de cómo está ordenado el universo. La teoría de la información (basada en la entropía) calcula el número de bits (información, preguntas sobre atributos) que hace falta suministrar para conocer la clase a la que pertenece un ejemplo. Cuanto menor sea el valor de la entropía, menor será la incertidumbre y más útil será el atributo para la clasificación. La definición de *entropía* que da Shannon en su *Teoría de la información* (1948) es: dado un conjunto de eventos $A=\{A_1, A_2, \dots, A_n\}$, con probabilidades $\{p_1, p_2, \dots, p_n\}$, la información en el conocimiento de un suceso A_i (bits) se define en la ecuación 6.39, mientras que la información media de A (bits) se muestra en la ecuación 6.40.

$$I(A_i) = \log_2 \left(\frac{1}{p_i} \right) = -\log_2(p_i) \quad \text{Ec. 6.39}$$

$$I(A) = \sum_{i=1}^n p_i I(A_i) = - \sum_{i=1}^n p_i \log_2(p_i) \quad \text{Ec. 6.40}$$

Si aplicamos la entropía a los problemas de clasificación se puede medir lo que se discrimina (se gana por usar) un atributo A_i empleando para ello la ecuación 6.41, en la que se define la ganancia de información.

$$G(A_i) = I - I(A_i) \quad \text{Ec. 6.41}$$

Siendo I la información antes de utilizar el atributo e $I(A_i)$ la información después de utilizarlo. Se definen ambas en las ecuaciones 6.42 y 6.43.

$$I = - \sum_{c=1}^{nc} \frac{n_c}{n} \log_2 \left(\frac{n_c}{n} \right) \quad \text{Ec. 6.42}$$

$$I(A_i) = \sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} I_{ij}; \quad I_{ij} = - \sum_{k=1}^{nc} \frac{n_{ijk}}{n_{ij}} \log_2 \left(\frac{n_{ijk}}{n_{ij}} \right) \quad \text{Ec. 6.43}$$

En estas ecuaciones, nc será el número de clases y n_c el número de ejemplares de la clase c , siendo n el número total de ejemplos. Será $nv(A_i)$ el número de valores del atributo A_i , n_{ij} el número de ejemplos con el valor j en A_i y n_{ijk} el número de ejemplos con valor j en A_i y que pertenecen a la clase k . Una vez explicada la heurística empleada para seleccionar el mejor atributo en un nodo del árbol de decisión, se muestra el algoritmo ID3:

1. Seleccionar el atributo A_i que maximice la ganancia $G(A_i)$.
2. Crear un nodo para ese atributo con tantos sucesores como valores tenga.
3. Introducir los ejemplos en los sucesores según el valor que tenga el atributo A_i .
4. Por cada sucesor:
 - a. Si sólo hay ejemplos de una clase, C_k , entonces etiquetarlo con C_k .
 - b. Si no, llamar a ID3 con una tabla formada por los ejemplos de ese nodo, eliminando la columna del atributo A_i .

Fig. 6.19. Pseudocódigo del algoritmo ID3.

Por último, en la figura 6.20 se representa el proceso de generación del árbol de decisión para el problema planteado en la tabla de la figura 6.15.

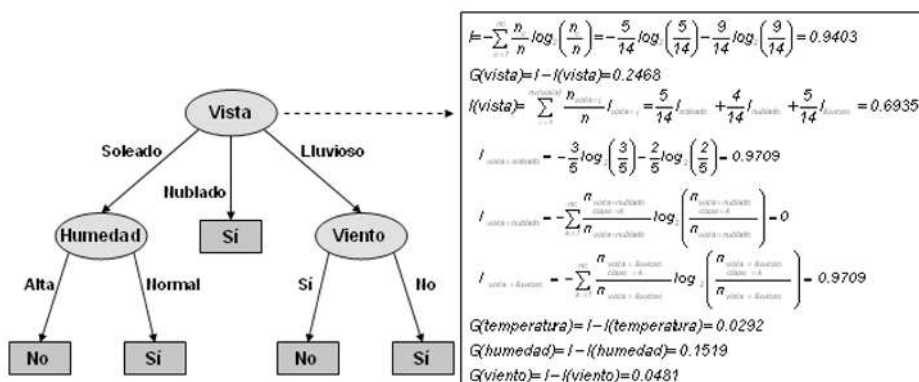


Fig. 6.20. Ejemplo de clasificación con ID3.

En la figura 6.20 se muestra el árbol de decisión que se generaría con el algoritmo ID3. Además, para el primer nodo del árbol se muestra cómo se llega a decidir que el mejor atributo para dicho nodo es “vista”. Se generan nodos para cada valor del atributo y, en el caso de “vista = Nublado” se llega a un nodo hoja, ya que todos los ejemplos de entrenamiento que llegan a dicho nodo son de clase “Si”. Sin embargo, para los otros dos casos se repite el proceso de elección con el resto de atributos y con los ejemplos de entrenamiento que se clasifican a través de ese nodo.

El algoritmo C4.5

El ID3 es capaz de tratar con atributos cuyos valores sean discretos o continuos. En el primer caso, el árbol de decisión generado tendrá tantas ramas como valores posibles tome el atributo. Si los valores del atributo son continuos, el ID3 no clasifica correctamente los ejemplos dados. Por ello, Quinlan [QUIN93] propuso el C4.5, como extensión del ID3, que permite:

- Empleo del concepto razón de ganancia (GR, *Gain Ratio*).
- Construir árboles de decisión cuando algunos de los ejemplos presentan valores desconocidos para algunos de los atributos.
- Trabajar con atributos que presenten valores continuos.
- La “poda” de los árboles de decisión [QUIN87, QR89].
- Obtención de reglas de clasificación.

Razón de ganancia

El test basado en el criterio de maximizar la ganancia tiene como sesgo la elección de atributos con muchos valores. Esto es debido a que cuanto más fina sea la participación producida por los valores del atributo, normalmente, la incertidumbre o entropía en cada nuevo nodo será menor y, por lo tanto, también será menor la media de la entro-

pía a ese nivel. C4.5 modifica el criterio de selección del atributo empleando en lugar de la ganancia la razón de ganancia, cuya definición se muestra en la ecuación 6.44.

$$GR(A_i) = \frac{G(A_i)}{I(\text{División } A_i)} = \frac{G(A_i)}{-\sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} \log_2 \left(\frac{n_{ij}}{n} \right)} \quad \text{Ec. 6.44}$$

Al término $I(\text{División } A_i)$ se le denomina *información de ruptura*. En esta medida, cuando n_{ij} tiende a n , el denominador se hace 0. Esto es un problema aunque, según Quinlan, la razón de ganancia elimina el sesgo.

Valores desconocidos

El sistema C4.5 admite ejemplos con atributos desconocidos tanto en el proceso de aprendizaje como en el de validación. Para calcular durante el proceso de aprendizaje la razón de ganancia de un atributo con valores desconocidos, se redefinen sus dos términos, la ganancia, ecuación 6.45, y la información de ruptura, ecuación 6.46.

$$G(A_i) = \frac{n_{ic}}{n} (I - I(A_i)) \quad \text{Ec. 6.45}$$

$$I(\text{División } A_i) = - \left(\sum_{j=1}^{nv(A_i)} \frac{n_{ij}}{n} \log_2 \left(\frac{n_{ij}}{n} \right) \right) - \frac{n_{id}}{n} \log_2 \left(\frac{n_{id}}{n} \right) \quad \text{Ec. 6.46}$$

En estas ecuaciones, n_{ic} es el número de ejemplos con el atributo i conocido, y n_{id} el número de ejemplos con valor desconocido en el mismo atributo. Además, para el cálculo de la entropía $I(A_i)$ se tendrán en cuenta únicamente los ejemplos en los que el atributo A_i tenga un valor definido.

No se toma el valor desconocido como significativo, sino que se supone una distribución probabilística del atributo de acuerdo con los valores de los ejemplos en la muestra de entrenamiento. Cuando se entrena, los casos con valores desconocidos se distribuyen con pesos de acuerdo a la frecuencia de aparición de cada posible valor del atributo en el resto de ejemplos de entrenamiento. El peso ω_{ij} con que un ejemplo i se distribuiría desde un nodo etiquetado con el atributo A hacia el hijo con valor j en dicho atributo se calcula mediante la ecuación 6.47, en la que ω_i es el peso del ejemplo i al llegar al nodo, esto es, antes de distribuirse, y $p(A=j)$ la suma de pesos de todos los ejemplos del nodo con valor j en el atributo A entre la suma total de pesos de todos los ejemplos del nodo (ω).

$$\omega_{ij} = \omega_i p(A = j) = \omega_i \frac{\omega_{A=j}}{\omega} \quad \text{Ec. 6.47}$$

En cuanto a la clasificación de un ejemplo de test, si se alcanza un nodo con un atributo que el ejemplo no tiene (desconocido), se distribuye el ejemplo (divide) en tantos casos como valores tenga el atributo, y se da un peso a cada resultado con el mismo criterio que en el caso del entrenamiento: la frecuencia de aparición de cada posible valor del atributo en los ejemplos de entrenamiento. El resultado de esta técnica es una clasificación con probabilidades, correspondientes a la distribución de ejemplos en cada nodo hoja.

Atributos continuos

El tratamiento que realiza C4.5 de los atributos continuos está basado en la ganancia de información, al igual que ocurre con los atributos discretos. Si un atributo continuo A_i presenta los valores ordenados v_1, v_2, \dots, v_n , se comprueba cuál de los valores $z_j = (v_j + v_{j+1})/2$; $1 \leq j < n$ supone una ruptura del intervalo $[v_1, v_n]$ en dos subintervalos $[v_1, z_j]$ y $[z_j, v_n]$ con mayor ganancia de información. El atributo continuo, ahora con dos únicos valores posibles, entrará en competencia con el resto de los atributos disponibles para expandir el nodo.

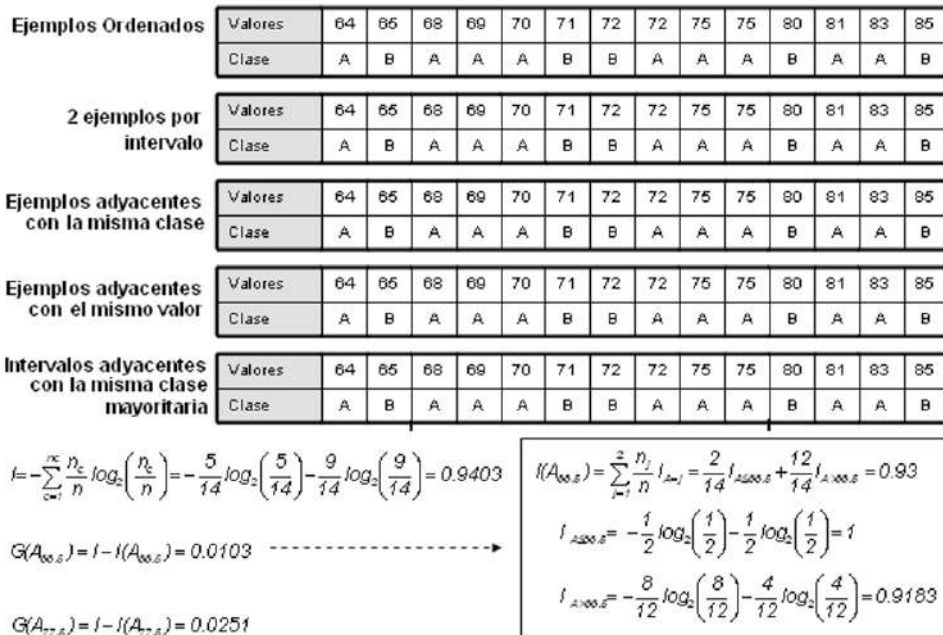


Fig. 6.21. Ejemplo de tratamiento de atributos continuos con C4.5.

Para mejorar la eficiencia del algoritmo no se consideran todos los posibles puntos de corte, sino que se tienen en cuenta las siguientes reglas:

- Cada subintervalo debe tener un número mínimo de ejemplos (por ejemplo, 2).
- No se divide el intervalo si el siguiente ejemplo pertenece a la misma clase que el actual.
- No se divide el intervalo si el siguiente ejemplo tiene el mismo valor que el actual.
- Se unen subintervalos adyacentes si tienen la misma clase mayoritaria.

Como se ve en el ejemplo de la figura 6.21, aplicando las reglas anteriores sólo es preciso probar dos puntos de corte (66,5 y 77,5), mientras que si no se empleara ninguna de las mejoras que se comentaron anteriormente se deberían haber probado un total de trece puntos. Como se ve en esta figura, finalmente se tomaría como punto de ruptura el 77,5, dado que obtiene una mejor ganancia. Una vez seleccionado el punto de corte, este atributo numérico competiría con el resto de atributos. Si bien aquí se ha empleado la ganancia, realmente se emplearía la razón de ganancia, pero no afecta a la elección del punto de corte. Cabe mencionar que ese atributo no deja de estar disponible en niveles inferiores como en el caso de los discretos, aunque con sus valores restringidos al intervalo que domina el camino.

Poda del árbol de decisión

El árbol de decisión ha sido construido a partir de un conjunto de ejemplos, por tanto, reflejará correctamente todo el grupo de casos. Sin embargo, como esos ejemplos pueden ser muy diferentes entre sí, el árbol resultante puede llegar a ser bastante complejo, con trayectorias largas y muy desiguales. Para facilitar la comprensión del árbol puede realizarse una poda del mismo. C4.5 efectúa la poda después de haber desarrollado el árbol completo (pospoda), a diferencia de otros sistemas que realizan la construcción del árbol y la poda a la vez (prepoda); es decir, estiman la necesidad de seguir desarrollando un nodo aunque no posea el carácter de hoja. En C4.5 el proceso de podado comienza en los nodos hoja y recursivamente continúa hasta llegar al nodo raíz. Se consideran dos operaciones de poda en C4.5: reemplazo de subárbol por hoja (*subtree replacement*) y elevación de subárbol (*subtree raising*). En la figura 6.22 se muestra en qué consiste cada tipo de poda.

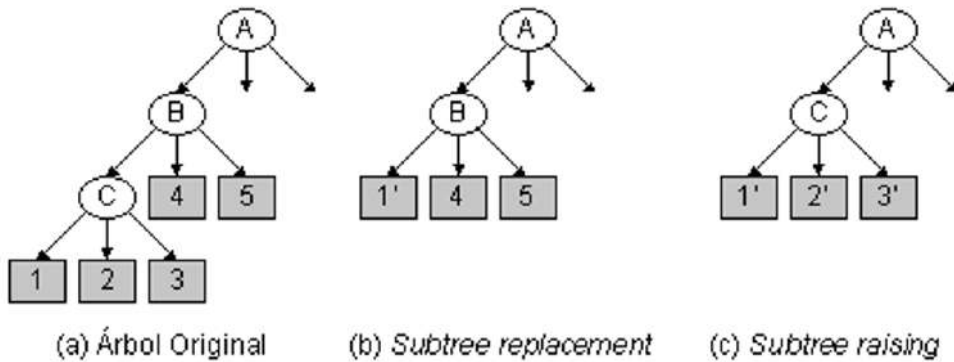


Fig. 6.22. Tipos de operaciones de poda en C4.5.

En esta figura tenemos el árbol original antes del podado (a), y las dos posibles acciones de podado a realizar sobre el nodo interno C. En (b) se realiza *subtree replacement*, en cuyo caso el nodo C es reemplazado por uno de sus subárboles. Por último, en (c) se realiza *subtree raising*: el nodo B es sustituido por el subárbol con raíz C. En este último caso, hay que tener en cuenta que habrá que reclasificar de nuevo los ejemplos a partir del nodo C. Además, *subtree raising* es muy costoso computacionalmente hablando, de forma que se suele restringir su uso al camino más largo a partir del nodo (hasta la hoja) que estamos podando. Como se comentó anteriormente, el proceso de podado comienza en las hojas y continúa hacia la raíz, pero la cuestión es cómo decidir reemplazar un nodo interno por una hoja (*replacement*) o reemplazar un nodo interno por uno de sus nodos hijo (*raising*). Lo que se hace es comparar el error estimado de clasificación en el nodo en el que nos encontramos y compararlo con el error en cada uno de sus hijos y en su padre para realizar alguna de las operaciones o ninguna. En la figura 6.23 se muestra el pseudocódigo del proceso de podado que se emplea en C4.5.

```

Podar (raíz) {
    Si raíz No es HOJA Entonces
        Para cada hijo H de raíz Hacer
            Podar (H)

    Obtener Brazo más largo (B) de raíz // raising
    ErrorBrazo = EstimarErrorArbol (B, raíz.ejemplos)

    ErrorHoja = EstimarError (raíz, raíz.ejemplos) // replacement

    ErrorÁrbol = EstimarErrorArbol (raíz, raíz.ejemplos)
}

```

```

    Si ErrorHoja <= ErrorÁrbol Entonces // replacement
        raíz es Hoja
    Fin Poda

    Si ErrorBrazo <= ErrorÁrbol Entonces // raising
        raíz = B
        Podar (raíz)
}

EstimarErrorArbol (raíz, ejemplos) {
    Si raíz es HOJA Entonces
        EstimarError (raíz, ejemplos)
    Si no
        Distribuir los ejemplos (ej[]) en los brazos
        Para cada brazo (B)
            error = error + EstimarErrorArbol (B, ej[B])
}

```

Fig. 6.23. Pseudocódigo del algoritmo de podado en C4.5.

De esta forma, el *subtree raising* se emplea únicamente para el subárbol más largo. Además, para estimar su error se emplean los ejemplos de entrenamiento, pero los del nodo origen, ya que si se eleva deberá clasificarlos él. En cuanto a la función *EstimarError*, es la función que estima el error de clasificación de una hoja del árbol. Así, para tomar la decisión debemos estimar el error de clasificación en un nodo determinado para un conjunto de test independiente. Habrá que estimarlo tanto para los nodos hoja como para los internos (suma de errores de clasificación de sus hijos). No se puede tomar como dato el error de clasificación en el conjunto de entrenamiento dado que, lógicamente, el error se subestimaría.

Una técnica para estimar el error de clasificación es la denominada *reduced-error pruning*, que consiste en dividir el conjunto de entrenamiento en n subconjuntos, $n-1$ de los cuáles servirán realmente para el entrenamiento del sistema y 1 para la estimación del error. Sin embargo, el problema es que la construcción del clasificador se lleva a cabo con menos ejemplos. Ésta no es la técnica empleada en C4.5. La técnica empleada en C4.5 consiste en estimar el error de clasificación basándose en los propios ejemplos de entrenamiento. Para ello, en el nodo donde queramos estimar el error de clasificación, se toma la clase mayoritaria de sus ejemplos como clase representante. Esto implica que habrá E errores de clasificación de un total de N ejemplos que se clasifican a través de dicho nodo. El error observado será $f=E/N$, siendo q la probabilidad

de error de clasificación del nodo y $p=1-q$ la probabilidad de éxito. Se supone que la función f sigue una distribución binomial de parámetro q . Y lo que se desea obtener es el error e , que será la probabilidad del extremo superior con un intervalo $[f-z, f+z]$ de confianza c . Dado que se trata de una distribución binomial, se obtendrá e mediante las ecuaciones 6.48 y 6.49.

$$P\left[\frac{f - q}{q(1 - q)/N} \leq z\right] = c \quad \text{Ec. 6.48}$$

$$e = \left(\frac{f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \right) \quad \text{Ec. 6.49}$$

Como factor c (factor de confianza) se suele emplear en C4.5 el 25 %, puesto que es el que mejores resultados suele dar y que corresponde a un $z=0.69$.

Obtención de reglas de clasificación

Cualquier árbol de decisión se puede convertir en reglas de clasificación, entendiendo como tal una estructura del tipo *Si <Condición> Entonces <Clase>*. El algoritmo de generación de reglas consiste básicamente en que, por cada rama del árbol de decisión, las preguntas y sus valores estarán en la parte izquierda de las reglas y la etiqueta del nodo hoja correspondiente en la parte derecha (clasificación). Sin embargo, este procedimiento generaría un sistema de reglas con mayor complejidad de la necesaria. Por ello, el sistema C4.5 [QUIN93] realiza un podado de las reglas obtenidas. En la figura 6.24 se muestra el algoritmo completo de obtención de reglas.

```

ObtenerReglas (árbol) {
    Convertir el árbol de decisión (árbol) a un conjunto de reglas, R
    error = error de clasificación con R
    Para cada regla Ri de R Hacer
        Para cada precondition pj de Ri Hacer
            nuevoError = error al eliminar pj de Ri
            Si nuevoError <= error Entonces
                Eliminar pj de Ri
                error = nuevoError
        Si Ri no tiene preconditiones Entonces
            Eliminar Ri
    }

```

Fig. 6.24. Pseudocódigo del algoritmo de obtención de reglas de C4.5.

En cuanto a la estimación del error, se realiza del mismo modo que para realizar el podado del árbol de decisión.

Decision stump (árbol de un solo nivel)

Todavía existe un algoritmo más sencillo que genera un árbol de decisión de un único nivel. Se trata de un algoritmo, *decision stump*, que utiliza un único atributo para construir el árbol de decisión. La elección del único atributo que formará parte del árbol se realizará basándose en la ganancia de información, y a pesar de su simplicidad, en algunos problemas puede llegar a conseguir resultados interesantes. No tiene opciones de configuración, pero la implementación es muy completa, dado que admite tanto atributos numéricos como simbólicos y clases de ambos tipos también. El árbol de decisión tendrá tres ramas: una de ellas será para el caso de que el atributo sea desconocido, y las otras dos serán para el caso de que el valor del atributo del ejemplo de test sea igual a un valor concreto del atributo o distinto a dicho valor, en caso de los atributos simbólicos, o que el valor del ejemplo de test sea mayor o menor a un determinado valor, en el caso de atributos numéricos. En el caso de los atributos simbólicos se considera cada valor posible del mismo y se calcula la ganancia de información con el atributo igual al valor, distinto al valor y valores desconocidos del atributo. En el caso de atributos numéricos se busca el mejor punto de ruptura, tal y como se vio en el sistema C4.5. Deben tenerse en cuenta cuatro posibles casos al calcular la ganancia de información: que sea un atributo simbólico y la clase sea simbólica o que la clase sea numérica, o que sea un atributo numérico y la clase sea simbólica o que la clase sea numérica. A continuación, se comenta cada caso por separado.

Atributo simbólico y clase simbólica

Se toma cada vez un valor v_x del atributo simbólico A_i como base y se consideran únicamente tres posibles ramas en la construcción del árbol: que el atributo A_i sea igual a v_x , que el atributo A_i sea distinto a v_x o que el valor del atributo A_i sea desconocido. Con ello, se calcula la entropía del atributo tomando como base el valor escogido tal y como se muestra en la ecuación 6.50, en la que el valor de j en el sumatorio va desde uno a tres porque los valores del atributo se restringen a tres: igual a v_x , distinto de v_x o valor desconocido. En cuanto a los parámetros, n_{ij} es el número de ejemplos con valor j en el atributo i , n el número total de ejemplos y n_{ijk} el número de ejemplos con valor j en el atributo i y que pertenece a la clase k .

$$I(A_{i|v_x}) = \frac{\sum_{j=1}^3 n_{ij} \log(n_{ij}) - I_{ij}}{n}; I_{ij} = - \sum_{k=1}^{nc} n_{ijk} \log(n_{ijk}) \quad \text{Ec. 6.50}$$

Atributo numérico y clase simbólica

Se ordenan los ejemplos según el atributo A_i y se considera cada valor v_x del atributo como posible punto de corte. Se consideran entonces como posibles valores del atributo el rango menor o igual a v_x , mayor a v_x y valor desconocido. Se calcula la entropía del rango tomando como base esos tres posibles valores restringidos del atributo.

Atributo simbólico y clase numérica

Se vuelve a tomar como base cada vez cada valor del atributo, tal y como se hacía en el caso atributo simbólico y clase simbólica, pero en éste se calcula la varianza de la clase para los valores del atributo mediante la ecuación 6.51, donde S_j es la suma de los valores de la clase de los ejemplos con valor j en el atributo i , SS_j es la suma de los valores de la clase al cuadrado y W_j es la suma de los pesos de los ejemplos (número de ejemplos si no se incluyen pesos) con valor j en el atributo.

$$\text{Varianza}(A_{iv_x}) = \sum_{j=1}^3 \left(SS_j - \frac{S_j^2}{W_j} \right) \quad \text{Ec. 6.51}$$

Atributo numérico y clase numérica

Se considera cada valor del atributo como punto de corte tal y como se hacía en el caso atributo numérico y clase simbólica. Posteriormente, se calcula la varianza tal y como se muestra en la ecuación 6.51.

En cualquiera de los cuatro casos que se han comentado, lo que se busca es el valor mínimo de la ecuación calculada, ya sea la entropía o la varianza. De esta forma se obtiene el atributo que será raíz del árbol de decisión y sus tres ramas. Lo único que se hará por último es construir dicho árbol: cada rama finaliza en un nodo hoja con el valor de la clase, que será la media o la moda de los ejemplos que se clasifican por ese camino, según se trate de una clase numérica o simbólica.

6.5.3 Reglas de clasificación

Las **técnicas de inducción de reglas** [QUIN87, QUIN93] surgieron hace más de dos décadas y permiten la generación y contraste de árboles de decisión, o reglas y patrones a partir de los datos de entrada. La información de entrada será un conjunto de casos donde se ha asociado una clasificación o evaluación a un conjunto de variables o atributos. Con esa información, estas técnicas obtienen el árbol de decisión o conjunto de reglas que soportan la evaluación o clasificación [CN89, HMM86]. En los casos en que la información de entrada posee algún tipo de "ruido" o defecto (insuficientes atributos o datos, atributos irrelevantes o errores u omisiones en los datos), estas técnicas pueden

habilitar métodos estadísticos de tipo probabilístico para generar árboles de decisión recortados o podados. También en estos casos pueden identificar los atributos irrelevantes, la falta de atributos discriminantes o detectar *gaps* o huecos de conocimiento. Esta técnica suele llevar asociada una alta interacción con el analista de forma que éste pueda intervenir en cada paso de la construcción de las reglas, bien para aceptarlas, bien para modificarlas [MM95].

La inducción de reglas se puede lograr fundamentalmente mediante dos caminos: generando un árbol de decisión y extrayendo de él las reglas [QUIN93], como puede hacer el sistema C4.5, o bien mediante una estrategia de *covering*, consistente en tener en cuenta cada vez una clase y buscar las reglas necesarias para cubrir (*cover*) todos los ejemplos de esa clase; cuando se obtiene una regla se eliminan todos los ejemplos que cubre y se continúa buscando más reglas hasta que no haya más ejemplos de la clase. A continuación se muestran una técnica de inducción de reglas basada en árboles de decisión, otra basada en *covering* y una más que mezcla las dos estrategias.

Algoritmo 1R

El más simple algoritmo de reglas de clasificación para un conjunto de ejemplos es el 1R [HOL93]. Este algoritmo genera un árbol de decisión de un nivel expresado mediante reglas. Consiste en seleccionar un atributo (nodo raíz) del cual nace una rama por cada valor, que va a parar a un nodo hoja con la clase más probable de los ejemplos de entrenamiento que se clasifican a través suyo. Este algoritmo se muestra en la figura 6.25.

```
1R (ejemplos) {
    Para cada atributo (A)
        Para cada valor del atributo (Ai)
            Contar el número de apariciones de cada clase con Ai
            Obtener la clase más frecuente (Cj)
            Crear una regla del tipo Ai -> Cj
        Calcular el error de las reglas del atributo A
    Escoger las reglas con menor error
}
```

Fig. 6.25. Pseudocódigo del algoritmo 1R.

La clase debe ser simbólica, mientras que los atributos pueden ser simbólicos o numéricos. También admite valores desconocidos, que se toman como otro valor más del atributo. En cuanto al error de las reglas de un atributo, consiste en la proporción entre los ejemplos que cumplen la regla y los ejemplos que cumplen la premisa de la regla. En el caso de los atributos numéricos, se generan una serie de **puntos de ruptura** (*breakpoint*), que discretizarán dicho atributo formando conjuntos. Para ello, se ordenan los ejemplos por el atributo numérico y se recorren. Se van contando las apariciones de cada clase hasta un número *m* que indica el mínimo número de ejemplos que pueden

pertenecer a un conjunto, para evitar conjuntos demasiado pequeños. Por último, se unen a este conjunto ejemplos con la clase más frecuente y ejemplos con el mismo valor en el atributo.

La sencillez de este algoritmo es un poco insultante. Su autor llega a decir [HOL93; pág. 64]: “*Program 1R is ordinary in most respects*”. Tanto es así que 1R no tiene ningún elemento de sofisticación y genera para cada atributo un árbol de profundidad 1, donde una rama está etiquetada por *missing* si es que aparecen valores desconocidos (*missing values*) en ese atributo en el conjunto de entrenamiento; el resto de las ramas tienen como etiqueta un intervalo construido de una manera muy simple, como se ha explicado antes, o un valor nominal, según el tipo de atributo del que se trate. Lo sorprendente de este sistema es su rendimiento. En [HOL93] se describen rendimientos que en media están por debajo de los de C4.5 en 5,7 puntos porcentuales de aciertos de clasificación. Para la realización de las pruebas, Holte, elige un conjunto de dieciséis problemas del almacén de la UCI [Blake, Keogh, Merz, 98] que desde entonces han gozado de cierto reconocimiento como conjunto de pruebas; en alguno de estos problemas introduce algunas modificaciones que también se han hecho estándar. El mecanismo de estimación consiste en separar el subconjunto de entrenamiento original en subconjuntos de entrenamiento y test en proporción 2/3 y 1/3 respectivamente y repetir el experimento veinticinco veces. Aunque la diferencia de 5,7 es algo elevada, en realidad, en catorce de los dieciséis problemas la diferencia es solo de 3,1 puntos. En la tabla de la figura 6.26 se presenta un ejemplo de 1R, basado en los ejemplos de la tabla de la figura 6.15.

atributo	reglas	errores	error total
vista	Soleado → no	2/5	4/14
	Nublado → sí	0/4	
	Lluvioso → sí	2/5	
temperatura	Alta → no	2/4	5/14
	Media → sí	2/6	
	Baja → sí	1/4	
humedad	Alta → no	3/7	4/14
	Normal → sí	1/7	
viento	Falso → sí	2/8	5/14
	Cierto → no	3/6	

Fig. 6.26. Resultados del algoritmo 1R.

Para clasificar según la clase “jugar”, 1R considera cuatro conjuntos de reglas, uno por cada atributo, que son las mostradas en la tabla anterior, en las que además aparecen los errores que se cometen. De esta forma se concluye que como los errores mínimos corresponden a las reglas generadas por los atributos “vista” y “humedad”, cualquiera de ellas es válida, de manera que arbitrariamente se puede elegir cualquiera de estos dos conjuntos de reglas como generador de 1R.

Algoritmo PRISM

PRISM [CEN87] es un algoritmo básico de aprendizaje de reglas que asume que no hay ruido en los datos. Sea t el número de ejemplos cubiertos por la regla y p el número de ejemplos positivos cubiertos por la regla. Lo que hace PRISM es añadir condiciones a reglas que maximicen la relación p/t (relación entre los ejemplos positivos cubiertos y ejemplos cubiertos en total). En la figura 6.27 se muestra el algoritmo de PRISM.

```
PRISM (ejemplos) {
  Para cada clase (C)
    E = ejemplos
    Mientras E tenga ejemplos de C
      Crea una regla R con parte izquierda vacía y clase C
      Hasta R perfecta Hacer
        Para cada atributo A no incluido en R y cada valor v de A
          Considera añadir la condición A=v a la parte izquierda de R
          Selecciona el par A=v que maximice p/t
          (en caso de empates, escoge la que tenga p mayor)
        Añadir A=v a R
      Elimina de E los ejemplos cubiertos por R
```

Fig. 6.27. Pseudocódigo del algoritmo PRISM.

Este algoritmo va eliminando los ejemplos que va cubriendo cada regla, de manera que las reglas tienen que interpretarse en orden. Se habla entonces de listas de reglas (*decision list*). En la figura 6.28 se muestra un ejemplo de cómo actúa el algoritmo. Concretamente se trata de la aplicación del mismo sobre el ejemplo de la tabla de la figura 6.15.

Regla 1. Clase "Sí".	
Añadir a	p/t
"If <vacío> Then Sí"	
Vista = Soleado	2/5
Vista = Nublado	4/4
Vista = Lluvioso	3/5
Temperatura = Alta	2/4
Temperatura = Media	4/6
Temperatura = Baja	3/4
Humedad = Alta	3/7
Humedad = Normal	6/7
Viento = Sí	3/6
Viento = No	6/8
If Vista = Nublado Then Sí	

Regla 2. Clase "Sí".	
Añadir a	p/t
"If <vacío> Then Sí"	
Vista = Soleado	2/5
Vista = Lluvioso	3/5
Temperatura = Alta	0/2
Temperatura = Media	3/5
Temperatura = Baja	2/3
Humedad = Alta	1/5
Humedad = Normal	4/5
Viento = Sí	1/4
Viento = No	4/6
If Humedad=Normal and Viento = No Then Sí	

Añadir a	p/t
"If Humedad=Normal Then Sí"	
Vista = Soleado	2/2
Vista = Lluvioso	2/3
Temperatura = Alta	0/0
Temperatura = Media	2/2
Temperatura = Baja	2/3
Viento = Sí	1/2
Viento = No	3/3

Lista de Decisión Completa:

If Vista = Nublado Then Sí

If Humedad=Normal and Viento = No Then Sí

If Temperatura = Media and Humedad = Normal Then Sí

If Vista = Lluvioso and Viento = No Then Sí

If Vista = Soleado and Humedad = Alta Then No

If Vista = Lluvioso and Viento = Sí Then Sí

Fig. 6.28. Ejemplo de PRISM.

En la figura 6.28 se muestra cómo el algoritmo toma en primer lugar la clase "Sí". Partiendo de todos los ejemplos de entrenamiento (un total de catorce) calcula el cociente p/t para cada par atributo-valor y escoge el mayor. En este caso, dado que la condición escogida hace la regla perfecta ($p/t = 1$), se eliminan los cuatro ejemplos que cubre dicha regla y se busca una nueva regla. En la segunda regla se obtiene en un primer momento una condición que no hace perfecta la regla, de modo que se continúa buscando con otra condición. Finalmente, se muestra la lista de decisión completa que genera el algoritmo.

Algoritmo PART

Uno de los sistemas más importantes de aprendizaje de reglas es el proporcionado por C4.5 [QUI93], explicado anteriormente. Este sistema, al igual que otros sistemas de inducción de reglas, realiza dos fases: primero genera un conjunto de reglas de clasificación y después refina estas reglas para mejorarlas, realizando así un proceso de optimización global de dichas reglas. Este proceso de optimización global es siempre muy complejo y costoso computacionalmente hablando. Por otro lado, el algoritmo PART [FRWI98] es un sistema que obtiene reglas sin dicha optimización global. Recibe el nombre de PART por su modo de actuación: *obtaining rules from PARTial decision trees*, y fue desarrollado por el grupo neozelandés que construyó el entorno Weka [WF98].

El sistema se basa en las dos estrategias básicas para la inducción de reglas: el *covering* y la generación de reglas a partir de árboles de decisión. Adopta la estrategia del *covering* (con lo que se obtiene una lista de decisión) dado que genera una regla, elimina los ejemplares que dicha regla cubre y continúa generando reglas hasta que no queden ejemplos por clasificar. Sin embargo, el proceso de generación de cada regla no es el usual. En este caso, para crear una regla, se genera un árbol de decisión podado, se obtiene la hoja que clasifique el mayor número de ejemplos, que se transforma en la regla, y posteriormente se elimina el árbol. Uniendo estas dos estrategias se consigue mayor flexibilidad y velocidad. Además, no se genera un árbol completo, sino un árbol parcial (*partial decision tree*). Un árbol parcial es un árbol de decisión que contiene brazos con subárboles no definidos. Para generar este árbol se integran los procesos de construcción y podado hasta que se encuentra un subárbol estable que no puede simplificarse más, en cuyo caso se para el proceso y se genera la regla a partir de dicho subárbol. Este proceso se muestra en la figura 6.29.

```
Expandir (ejemplos) {
  elegir el mejor atributo para dividir en subconjuntos
  Mientras (subconjuntos No expandidos)
    Y (todos los subconjuntos expandidos son HOJA)
      Expandir (subconjunto)
  Si (todos los subconjuntos expandidos son HOJA)
    Y (errorSubárbol >= errorNodo)
      deshacer la expansión del nodo y nodo es HOJA
```

Fig. 6.29. Pseudocódigo de expansión de PART.

El proceso de elección del mejor atributo se hace como en el sistema C4.5, esto es, basándose en la razón de ganancia. La expansión de los subconjuntos generados se realiza en orden, comenzando por el que tiene menor entropía y finalizando por el que tiene mayor. La razón de realizarlo así es porque si un subconjunto tiene menor entropía hay más probabilidades de que se genere un subárbol menor y consecuentemente se cree una regla más general. El proceso continúa recursivamente expandiendo los subconjuntos hasta que se obtienen hojas, momento en el que se realizará una vuelta atrás (*backtracking*). Cuando se realiza dicha vuelta atrás y los hijos del nodo en cuestión son hojas, comienza el podado tal y como se realiza en C4.5 (comparando el error esperado del subárbol con el del nodo), pero únicamente se realiza la función de reemplazamiento del nodo por hoja (*subtree replacement*). Si se lleva a cabo el podado se realiza otra vuelta atrás hacia el nodo padre, que sigue explorando el resto de sus hijos, pero si no se puede realizar el podado, el padre no continuará con la exploración del resto de nodos hijos (ver segunda condición del bucle “mientras” en la figura 6.29). En este momento finalizará el proceso de expansión y generación del árbol de decisión.

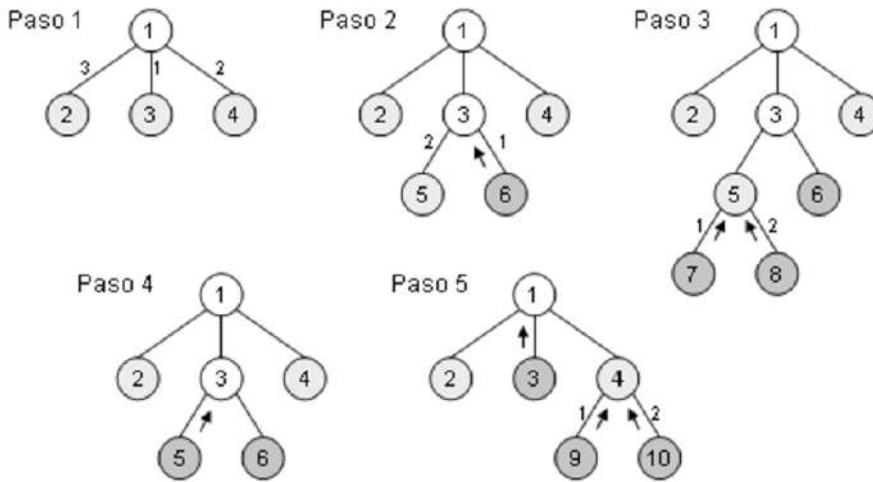


Fig. 6.30. Ejemplo de generación de árbol parcial con PART.

En la figura 6.30 se presenta un ejemplo de generación de un árbol parcial donde, junto a cada brazo de un nodo, se muestra el orden de exploración (orden ascendente según el valor de la entropía). Los nodos con relleno gris claro son los que aún no se han explorado y los nodos con relleno gris oscuro los nodos hoja. Las flechas ascendentes representan el proceso de *backtracking*. Por último, en el paso 5, cuando el nodo 4 es explorado y los nodos 9 y 10 pasan a ser hoja, el nodo padre intenta realizar el proceso de podado, pero no se realiza el reemplazo (representado con el 4 en negrita), con lo que el proceso, al volver al nodo 1, finaliza sin explorar el nodo 2.

Una vez generado el árbol parcial se extrae una regla del mismo. Cada hoja se corresponde con una posible regla, y lo que se busca es la mejor hoja. Si bien se pueden considerar otras heurísticas, en el algoritmo PART se considera mejor hoja aquella que cubre un mayor número de ejemplos. Se podría haber optado, por ejemplo, por considerar mejor aquella que tiene un menor error esperado, pero tener una regla muy precisa no significa lograr un conjunto de reglas muy preciso. Por último, PART permite que haya atributos con valores desconocidos tanto en el proceso de aprendizaje como en el de validación y atributos numéricos, tratándolos exactamente como el sistema C4.5.

6.5.4 Clasificación bayesiana

El clasificador bayesiano [DH73] se introdujo en el capítulo anterior como método clásico que nos permite predecir tanto las probabilidades del número de miembros de clase,

como la probabilidad de que una muestra dada pertenezca a una clase particular. La clasificación bayesiana se basa en el teorema de Bayes, y los clasificadores bayesianos han demostrado una alta exactitud y velocidad cuando se han aplicado a grandes bases de datos. Diferentes estudios comparando los algoritmos de clasificación han determinado que un clasificador bayesiano sencillo conocido como el clasificador *naive bayesiano* [JOH97] es comparable en rendimiento a un árbol de decisión y a clasificadores de redes de neuronas. A continuación se explica los fundamentos de los clasificadores bayesianos y, más concretamente, del clasificador *naive bayesiano*. Tras esta explicación se comentará otro clasificador que, si bien no es un clasificador bayesiano, está relacionado con él, dado que se trata también de un clasificador basado en la estadística.

Clasificador *naive bayesiano*

Lo que normalmente se quiere saber en aprendizaje es cuál es la mejor hipótesis (más probable) dados los datos. Si denotamos $P(D)$ como la probabilidad *a priori* de los datos (por ejemplo, cuáles datos son más probables que otros), $P(D|h)$ la probabilidad de los datos dada una hipótesis, lo que queremos estimar es: $P(h|D)$, la probabilidad posterior de h dados los datos. Esto se puede estimar con el teorema de Bayes, ecuación 6.52.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \quad \text{Ec. 6.52}$$

Para estimar la hipótesis más probable (MAP, *Maximum A Posteriori*) se busca el mayor $P(h|D)$ como se muestra en la ecuación 6.53.

$$\begin{aligned} h_{\text{MAP}} &= \operatorname{argmax}_{h \in H} (P(h|D)) \\ &= \operatorname{argmax}_{h \in H} \left(\frac{P(D|h)P(h)}{P(D)} \right) \\ &= \operatorname{argmax}_{h \in H} (P(D|h)P(h)) \end{aligned} \quad \text{Ec. 6.53}$$

Ya que $P(D)$ es una constante independiente de h . Si se asume que todas las hipótesis son igualmente probables, entonces resulta la hipótesis de máxima verosimilitud (ML, *Maximum Likelihood*) de la ecuación 6.54.

$$h_{ML} = \operatorname{argmax}_{h \in H} (P(D|h)) \quad \text{Ec. 6.54}$$

El clasificador *naive* (ingenuo) bayesiano se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos (a_i 's) en un conjunto finito de clases (V). Clasificar un nuevo ejemplo de acuerdo con el valor más probable dados los valores de sus atributos. Si se aplica la ecuación 6.54 al problema de la clasificación se obtendrá la ecuación 6.55.

$$\begin{aligned} v_{MAP} &= \operatorname{argmax}_{v_j \in V} (P(v_j | a_1, \dots, a_n)) \\ &= \operatorname{argmax}_{v_j \in V} \left(\frac{P(a_1, \dots, a_n | v_j) P(v_j)}{P(a_1, \dots, a_n)} \right) \\ &= \operatorname{argmax}_{v_j \in V} (P(a_1, \dots, a_n | v_j) P(v_j)) \end{aligned} \quad \text{Ec. 6.55}$$

Además, el clasificador *naive* bayesiano asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase, de modo que se hace cierta la ecuación 6.56 y con ella la 6.57.

$$P(a_1, \dots, a_n | v_j) = \prod_i P(a_i | v_j) \quad \text{Ec. 6.56}$$

$$P(v_j | a_1, \dots, a_n) = P(v_j) \times \prod_i P(a_i | v_j) \quad \text{Ec. 6.57}$$

Los clasificadores *naive* bayesianos asumen que el efecto de un valor del atributo en una clase dada es independiente de los valores de los otros atributos. Esta suposición se llama *independencia condicional de clase*. Ésta simplifica los cálculos involucrados y, en este sentido, es considerado "ingenuo" (*naive*). Esta asunción es una simplificación de la realidad. A pesar del nombre del clasificador y de la simplificación realizada, el *naive* bayesiano funciona muy bien, sobre todo cuando se filtra el conjunto de atributos seleccionado para eliminar redundancia, con lo que se elimina también dependencia entre datos. En la figura 6.31 se muestra un ejemplo de aprendizaje con el clasificador *naive* bayesiano, así como una muestra de cómo se clasificaría un ejemplo de test. Como ejemplo se empleará el de la tabla de la figura 6.15.

Proceso de Aprendizaje

Vista			Temperatura			Humedad			Viento			Jugar		
	Si	No		Si	No		Si	No		Si	No		Si	No
Soleado	2	3	Alta	2	2	Alta	3	4	Si	3	3	9	5	
Nublado	4	0	Media	4	2	Normal	6	1	No	6	2			
Lluvioso	3	2	Baja	3	1									
Soleado	2/9	3/5	Alta	2/9	2/5	Alta	3/9	4/5	Si	3/9	3/5	9/14	5/14	
Nublado	4/9	0/5	Media	4/9	2/5	Normal	6/9	1/5	No	6/9	2/5			
Lluvioso	3/9	2/5	Baja	3/9	1/5									

Clasificación de un ejemplo de test

Vista	Temperatura	Humedad	Viento	Jugar
Soleado	Fria	Alta	Si	¿

$$P(Si | E) = P(Si) \times \prod_i P(a_i | Si) = \frac{9}{14} \times \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} = 0,0053$$

$$P(No | E) = P(No) \times \prod_i P(a_i | No) = \frac{5}{14} \times \frac{3}{9} \times \frac{1}{9} \times \frac{4}{9} = 0,0206$$

$$\text{Normalizado} \begin{cases} P(Si | E) = \frac{0,0053}{0,0053 + 0,0206} = 20,9\% \\ P(No | E) = \frac{0,0206}{0,0053 + 0,0206} = 79,9\% \end{cases}$$

Fig. 6.31. Ejemplo de aprendizaje y clasificación con *naive bayesiano*.

En este ejemplo se observa que en la fase de aprendizaje se obtienen todas las probabilidades condicionadas $P(a_i | v_j)$ y las probabilidades $P(v_j)$. En la clasificación se realiza el productorio y se escoge como clase del ejemplo de entrenamiento la que obtenga un mayor valor. Algo que puede ocurrir durante el entrenamiento con este clasificador es que para cada valor de cada atributo no se encuentren ejemplos para todas las clases. Supóngase que para el atributo a_i y el valor j de dicho atributo no hay ningún ejemplo de entrenamiento con clase k . En este caso, $P(a_{ij} | k) = 0$. Esto hace que si se intenta clasificar cualquier ejemplo con el par atributo-valor a_{ij} , la probabilidad asociada para la clase k será siempre 0, ya que hay que realizar el productorio de las probabilidades condicionadas para todos los atributos de la instancia. Para resolver este problema se parte de que las probabilidades se contabilizan a partir de las frecuencias de aparición de cada evento o, en nuestro caso, las frecuencias de aparición de cada terna atributo-valor-clase. El **estimador de Laplace**, consiste en comenzar a contabilizar la frecuencia de aparición de cada terna a partir del 1 y no del 0, con lo que ninguna probabilidad condicionada será igual a 0.

Una ventaja de este clasificador es la cuestión de los valores perdidos o desconocidos: en el clasificador *naive bayesiano*, si se intenta clasificar un ejemplo con un atributo sin valor, simplemente el atributo en cuestión no entra en el productorio que sirve para calcular las probabilidades. Respecto a los atributos numéricos, se suele suponer que siguen una distribución normal o gaussiana. Para estos atributos se calcula la media μ y la desviación típica σ obteniendo los dos parámetros de la distribución $N(\mu, \sigma)$, que

sigue la expresión de la ecuación 6.58, donde el parámetro x será el valor del atributo numérico en el ejemplo que se quiere clasificar.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{Ec. 6.58}$$

Votación por intervalos de características

Este algoritmo es una técnica basada en la proyección de características. Se le denomina *votación por intervalos de características* (VFI, *Voting Feature Interval*) porque se construyen intervalos para cada característica (*feature*) o atributo en la fase de aprendizaje y el intervalo correspondiente en cada característica “vota” para cada clase en la fase de clasificación. Al igual que en el clasificador *naive* bayesiano, cada característica es tratada de forma individual e independiente del resto. Se diseña un sistema de votación para combinar las clasificaciones individuales de cada atributo por separado.

Mientras que en el clasificador *naive* bayesiano cada característica participa en la clasificación asignando una probabilidad para cada clase, y la probabilidad final para cada clase consiste en el producto de cada probabilidad dada por cada característica, en el algoritmo VFI cada característica distribuye sus votos para cada clase y el voto final de cada clase es la suma de los votos obtenidos por cada característica. Una ventaja de estos clasificadores, al igual que ocurría con el clasificador *naive* bayesiano, es el tratamiento de los valores desconocidos tanto en el proceso de aprendizaje como en el de clasificación: simplemente se ignoran, dado que se considera cada atributo como independiente del resto.

En la fase de aprendizaje del algoritmo VFI se construyen intervalos para cada atributo contabilizando, para cada clase, el número de ejemplos de entrenamiento que aparecen en dicho intervalo. En la fase de clasificación, cada atributo del ejemplo de test añade votos para cada clase dependiendo del intervalo en el que se encuentre y el conteo de la fase de aprendizaje para dicho intervalo en cada clase. En la figura 6.32 se muestra este algoritmo.

```

Aprendizaje (ejemplos) {
  Para cada atributo (A) Hacer
    Si A es NUMÉRICO Entonces
      Obtener mínimo y máximo de A para cada clase en ejemplos
      Ordenar los valores obtenidos (I intervalos)
    Si no /* es SIMBÓLICO */
      Obtener los valores que recibe A para cada clase en ejemplos
      Los valores obtenidos son puntos (I intervalos)

  Para cada intervalo I Hacer
    Para cada clase C Hacer
      contadores [A, I, C] = 0

```

```

Para cada ejemplo E Hacer
  Si A es conocido Entonces
    Si A es SIMBÓLICO Entonces
      contadores [A, E.A, E.C] += 1
    Si no /* es NUMÉRICO */
      Obtener intervalo I de E.A
      Si E.A = extremo inferior de intervalo I Entonces
        contadores [A, I, E.C] += 0.5
        contadores [A, I-1, E.C] += 0.5
      Si no
        contadores [A, I, E.C] += 1

Normalizar contadores[] /*  $\sum_c \text{contadores}[A, I, C] = 1$  */
}

clasificar (ejemplo E) {
  Para cada atributo (A) Hacer
    Si E.A es conocido Entonces
      Si A es SIMBÓLICO
        Para cada clase C Hacer
          voto[A, C] = contadores[A, E.A, C]
      Si no /* es NUMÉRICO */
        Obtener intervalo I de E.A
        Si E.A = limite inferior de I Entonces
          Para cada clase C Hacer
            voto[A, C] = 0.5*contadores[A,I,C] +
                          0.5*contadores[A,I-1,C]
        Si no
          Para cada clase C Hacer
            voto[A, C] = contadores [A, I, C]

    voto[C] = voto[C] + voto[A, C]

Normalizar voto[] /*  $\sum_c \text{voto}[C] = 1$  */
}

```

Fig. 6.32. Pseudocódigo del algoritmo VFI.

En la figura 6.33 se presenta un ejemplo de entrenamiento y clasificación con el algoritmo VFI, en el que se muestra una tabla con los ejemplos de entrenamiento y cómo el proceso de aprendizaje consiste en el establecimiento de intervalos para cada atributo con el conteo de ejemplos que se encuentran en cada intervalo. Se muestra entre paréntesis el número de ejemplos que se encuentran en la clase e intervalo concreto, mientras que fuera de los paréntesis se encuentra el valor normalizado. Para el atributo simbólico simplemente se toma como intervalo (punto) cada valor de dicho atributo y se cuenta el número de ejemplos que tiene un valor determinado en el atributo para la clase del ejemplo en cuestión. En el caso del atributo numérico, se obtiene el máximo y el mínimo valor del atributo para cada clase que en este caso son 4 y 7 para la clase A, y 1 y 5 para la clase B. Se ordenan los valores formándose un total de cinco intervalos y se cuenta el número de ejemplos que se encuentran en un intervalo determinado para su clase, teniendo en cuenta que si se encuentra en el punto compartido por dos intervalos se contabiliza la mitad para cada uno de ellos. También se muestra un ejemplo

de clasificación: en primer lugar, se obtienen los votos que cada atributo por separado concede a cada clase, que será el valor normalizado del intervalo (o punto si se trata de atributos simbólicos) en el que se encuentre el valor del atributo, y posteriormente se suman los votos (que se muestra entre paréntesis) y se normaliza. La clase con mayor porcentaje de votos (en el ejemplo la clase A) gana.

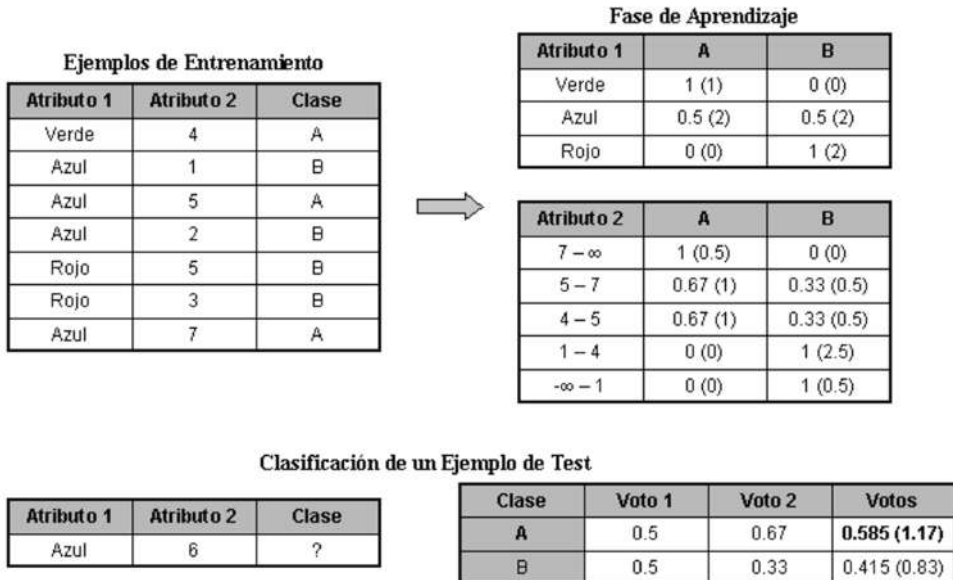


Fig. 6.33. Ejemplo de aprendizaje y clasificación con VFI.

6.5.5 Aprendizaje basado en ejemplares

El aprendizaje basado en ejemplares o instancias [BRIS96] tiene como principio de funcionamiento, en sus múltiples variantes, el almacenamiento de ejemplos: en unos casos todos los ejemplos de entrenamiento, en otros sólo los más representativos, en otros los incorrectamente clasificados cuando se clasifican por primera vez, etc. La clasificación posterior se realiza por medio de una función que mide la proximidad o parecido. Dado un ejemplo para clasificar se le clasifica de acuerdo al ejemplo o ejemplos más próximos. El *bias* (sesgo) que rige este método es la proximidad; es decir, la generalización se guía por la proximidad de un ejemplo a otros. Algunos autores consideran este *bias* más apropiado para el aprendizaje de conceptos naturales que el correspondiente al proceso inductivo (Bareiss *et al.* en [KODR90]), por otra parte, también se ha estudiado la relación entre este método y los que generan reglas (Clark, 1990).

Se han enumerado ventajas e inconvenientes del aprendizaje basado en ejemplares [BRIS96], pero se suele considerar no adecuado para el tratamiento de atributos no

numéricos y valores desconocidos. Las mismas medidas de proximidad sobre atributos simbólicos suelen proporcionar resultados muy dispares en problemas diferentes. A continuación se muestran dos técnicas de aprendizaje basado en ejemplares: el método de los k -vecinos más próximos y el k estrella.

6.5.5.1 Algoritmo de los k -vecinos más próximos

El método de los k -vecinos más próximos [MITC97] (KNN, *k-Nearest Neighbor*) está considerado como un buen representante de este tipo de aprendizaje, y es de gran sencillez conceptual. Se suele denominar *método* porque es el esqueleto de un algoritmo que admite el intercambio de la función de proximidad dando lugar a múltiples variantes. La función de proximidad puede decidir la clasificación de un nuevo ejemplo atendiendo a la clasificación del ejemplo o de la mayoría de los k ejemplos más cercanos. Admite también funciones de proximidad que consideren el peso o coste de los atributos que intervienen, lo que permite, entre otras cosas, eliminar los atributos irrelevantes. Una función de proximidad clásica entre dos instancias x_i y x_j , si suponemos que un ejemplo viene representado por una n -tupla de la forma $(a_1(x), a_2(x), \dots, a_n(x))$ en la que $a_r(x)$ es el valor de la instancia para el atributo a_r , es la distancia euclídea, que se muestra en la ecuación 6.59.

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^n (x_{il} - x_{jl})^2} \quad \text{Ec. 6.59}$$

En la figura 6.34 se muestra un ejemplo del algoritmo k -NN para un sistema de dos atributos, representándose por ello en un plano. En este ejemplo se ve como el proceso de aprendizaje consiste en el almacenamiento de todos los ejemplos de entrenamiento. Se han representado los ejemplos de acuerdo a los valores de sus dos atributos y la clase a la que pertenecen (las clases son $+$ y $-$). La clasificación consiste en la búsqueda de los k ejemplos (en este caso tres) más cercanos al ejemplo a clasificar. Concretamente, el ejemplo a se clasificaría como $-$, y el ejemplo b como $+$.

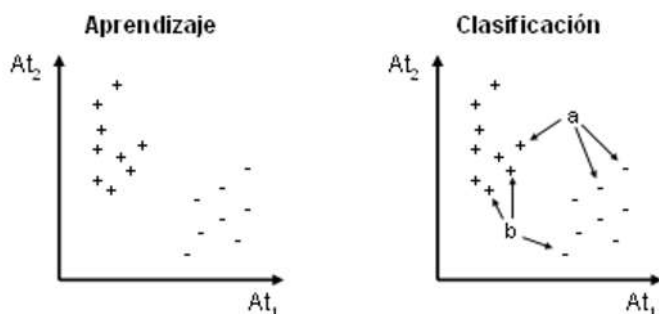


Fig. 6.34. Ejemplo de aprendizaje y clasificación con k -NN.

Dado que el algoritmo k-NN permite que los atributos de los ejemplares sean simbólicos y numéricos, así como que haya atributos sin valor (*missing values*), el algoritmo para el cálculo de la distancia entre ejemplares se complica ligeramente. En la figura 6.35 se muestra el algoritmo que calcula la distancia entre dos ejemplares cualesquiera.

```

Distancia (E1, E2) {
    dst = 0
    n = 0
    Para cada atributo A Hacer {
        dif = Diferencia(E1.A, E2.A)
        dst = dst + dif * dif
        n = n + 1
    }
    dst = dst / n
    Devolver dst
}

Diferencia (A1, A2) {
    Si A1.nominal Entonces {
        Si SinValor(A1) O SinValor(A2) O A1 <> A2 Entonces
            Devolver 1
        Si no
            Devolver 0
    } Si no {
        Si SinValor(A1) O SinValor(A2) Entonces {
            Si SinValor(A1) Y SinValor(A2) Entonces
                Devolver 1
            Si SinValor(A1) Entonces
                dif = A2
            Si no Entonces
                dif = A1
            Si dif < 0.5 Entonces
                Devolver 1 - dif
            Si no
                Devolver dif
        } Si no
            Devolver abs(A1 - A2)
    }
}

```

Fig. 6.35. Pseudocódigo del algoritmo empleado para definir la distancia entre dos ejempls.

Además de los distintos tipos de atributos, hay que tener en cuenta también, en el caso de los atributos numéricos, los rangos en los que se mueven sus valores. Para evitar que atributos con valores muy altos tengan mucho mayor peso que atributos con valores bajos, se normalizarán dichos valores con la ecuación 6.60.

$$\frac{x_{ij} - \min_i}{\text{Max}_i - \min_i} \quad \text{Ec. 6.60}$$

En esta ecuación x_{if} será el valor i del atributo f , siendo \min_f el mínimo valor del atributo f y \max_f el máximo. Por otro lado, el algoritmo permite dar mayor preferencia a aquellos ejemplares más cercanos al que deseamos clasificar. En ese caso, en lugar de emplear directamente la distancia entre ejemplares, se utilizará la ecuación 6.61.

$$\frac{1}{1 + d(x_i, x_j)} \quad \text{Ec. 6.61}$$

6.5.5.2 Algoritmo k-estrella

El algoritmo k-estrella o K^* [CLTR95] es una técnica de *data mining* basada en ejemplares en la que la medida de la distancia entre ejemplares se basa en la teoría de la información. Una forma intuitiva de verlo es que la distancia entre dos ejemplares se define como la complejidad de transformar un ejemplar en el otro. El cálculo de la complejidad se basa en primer lugar en definir un conjunto de transformaciones $T = \{t_1, t_2, \dots, t_n, \sigma\}$ para pasar de un ejemplo (valor de atributo) a a uno b . La transformación σ es la de parada y es la transformación identidad ($\sigma(a) = a$). El conjunto P es el conjunto de todas las posibles secuencias de transformaciones descritos en T^* que terminan en σ , y $\bar{t}(a)$ es una de estas secuencias concretas sobre el ejemplo a . Esta secuencia de transformaciones tendrá una probabilidad determinada $p(\bar{t})$, definiéndose la función de probabilidad $P^*(b/a)$ como la probabilidad de pasar del ejemplo a al ejemplo b a través de cualquier secuencia de transformaciones, tal y como se muestra en la ecuación 6.62.

$$P^*(b/a) = \sum_{\bar{t} \in P: \bar{t}(a)=b} p(\bar{t}) \quad \text{Ec. 6.62}$$

Esta función de probabilidad cumplirá las propiedades que se muestran en la ecuación 6.63.

$$\sum_b P^*(b/a) = 1; 0 \leq P^*(b/a) \leq 1 \quad \text{Ec. 6.63}$$

La función de distancia K^* se define entonces tomando logaritmos, tal y como se muestra en la ecuación 6.64.

$$K^*(b/a) = -\log_2 P^*(b/a) \quad \text{Ec. 6.64}$$

Realmente K^* no es una función de distancia dado que, por ejemplo $K^*(a/a)$ generalmente no será exactamente 0, además de que el operador $/$ no es simétrico, esto es,

$K^*(a/b)$ no es igual que $K^*(b/a)$. Sin embargo, esto no interfiere en el algoritmo K^* . Además, la función K^* cumple las propiedades que se muestran en la ecuación 6.65.

$$K^*(b|a) \geq 0; K^*(c|b) + K^*(b|a) \geq K^*(c|a); \quad \text{Ec. 6.65}$$

Una vez explicado cómo se obtiene la función K^* y cuáles son sus propiedades, se presenta a continuación la expresión concreta de la función P^* , de la que se obtiene K^* , para los tipos de atributos admitidos por el algoritmo: numéricos y simbólicos.

6.5.5.3 Probabilidad de transformación para los atributos permitidos

En cuanto a los atributos numéricos, las transformaciones consideradas serán restar del valor a un número n o sumar al valor a un número n , siendo n un número mínimo. La probabilidad de pasar de un ejemplo con valor a a uno con valor b vendrá determinada únicamente por el valor absoluto de la diferencia entre a y b , que se denominará x . Se escribirá la función de probabilidad como una función de densidad, tal y como se muestra en la ecuación 6.66, donde x_0 será una medida de longitud de la escala, por ejemplo, la media esperada para x sobre la distribución P^* . Es necesario elegir un x_0 razonable. Posteriormente se mostrará un método para elegir este factor. Para los simbólicos, se considerarán las probabilidades de aparición de cada uno de los valores de dicho atributo.

$$P^*(x) = \frac{1}{2x_0} e^{-\frac{x}{x_0}} dx \quad \text{Ec. 6.66}$$

Si el atributo tiene un total de n posibles valores, y la probabilidad de aparición del valor i del atributo es p_i (obtenido a partir de las apariciones en los ejemplos de entrenamiento), se define la probabilidad de transformación de un ejemplo con valor i a uno con valor j como se muestra en la ecuación 6.67.

$$P^*(j|i) = \begin{cases} (1-s)p_j & \text{si } i \neq j \\ s + (1-s)p_i & \text{si } i = j \end{cases} \quad \text{Ec. 6.67}$$

En esta ecuación, s es la probabilidad del símbolo de parada (σ). De esta forma, se define la probabilidad de cambiar de valor como la probabilidad de que no se pare la transformación multiplicado por la probabilidad del valor de destino, mientras la probabilidad de continuar con el mismo valor es la probabilidad del símbolo de parada más la probabilidad de que se continúe transformando multiplicado por la probabilidad del valor de destino. También es importante, al igual que con el factor x_0 , definir correctamente la probabilidad s . Y como ya se comentó con x_0 , posteriormente se comentará un método para obtenerlo. También debe tenerse en cuenta la posibilidad de los atributos con valores desconocidos. Cuando los valores desconocidos aparecen en los ejemplos de entrenamiento se propone como solución el considerar que el atributo desconocido se determina a través del resto de ejemplares de entrenamiento. Esto se muestra en la ecuación 6.68, donde n es el número de ejemplos de entrenamiento.

$$P^*(?|a) = \sum_{b=1}^n \frac{P^*(b|a)}{n} \quad \text{Ec. 6.68}$$

6.5.5.4 Combinación de atributos

Ya se han definido las funciones de probabilidad para los tipos de atributos permitidos. Pero los ejemplos reales tienen más de un atributo, de modo que es necesario combinar los resultados obtenidos para cada atributo. Y para combinarlos, y definir así la distancia entre dos ejemplos, se entiende la probabilidad de transformación de un ejemplar en otro como la probabilidad de transformar el primer atributo del primer ejemplo en el del segundo, seguido de la transformación del segundo atributo del primer ejemplo en el del segundo, etc. De esta forma, la probabilidad de transformar un ejemplo en otro viene determinado por la multiplicación de las probabilidades de transformación de cada atributo de forma individual, tal y como se muestra en la ecuación 6.69. En esta ecuación, m será el número de atributo de los ejemplos. Y con esta definición, la distancia entre dos ejemplos se define como la suma de distancias entre cada atributo de los ejemplos.

$$P^*(E_2|E_1) = \prod_{i=1}^m P^*(v_{2i}|v_{1i}) \quad \text{Ec. 6.69}$$

6.5.5.5 Selección de los parámetros aleatorios

Para cada atributo debe determinarse el valor para los parámetros s o x_0 según se trate de un atributo simbólico o numérico respectivamente. Y el valor de este atributo es muy importante. Por ejemplo, si a s se le asigna un valor muy bajo, las probabilidades de transformación serán muy altas, mientras que si s se acerca a 0 las probabilidades de transformación serán muy bajas. Y lo mismo ocurriría con el parámetro x_0 . En ambos casos se puede observar cómo varía la función de probabilidad P^* según se varía el número de ejemplos incluidos partiendo desde 1 (vecino más cercano) hasta n (todos los ejemplares con el mismo peso). Se puede calcular para cualquier función de probabilidad el número efectivo de ejemplos como se muestra en la ecuación 6.70, en la que n es el número de ejemplos de entrenamiento y n_0 es el número de ejemplos con la distancia mínima al ejemplo a (para el atributo considerado). El algoritmo K^* escogerá para x_0 (o s) un número entre n_0 y n .

$$n_0 \leq \frac{(\sum_{b=1}^n P^*(b|a))^2}{\sum_{b=1}^n P^*(b|a)^2} \leq n \quad \text{Ec. 6.70}$$

Por conveniencia se expresa el valor escogido como un **parámetro de mezclado** (*blending*) b , que varía entre $b=0\%$ (n_0) y $b=100\%$ (n). La configuración de este parámetro se puede ver como una **esfera de influencia** que determina cuántos vecinos de a deben considerarse importantes. Para obtener el valor correcto para el parámetro x_0 (o s) se realiza un proceso iterativo en el que se obtienen las esferas de influencia máxima (x_0 o s igual a 0) y mínima (x_0 o s igual a 1), y se aproximan los valores para que dicha esfera se acerque a la necesaria para cumplir con el parámetro de mezclado.

En la figura 6.36 se presenta un ejemplo práctico de cómo obtener los valores para los parámetros x_0 o s . Se va a utilizar para ello el problema que se presentó en la tabla de la figura 6.15, y más concretamente el atributo “vista” con el valor igual a “lluvioso” de dicho problema.

Obtención de s para Vista = Lluvioso

Objetivo: $esfera = \frac{b}{100} * n + n_{lluvioso} \quad \forall i \rightarrow lluvioso = 0,2 * 9 + 5 = 6,8$

Iteración 0 (Inicio): $vInferior = 0 + EPSILON/2 = 0,005 \Rightarrow esfera = 13,99928$
 $vSuperior = 1 - EPSILON/2 = 0,995 \Rightarrow esfera = 5,03012$
 $valor_0 = 1 - \frac{b}{100} = 0,8 \Rightarrow esfera = 6,41218$

Iteración 1: $vInferior = vInferior; vSuperior = valor_0$
 $valor_1 = \frac{vInferior + vSuperior}{2} = 0,4025 \Rightarrow esfera = 10,63580$

Iteración 2: $vInferior = valor_1; vSuperior = vSuperior$
 $valor_2 = \frac{vInferior + vSuperior}{2} = 0,60125 \Rightarrow esfera = 8,29876$

$$esfera = \frac{\sum_{i=1}^n P(b|lluv)}{\sum_{i=1}^n P(b|lluv)} = \frac{(P(sol|lluv) * n_{sol} + P(nub|lluv) * n_{nub} + P(lluv|lluv) * n_{lluv})}{P(sol|lluv) * n_{sol} + P(nub|lluv) * n_{nub} + P(lluv|lluv) * n_{lluv}} =$$

$$= \frac{(0,00949 * 5 + 0,00949 * 4 + 0,05244 * 5)}{0,00949 * 5 + 0,00949 * 4 + 0,05244 * 5} = \frac{0,12083}{0,01456} = 8,29876$$

$$P(sol|lluv) = \frac{1-s}{nv/m} = \frac{1-0,60125}{3/14} = 0,00949$$

$$P(nub|lluv) = \frac{1-0,60125}{3/14} = 0,00949$$

$$P(lluv|lluv) = \frac{s + \frac{1-s}{nv/m}}{0,60125 + \frac{1-0,60125}{3/14}} = 0,05244$$

Iteración 3: $vInferior = valor_2; vSuperior = vSuperior$
 $valor_3 = \frac{vInferior + vSuperior}{2} = 0,70062 \Rightarrow esfera = 7,29193$

...

Iteración 8: $vInferior = vInferior = 0,75031$
 $vSuperior = valor_8 = 0,75652$
 $valor_8 = \frac{vInferior + vSuperior}{2} = 0,75341 \Rightarrow esfera = 6,80871$

$abs(esfera - objetivo) < EPSILON \Rightarrow \text{Conseguido!} \Rightarrow s = 0,75341$

Fig. 6.36. Ejemplo de obtención del parámetros de un atributo simbólico con el algoritmo K*.

En la figura 6.36 se muestra cómo el objetivo es conseguir un valor para s tal que se obtenga una esfera de influencia de 6,8 ejemplos. Los parámetros de configuración necesarios para el funcionamiento del sistema son: el parámetro de mezclado b , en este caso igual a 20 %; una constante denominada *EPSILON*, en este caso igual a 0,01, que determina entre otras cosas cuándo se considera alcanzada la esfera de influencia deseada. En cuanto a la nomenclatura empleada, n será el número total de ejemplos de entrenamiento, nv el número de valores que puede adquirir el atributo, y se han empleado abreviaturas para denominar los valores del atributo: *lluv* por lluvioso, *nub* por nublado y *sol* por soleado.

Tal y como puede observarse en la figura 6.36, las ecuaciones empleadas para el cálculo de la esfera y de P^* no son exactamente las definidas en las ecuaciones definidas anteriormente. Sin embargo, en el ejemplo se han empleado las implementadas en la herramienta Weka por los creadores del algoritmo. En cuanto al ejemplo en sí, se muestra como son necesarias ocho iteraciones para llegar a conseguir el objetivo planteado, siendo el resultado de dicho proceso, el valor de s , igual a 0,75341.

6.5.5.6 Clasificación de un ejemplo

Se calcula la probabilidad de que un ejemplo a pertenezca a la clase c sumando la probabilidad de a a cada ejemplo que es miembro de c , tal y como se muestra en la ecuación 6.71.

$$P^*(c|a) = \sum_{b \in c} P^*(b|a) \quad \text{Ec. 6.71}$$

Se calcula la probabilidad de pertenencia a cada clase y se escoge la que mayor resultado haya obtenido como predicción para el ejemplo.

Clasificación de un ejemplo de Test

Vista	Temperatura	Humedad	Viento
Lluvioso	67	91	Sí

↓

$$P^*(\text{sí} | T) = \sum_{b \in \text{sí}} P^*(b | T) = P^*(b_1 | T) + P^*(b_2 | T) + P^*(b_3 | T) + P^*(b_4 | T) + P^*(b_5 | T) \dots + P^*(b_n | T) =$$

$$= 2,32991 \times 10^{-11} + 4,07085 \times 10^{-5} + 1,07623 \times 10^{-5} + 5,77568 \times 10^{-10} + 2,07353 \times 10^{-9} + \dots + 4,33407 \times 10^{-13}$$

$$= 1,03910 \times 10^{-5}$$

↓

$$P^*(b_1 | T) = \prod_{i=1}^m P^*(v_{b_i} | v_{t_i}) = P^*_{\text{Vista}}(\text{nub} | \text{lluv}) * P^*_{\text{Temperatura}}(83 | 67) * P^*_{\text{Humedad}}(86 | 91) * P^*_{\text{Viento}}(\text{no} | \text{sí}) =$$

$$= 0,08219 * 1,67060 \times 10^{-7} * 0,01867 * 0,09086 = 2,32991 \times 10^{-11}$$

$$P^*_{\text{Vista}}(\text{nub} | \text{lluv}) = \frac{1-s}{nv} = \frac{1-0,75342}{3} = 0,08219$$

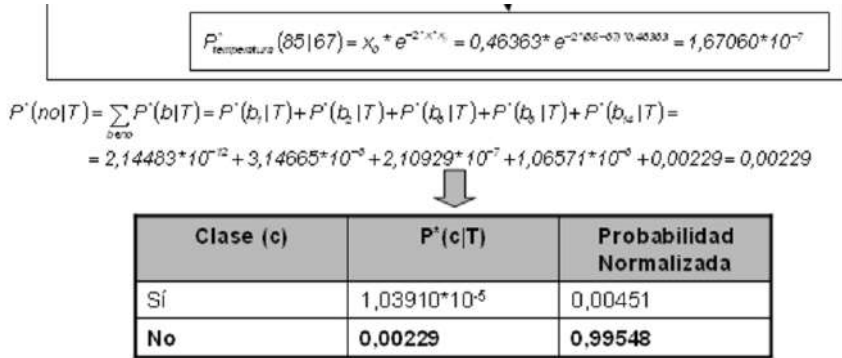


Fig. 6.37. Ejemplo de clasificación con K*.

Una vez definido el modo en que se clasifica un determinado ejemplo de test mediante el algoritmo K*, en la figura 6.37 se muestra un ejemplo concreto en el que se emplea dicho algoritmo. En él se clasifica un modelo de test tomando como ejemplos de entrenamiento los que se mostraron en la tabla de la figura 6.15, tomando los atributos “temperatura” y “humedad” como numéricos. El proceso que se sigue para determinar a qué clase pertenece un ejemplo de test concreto es el siguiente: en primer lugar, habría que calcular los parámetros x_0 y s que aún no se conocen para los pares atributo-valor del ejemplo de test. Posteriormente se aplican las ecuaciones, que de nuevo no son exactamente las definidas anteriormente: se han empleado las que los autores del algoritmo implementan en la herramienta Weka. Una vez obtenidas las probabilidades, se normalizan y se escoge la mayor de las obtenidas. En este caso, hay más del 99% de probabilidad a favor de la clase “No”. Esto se debe a que el último ejemplo del conjunto de entrenamiento (fila 14 de la figura 6.15) es casi idéntico al ejemplo de test por clasificar. En este ejemplo no se detallan todas las operaciones realizadas, sino un ejemplo de cada tipo: un ejemplo de la obtención de P^* para un atributo simbólico, otro de la obtención de P^* para un atributo numérico y otro para la obtención de la probabilidad de transformación del ejemplo de test en un ejemplo de entrenamiento.

6.5.6 Máquinas de vectores de soporte (SVM)

La máquina de vectores de soporte es un tipo de algoritmo de aprendizaje para problemas de regresión y reconocimiento de patrones que construyen su solución basándose en un subconjunto de los datos de entrenamiento, los “vectores de soporte” [BGV92]. Constituye una herramienta prometedora para estudiar regularidades en la clasificación de patrones, destacando dos características principales:

- Permite la construcción de varios algoritmos simultáneos para su elección en diferentes productos. Así la influencia del conjunto de funciones que pueden ser implementadas por una máquina de aprendizaje específica puede ser estudiada en un marco unificado.
- Construye basándose en resultados de la teoría de aprendizaje estadístico, llamada *principio de minimización del riesgo estructural* [VAPN79] (descrito más adelante) garantizando alta capacidad de generalización. Así hay razón para creer que las reglas de decisión construidas por el algoritmo del vector de soporte no reflejan incapacidades de la *learning machine*, pero sí algo los patrones seguidos por los datos.

La mayor diferencia entre las máquinas de vectores de soporte y otros métodos de aprendizaje es que las SVM no siguen el principio de minimización del riesgo empírico (ERM, *Empirical Risk Minimization*), que consiste en construir modelos que cometen pocos errores sobre los datos de entrenamiento, esperando que se comporten de igual forma ante datos futuros. Lo que pretenden las SVM es buscar modelos confiables, es decir, que produzcan predicciones en las que se pueda tener mucha confianza, aun a costa de cometer ciertos errores sobre los datos de entrenamiento. Ese principio recibe el nombre de *minimización del riesgo estructural* (SRM, *Structural Risk Minimization*). Las SVM buscan un modelo que estructuralmente tenga poco riesgo de cometer errores ante datos futuros, aunque puedan cometer más errores sobre los datos de entrenamiento.

Este principio inductivo (desarrollado por Vapnik y Chervonenkis, [VAPN74]), que ha demostrado ser superior al ERM, debe seleccionar un modelo generalizado de un conjunto finito de datos, con el problema consiguiente de *overfitting* o sobreajuste (modelo que convertía de un modo excesivamente fuerte las particularidades del sistema de entrenamiento y que generalizaba mal a los nuevos datos). El principio de SRM acomete este problema equilibrando la complejidad del modelo con su éxito en ajustar los datos del entrenamiento. Podemos afirmar pues que este principio trata de minimizar: el error en las entradas y el margen de separación entre clases [MRE].

6.5.6.1 SVM lineal

Supongamos que tenemos un conjunto de puntos etiquetados para entrenamiento, linealmente separables, como se aprecia en la figura siguiente.

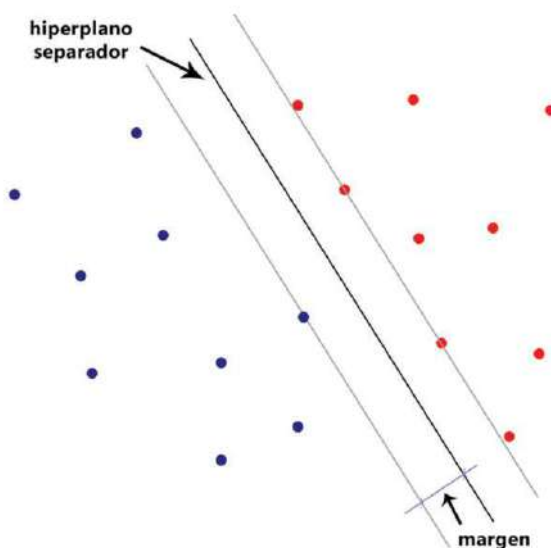


Fig. 6.38. Datos linealmente separables en el espacio de entradas.

Cada punto de entrenamiento, $x_i \in \mathbb{R}^n$, pertenece a alguna de las dos clases y_i , considerando un problema de clasificación binaria, se le ha dado una etiqueta y_i : $\{x_i, y_i\}$ $y_i \in \{-1, 1\}$ para $i = 1, \dots, n$. En la mayoría de los casos, la búsqueda del hiperplano adecuado en un espacio de entrada es demasiado restrictiva para ser de uso práctico. Como veremos más adelante en detalle, una solución a esta situación es mapear el espacio de entrada en un espacio de características de una dimensión mayor.

En todo caso desearemos encontrar el hiperplano que cumpla la ecuación:

$$x \cdot w + b = 0 \quad \text{Ec. 6.72}$$

Definido por el par (w, b) , tal que podamos separar el punto x_i :

- w es la normal que lo define.
- b es su distancia al origen de coordenadas.

Por tanto, buscamos la función que permita separar los datos:

$$f(x_i) = \text{signo}(w \cdot x + b) = \begin{cases} +1, & y_i = +1 \\ -1, & y_i = -1 \end{cases} \quad \text{Ec. 6.73}$$

Para el caso linealmente separable de S , podemos encontrar un único hiperplano óptimo, para el cual, el margen entre las proyecciones de los puntos de entrenamiento de dos diferentes clases es maximizado. Como vemos en la figura 6.39 la frontera de decisión debe estar tan lejos de los datos de ambas clases como sea posible.

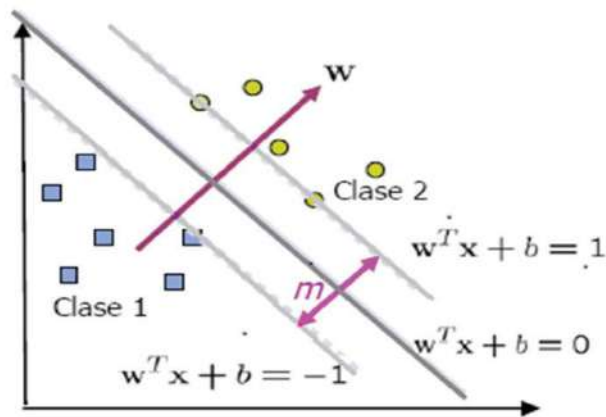


Fig. 6.39. Frontera de decisión lineal y margen.

Con este planteamiento, puede demostrarse de manera sencilla que el margen de separación viene dado por la expresión $1/\|w\|$, de modo que determinar el plano que maximice la distancia entre los puntos de las dos clases más próximos entre ellos puede formularse así:

$$\min (\|w\|^2) \quad \text{sujeto a: } y_i \cdot (x_i \cdot w + b) - 1 \geq 0, \forall i \in \{1, \dots, l\} \quad \text{Ec. 6.74}$$

Es, por tanto, un problema de programación cuadrática (QP) y puede ser resuelto por su problema equivalente con multiplicadores de Lagrange α_i :

$$\max \left(-\frac{1}{2} \|w\|^2 + \sum_{i,j=1}^l \alpha_i y_i (w x_i + b) - \sum_{i=1}^l \alpha_i \right) \quad \text{sujeto a } \alpha_i \geq 0 \quad \text{Ec. 6.75}$$

Al imponer como condición necesaria las derivadas parciales nulas con respecto a α_i tenemos:

$$\sum_{i,j=1}^l \alpha_i y_i = 0; \quad w = \sum_{i=1}^l \alpha_i y_i x_i \quad \text{Ec. 6.76}$$

Que a su vez permiten transformar la formulación al problema dual, también conocida como *formulación de Wolfe*:

$$\max \left(-\frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j (x_i \cdot x_j) + \sum_{i=1}^l \alpha_i \right) \quad \text{sueto a } \alpha_i \geq 0 \quad \text{Ec. 6.77}$$

El problema planteado es un caso particular de optimización restringida con inecuaciones lineales, normalmente referido como *el problema de optimización con condiciones de Karush-Kuhn-Tucker* (KKT) [VAPN74]. Estas condiciones desempeñan un papel central en la teoría y la práctica de la optimización restringida. Las condiciones KKT se satisfacen en la solución de cualquier problema de optimización restringida mediante funciones regulares, con cualquier clase de restricción. El problema formulado para las SVM es de tipo convexo, y para los problemas convexos, las condiciones de KKT son necesarias y suficientes para w, b . Las condiciones de Karush-Kuhn-Tucker tienen como consecuencia que una solución sólo puede estar en una situación con respecto a cada restricción en forma de desigualdad: o bien está en el interior de la región definida por la desigualdad (restricción inactiva), o bien está en la frontera (restricción activa). La eliminación de una restricción inactiva no tiene ningún efecto en la solución del problema. Así, solucionar el problema de SVM es equivalente a encontrar una solución a las condiciones de KKT, que queda expresado con la formulación:

$$\begin{aligned} y_i \cdot (x_i \cdot w + b) - 1 &\geq 0 \\ \alpha_i &\geq 0, \forall i \\ \alpha_i \{y_i \cdot (x_i \cdot w + b) - 1\} &= 0, \forall i \end{aligned} \quad \text{Ec. 6.78}$$

De estas condiciones se deduce que los únicos valores $\alpha_i \neq 0$ son aquellos en los que las condiciones son satisfechas con el signo de igualdad (restricción activa). Cada punto x_i correspondiente con $\alpha_i > 0$ es llamado *vector de soporte*:

$$y_i \cdot (x_i \cdot w + b) - 1 = 0 \quad \text{Ec. 6.79}$$

Habitualmente, la solución a estas ecuaciones del problema dual, más manejable que la formulación original, se encuentra con métodos numéricos. La convexidad garantiza una solución única (esto supone una ventaja con respecto al modelo general de redes neuronales) y las implementaciones actuales permiten una eficiencia razonable para problemas reales con muchos ejemplos y atributos. Por último, es interesante apuntar que la descripción dada por los puntos de los vectores soporte es capaz de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos de fuera de la frontera.

6.5.6.2 SVM lineal de margen blando (*soft margin*)

Cuando el conjunto S no es linealmente separable porque algunos datos quedan al otro lado de cualquier frontera lineal, existe una extensión directa que permite violaciones a la clasificación de la SVM manteniendo la estructura lineal.

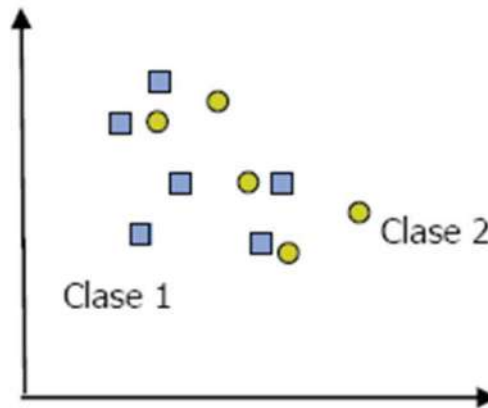


Fig. 6.40. Datos no linealmente separables en el espacio de entradas.

Para tratar con datos que no son linealmente separables, el análisis previo puede ser generalizado introduciendo algunas variables no-negativas $\xi_i \geq 0$, de tal modo que la formulación es modificada a:

$$\min \left(\|w\|^2 + C \cdot \sum_{i=0}^l \xi_i \right) \quad \text{sujeto a: } y_i \cdot (x_i \cdot w + b) + \xi_i - 1 \geq 0, \forall i \quad \text{Ec. 6.80}$$

Los términos $\xi_i \neq 0$ en esta ecuación son aquellos para los cuales el punto x_i no satisface la ecuación. Por tanto, el término $\sum_{i=0}^l \xi_i$ puede ser tomado como una medida de error permitido de la clasificación a costa de generalizar bien la frontera de decisión. El parámetro C puede ser definido como una constante de regularización. Éste es el único parámetro libre de ser ajustado en la formulación de la SVM. El ajuste de este parámetro puede hacer un balance entre la maximización del margen y la violación a la clasificación.

De nuevo este problema admite una formulación dual con los multiplicadores de Lagrange y la transformación de Wolfe:

$$\max \left\{ -\frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j (x_i \cdot x_j) + \sum_{i=1}^l \alpha_i \right\}$$

Ec. 6.81

$$\text{sujeto a } 0 \leq \alpha_i \leq C, \sum_{i=1}^l \alpha_i y_i = 0$$

La ventaja de esta formulación es que no depende de los términos ξ_i , ni tampoco de sus multiplicadores de Lagrange, μ_i . Si ahora buscamos el hiperplano óptimo de la ecuación anterior, volvemos a encontrarnos con un problema de programación cuadrática con restricciones de desigualdad, de nuevo sujeto a las condiciones de Karush-Kuhn-Tucker, llegando a las condiciones:

$$0 \leq \alpha_i \leq C, \forall i$$

$$y_i \cdot (x_i \cdot w + b) + \xi_i - 1 \geq 0, \forall i$$

$$\alpha_i \{y_i \cdot (x_i \cdot w + b) + \xi_i - 1\} = 0, \forall i$$

$$\xi_i \geq 0, \forall i$$

Ec. 6.82

Puede verse que la única diferencia con el caso separable es que los multiplicadores ahora están limitados por la constante C . Como en el caso anterior, la condición $\alpha_i \neq 0$ se da únicamente para los llamados *vectores de soporte*, pero ahora nos encontramos con dos tipos de vectores de soporte:

- Para $0 < \alpha_i < C$, el correspondiente vector de soporte x_i satisface las igualdades.

$$y_i(x_i \cdot w + b) - 1 = 0; \text{ y } \xi_i = 0.$$

- Para $\alpha_i = C$, el correspondiente vector de soporte x_i no satisface la desigualdad con $\xi_i > 0$, y tendremos un error de clasificación si además se da $\xi_i > 1$.

Para el segundo tipo, nos referimos a estos vectores de soporte como errores, mientras que cada punto x_i con $\alpha_i = 0$ es clasificado correctamente y está claramente alejado del margen de decisión.

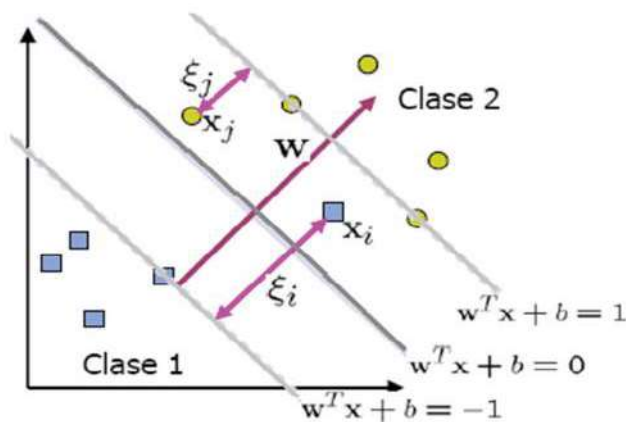


Fig. 6.41. ξ_i en el error de clasificación.

Finalmente, para construir el hiperplano óptimo $w \cdot x + b$ se utiliza de nuevo la expresión:

$$w = \sum_{i=1}^l \alpha_i y_i x_i$$

Y el escalar b puede ser determinado de las condiciones Karush-Kuhn-Tucker. La función de decisión generalizada queda formulada de nuevo así:

$$f(x) = \text{sign}(wx + b) = \text{sign}\left(\sum_{i=1}^l \alpha_i y_i x_i x + b\right) \quad \text{Ec. 6.83}$$

6.5.6.3 SVM no lineal. Funciones *kernel*

El algoritmo SVM presentado hasta ahora resuelve el problema de los datos no separables hasta un cierto grado, incluyendo un margen de error C en la clasificación. En general, puede resolverse la limitación de separación lineal mediante un mapeo de los puntos de entrada a un espacio de características de una dimensión mayor (por ejemplo, si los puntos están en \mathbb{R}^2 entonces son mapeados por la SVM a \mathbb{R}^3) y encuentra un hiperplano que los separe y maximice el margen m (distancia entre el hiperplano separador y el punto más cercano) entre las clases en este espacio. Sin ningún conocimiento del mapeo, la SVM encuentra el hiperplano óptimo utilizando el producto escalar con funciones

en el espacio de características que son llamadas *funciones kernel*. El planteamiento dual de las SVM permite trabajar con estas funciones de manera eficiente sin calcular explícitamente las representaciones de los ejemplos en el espacio de características.

Hemos visto que los datos aparecen en el problema del entrenamiento en forma de productos, $x_i \cdot x_j$. Ahora suponemos que primero mapeamos los datos a un cierto espacio euclídeo H , usando un mapeo que llamaremos Φ :

$$\Phi: R^2 \rightarrow H \quad \text{Ec. 6.84}$$

Entonces, el algoritmo del entrenamiento dependería solamente de los datos a través de productos en el espacio H , es decir, en las funciones de la forma $\Phi(x_i) \cdot \Phi(x_j)$. Ahora, si hubiera una función *kernel* K tal que $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, necesitaríamos solamente el uso de K en el algoritmo del entrenamiento, e incluso nunca necesitaríamos saber explícitamente cuál es el valor de Φ , puesto que únicamente utilizaríamos los productos escalares dados por la función $K(\cdot)$.

En la fase de prueba, una SVM utiliza los productos escalares calculados sobre cada punto de prueba dado, x , con los coeficientes w , o más específicamente calculando el signo resultante de la suma del producto escalar sobre los vectores de soporte:

$$f(x) = \text{sign}(wx + b) = \sum_{i,j=1}^l \alpha_i y_i (\Phi(x_i) \cdot \Phi(x)) + b = \sum_{i,j=1}^l \alpha_i y_i K(x_i, x) + b \quad \text{Ec. 6.85}$$

Donde x_i son los vectores de soporte. Por tanto, podemos evitar calcular $\Phi(x)$ explícitamente y utilizar en su lugar la función *Kernel*: $K(x_i, x) = \Phi(x_i) \cdot \Phi(x)$.

Vemos un ejemplo muy simple de un *kernel* permitido, ilustrado en la figura 6.42, para el cual podemos construir el mapeo de Φ . Suponemos que los datos son vectores en R^2 , y elegimos $K(x_i, x_j) = (x_i \cdot x_j)^2$.

Necesitamos un espacio H , y un mapeo de Φ desde R^2 a H , tal que $(x \cdot y)^2 = \Phi(y) \cdot \Phi(x)$. Por ejemplo, nos vale para este propósito:

$$H = R^3 \text{ y } \Phi(x) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^t \quad \text{Ec. 6.86}$$

Para los datos en L definido en el cuadrante $[-1, 1] \times [-1, 1] \in R^2$, la imagen del mapeo de Φ se indica en la figura 6.42, donde se muestra claramente la separabilidad lineal de los datos en el nuevo espacio H .

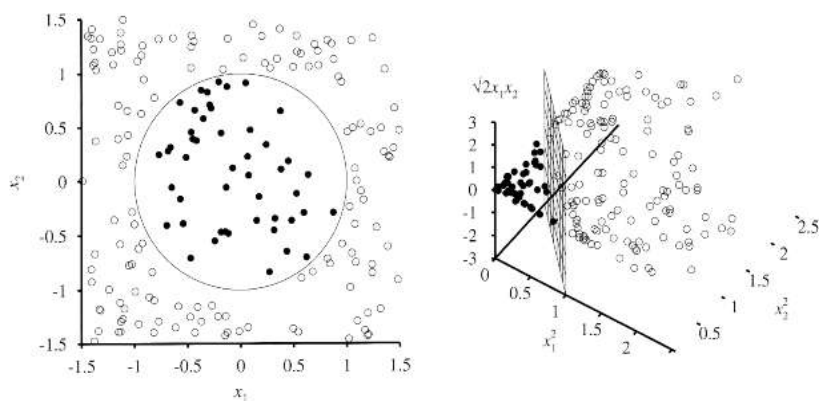


Fig. 6.42. Ejemplo de proyección de un problema 2D a un espacio 3D linealmente separable.

Para construir un clasificador SVM podemos reutilizar la formulación anterior únicamente incorporando la función de *kernel*:

$$\begin{aligned} &\text{maximizar} \quad -\frac{1}{2} \sum_{i,j=1}^l \alpha_i y_i \alpha_j y_j K(x_i, x_j) + \sum_{i=1}^l \alpha_i \\ &\text{sujeto a} \quad 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^l \alpha_i y_i = 0 \end{aligned} \quad \text{Ec. 6.87}$$

Y la función de decisión queda ahora de la siguiente forma:

$$f(x) = \text{sign}(wx + b) = \text{sign}\left(\sum_{i=1}^l \alpha_i y_i K(x_i, x) + b\right) \quad \text{Ec. 6.88}$$

Entre las funciones *kernel* más habituales tenemos las siguientes:

- **Kernel lineal.**

$$K(x_i, x_j) = x_i \cdot x_j \quad \text{Ec. 6.89}$$

Es la función *kernel* más simple que se corresponde con el caso lineal presentado al principio.

- **Kernel con base radial (RBF).**

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \quad \text{Ec. 6.90}$$

Siendo γ una constante de proporcionalidad cuyo rango de valores útiles debe ser estimado para cada aplicación en particular. Ésta es la más popular de las usadas en SVM.

- **Kernel polinómica.**

$$K(x_i, x_j) = (ax_i \cdot x_j + b)^p \quad \text{Ec. 6.91}$$

- **Kernel sigmoide.**

$$K(x_i, x_j) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}; \quad z = \gamma x_i \cdot x_j + k \quad \text{Ec. 6.92}$$

Por tanto, la extensión de la SVM al problema no lineal requiere de una configuración y ajuste cuidadosos, que puede resumirse en estos pasos:

- Normalizar los atributos.
- Encontrar los parámetros más apropiados.
 - Parámetro C (coste, complejidad).
 - ◊ Valor alto: complejidad mayor, posible sobreajuste (*overfitting*).
 - ◊ Valor bajo: complejidad menor, tal vez mejor generalización pero a costa de sesgo (*underfitting*).
 - Funciones *kernel*.
 - ◊ Lineal: sin parámetro.
 - ◊ Polinómico: exponente (d).
 - ◊ Gaussiano: γ .

(Una estrategia puede ser comenzar con un *kernel* lineal y después comparar con polinómico y gaussiano).
- Construir el modelo con los mejores parámetros y todo el conjunto de entrenamiento.

Como cada evaluación supone una validación cruzada, y el espacio de parámetros es multidimensional (por ejemplo C , γ), la optimización de la SVM es un proceso costoso computacionalmente.

6.5.6.4 Clasificación multiclase

Aunque se ha presentado el desarrollo teórico de las técnicas SVM para el problema de la clasificación binaria, estos algoritmos también se pueden aplicar para trabajar con más de dos clases. Tal como ya se presentó con la clasificación mediante regresión, se pueden utilizar dos estrategias:

- **Uno contra el resto:**
 - Se entrenan k clasificadores (una clase es la positiva y el resto la negativa).
 - Se predice la clase para todos los clasificadores.
 - La clase asignada es aquella con la que se consiguió mayor margen (en el caso en que se clasifique como positiva en más de un clasificador).
 - Por ejemplo, con tres clases (A, B, C), generar:
 - ◊ Separador A versus B+C.
 - ◊ Separador B versus A+C.
 - ◊ Separador C versus A+B.
 - ◊ En test, escoger el clasificador que dé más valor (la salida de la función de clasificación SVM es la distancia al hiperplano).
- **Uno contra uno:**
 - Se construyen $k(k-1)/2$ clasificadores, cada uno entrena datos de dos diferentes.
 - Se usa la estrategia de votación para clasificar: cada clasificador binario se considera como un voto y se toma la clase con mayor número de votos.
 - Por ejemplo, con tres clases (A, B, C), generar:
 - ◊ Separador A versus B.
 - ◊ Separador A versus C.
 - ◊ Separador B versus C.
 - ◊ En test, escoger el clasificador que reciba más puntos por votación.

Finalmente, para concluir esta sección a modo de resumen, se puede destacar que la principal fortaleza de las técnicas SVM es su capacidad de escalar relativamente bien para datos en espacios dimensionalmente altos, con un entrenamiento relativamente sencillo, y la existencia de óptimo global, a diferencia de los óptimos locales en las redes de neuronas. Su principal debilidad es la complejidad para su ajuste y optimización,

que como se ha indicado implica seleccionar la función *kernel* más apropiada a cada problema y una metodología eficiente para ajustar los parámetros de inicialización con un procedimiento iterativo en un conjunto de test independiente del utilizado para el entrenamiento.

6.5.7 Redes de neuronas

Las redes de neuronas constituyen una técnica inspirada en los trabajos de investigación, iniciados en 1930, que pretendían modelar computacionalmente el aprendizaje humano llevado a cabo a través de las neuronas en el cerebro [RM86, CR95]. Posteriormente se comprobó que tales modelos no eran del todo adecuados para describir el aprendizaje humano. Las redes de neuronas constituyen una nueva forma de analizar la información con una diferencia fundamental con respecto a las técnicas tradicionales: son capaces de detectar y aprender complejos patrones y características dentro de los datos [SN88, FU94]. Se comportan de forma parecida a nuestro cerebro aprendiendo de la experiencia y del pasado, y aplicando tal conocimiento a la resolución de problemas nuevos. Este aprendizaje se obtiene como resultado del entrenamiento (*training*) y éste permite la sencillez y la potencia de adaptación y evolución ante una realidad cambiante y muy dinámica. Una vez adiestradas las redes de neuronas, pueden hacer previsiones, clasificaciones y segmentación. Presentan, además, una eficiencia y fiabilidad similar a los métodos estadísticos y sistemas expertos, si no mejor, en la mayoría de los casos. En aquellos casos de muy alta complejidad, las redes neuronales se muestran como especialmente útiles dada la dificultad de modelado que supone para otras técnicas. Sin embargo, las redes de neuronas tienen el inconveniente de la dificultad de acceder y comprender los modelos que generan, y presentan dificultades para extraer reglas de tales modelos. Otra característica es que son capaces de trabajar con datos incompletos e, incluso, contradictorios lo que, dependiendo del problema, puede resultar una ventaja o un inconveniente. Las redes de neuronas poseen las dos formas de aprendizaje, supervisado y no supervisado, ya comentadas [WI98], derivadas del tipo de paradigma que usan: el no supervisado (usa paradigmas como los ART, *Adaptive Resonance Theory*), y el supervisado, que suele usar el paradigma del *backpropagation* [RHW86].

Las redes de neuronas están siendo utilizadas en distintos y variados sectores como la industria, el gobierno, el ejército, las comunicaciones, la investigación espacial, la banca y las finanzas, los seguros, la medicina, la distribución, la robótica, el *marketing*, etc. En la actualidad se está estudiando la posibilidad de utilizar técnicas avanzadas y novedosas como los algoritmos genéticos para crear nuevos paradigmas que mejoren el entrenamiento y la propia selección y diseño de la arquitectura de la red (número de capas y neuronas), diseño que ahora debe realizarse en base a la experiencia del analista y para cada problema concreto.

6.5.7.1 Estructura de las redes de neuronas

Las redes neuronales se construyen estructurando en una serie de niveles o capas (al menos tres: entrada, procesamiento u oculta y salida) compuestas por nodos o “neuronas”, que tienen la estructura que se muestra en la figura 6.43.

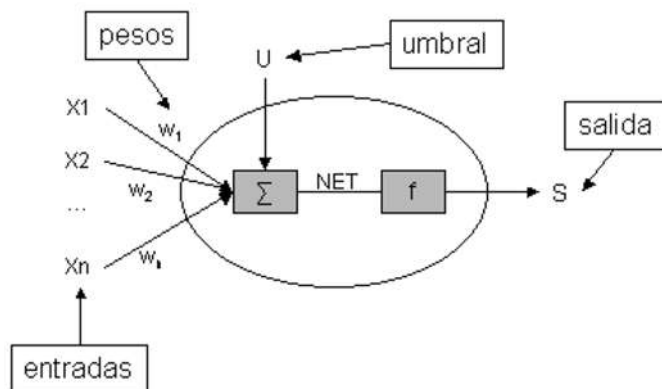


Fig. 6.43. Estructura de una neurona.

Tanto el umbral como los pesos son constantes que se inicializarán aleatoriamente y durante el proceso de aprendizaje serán modificadas. La salida de la neurona se define tal y como se muestra en las ecuaciones 6.93 y 6.94.

$$NET = \sum_{i=1}^N X_i w_i + U \quad \text{Ec. 6.93}$$

$$S = f(NET) \quad \text{Ec. 6.94}$$

Como función f se suele emplear una función sigmoideal, bien definida entre 0 y 1 (ecuación 6.95) o entre -1 y 1 (ecuación 6.96).

$$f(x) = \frac{1}{1 + e^{-x}} \quad \text{Ec. 6.95}$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \text{Ec. 6.96}$$

Cada neurona está conectada a todas las neuronas de las capas anterior y posterior a través de los pesos o “dendritas”, tal y como se muestra en la figura 6.44.

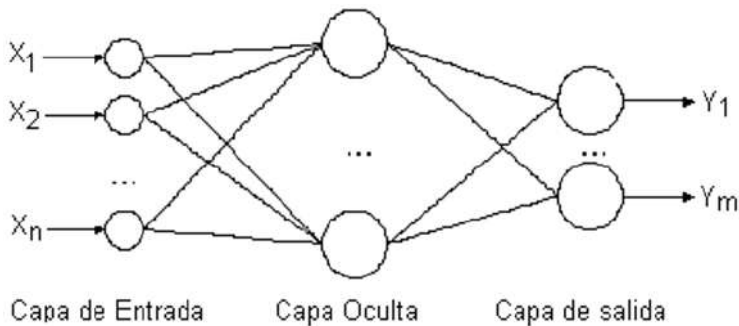


Fig. 6.44. Estructura de la red de neuronas.

Cuando un nodo recibe las entradas o “estímulos” de otras los procesa para producir una salida que transmite a la siguiente capa de neuronas. La señal de salida tendrá una intensidad fruto de la combinación de la intensidad de las señales de entrada y de los pesos que las transmiten. Los pesos o dendritas tienen un valor distinto para cada par de neuronas que conectan, pudiendo así fortalecer o debilitar la conexión o comunicación entre neuronas particulares. Los pesos son modificados durante el proceso de entrenamiento.

El diseño de la red de neuronas consistirá, entre otras cosas, en la definición del número de neuronas de las tres capas de la red. Las neuronas de la capa de entrada y las de la capa de salida vienen dadas por el problema a resolver, dependiendo de la codificación de la información. En cuanto al número de neuronas ocultas (o número de capas ocultas) se determinará por prueba y error. Por último, debe tenerse en cuenta que la estructura de las neuronas de la capa de entrada se simplifica, dado que su salida es igual a su entrada: no hay umbral ni función de salida.

6.5.7.2 Proceso de entrenamiento (retropropagación)

Existen distintos métodos o paradigmas mediante los cuales estos pesos pueden ser variados durante el adiestramiento, de los cuales el más utilizado es el de retropropagación (*backpropagation*) [RHW86]. Este paradigma varía los pesos de acuerdo a las diferencias encontradas entre la salida obtenida y la que debería obtenerse. De esta forma, si las diferencias son grandes, se modifica el modelo de forma importante y, según van siendo menores, se va convergiendo a un modelo final estable. El error en una red de neuronas para un patrón $[x = (x_1, x_2, \dots, x_n), t(x)]$, siendo x el patrón de entrada, $t(x)$ la salida deseada e $y(x)$ la proporcionada por la red, se define como se muestra en la ecuación 6.97 para m neuronas de salida y como se muestra en la ecuación 6.98 para una neurona de salida.

$$e(x) = \|t(x) - y(x)\|^2 = \frac{1}{2} \sum_{i=1}^m (t_i(x) - y_i(x))^2 \quad \text{Ec. 6.97}$$

$$e(x) = \frac{1}{2} (t(x) - y(x))^2 \quad \text{Ec. 6.98}$$

El método de descenso de gradiente consiste en modificar los parámetros de la red siguiendo la dirección negativa del gradiente del error. Lo que se realizaría mediante la ecuación 6.99.

$$w^{\text{nuevo}} = w^{\text{anterior}} + \alpha \left(-\frac{\partial e}{\partial w} \right) = w^{\text{anterior}} - \alpha \frac{\partial e}{\partial w} \quad \text{Ec. 6.99}$$

En esta ecuación, w es el peso a modificar en la red de neuronas (pasando de w^{anterior} a w^{nuevo}) y α es la razón de aprendizaje, que se encarga de controlar cuánto se desplazan los pesos en la dirección negativa del gradiente. Influye en la velocidad de convergencia del algoritmo, puesto que determina la magnitud del desplazamiento. El algoritmo de retropropagación es el resultado de aplicar el método de descenso del gradiente a las redes de neuronas. El algoritmo completo de retropropagación se muestra en la figura 6.45.

- Paso 1:** Inicialización aleatoria de los pesos y umbrales.
- Paso 2:** Dado un patrón del conjunto de entrenamiento $(x, t(x))$, se presenta el vector x a la red y se calcula la salida de la red para dicho patrón, $y(x)$.
- Paso 3:** Se evalúa el error $e(x)$ cometido por la red.
- Paso 4:** Se modifican todos los parámetros de la red utilizando la ec.2.88.
- Paso 5:** Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje.
- Paso 6:** Se realizan n ciclos de aprendizaje (pasos 2, 3, 4 y 5) hasta que se verifique el criterio de parada establecido.

Fig. 6.45. Pseudocódigo del algoritmo de retropropagación.

En cuanto al criterio de parada, se debe calcular la suma de los errores en los patrones de entrenamiento. Si el error es constante de un ciclo a otro, los parámetros dejan de sufrir modificaciones y se obtiene así el error mínimo. Por otro lado, también se debe tener en cuenta el error en los patrones de validación, que se presentarán a la red tras n ciclos de aprendizaje. Si el error en los patrones de validación evoluciona favorablemente se continúa con el proceso de aprendizaje. Si el error no desciende, se detiene el aprendizaje.

A modo de resumen de las dos secciones anteriores, la tabla de la figura 6.46 muestra una comparativa entre las redes neuronales artificiales y los algoritmos SVM [ALBA17].

Comparación de SVM y ANN	
SVM	ANN
<ul style="list-style-type: none"> • Separación mediante función <i>kernel</i> que transforma a espacios de dimensión superior. • Espacio de búsqueda con un único mínimo global. • Entrenamiento eficiente. • Clasificación eficiente. • Clave de diseño: diseño de función <i>kernel</i> y parámetro de coste. 	<ul style="list-style-type: none"> • Separación mediante capas ocultas para transformar el espacio de entrada a otros intermedios. • Espacio de búsqueda con múltiples mínimos locales. • Entrenamiento costoso. • Clasificación eficiente. • Clave de diseño: número de nodos y capas ocultas.

Fig. 6.46. Comparativa entre las redes neuronales artificiales y los algoritmos SVM.

6.5.8 Lógica borrosa (*fuzzy logic*)

La lógica borrosa surge de la necesidad de modelar la realidad de una forma más exacta evitando precisamente el determinismo o la exactitud [ZAD65, CPS98]. En palabras menos pretenciosas lo que la lógica borrosa permite es el tratamiento probabilístico de la categorización de un colectivo [ZAD65].

Así, para establecer una serie de grupos, segmentos o clases en los cuales se puedan clasificar a las personas por la edad, lo inmediato sería proponer unas edades límite para establecer tal clasificación de forma disjunta. Así los niños serían aquellos cuya edad fuera menor a los doce años, los adolescentes aquellos entre doce y diecisiete años, los jóvenes aquellos entre dieciocho y treinta y cinco, las personas maduras entre treinta y seis y cuarenta y cinco años y así sucesivamente. Se habrían creado unos grupos disjuntos cuyo tratamiento, a efectos de clasificación y procesamiento, es muy sencillo: basta comparar la edad de cada persona con los límites establecidos. Sin

embargo, enseguida se observa que esto supone una simplificación enorme dado que una persona de dieciséis años, once meses y veinte días pertenecería al grupo de los adolescentes y, seguramente, es más parecido a una persona de dieciocho (miembro de otro grupo) que a uno de doce (miembro de su grupo). Lógicamente no se puede establecer un grupo para cada año, dado que sí se reconocen grupos, y no muchos, con comportamientos y actitudes similares en función de la edad. Lo que implícitamente se está descubriendo es que las clases existen, pero que la frontera entre ellas no es clara ni disjunta sino “difusa” y que una persona puede tener aspectos de su mentalidad asociados a un grupo y otros asociados a otro grupo, es decir, que implícitamente se está distribuyendo la pertenencia entre varios grupos. Cuando esto se lleva a una formalización matemática surge el concepto de *distribución de posibilidad*, de forma que lo que entendería como función de pertenencia a un grupo de edad serían unas curvas de posibilidad. Por tanto, la lógica borrosa es aquella técnica que permite y trata la existencia de barreras difusas o suaves entre los distintos grupos en los que se categoriza un colectivo o entre los distintos elementos, factores o proporciones que concurren en una situación o solución [BS97].

Para identificar las áreas de utilización de la lógica difusa basta con determinar cuántos problemas hacen uso de la categorización disjunta en el tratamiento de los datos para observar la cantidad de posibles aplicaciones que esta técnica puede tener [ZAD65]. Sin embargo, el tratamiento ortodoxo y purista no siempre está justificado dada la complejidad que induce en el procesamiento (pasamos de valores a funciones de posibilidad) y un modelado sencillo puede ser más que suficiente. Aun así, existen problemáticas donde este modelado sí resulta justificado, como en el control de procesos y la robótica, entre otros. Tal es así que un país como Japón, líder en la industria y la automatización, dispone del Laboratory for International Fuzzy Engineering Research (LIFE) y empresas como Yamaichi Securities y Canon hacen un extenso uso de esta técnica.

6.5.9 Técnicas genéticas: algoritmos genéticos (*genetic algorithms*)

Los algoritmos genéticos son otra técnica que tiene su inspiración en la biología como las redes de neuronas [GOLD89, MIC92, MITC96]. Estos algoritmos representan el modelado matemático de cómo los cromosomas en un marco evolucionista alcanzan la estructura y composición más óptima en aras de la supervivencia. Entendiendo la evolución como un proceso de búsqueda y optimización de la adaptación de las especies que se plasma en mutaciones y cambios de los genes o cromosomas, los algoritmos genéticos hacen uso de las técnicas biológicas de reproducción (mutación y cruce) para ser utilizadas en todo tipo de problemas de búsqueda y optimización. Se da la mutación

cuando alguno o algunos de los genes cambian bien de forma aleatoria o de forma controlada vía funciones y se obtiene el cruce cuando se construye una nueva solución a partir de dos contribuciones procedentes de otras soluciones “padre”. En cualquier caso, tales transformaciones se realizan sobre aquellos especímenes o soluciones más aptas o mejor adaptadas. Dado que los mecanismos biológicos de evolución han dado lugar a soluciones (los seres vivos) realmente idóneas, cabe esperar que la aplicación de tales mecanismos a la búsqueda y optimización de otro tipo de problemas tenga el mismo resultado. De esta forma, los algoritmos genéticos transforman los problemas de búsqueda y optimización de soluciones en un proceso de evolución de unas soluciones de partida. Las soluciones se convierten en cromosomas, transformación que se realiza pasando los datos a formato binario, y a los mejores se les van aplicando las reglas de evolución (funciones probabilísticas de transición) hasta encontrar la solución óptima. En muchos casos, estos mecanismos brindan posibilidades de convergencia más rápidos que otras técnicas.

El uso de estos algoritmos no está tan extendido como otras técnicas, pero van siendo cada vez más utilizados directamente en la solución de problemas, así como en la mejora de ciertos procesos presentes en otras herramientas. Así, por ejemplo, se usan para mejorar los procesos de entrenamiento y selección de la arquitectura de las redes de neuronas, para la generación e inducción de árboles de decisión y para la síntesis de programas a partir de ejemplos (*genetic programming*).

7.1 Agrupamiento. *Clustering*

En el capítulo 5 se introdujeron los problemas de clasificación. Éstos consisten en determinar un modelo que permite etiquetar una instancia de datos asociándola a una categoría o clase existente.

Pero ¿qué sucede cuando no se tienen clases previas? Esta situación es muy habitual, determinar los tipos de consumidores de un determinado producto o en cuántos grupos pueden dividirse los conductores en función de su comportamiento o tipos de vulnerabilidades que pueden comprometer la integridad de un sistema o si las personas que sufren una misma dolencia son un grupo homogéneo o pueden establecerse varias tipologías que conducen a la misma enfermedad, etc. Para todos estos casos y muchísimos otros similares, se puede tener una intuición aproximada de la cantidad de agrupaciones que pueden realizarse, pero ésta puede estar equivocada, incluso sesgada por juicios *a priori*. Existe un conjunto de técnicas, de *clustering* en inglés, que permiten encontrar las propiedades comunes que agrupan un conjunto de datos. Todas ellas utilizan alguna medida que tiene por objetivo maximizar la distancia de las instancias al agrupamiento al que no pertenecen, minimizando la distancia al propio. En función del tipo de distancia utilizada o cómo se aplica ésta para obtener las agrupaciones se distinguen las siguientes principales técnicas de agrupamiento:

- Jerárquico.
- Particionamiento.
- Basadas en densidad.

- Basadas en modelo.
- *Fuzzy* (borroso).

Para aplicar las técnicas de agrupamiento se requiere que los datos, en la medida de lo posible, estén completos, los atributos nominales deben ser binarizados y todos los atributos normalizados o estandarizados. De no estar estandarizados, se produce un sesgo importante, las variaciones de atributos que tienen sus valores en grandes magnitudes pesarán mucho más que los de las pequeñas.

Se van a utilizar los atributos del primer componente principal que se obtuvo al aplicar PCA en el capítulo dedicado a la clasificación junto con los de las condiciones meteorológicas y si hubo o no lluvia. No se ha mencionado, pero, por supuesto, previa a la aplicación de los algoritmos de agrupamiento, hay que tratar de reducir la dimensionalidad, eliminar atributos correlados y eliminar datos atípicos que pueden sesgar el resultado.

Para el ejemplo que estamos tratando, habrá que binarizar el atributo Lluvia_SN y estandarizar el resto. Si se hubiera considerado el atributo día de la semana, entonces habría que haber añadido siete nuevos atributos *dummies* binarios.

```
dfMedidas_CLUS <- dfMedidas[, names(dfMedidas) %in% c("Lluvia_SN", "Lluvia", "T_MAX",
"T_MIN",
"Viento_MAX", "Viento_MED", "CO", "BEN",
"NO", "NO2", "TOL", "SO2", "EBE", "PM2.5",
"PM10", "PM25", "TCH", "PM10", "O3")]

dfMedidas_CLUS$Lluvia_SN <- as.integer(dfMedidas_CLUS$Lluvia_SN) - 1

dfMedidas_CLUS <- na.omit(dfMedidas_CLUS) #eliminamos instancias si no tienen algún
valor

dfMedidas_CLUSscale <- as.data.frame(scale(dfMedidas_CLUS)) # estandarizamos los atri-
butos columna
```

Ahora todos los atributos tienen media 0 y desviación estándar 1.

7.1.1 Agrupamiento jerárquico

El concepto de la agrupación jerárquica es muy simple. Definida una métrica, se agrupan las dos instancias de datos más próximas, el valor medio de ambas (u otra estrategia) las reemplaza, el proceso continúa hasta que todas las instancias han sido agrupadas en un solo grupo. El resultado es una agrupación jerárquica de las instancias de datos.

```

distancia <- dist(dfMedidas_CLUSScale, method = "euclidean") #matriz de distancias
euclideas
clus_hc <- hclust(distancia, method="ward")
plot(clus_hc)

```

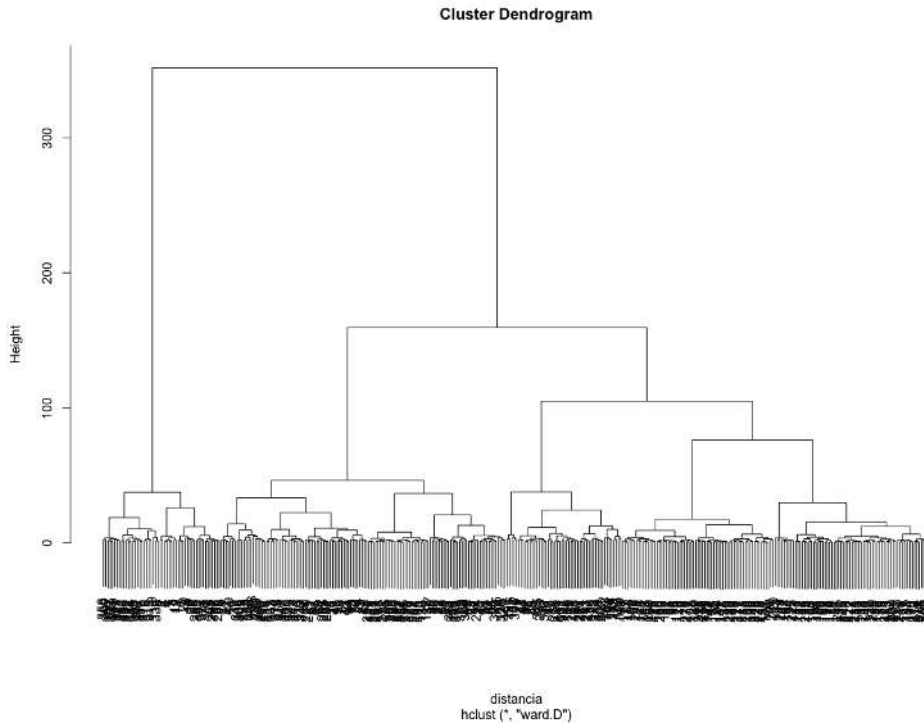


Fig. 7.1. Dendrograma del agrupamiento jerárquico.

Si se elige otra métrica o la forma de considerar ésta para realizar los grupos (en este caso se ha aplicado la técnica de Ward), el agrupamiento cambia.

Se aprecia en la figura como se han ido agrupando todas las instancias de datos. En función del corte en la altura, se establecerá un número u otro de agrupaciones.

```

summary(C5.0(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED,
data = setEntrena, rules = TRUE))

```

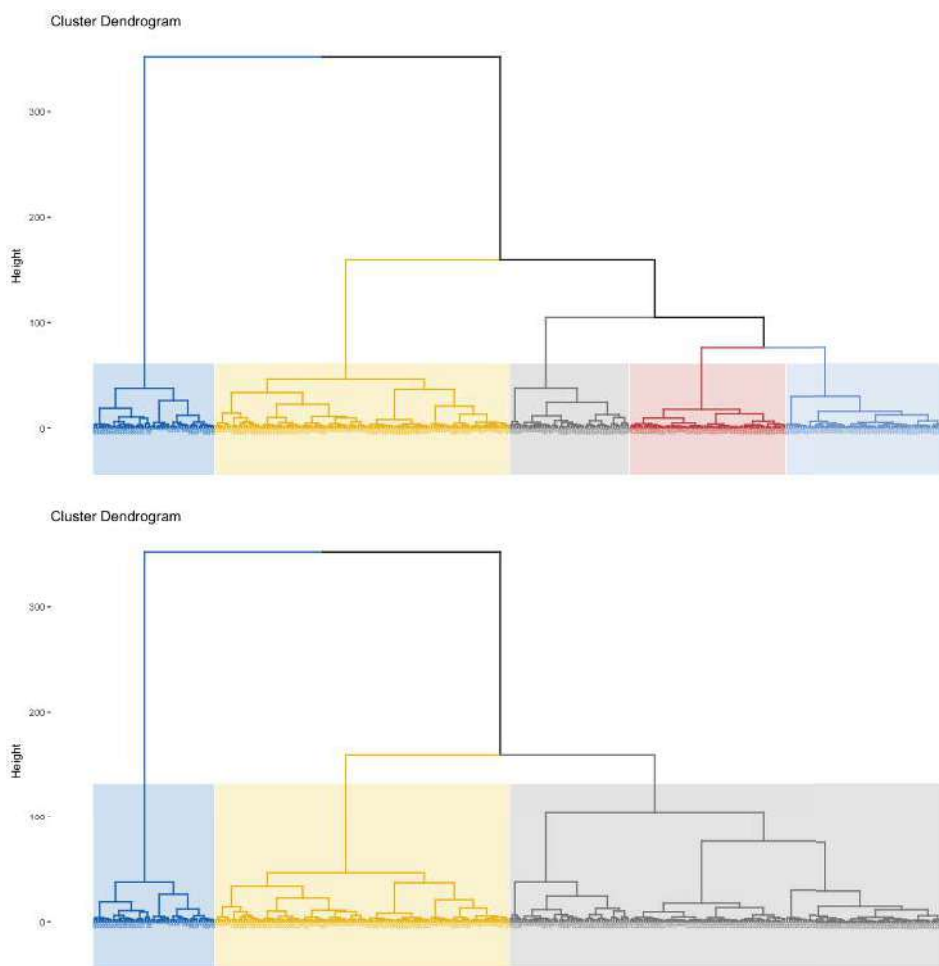


Fig. 7.2. Dendrogramas para dos agrupaciones diferentes, resultado del agrupamiento jerárquico.

En este momento se pone de manifiesto una de las cuestiones más importantes cuando se utilizan las técnicas de agrupamiento. ¿Cuál es el número adecuado de agrupamientos?

Antes de dar respuesta a esta cuestión, se muestra otra aplicación del agrupamiento jerárquico. Si la agrupación se realiza sobre el conjunto de datos transpuesto, es decir, en las filas están los atributos y en las columnas las instancias de datos, entonces se obtiene una relación de proximidad entre los atributos.

```

distanciaT <- dist(t(dfMedidas_CLUSscale), method = "euclidean") #matriz de distan-
cias euclídeas

clus_hcT <- hclust(distanciaT, method="ward")

fviz_dend(clus_hcT)

```

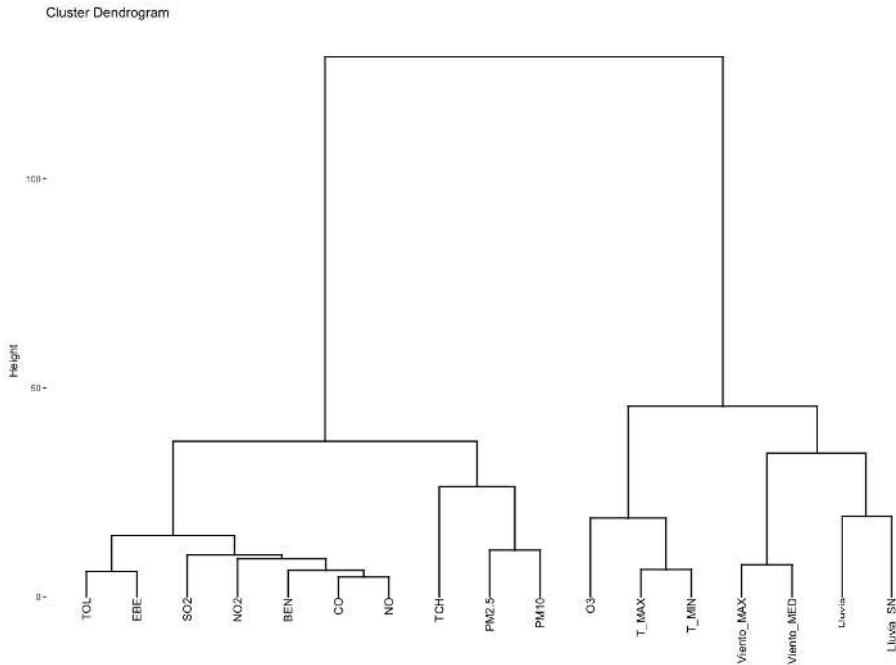


Fig. 7.3. Agrupamiento jerárquico de los atributos.

Del agrupamiento jerárquico de los atributos se extraen algunos resultados interesantes. El agrupamiento mayor se realiza separando atributos meteorológicos de atributos de concentración de sustancias, salvo el ozono, que es agrupado junto con las temperaturas. Este tipo de información permite identificar causas comunes para la producción de determinadas sustancias o relaciones de causalidad entre ellas y descartar otras.

Volviendo a la cuestión acerca del número de agrupaciones óptimas, existen diferentes estadísticos que permiten realizar una aproximación al mismo.

7.1.2 Número óptimo de agrupaciones

Para situaciones en las que no se conoce *a priori* o no se quiere imponer el número de agrupamientos, es necesario aplicar diferentes técnicas que den una indicación de cuál es la mejor partición en grupos para el conjunto de datos.

La primera cuestión por resolver es si el conjunto de datos es susceptible de poder agruparse. Pueden utilizarse estadísticos y diferentes representaciones gráficas para tratar de dar respuesta a esa pregunta.

```
fviz_dist(distancia, show_labels = FALSE)
```

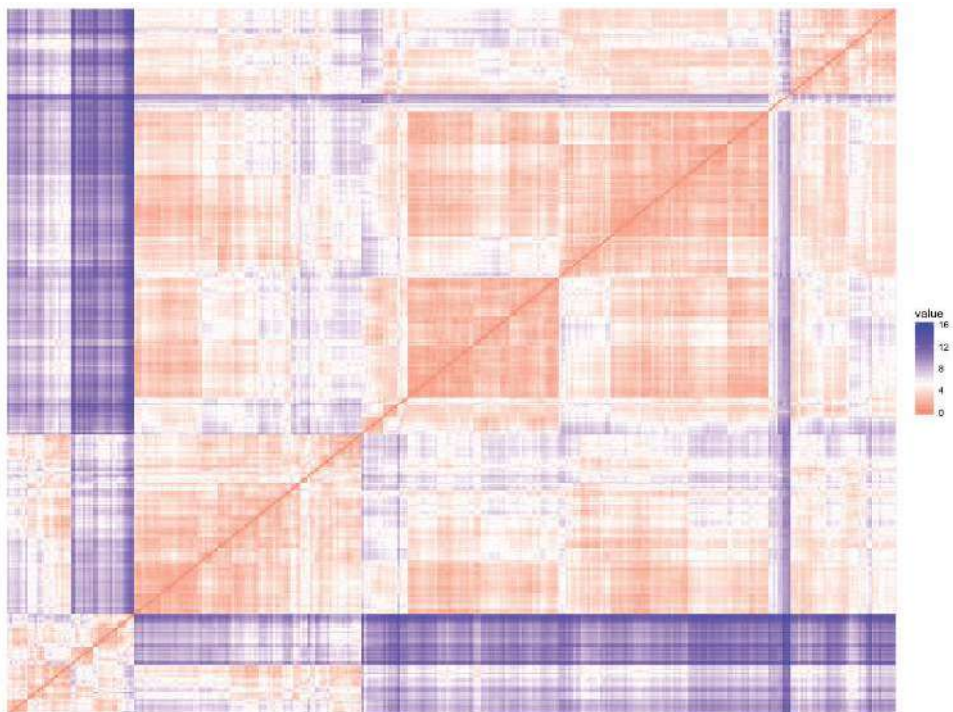


Fig. 7.4. Mapa de calor de la matriz de disimilitud.

Representando la matriz de disimilitudes en un mapa de calor, pueden apreciarse algunas estructuras (agrupaciones de colores), aunque no de una forma muy evidente, y es difícil establecer el número.

Se puede utilizar la representación de las instancias en los ejes de los dos primeros componentes principales.

```
fviz_pca_ind(prcomp(dfMedidas_CLUSscale), title = "Componentes Principales", geom = "point")
```

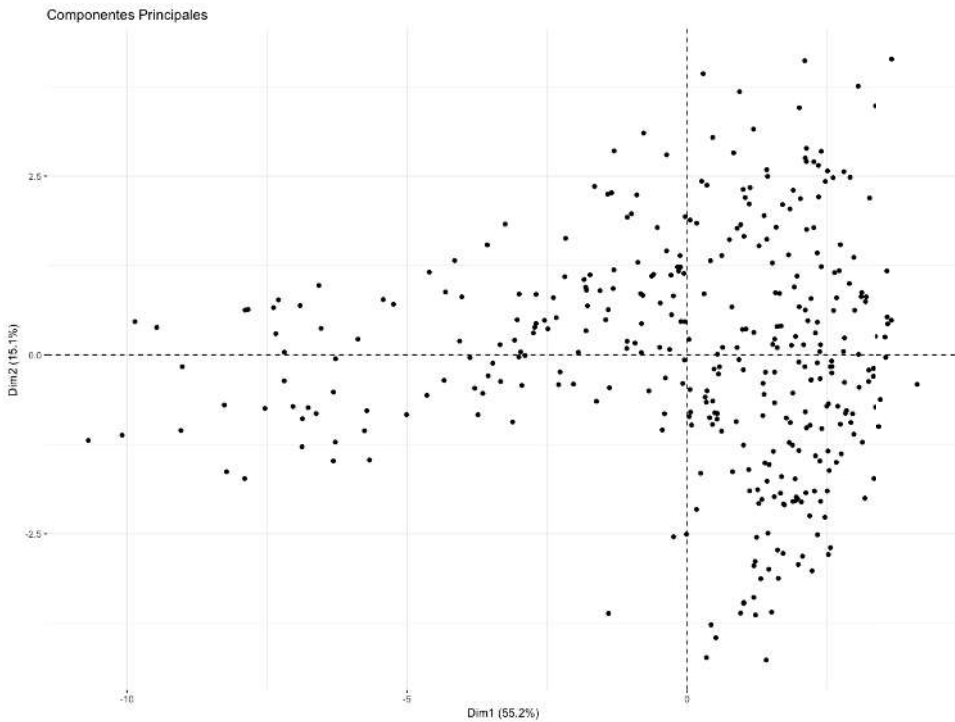


Fig. 7.5. Instancias proyectadas en los dos primeros componentes principales.

No se presentan agrupaciones claras, aunque también es evidente que no están distribuidas aleatoriamente por el plano.

El estadístico de Hopkins permite determinar si hay agrupaciones. Un valor próximo a 0.5 mostraría un comportamiento cercano al aleatorio, entre 0.01 y 0.3, los datos están distribuidos uniformemente y entre 0.7 y 0.99 mostrarían una tendencia clara a estar agrupados.

```
get_clust_tendency(dfMedidas_CLUSscale, n = nrow(dfMedidas_CLUSscale)-1, graph = FALSE)

$hopkins_stat
[1] 0.1529678
```

El resultado del test de Hopkins muestra que los datos se asemejan a los obtenidos mediante una distribución uniforme. Habría que hacer un test específico para demostrar que esta hipótesis es correcta. Otra posibilidad (la más realista) es que son insuficientes instancias considerando que se están utilizando diecisiete atributos y el test muestra que, para obtener agrupaciones consistentes, deberían o aumentarse el número de instancias o disminuir el de atributos (para este ejemplo, drásticamente en ambos casos).

Hay diferentes métodos para determinar el número óptimo de agrupaciones. Uno de los más simples y conocidos es el **método del codo**.

```
fviz_nbclust(dfMedidas_CLUSscale, kmeans, method = "wss") +
geom_vline(xintercept = 3, linetype = 2) +
labs(title = "Número óptimo de agrupaciones", subtitle = "Método \"Elbow\"")
```

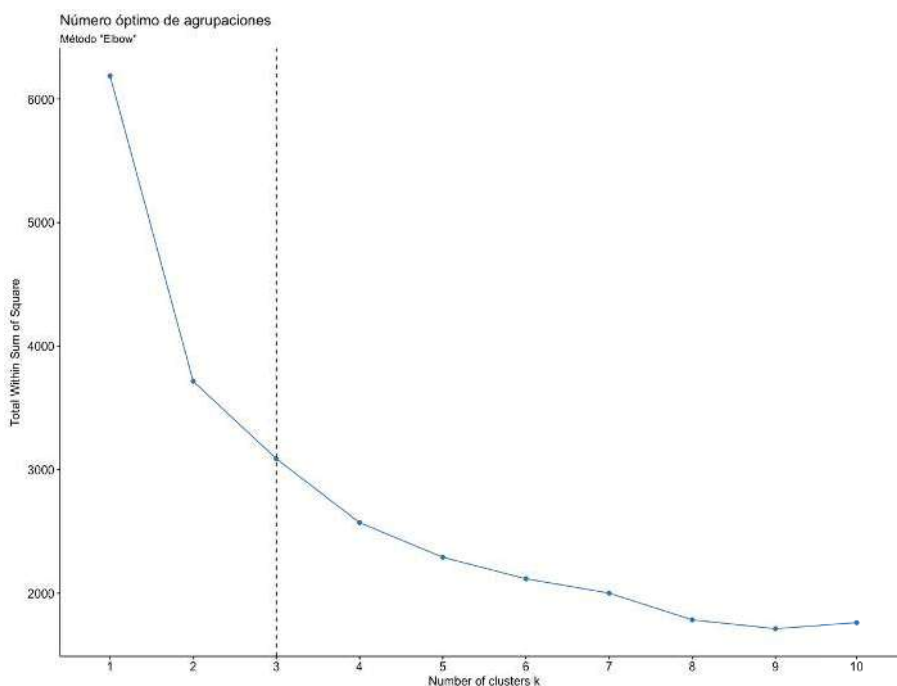


Fig. 7.6. Método del codo para determinar el número de agrupamientos óptimo.

El método del codo determina el momento en el que al aumentar el número de agrupaciones no supone una mejora sustancial en una medida de calidad. Las medidas que se toman para determinar la bondad de un agrupamiento tienen que ver con el nivel de cohesión de las instancias en los grupos y el de separación entre ellos. El nivel de cohesión se suele medir con el *Sum of Squares Within* (SSW), que es el que se muestra en la figura anterior. El punto codo es el lugar en el que hay un significativo cambio de tendencia. En el ejemplo se ha tomado el valor 3, pero bien podrían valer 2 y 4 como puntos codo. Este método tiene la ventaja de ser sencillo, pero el inconveniente de que no siempre se puede determinar el punto codo. Se puede usar en lugar de SSW el valor de separación interclúster con *Sum of Squares Between* (SSB) o con otros indicadores

que agregan la medida de cohesión y la de separación en un nuevo índice, como por ejemplo el índice de Hartigan o el de Calinski y Harabasz.

Otra medida es el llamado *método del coeficiente Silhouette*, que mide, con distancias promedios, la cohesión y separación integrándolas en un indicador. Se toma el valor máximo del coeficiente Silhouette frente al número de clústeres.

```
fviz_nbclust(dfMedidas_CLUSscale, kmeans, method = "silhouette") +
labs(title = "Número óptimo de agrupaciones", subtitle = "Método \"Silhouette\"")
```

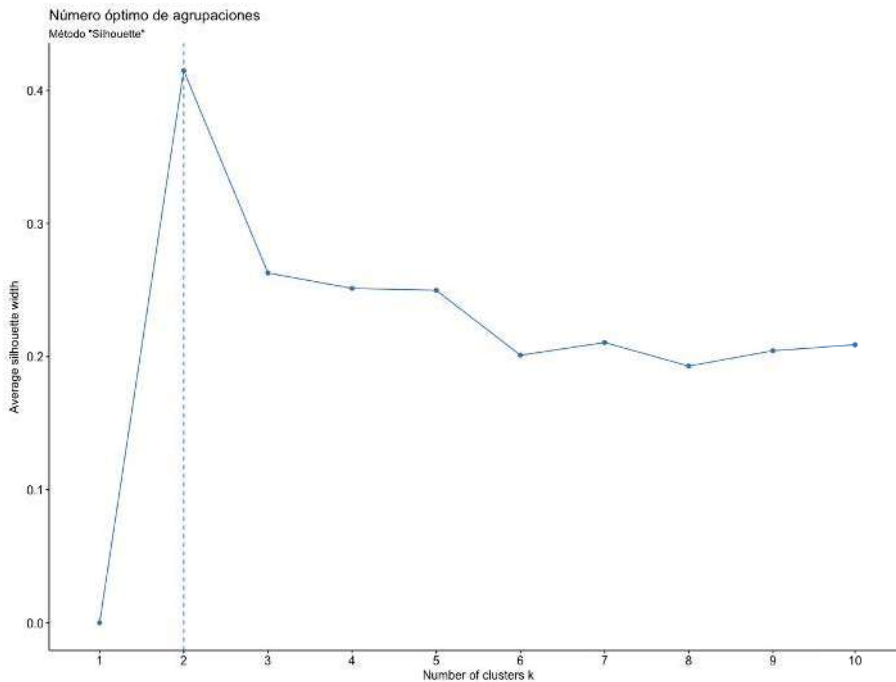


Fig. 7.7. Variación del coeficiente Silhouette.

Para este caso, el óptimo sería realizar dos agrupaciones.

Otra medida es el punto codo pero aplicado sobre el estadístico *gap*. El estadístico *gap* compara el valor de cohesión de una agrupación con el valor de cohesión de una distribución aleatoria de datos.

```
set.seed(123)
fviz_nbclust(dfMedidas_CLUSscale, kmeans, nstart = 2, method = "gap_stat", nboot = 500) +
labs(title = "Número óptimo de agrupaciones", subtitle = "Método del estadístico \"Gap\"")
```

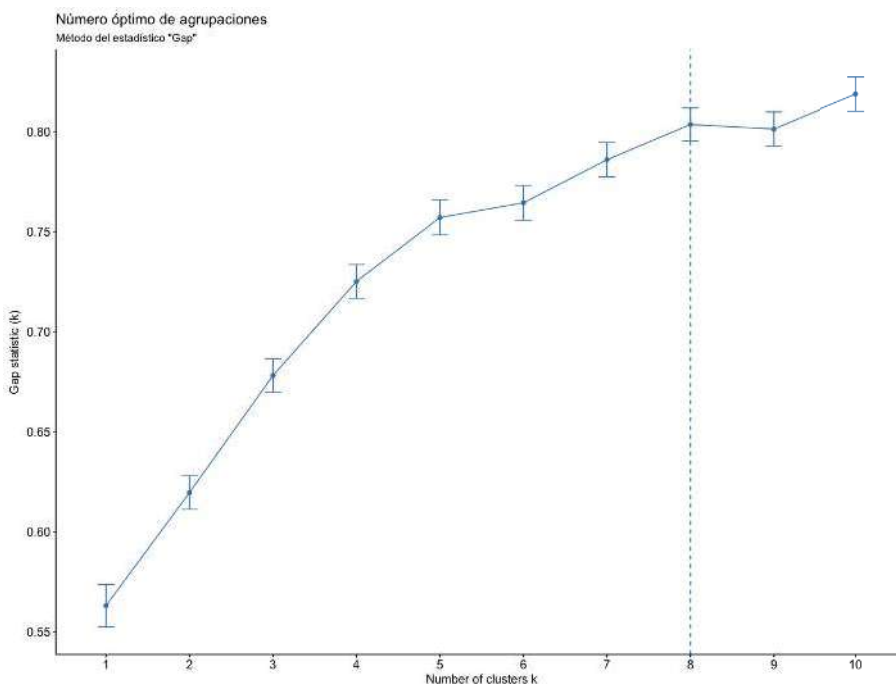


Fig. 7.8. Evolución del estadístico *gap* según el número de agrupamientos.

Como podemos observar, cada método arroja una cantidad diferente para el valor óptimo de agrupamientos. Puede ser una buena estrategia tomar la decisión por votación mayoritaria; en este caso, no se resolvería el problema dado que los tres han dado resultados diferentes, aunque podría descartarse el 8 por estar bastante alejado de los otros dos valores obtenidos. En cualquier caso, la librería *NbClust* incorpora treinta medidas diferentes para determinar el número óptimo de agrupaciones; con todas ellas, está bastante asegurado el desempeño.

```
library(NbClust)
set.seed(123)
fviz_nbclust(NbClust(dfMedidas_CLUSscale, distance = "euclidean", min.nc = 2, max.nc
= 10, method = "kmeans", index = "all"))
```

```

*****
Among all indices:
=====
* 2 proposed 0 as the best number of clusters
* 12 proposed 2 as the best number of clusters
* 4 proposed 3 as the best number of clusters
* 4 proposed 4 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 10 as the best number of clusters
Conclusion
=====
* According to the majority rule, the best number of clusters is 2.

```

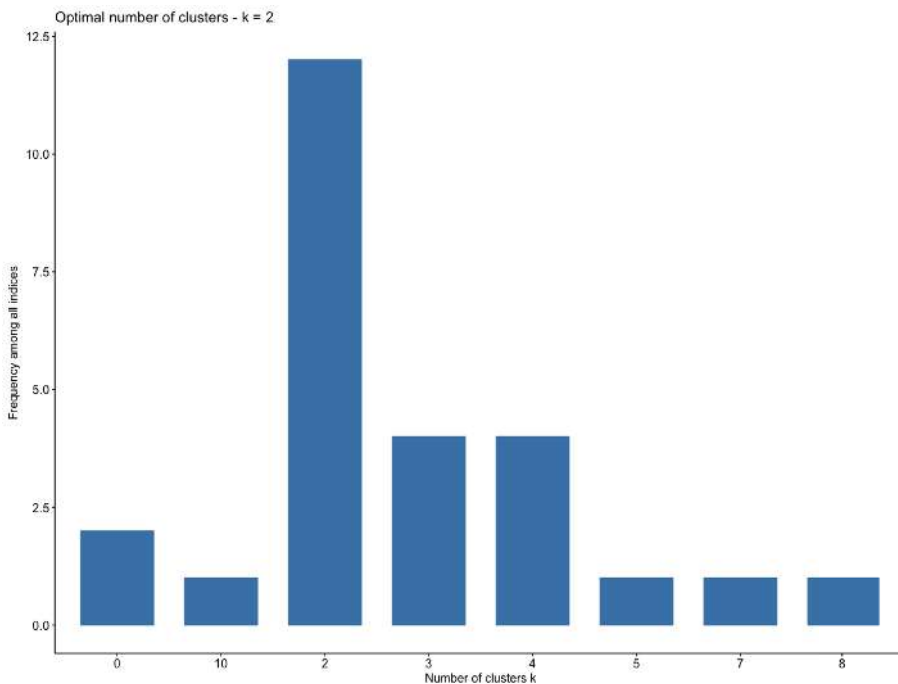


Fig. 7.9. Resultado de aplicar diferentes medidas que estiman el número óptimo de agrupamientos con la librería *NbClust*.

El resultado ha sido que la mayoría determina que dos grupos es la partición óptima, el resto, a bastante distancia, estiman que tres o cuatro serán los adecuados.

Hay un detalle que se debe considerar. Observando todas las llamadas a las funciones que tratan de estimar el número óptimo de agrupaciones, incluyen un parámetro, para todos ha sido *k-means*. Éste es el algoritmo de agrupamiento aplicado sobre el que posteriormente se calculan los índices de desempeño. Este algoritmo puede ser cambiado por otro, de hecho, *k-means*, que será tratado a continuación, es bastante sensible a datos atípicos. Conviene aplicar diferentes algoritmos de agrupamiento (dependiendo del problema de distancia) y tomar la opción del número de agrupaciones más frecuente. Para el ejemplo que se está tratando sería dos.

Aplicando el agrupamiento jerárquico para tener dos grupos, el resultado, es decir, etiquetar cada instancia con el grupo al que pertenece, se podría obtener con la función siguiente:

```
res_HC <- eclust(dfMedidas_CLUSscale, "hclust", k = 2, graph = TRUE)
fviz_dend(res_HC, palette = "jco", rect = TRUE, show_labels = FALSE)
```

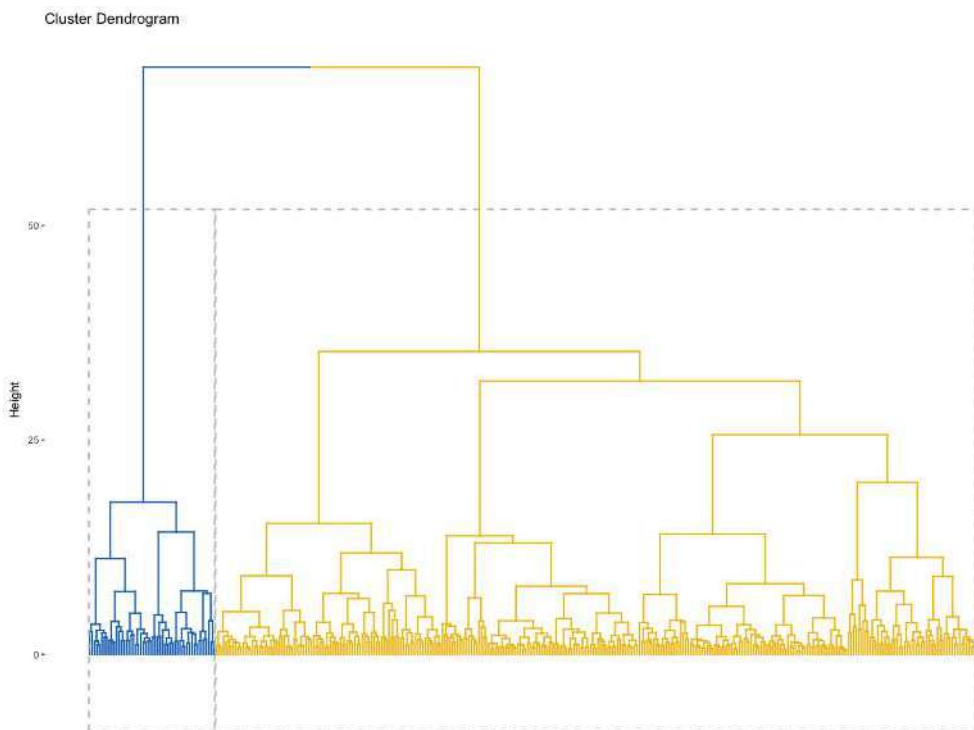


Fig. 7.10. Agrupación en dos clases aplicando agrupamiento jerárquico.

Se puede representar en los ejes de los dos primeros componentes principales la asociación de los atributos a las clases obtenidas.

```
fviz_cluster(res_HC, data = dfMedidas_CLUSscale, ellipse.type = "convex",
             palette = "jco", labelsize = 8)
```

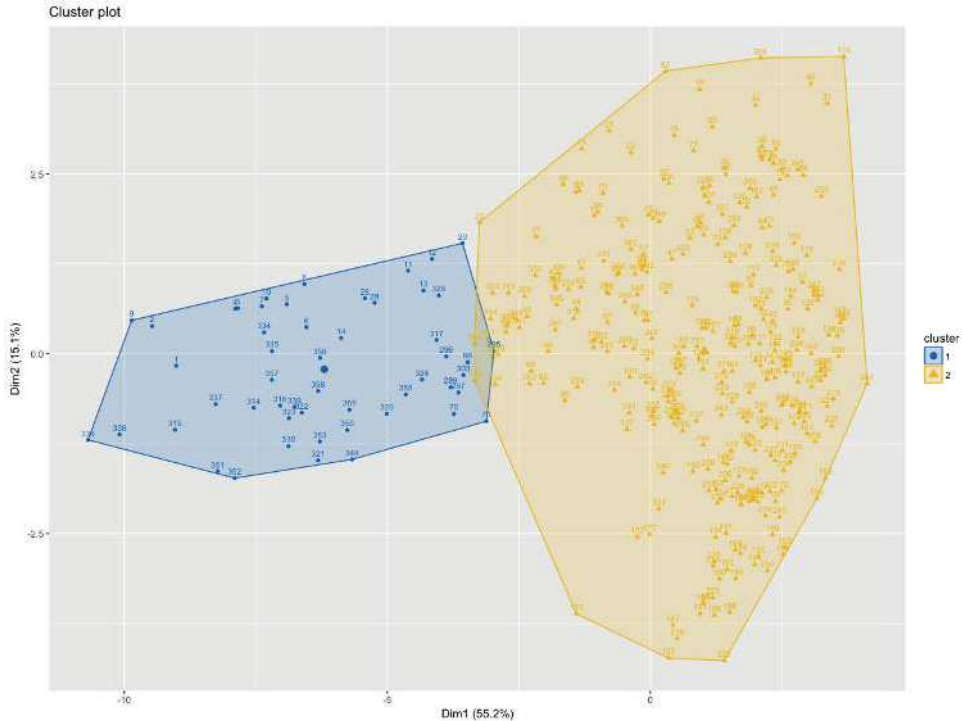


Fig. 7.11. Representación de los grupos obtenidos con agrupamiento jerárquico en PCA.

Se aprecia en la figura anterior que las clases se solapan en un conjunto pequeño de instancias. Si se calcula para las instancias que se solapan, ¿la distancia a los centroides a la clase a la que pertenecen es menor que para la otra clase? Es decir, ¿están correctamente asignadas estas instancias?

Una forma de responder a la cuestión anterior es utilizar el gráfico de Silhouette.

```
fviz_silhouette(res_HC, palette = "jco")
```

cluster	size	ave.sil.width
1	52	0.45
2	313	0.46

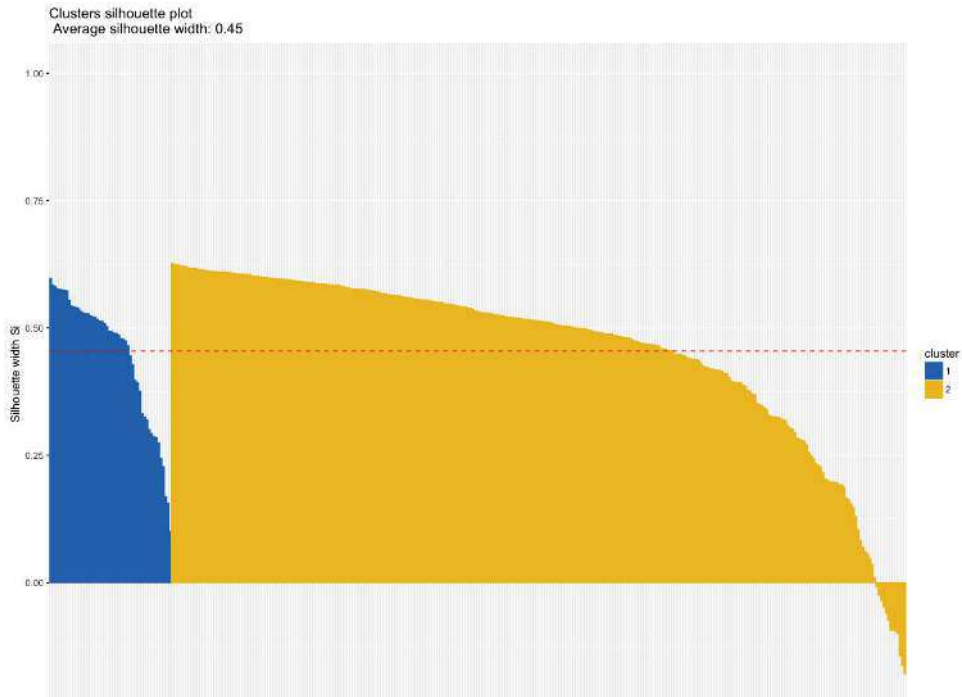


Fig. 7.12. Gráfico de Silhouette para la asignación de instancias con agrupamiento jerárquico.

Efectivamente, las instancias con valores negativos tienen una distancia menor al centroide de la clase 1 que al que han sido asignadas.

Se va a reasignar la clase a esas instancias con valor de Silhouette negativo y se representan de nuevo las agrupaciones.

```
res_HC_sil <- res_HC$silinfo$widths
# Instancias con valor negativo en silhouette
neg_sil_index <- which(res_HC_sil[, "sil_width"] < 0)
n <- res_HC_sil[neg_sil_index, , drop = FALSE]
res_HC_corregido <- res_HC
res_HC_corregido$cluster[as.integer(row.names(n))] <- 1
fviz_cluster(res_HC_corregido, data = dfMedidas_CLUSscale, ellipse.type =
"convex", palette = "jco", labels = 8)
```

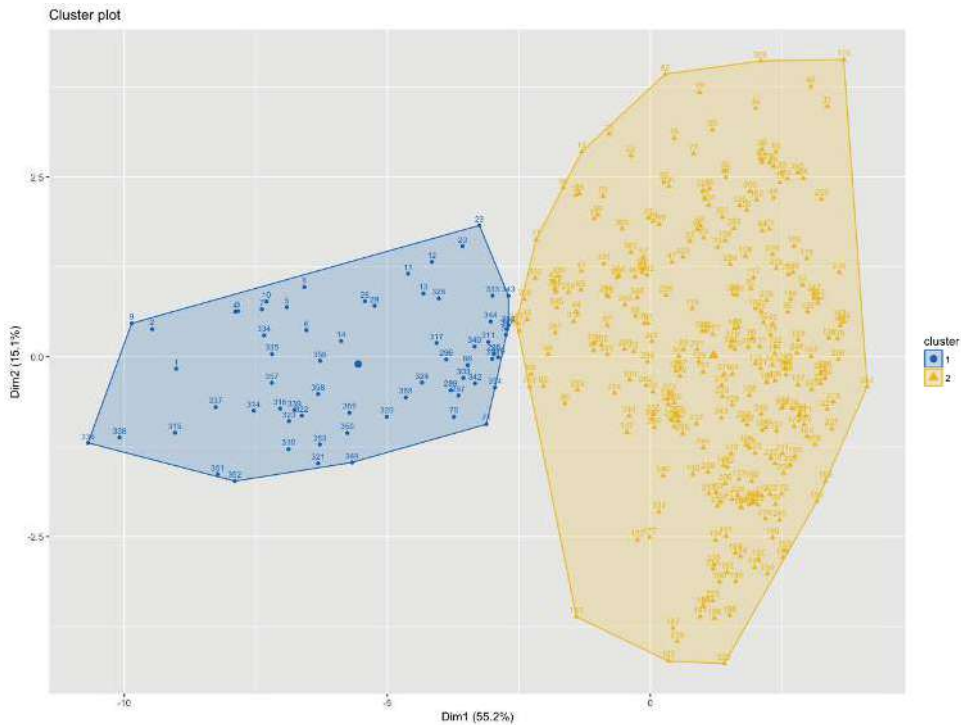


Fig. 7.13. Resultado de reasignar las instancias con valor de Silhouette negativo.

Efectivamente, ahora no hay solapamiento entre las clases.

A continuación se va a aplicar otro tipo de método de agrupamiento, es muy posible que el resultado difiera del obtenido.

7.1.3 Agrupamiento por particionamiento

Los métodos por particionamiento requieren de establecer *a priori* el número de agrupamientos en los que se va a dividir el conjunto de datos. Cada grupo es representado por un prototipo y la asignación de los datos se realiza asignando cada uno de ellos al prototipo más próximo. Las diferentes técnicas de particionamiento surgen de la forma en la que cada una de ellas establece el prototipo de la clase y la forma de medir la distancia de los prototipos a las instancias de datos.

El más conocido de los algoritmos de particionamiento es *k-means*. Los prototipos de las clases en *k-means* son los centroides que van modificándose a medida que van agregando las instancias más proximas a ellos. Los centroides se asignan aleatoriamente la primera vez, por tanto, se recomienda generar bastantes valores para mejorar la robustez de los resultados.

```
set.seed(123)
res_KM <- kmeans(dfMedidas_CLUSscale, 2, nstart = 25)
print(res_KM$centers)
```

	Lluvia	Lluvia_SN	T_MAX	T_MIN	Viento_MAX	Viento_MED	SO2	CO
1	0.06129115	0.09956742	0.1941037	0.2246980	0.2973405	0.2816717	-0.4009003	-0.4283317
2	-0.23699243	-0.38499403	-0.7505344	-0.8688322	-1.1497167	-1.0891306	1.5501477	1.6562157
	NO	NO2	PM2.5	PM10	O3	TOL	BEN	EBE
1	-0.4313999	-0.3985483	-0.3398866	-0.1857292	0.3596324	-0.3908253	-0.4193721	-0.3849633
2	1.6680797	1.5410535	1.3142280	0.7181528	-1.3905786	1.5111913	1.6215721	1.4885246
	TCH							
1	-0.1934967							
2	0.7481871							

Los datos que se muestran en las filas son el punto donde se encuentra el centroide para la clase. Los valores de los centroides están en los valores de los datos escalados, por ese motivo son difíciles de interpretar. Se pone de manifiesto en una primera impresión que, para todos ellos, las agrupaciones tienen valores muy diferentes.

Para realizar la representación de los centroides sobre los datos sin escalar se puede aplicar la siguiente sentencia.

```
aggregate(dfMedidas_CLUS, by=list(cluster=res_KM$cluster), mean)
summary(dfMedidas_CLUS)
```

cluster	Lluvia	Lluvia_SN	T_MAX	T_MIN	Viento_MAX	Viento_MED	SO2	CO	
1	1	0.921724138	0.2310345	23.88207	13.204483	34.97586	15.593103	5.895364	0.2952960
2	2	0.005333333	0.0400000	15.36667	5.698667	18.12000	8.906667	10.576148	0.6444741
	NO	NO2	PM2.5	PM10	O3	TOL	BEN	EBE	TCH
1	11.90216	33.20553	9.456839	19.1072	59.75720	2.265402	0.5563448	0.3366782	1.487678
2	84.02447	70.98184	19.838222	28.7800	15.17934	6.560622	1.5354222	1.1134889	1.603889
	Lluvia	Lluvia_SN	T_MAX	T_MIN	Viento_MAX				
Min.	: 0.0000	Min. :0.0000	Min. : 3.50	Min. : -1.50	Min. :10.00				
1st Qu.:	0.0000	1st Qu.:0.0000	1st Qu.:14.50	1st Qu.: 6.40	1st Qu.:23.00				
Median :	0.0000	Median :0.0000	Median :21.40	Median :11.30	Median :31.00				
Mean :	0.7334	Mean :0.1918	Mean :22.13	Mean :11.66	Mean :31.51				
3rd Qu.:	0.0000	3rd Qu.:0.0000	3rd Qu.:29.50	3rd Qu.:17.10	3rd Qu.:39.00				
Max.	:26.6000	Max. :1.0000	Max. :39.90	Max. :26.00	Max. :75.00				
Viento_MED	SO2	CO	NO	NO2					
Min.	: 4.00	Min. : 3.778	Min. :0.180	Min. : 2.333	Min. : 9.583				

1st Qu.:11.00	1st Qu.: 5.200	1st Qu.:0.260	1st Qu.: 6.292	1st Qu.: 25.583
Median :14.00	Median : 6.000	Median :0.310	Median : 11.125	Median : 37.083
Mean :14.22	Mean : 6.857	Mean :0.367	Mean : 26.722	Mean : 40.968
3rd Qu.:17.00	3rd Qu.: 7.800	3rd Qu.:0.420	3rd Qu.: 30.333	3rd Qu.: 51.261
Max. :28.00	Max. :15.600	Max. :0.990	Max. :168.000	Max. :104.875
PM2.5	PM10	O3	TOL	BEN
Min. : 2.80	Min. : 5.727	Min. : 7.429	Min. : 0.4833	Min. :0.1600
1st Qu.: 7.00	1st Qu.:13.500	1st Qu.: 28.643	1st Qu.: 1.6667	1st Qu.:0.4333
Median :10.40	Median :19.273	Median : 53.538	Median : 2.4800	Median :0.6000
Mean :11.59	Mean :21.095	Mean : 50.597	Mean : 3.1480	Mean :0.7575
3rd Qu.:14.20	3rd Qu.:25.750	3rd Qu.: 70.929	3rd Qu.: 3.9167	3rd Qu.:0.9333
Max. :36.17	Max. :69.000	Max. :108.357	Max. :14.1667	Max. :2.6333
EBE	TCH			
Min. :0.1000	Min. :1.240			
1st Qu.:0.2333	1st Qu.:1.440			
Median :0.3500	Median :1.487			
Mean :0.4963	Mean :1.512			
3rd Qu.:0.6167	3rd Qu.:1.563			
Max. :2.4500	Max. :2.077			

El agrupamiento 1 se corresponde con días más calurosos, la temperatura máxima está en el tercer cuartil y la mínima en el segundo, mientras que para el agrupamiento 2, la temperatura máxima está en el segundo cuartil y la mínima en el primero, es decir, días más fríos. Las concentraciones de gases son para el agrupamiento 1 inferiores que para el agrupamiento 2, suelen estar en el primer o segundo cuartil, mientras que para el agrupamiento 2 suelen estar en el cuarto cuartil, salvo en el caso del ozono que se invierte. Por tanto, el agrupamiento 1 es el de días más calurosos en los que no hay riesgo de alcanzar altas concentraciones de gases, el único peligro es la concentración de ozono, todo lo contrario al agrupamiento 2, sin riesgo para el ozono, pero en peligro para el resto de gases, y esto ocurre en los días fríos.

Éste es el tipo de análisis posterior que hay que realizar siempre para interpretar las agrupaciones obtenidas. Hay que dotarlas de la semántica que permita explicar relaciones entre los atributos y las clases obtenidas. Se volverá a esta discusión posteriormente.

Se representa en la siguiente figura las agrupaciones obtenidas con *k-means*.

```
fviz_cluster(res_KM, data = dfMedidas_CLUSscale, ellipse.type = "convex",
             palette = "jco", labelsize = 8)
```

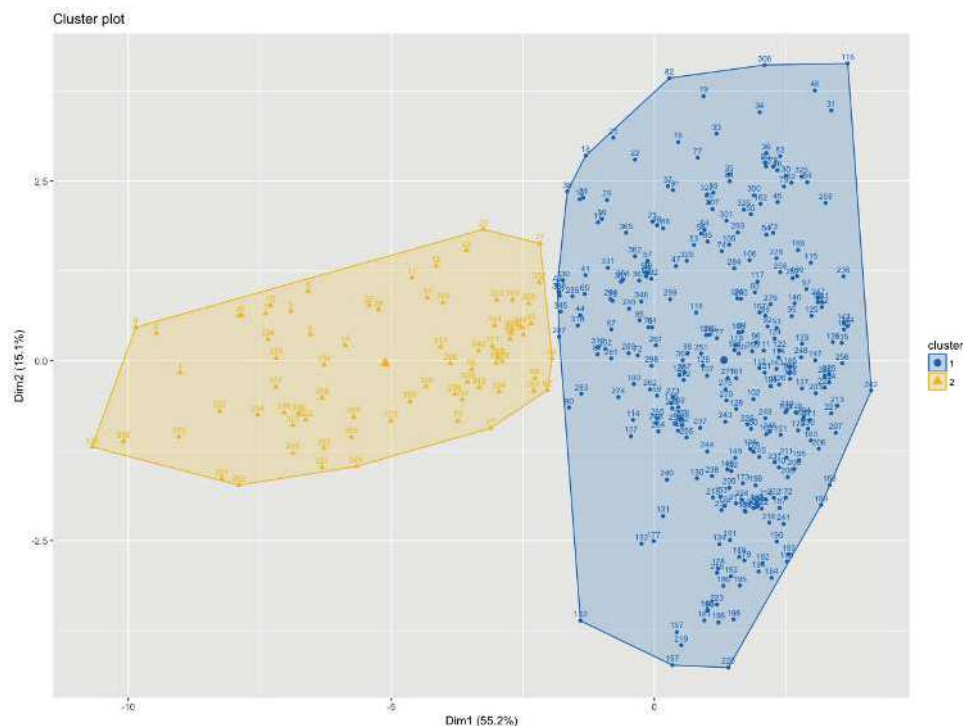


Fig. 7.14. Agrupaciones obtenidas con *k-means*.

Comparando con las agrupaciones que se obtuvieron con el agrupamiento jerárquico, se aprecia que las clases no están solapadas y algunas de las instancias que fueron clasificadas en un grupo el algoritmo *k-means* las ha clasificado en otro.

Otro algoritmo de particionamiento que mejora su robustez frente a datos atípicos es PAM (*Partitioning Around Medoids*) o *k-medoids*. Las principales diferencias con *k-means* es que el prototipo de la clase no es el centroide, es una instancia del conjunto de datos y la métrica de distancia no es la euclídea.

Se vuelve a calcular el número óptimo de agrupaciones usando la métrica Silhouette y con el método de agrupamiento PAM.

```
library(cluster)
set.seed(123)
fviz_nbclust(dfMedidas_CLUSscale, pam, method = "silhouette")
```

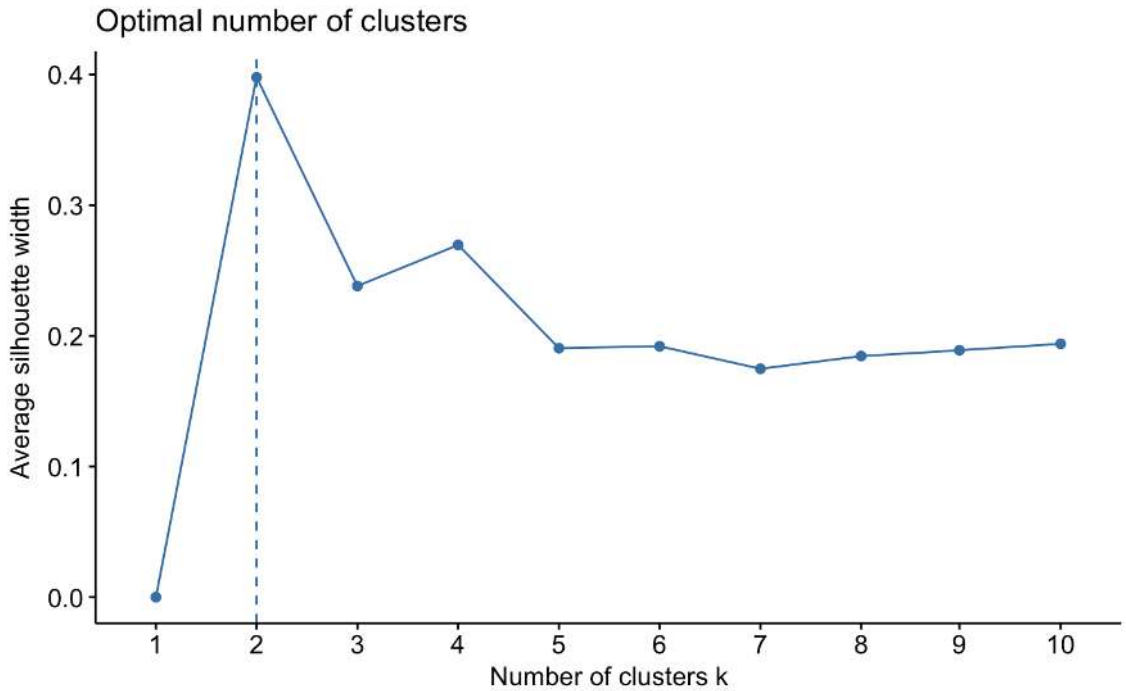


Fig. 7.15. Número óptimo de agrupamientos usando PAM.

El resultado muestra que el número óptimo de particiones es dos. Viene a reafirmar el resultado obtenido anteriormente.

```
res_PAM <- pam(dfMedidas_CLUSscale, 2)
print(res_PAM$medoids)
dfMedidas_CLUS[297,]
dfMedidas_CLUS[275,]
```

	Lluvia	Lluvia_SN	T_MAX	T_MIN	Viento_MAX	Viento_MED	SO2	CO		
297	-0.2387284	-0.4864546	-0.2254521	0.04921563	-1.2458674	-1.2750084	0.8931764			
1.0325192										
275	-0.2387284	-0.4864546	0.3070257	0.44258143	0.2994127	0.5701029	-0.5240141			
-0.5793453										
	NO	NO2	PM2.5	PM10	O3	TOL	BEN			
EBE										
297	0.9484773	0.7941437	0.7557776	0.3415665	-1.2854431	1.1743713	1.2350486			
1.5162867										
275	-0.4358293	-0.3192484	-0.4126769	-0.1412365	0.3495343	-0.2500386	-0.3978556			
-0.3528341										
	TCH									
297	0.63564202									
275	0.06841505									
	Lluvia	Lluvia_SN	T_MAX	T_MIN	Viento_MAX	Viento_MED	SO2	CO	NO	NO2
PM2.5										
297	0	0	20.1	12	17	8	9	0.54	59.30435	
56.43478	16.33333									
	PM10	O3	TOL	BEN	EBE	TCH				
297	24.75	17.85714	5.8	1.35	1.125	1.59				
	Lluvia	Lluvia_SN	T_MAX	T_MIN	Viento_MAX	Viento_MED	SO2	CO	NO	NO2
PM10										PM2.5
275	0	0	24.9	14.7	35	17	5.6	0.27	11.75	34.75
9	19.58333									
	O3	TOL	BEN	EBE	TCH					
275	59.5	2.583333	0.5666667	0.35	1.52					

Se muestran los prototipos de las clases, son los que tienen índices 297 y 275. Ahora habría que interpretar, como se hizo para *k-means*, qué instancias agrupan cada clase. Omitimos la discusión y la dejamos propuesta al lector.

Representamos las clases en PCA:

```
fviz_cluster(res_PAM, data = dfMedidas_CLUSscale, ellipse.type = "convex",
             palette = "jco", labelsize = 8)
```

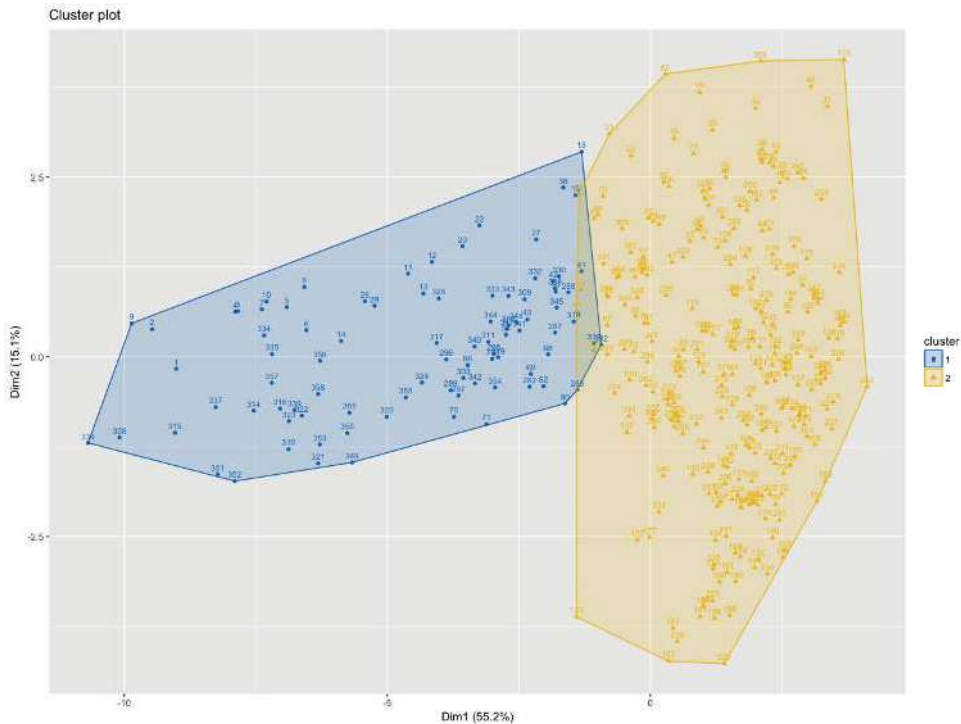


Fig. 7.16. Clases obtenidas con PAM proyectadas en las coordenadas de PCA.

Las clases son distintas a las obtenidas con los anteriores algoritmos, aunque sólo ligeramente. Se puede apreciar, en los datos frontera que se encuentran en la zona en la que las dos clases se solapan, que dependiendo del algoritmo, éstos se clasifican en una u otra clase.

Que se produzca solapamiento entre clases en la representación no es indicación de una mala clasificación, hay que considerar que los datos de doce dimensiones se están proyectando en dos, en dimensiones superiores es muy posible que las clases no se intersequen.

Hay una modificación muy adecuada de PAM, llamada CLARA (*Clustering Large Applications*), para conjuntos muy grandes de datos. No es de aplicación para el problema que se está tratando, dado que el conjunto de datos es muy, muy pequeño.

7.1.4 Agrupamiento basado en modelos

En el agrupamiento basado en modelos se asume que cada clase puede ser representada mediante una función de distribución gaussiana. De forma que una clase estará definida por un vector de medias, la matriz de covarianzas y la probabilidad de que

una instancia pertenezca a esa clase. La estimación de los parámetros se realiza usando el algoritmo de esperanza-maximización (EM, *Expectation-Maximization*). Una de las ventajas de este tipo de métodos de agrupamiento es que no necesita fijar *a priori* el número de clases, éstas son también un resultado del agrupamiento basado en modelos.

Se van a utilizar los algoritmos EM y DBSCAN, dos de los más conocidos métodos de agrupamiento basados en modelos.

```
library(mclust)

res_EM <- Mclust(dfMedidas_CLUSscale) #Agrupamiento basado en modelos, EM

summary(res_EM)
```

```
fitting ...
|=====
| 100%

-----

Gaussian finite mixture model fitted by EM algorithm
-----

Mclust VEV (ellipsoidal, equal shape) model with 3 components:

log.likelihood   n   df          BIC          ICL
-553.4406 365 480 -3938.832 -3940.515

Clustering table:
  1   2   3
121  75 169
```

El resultado muestra que el número óptimo de agrupamiento para EM es de tres. Se han utilizado distribuciones gaussianas elipsoidales de igual forma (está etiquetado en el resultado como VEV).

El método que usa *mclust* realiza una búsqueda en rejilla para las diferentes posibilidades con el fin de formar las distribuciones gaussianas y el número de agrupaciones

óptimo. Se ha tomado aquella combinación que maximiza BIC (*Bayesian Information Criterion*, “criterio de información bayesiano”).

```
fviz_mclust(res_EM, "BIC", palette = "jco")
```

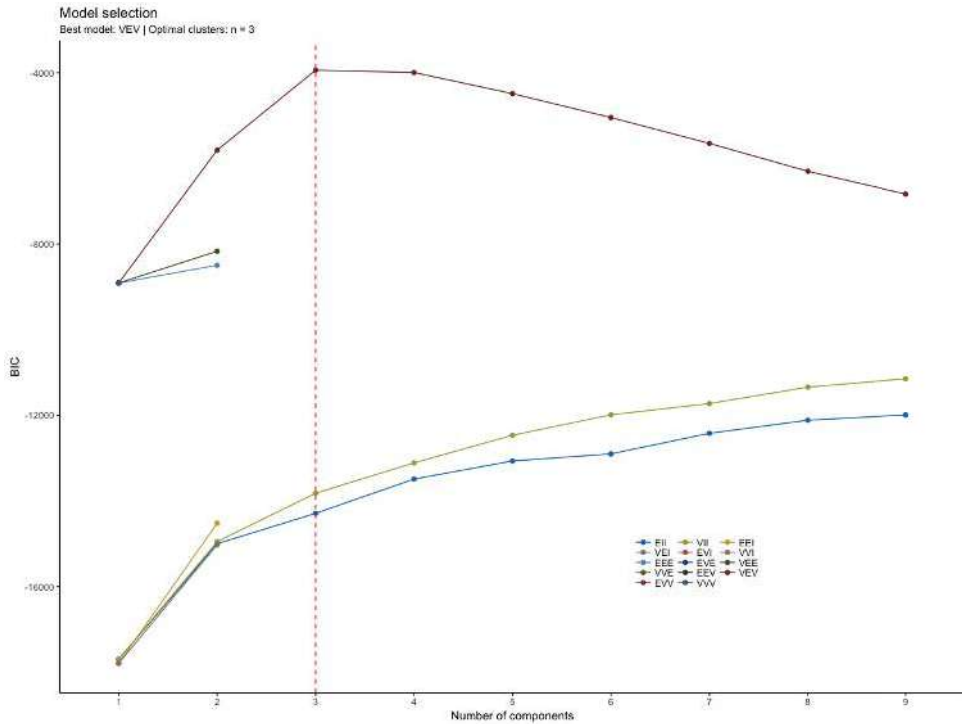


Fig. 7.17. Evolución de BIC para diferente número de agrupaciones y modelos de las distribuciones gaussianas que modelan las clases.

Representando el resultado del ajuste se aprecian las distribuciones gaussianas para cada una de las clases.

```
fviz_mclust(res_EM, "classification", geom = "point", pointsize = 1.5, palette = "jco")
```

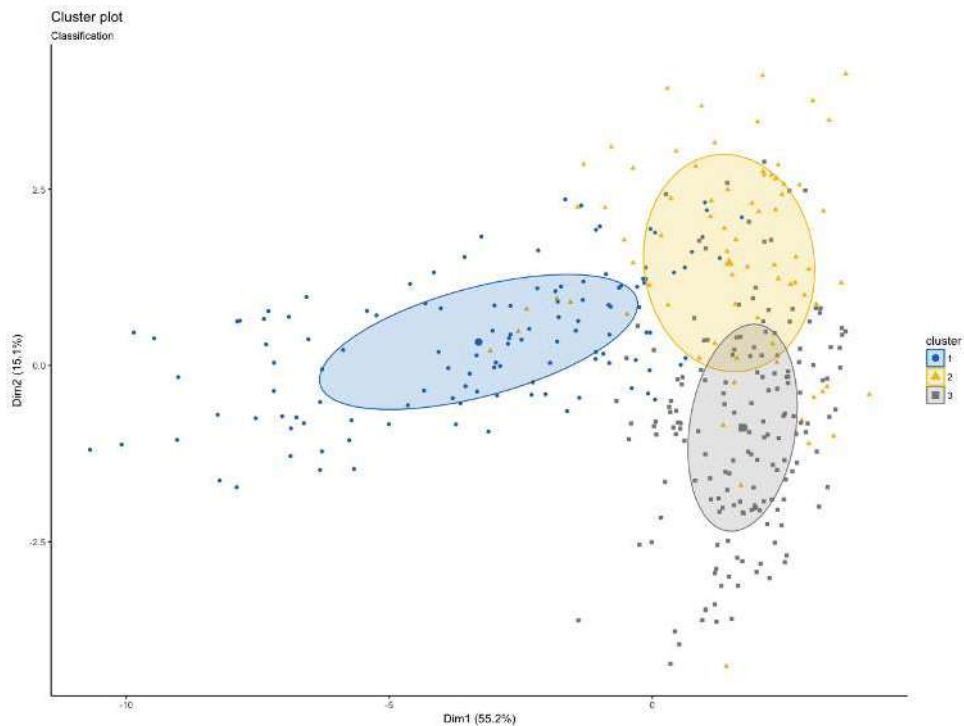


Fig. 7.18. Representación de las distribuciones para las clases obtenidas resultado de la aplicación del método EM para agrupamiento.

Se mantiene aproximadamente el primer agrupamiento que se ha obtenido previamente con otras técnicas y el segundo ha sido desdoblado en dos agrupamientos.

En la figura siguiente se ha representado la asignación de las instancias a cada clase. Esta asignación se resuelve asociando la instancia a la clase con mayor probabilidad.

```
fviz_cluster(res_EM, data = dfMedidas_CLUSscale, ellipse.type = "convex",
              palette = "jco", labelsize = 8)
```

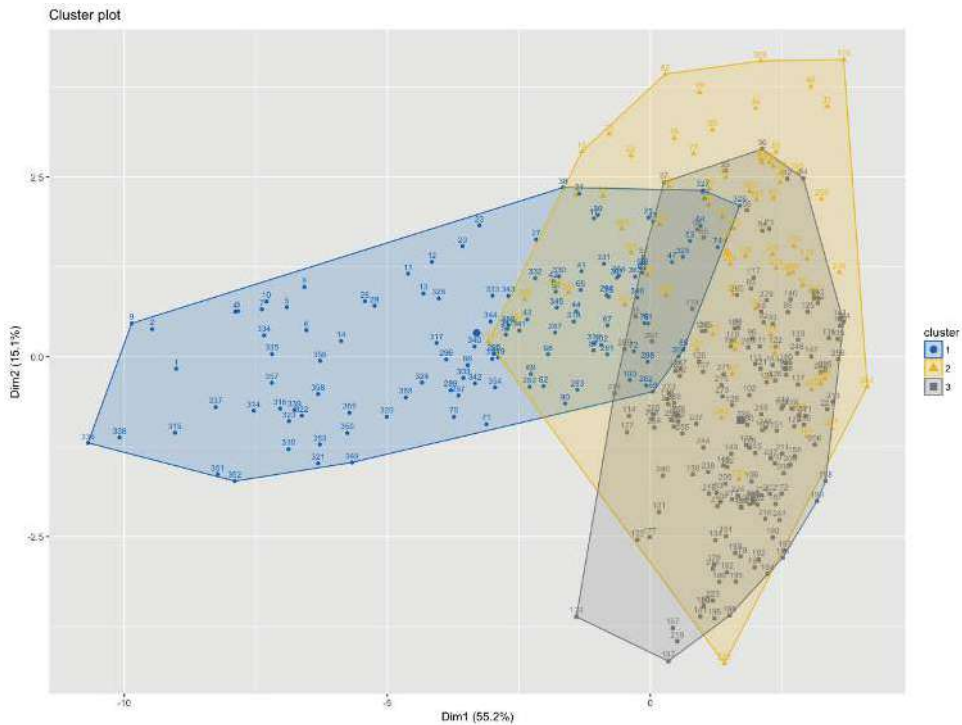


Fig. 7.19. Asignación de cada instancia de datos a las clases obtenidas con EM.

Se presenta para la solución obtenida con EM una cuestión. Para algunas instancias la probabilidad de pertenencia a una clase concreta podrá ser alta y muy baja para el resto, y para otras ese valor de probabilidad puede ser muy parecido para varias clases. Se puede representar el valor de incertidumbre asociado al hecho de que una instancia tiene una probabilidad de pertenencia alta a varias clases. En la figura siguiente se muestra esa incertidumbre.

```
fviz_mclust(res_EM, "uncertainty", palette = "jco")
```

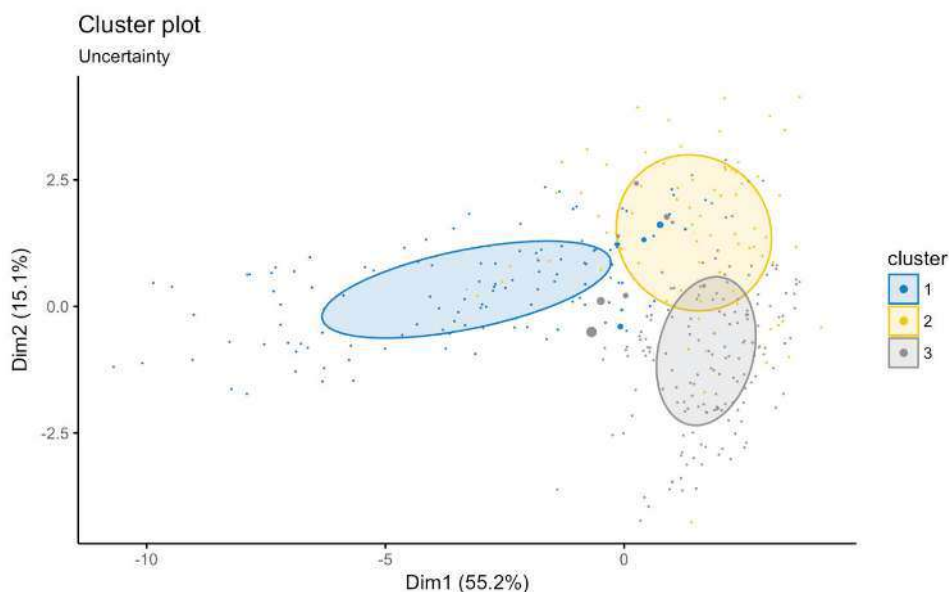


Fig. 7.20. Representación de la incertidumbre asociada a cada instancia en una agrupación EM.

El tamaño del punto en la figura anterior representa la incertidumbre. Estos puntos de alta incertidumbre sí representan solapamientos entre clases.

Todas las técnicas de agrupamiento que se han aplicado hasta este punto basadas en agrupamiento jerárquico, particionamiento o basadas en modelos asumen que las agrupaciones que determinan las clases son convexas, compactas y razonablemente separadas unas de otras. En definitiva, pueden ser representadas mediante esferas o elipsoides que no se intersecan. Con DBSCAN (*Density-Based Spatial Clustering and Application with Noise*) no se asume previamente ninguna forma para las agrupaciones.

El concepto que subyace en DBSCAN es el de encontrar regiones densas de instancias. Para definir la densidad, se necesitan dos parámetros, uno para indicar el número de vecinos (*MinPts*) y otro para el radio de vecindario (*eps*). El radio de vecindario debe ajustarse de forma adecuada y el número de agrupaciones que se obtengan va a depender fuertemente de este parámetro.

Una forma de estimar el mejor valor para *eps* es encontrar el punto codo representando la distancia media de un punto a sus *k*-vecinos más próximos.

```
library(fpc)
library(dbSCAN)
dbSCAN::kNNdistplot(dfMedidas_CLUSscale, k = 5)
abline(h = 2.6, lty = 2)
```

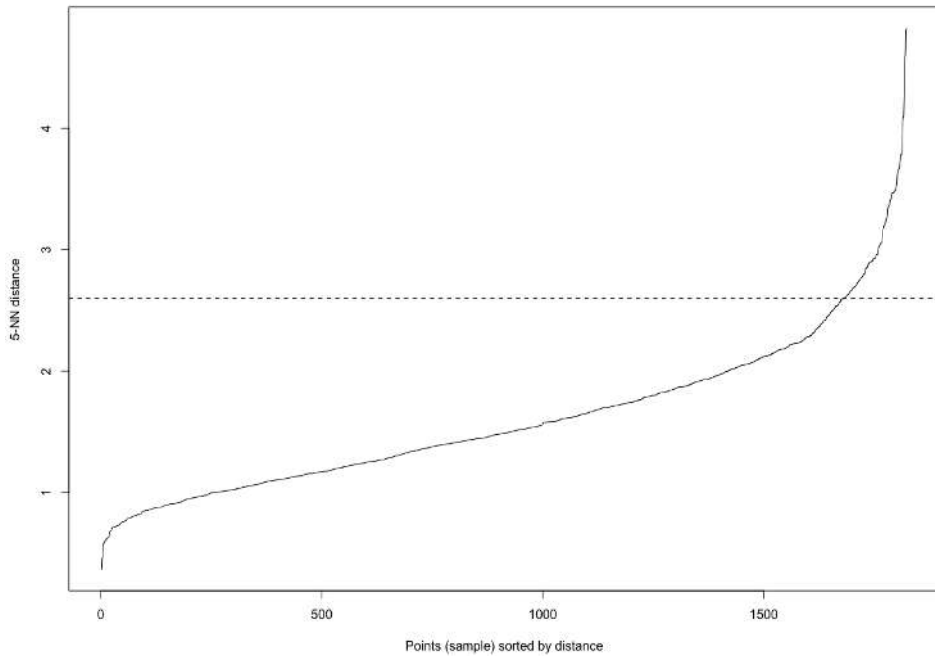


Fig. 7.21. Estimación del valor óptimo de *eps* para DBSCAN.

El valor del punto codo se ha situado en 2,6.

```
res_DBSCAN <- fpc::dbSCAN(dfMedidas_CLUSscale, eps = 2.6, MinPts = 5)
print(res_DBSCAN)
fviz_cluster(res_DBSCAN, data = dfMedidas_CLUSscale, ellipse = TRUE, geom =
"point", palette = "jco")
```

```
dbSCAN Pts=365 MinPts=5 eps=2.6
      0  1  2
border 18 16  5
seed    0 276 50
total  18 292 55
```

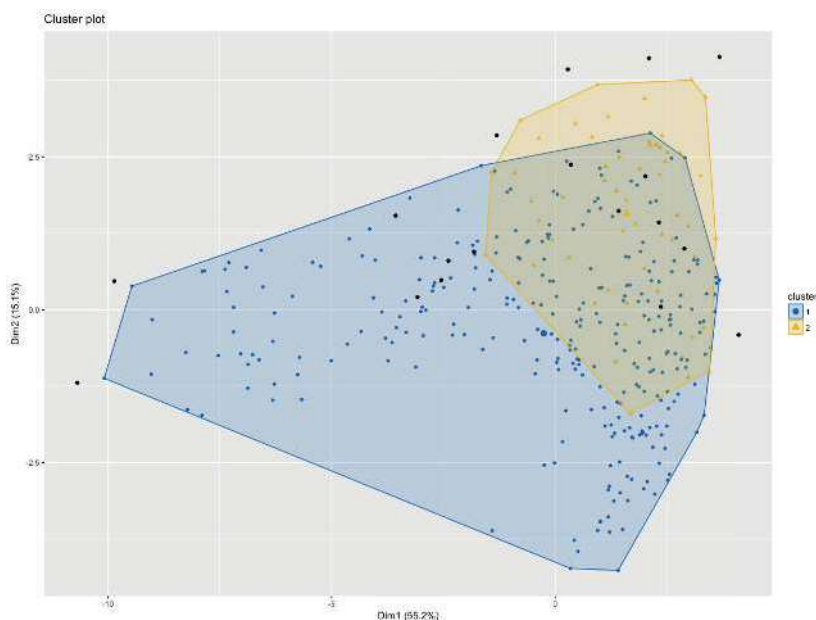


Fig. 7.22. Resultado de agrupamientos obtenidos con DBSCAN.

Los puntos en negro son considerados valores atípicos.

Una extensión de DBSCAN es OPTICS (*Ordering Points To Identify the Clustering Structure*), tiene la ventaja de que no requiere definir el radio de vecindad.

7.1.5 Agrupamiento borroso (*fuzzy*)

A diferencia del agrupamiento por particiones o jerárquico, donde la pertenencia a una clase es excluyente, una instancia pertenece sólo a una clase. El agrupamiento borroso es similar al agrupamiento basado en modelos en el sentido de que una instancia tiene un valor de pertenencia a las clases distinto de cero.

Fuzzy C-Means (FCM) es uno de los principales algoritmos de agrupamiento borroso. La diferencia con *k-means* consiste en incorporar un término con el valor de pertenencia a cada agrupamiento a la función objetivo.

Se va a utilizar la implementación de la función FCM de la librería *ppclust*.

```
library(ppclust)
res_FCM <- fcm(dfMedidas_CLUSscale, centers = 2, nstart=5) #Determinados 2 agrupaciones
```

La consulta de la asignación de una instancia a las clases devuelve la probabilidad de pertenencia a cada una de ellas.

```
round(head(res_FCM$u), 2)
```

	Cluster 1	Cluster 2
1	0.74	0.26
2	0.74	0.26
3	0.76	0.24
4	0.78	0.22
5	0.79	0.21
6	0.78	0.22

Para representar en un gráfico los componentes principales, se requiere convertir el resultado, que es un objeto de tipo *ppclust*, a otro que la librería *factoextra* pueda representar.

```
res_FCMfe <- ppclust2(res_FCM, "kmeans")
fviz_cluster(res_FCMfe, data = dfMedidas_CLUSscale, ellipse = TRUE, geom = "point",
  palette = "jco")
```

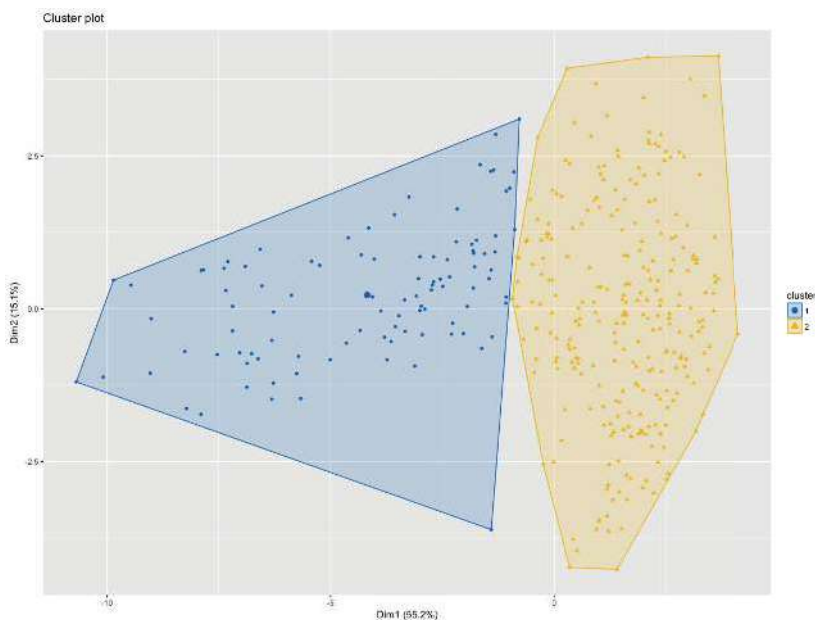


Fig. 7.23. Resultado del agrupamiento obtenido con FCM en coordenadas de PCA.

Al igual que se hizo en el particionamiento basado en modelos, la asignación de una instancia se realiza a la clase con la probabilidad mayor de pertenencia.

Hay otras técnicas de agrupamiento no encuadradas en la clasificación de las que se han hablado al principio de esta sección: basadas en redes de neuronas y modelos híbridos.

7.1.6 Otras técnicas de agrupamiento

Los **mapas autoorganizados** es una técnica para representación multidimensional que permite encontrar relaciones entre los datos. Se basa en un tipo de red de neuronas artificiales de aprendizaje no supervisado (redes de Kohonen).

El primer paso consiste en definir el tamaño de una rejilla, puede ser hexagonal o rectangular. Después se realiza el entrenamiento sobre la rejilla creada.

```
library(kohonen)
set.seed(123)
rejilla_SOM <- somgrid(xdim = 10, ydim = 10, topo = "hexagonal")
res_SOM <- som(as.matrix(dfMedidas_CLUSscale),
               grid = rejilla_SOM,
               rlen = 1000,
               alpha = c(0.05, 0.01),
               keep.data = TRUE )
par(mfrow=c(1,2))
plot(res_SOM, type="changes")
plot(res_SOM, type="count", main="Node Counts")
```

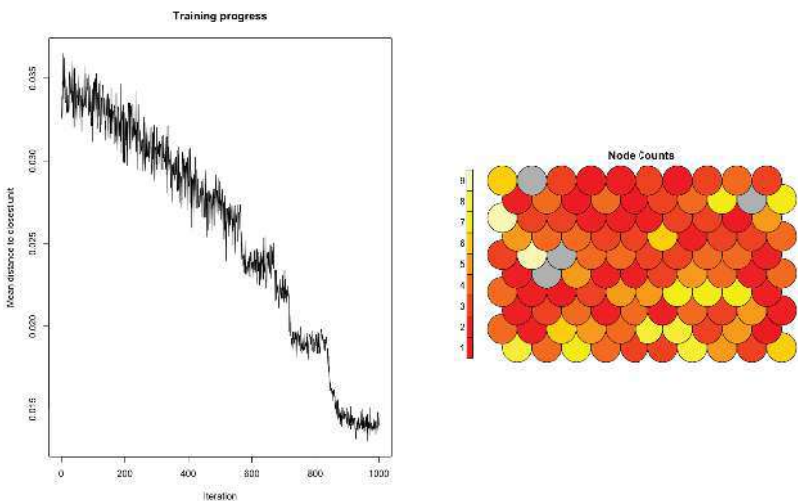


Fig. 7.24. Evolución del entrenamiento y número de instancias por nodo.

Se muestra en la figura 7.24 la evolución del proceso de entrenamiento. Cuando la distancia media de los nodos a las instancias se estabiliza después de decrecer, podemos considerar entonces que la red está entrenada. En la figura de la derecha se indica el número de instancias por nodo. Se aprecian nodos de color gris, es decir, están vacíos. Esto indica que la rejilla es demasiado grande para el conjunto de datos.

Se debe repetir el proceso con una rejilla de menor tamaño.

```
set.seed(123)

rejilla_SOM <- somgrid(xdim = 6, ydim = 6, topo = "hexagonal")

res_SOM <- som(as.matrix(dfMedidas_CLUSscale),
               grid = rejilla_SOM,
               rlen = 1000,
               alpha = c(0.05, 0.01),
               keep.data = TRUE )

par(mfrow=c(1,2))
plot(res_SOM, type="changes")
plot(res_SOM, type="count", main="Node Counts")
```

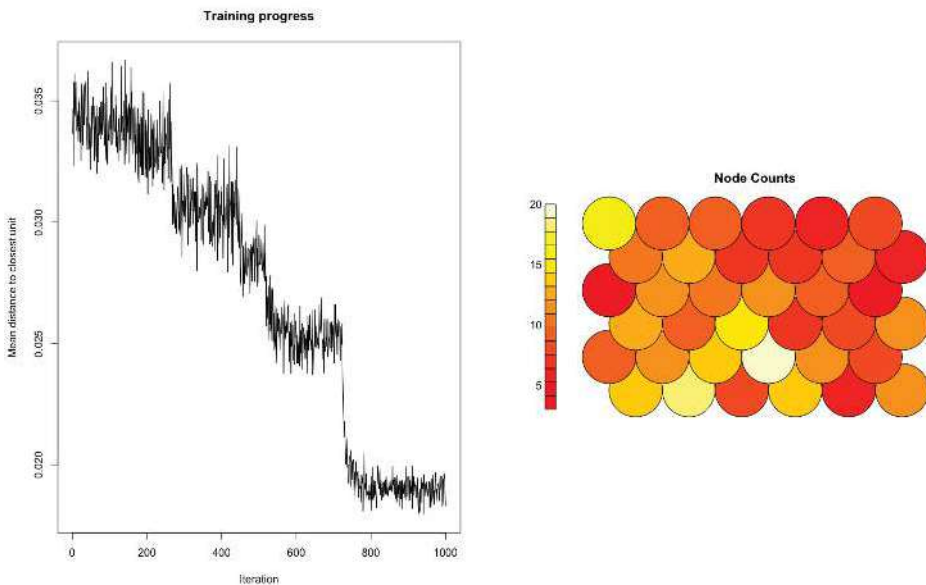


Fig. 7.25. Entrenamiento de SOM con una rejilla de 6x6.

Ahora en la rejilla se aprecia que no hay nodos vacíos.

La representación de la distancia de un nodo a los nodos vecinos da una apreciación de los grupos que pueden extraerse del SOM.

```
paleta_gris <- function(n, alpha = 1) {gray.colors(n, start = 1, end = 0, alpha =
alpha)[n:1]}
plot(res_SOM, type="dist.neighbours", main = "SOM neighbour distances", palette.name
= paleta_gris)
```

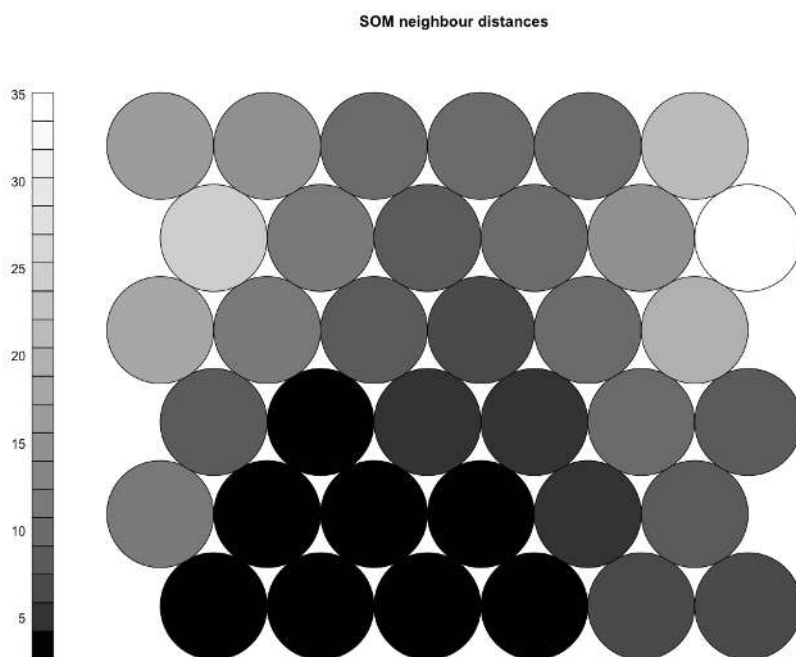


Fig. 7.26. Mapa de distancias de un nodo a sus vecinos.

Nodos con distancias pequeñas estarán agrupados, mientras que distancias grandes serán indicación de la frontera entre agrupaciones.

Si se aplica un agrupamiento jerárquico sobre el resultado de SOM, se obtiene una agrupación por regiones.

```

paleta <- c("#0073C2FF", "#EFC000FF", "#868686FF", "#CD534CFF", "#7AA6DCFF")

par(mfrow=c(2,2))

for (i in 5:2){

  som_cluster <- cutree(hclust(object.distances(res_SOM, "codes")), i)

  plot(res_SOM, type="mapping", bgcol = paleta[som_cluster], main = paste0("Agrupaciones con k:",i))

  add.cluster.boundaries(res_SOM, som_cluster)

}

```

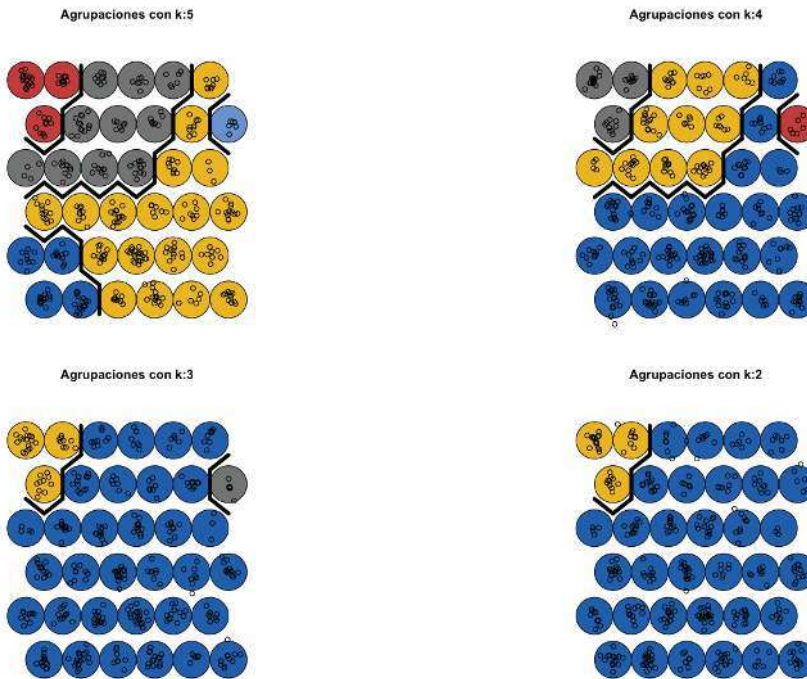


Fig. 7.27. Agrupaciones obtenidas cambiando la cantidad de agrupaciones en SOM.

La agrupación que aparece en el lado superior izquierdo es muy fuerte, al aumentar el número de agrupaciones, ésta no se subdivide. Representando el promedio de valores que se están mapeando en cada celda, se puede comprobar el parecido entre las que conforman una agrupación.

```

som_cluster <- cutree(hclust(object.distances(res_SOM, "codes")), 4)

plot(res_SOM, type="codes")

add.cluster.boundaries(res_SOM, som_cluster)

```

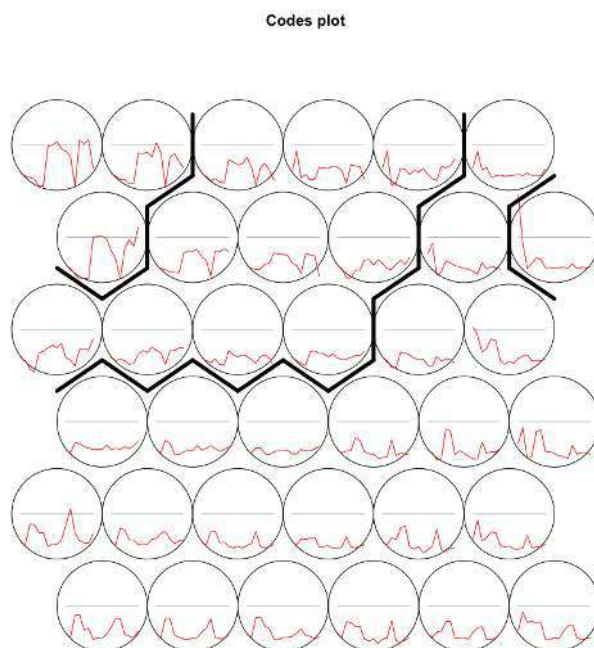


Fig. 7.28. Representación de los valores de los datos mapeados sobre cada celda de la red SOM.

En la figura 7.27 se muestran las fronteras para cuatro agrupaciones y se aprecia como los datos de celdas del mismo grupo se asemejan entre sí y la frontera representa un cambio abrupto en los datos.

Ahora, ver la figura 7.28, se puede inspeccionar para cada atributo como se han mapeado en las celdas de la red y dar significado a las agrupaciones.

```
plot(res_SOM, type = "property", property = getCodes(res_SOM)[,7],
     main = colnames(getCodes(res_SOM))[7])
add.cluster.boundaries(res_SOM, som_cluster)
```

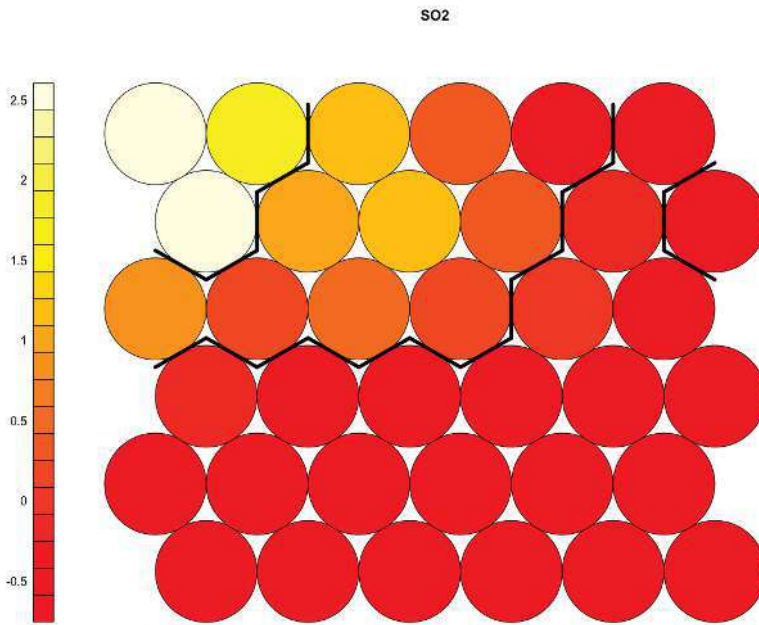


Fig. 7.29. Mapeado de la concentración escalada de SO₂ en SOM.

En la figura 7.29, la agrupación superior izquierda se corresponde con altas concentraciones de SO₂. Al desplazarnos en la red hacia la derecha y hacia abajo, la concentración de SO₂ disminuye. Por ejemplo, la celda que aparece en el lado superior derecho que constituye ella sola una agrupación, no podría explicarse por la concentración de SO₂, de modo que habrá que visualizar el resto de atributos para entender qué la caracteriza.

Representemos el resto de atributos en la escala de datos original, es decir, deshaciendo el escalado. De esta forma se podrá entender más adecuadamente qué conforma un agrupamiento.

```
par(mfrow=c(4,4))
for (i in 1:16){
  var_desescala <- aggregate(as.numeric(dfMedidas_CLUS[,i]), by=list(res_SOM$unit.
    classif),
                             FUN=mean, simplify=TRUE)[,2]
  plot(res_SOM, type = "property", property=var_desescala, main=colnames(getCodes(res_
    SOM))[i])
  add.cluster.boundaries(res_SOM, som_cluster)
}
```

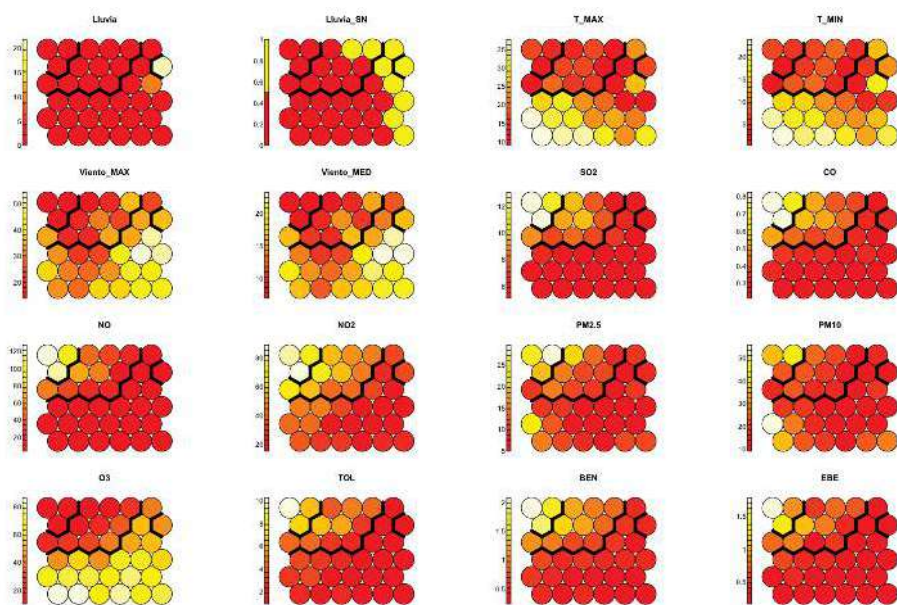


Fig. 7.30. Mapas de calor de los atributos en su proyección a SOM.

La visualización de los atributos en mapas de color en la red obtenida con SOM arroja mucha información acerca de la caracterización de las agrupaciones obtenidas. En la figura 7.30, la agrupación que aparece arriba y a la derecha con una sola celda se corresponde con días de lluvia de alta precipitación, pero no tiene una relación única que pueda explicar el resto de atributos. Por tanto, este agrupamiento, aun teniendo una estructura clara, carece de interés y debe ser descartado, asimilado en la agrupación que le rodea. El agrupamiento de la parte superior izquierda es compatible con altas concentraciones de gases, salvo para el ozono, cuyas altas concentraciones están bien reflejadas por el agrupamiento más grande. El análisis semántico de las clases es bastante sencillo utilizando SOM.

Las aproximaciones híbridas pueden ser muy variadas. Una muy común es utilizar los valores medios de las agrupaciones obtenidas con un agrupamiento jerárquico como los centroides iniciales de *k-means*. Es decir, el resultado de aplicar un método sirve como entrada o para refinar los resultados de otro. Otro ejemplo sería encontrar las agrupaciones de datos, pero transformados previamente mediante PCA, llamado HCPC (*Hierarchical Clustering on Principal Components*), tal como se muestra a continuación.

```
library(FactoMineR)
res_PCA <- PCA(dfMedidas_CLUSscale, ncp = 3, graph = FALSE)
res_HCPC <- HCPC(res_PCA, graph = FALSE)
plot(res_HCPC, choice = "3D.map")
```

Hierarchical clustering on the factor map

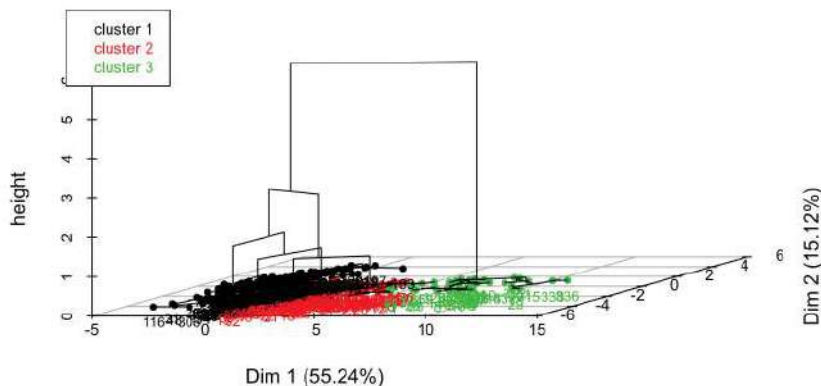


Fig. 7.31. Agrupamiento jerárquico sobre componentes principales.

En la figura 7.31 se muestra la estructura jerárquica de las agrupaciones que visualiza el hecho de que las agrupaciones 1 y 2 están más próximas que la 3. Una ventaja de este procedimiento es que no requiere determinar *a priori* el número de agrupamientos. En este caso, el algoritmo ha estimado 3 como el valor adecuado. Veamos a continuación la proyección sobre el plano.

```
fviz_cluster(res_HCPC, ellipse.type = "convex", palette = "jco", labelsiz = 8)
```

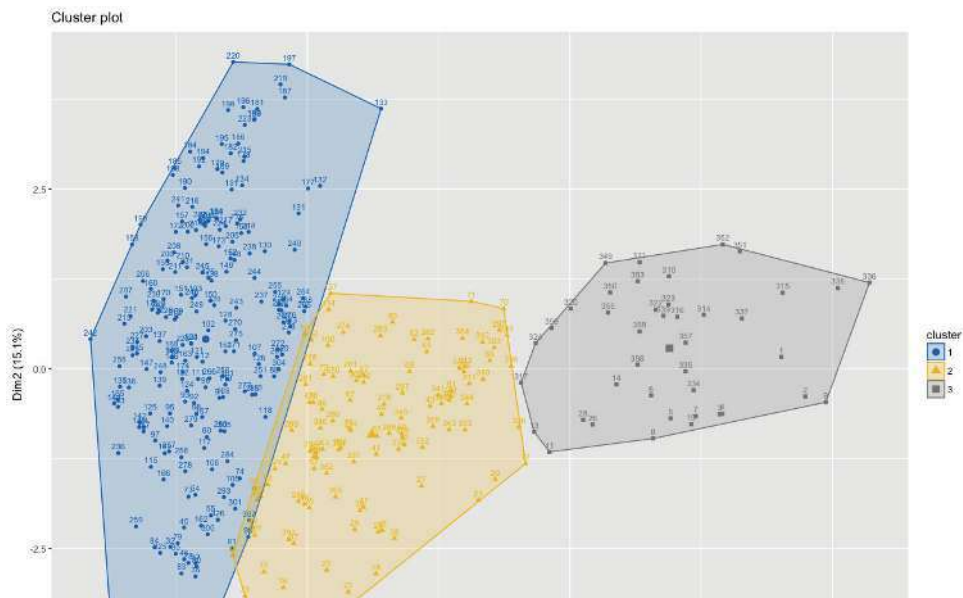


Fig. 7.32. Obtención de agrupamiento mediante HCPC.

Un resultado muy útil que puede obtenerse es la relación de atributos que mejor describe cada agrupamiento.

res_HCPC\$desc.var\$quanti					
\$`1`					
v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
O3	15.397520	0.6731793	6.454122e-18	0.6122111	0.9986292
1.700668e-53					
T_MIN	12.751788	0.5575079	8.199301e-17	0.8390080	0.9986292
3.046755e-37					
T_MAX	12.302597	0.5378693	7.955014e-17	0.8854508	0.9986292
8.770795e-35					
Viento_MAX	9.953659	0.4351738	1.471616e-16	0.8532085	0.9986292
2.430783e-23					
Viento_MED	9.572033	0.4184891	-1.851259e-16	0.8437152	0.9986292
1.048240e-21					
...					
\$`2`					
	v.test	Mean in category	Overall mean	sd in category	Overall sd
p.value					
NO2	5.304186	0.4254799	1.127428e-16	0.5483711	0.9986292
1.131768e-07					
CO	4.335545	0.3477796	-7.878021e-17	0.4877860	0.9986292
1.453991e-05					
BEN	4.232941	0.3395490	3.364888e-18	0.4859208	0.9986292
2.306553e-05					
SO2	3.953682	0.3171480	-1.420097e-16	0.6228833	0.9986292
7.695782e-05					
EBE	3.464111	0.2778767	2.915286e-17	0.6438859	0.9986292
5.319875e-04					
...					
\$`3`					
	v.test	Mean in category	Overall mean	sd in category	Overall sd
p.value					
NO	16.452898	2.4208984	4.600582e-18	7.647042e-01	0.9986292
7.994921e-61					
CO	15.458240	2.2745431	-7.878021e-17	6.772411e-01	0.9986292
6.638824e-54					
BEN	15.246807	2.2434327	3.364888e-18	6.892839e-01	0.9986292
1.728833e-52					
SO2	14.858776	2.1863374	-1.420097e-16	7.699284e-01	0.9986292
6.103022e-50					
TOL	14.640577	2.1542314	9.685935e-17	9.850886e-01	0.9986292
1.547488e-48					
...					

Podemos apreciar que el agrupamiento 1 está relacionado con la concentración de O3 y los atributos meteorológicos, mientras que para los agrupamientos 2 y 3 los principales atributos son las concentraciones de gases. Distinguiendo el agrupamiento 2 y 3 por la concentración de NO2 y de NO respectivamente.

Se ha visto que para determinadas técnicas es bastante sencillo extraer la información relativa a caracterizar cada agrupamiento porque incorporan funciones que la calculan o directamente ésta se encuentra en la estructura de datos resultado. En cualquier caso, es conveniente realizar medidas de estadística descriptiva sobre las clases para comprender mejor la información que están agrupando.

7.1.7 Representación y análisis de las clases

Se va a mostrar para los resultados obtenidos con *k-means*. Para el resto de técnicas el procedimiento es el mismo, simplemente basta con recuperar la asignación de las instancias a cada agrupamiento.

```
library(caret)

featurePlot(x = dfMedidas_CLUS, y = as.factor(res_KM$cluster), plot = "box",
  col = c("#0073C2FF", "#EFC000FF", "#868686FF"),
  scales = list(x = list(relation = "free"),
    y = list(relation = "free")))
```



Fig. 7.33. Boxplot de los atributos mostrando los valores segregados por la clase obtenida con *k-means*.

Se muestra la diferencia entre la clase 1 y la 2. Claramente, la concentración de O3 es superior para la clase 1, la media es de 60, mientras que para la clase 2 la media no alcanza 20°C. Esto ocurre para días con temperatura máxima media mayor, superior a 20°C, y temperatura mínima media cercana a 15°C. La lluvia no tiene impacto en las clases. Veamos algo más de detalle con las distribuciones de los atributos por clase.

```
featurePlot(x = dfMedidas_CLUS, y = as.factor(res_KM$cluster),
plot = "density",
col = c("#0073C2FF", "#EFC000FF", "#868686FF"),
scales = list(x = list(relation = "free"),
y = list(relation = "free")))
```

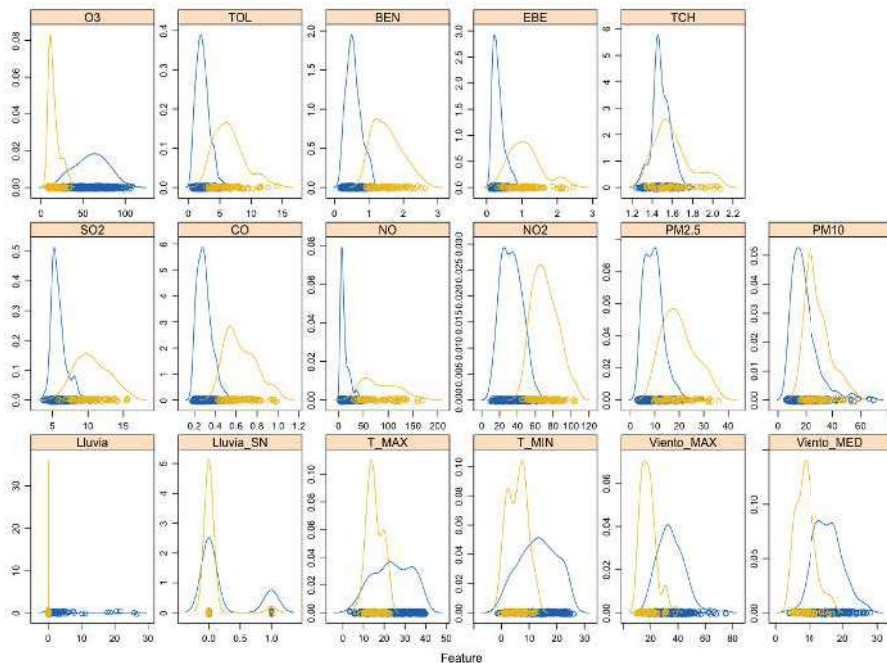


Fig. 7.34. Distribuciones de los atributos segregados por las clases obtenidas con *k-means*.

En las figuras anteriores se puede apreciar la discriminación de los atributos por las clases. Salvo algunos atributos, para la mayoría, las clases tienen distribuciones diferenciadas. De nuevo, los correspondientes a la lluvia muestran que no tienen impacto en la discriminación de clases. Por tanto, sería conveniente eliminar los atributos de lluvia y repetir el proceso de encontrar agrupaciones.

Para establecer relaciones entre las clases y varios atributos se pueden obtener representando gráficos de dispersión.

```
library(gridExtra)

df <- cbind(dfMedidas_CLUS, as.factor(res_KM$cluster))
colnames(df)[18] <- "Clase"
par(mfrow=c(1,2))
grid.arrange(
  ggplot(df, aes(x=O3, y=T_MAX, color=Clase)) + geom_point() + geom_rug() +
    stat_ellipse(type = "norm") +
    scale_colour_manual(name="", values =
c("1"="#0073C2FF", "2"="#EFC000FF")),
  ggplot(df, aes(x=SO2, y=T_MAX, color=Clase)) + geom_point() + geom_rug() +
    stat_ellipse(type = "norm") +
    scale_colour_manual(name="", values =
c("1"="#0073C2FF", "2"="#EFC000FF")),
  ncol = 2)

```

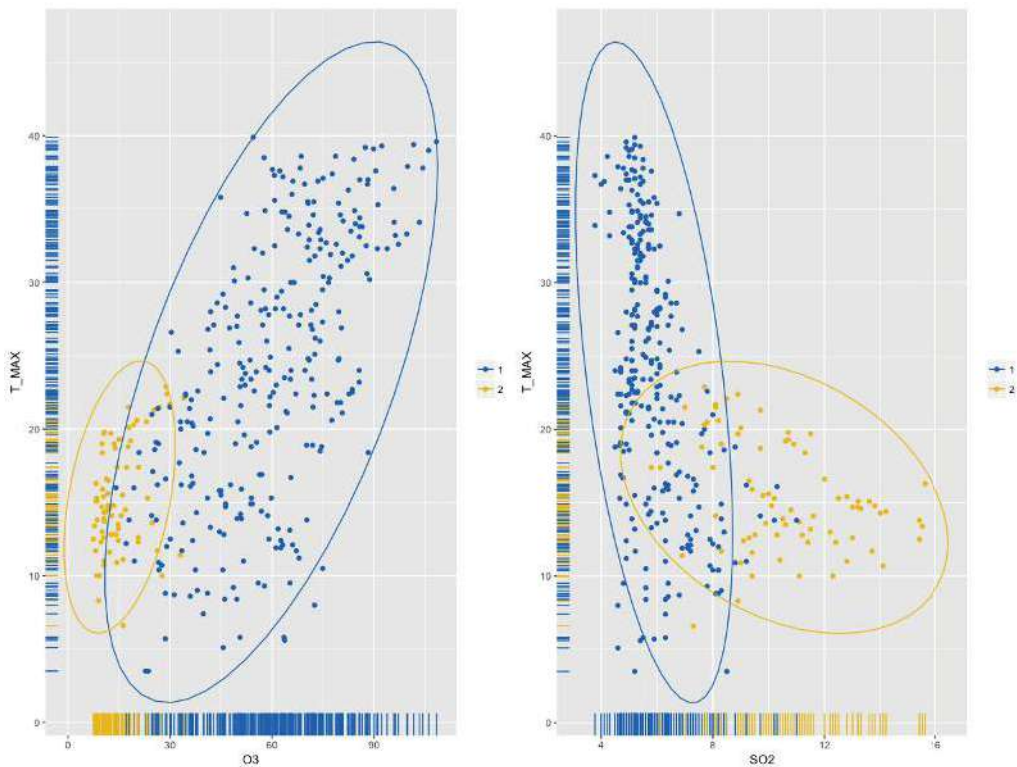


Fig. 7.35. Gráficos de dispersión de la temperatura máxima frente a las concentraciones de O3 y SO2 según las clases.

En la figura 7.35 se pone de manifiesto la discriminación de las clases, la clase 1 está relacionada con altas concentraciones de O₃ y bajas de SO₂, el inverso se produce para la clase 2. El efecto de la temperatura máxima es que la concentración alta de O₃ ocurre con temperaturas máximas altas, mientras que las altas concentraciones de SO₂ ocurren cuando las temperaturas máximas son bajas.

Habría que continuar representando diferentes atributos para poder explicar las clases, ésta es una etapa que siempre hay que realizar después de encontradas las clases con técnicas de agrupamiento no supervisado. Es muy posible también que del análisis de los resultados se proponga la eliminación, transformación o inclusión de atributos, teniendo que repetirse el proceso de agrupamiento. El proceso de análisis de datos es cíclico y requiere de varias iteraciones hasta poder llegar a una solución final.

Se ha dejado para el final de este capítulo una pregunta sin responder. Se ha visto que los resultados de aplicar diferentes técnicas de agrupamiento han producido diferentes resultados. ¿Qué técnica es mejor, cuál es la que tiene mejores resultados?

7.1.8 Validación de resultados

No hay una técnica mejor aplicable a cualquier problema. El artículo del año 1997 de David Wolpert y William Macready, *No Free Lunch Theorems for Optimization*, determinaba que sin añadir información del problema que se quiere resolver, utilizando las técnicas de optimización como una caja negra, todas ellas tienen el mismo comportamiento promedio sobre problemas elegidos al azar. Por tanto, para un problema nuevo habrá que probar con diferentes técnicas y evaluar su rendimiento. Si el problema que se pretende abordar es similar a algún otro que ya ha sido tratado, entonces la mejor estrategia es aplicar las técnicas que mejores resultados obtuvieron, es decir, incorporar al problema de búsqueda de soluciones información del dominio, en este caso la experiencia previa.

Cuando se trata un nuevo problema, el procedimiento es aplicar las diferentes técnicas y medir y comparar su rendimiento, para el problema que nos ocupa, el rendimiento al obtener agrupaciones. Algunas de las medidas ya se han aplicado, por ejemplo, para obtener el número óptimo de agrupaciones. No se realizaba una comparación con otras técnicas, se aplicaba la misma técnica cambiando el número de clases a obtener y una medida de calidad determinaba el óptimo.

Posibles medidas del rendimiento son:

- **Clases compactas.** Variación de las instancias que pertenecen a la misma clase. Dan lugar a varios índices dependiendo si usan la varianza, las distancias medias entre las instancias, entre las instancias y los centroides, etc.

- **Separación.** Medida de separación entre las clases. De nuevo, igual que para medir cómo de compactas son las clases, se puede usar la distancia entre centroides, entre instancias y centroides, etc.
- **Híbridos.** De las anteriores medidas se realizan combinaciones para obtener índices sintéticos. Por ejemplo, el índice de Dunn o el índice Silhouette.
- **Conectividad.** Densidad de instancias en las clases.
- **Pureza.** Si se conocen las clases *a priori* pueden realizarse las medidas de bondad propias de los problemas de clasificación: precisión, pureza, especificidad, etc.

La librería *fpc* incorpora una función para calcular muchos índices de desempeño, tomando la matriz de disimilitud y el resultado de asignar a cada instancia a una clase.

```
library(fpc)

km_stats <- cluster.stats(distancia, res_KM$cluster)

km_stats$average.between #distancia media entre clases
km_stats$average.within #distancia media intra-clases
km_stats$avg.silwidth #silhouette
km_stats$pearsongamma
km_stats$dunn #índice de Dunn
km_stats$entropy #entropía de la distribución de pertenencia a las clases
km_stats$wb.ratio #relación entre average.within/average.between
km_stats$ch #índice de Calinski and Harabasz
km_stats$widestgap #máxima distancia entre clases
km_stats$sindex #separación
```

```
[1] 7.552691
[1] 4.151023
[1] 0.4239778
[1] 0.6368811
[1] 0.0976847
[1] 0.5079054
[1] 0.5496084
[1] 241.6616
[1] 3.475233
[1] 1.703626
```

Éstos son algunos de los índices que se obtienen con la función *cluster.stats*, vamos a tomar algunos de ellos y compararlos con los obtenidos con otros algoritmos de agrupamiento.

```

compara_CLUS <- data.frame(res_stats$avg.silwidth, res_stats$dunn, res_stats$sindex,
res_stats$wb.ratio, row.names = "K-means") colnames(compara_CLUS) <- c("SIL", "DUNN",
"SEP", "WB")

add_resultado <- function(dfcompara, vector_clases, nombre){
  res_stats <- cluster.stats(distancia, vector_clases)
  dfcompara <- rbind(dfcompara, c(res_stats$avg.silwidth, res_stats$dunn, res_stats$sindex,
res_stats$wb.ratio))
  row.names(dfcompara)[nrow(dfcompara)] <- nombre
  return(dfcompara)
}

compara_CLUS <- add_resultado(compara_CLUS, res_HC$cluster, "HC")
compara_CLUS <- add_resultado(compara_CLUS, res_HC_corregido$cluster, "HC_corr")
compara_CLUS <- add_resultado(compara_CLUS, res_EM$classification, "EM")
compara_CLUS <- add_resultado(compara_CLUS, res_FCM$cluster, "FCM")
compara_CLUS <- add_resultado(compara_CLUS, res_DBSCAN$cluster, "DBSCAN")
compara_CLUS <- add_resultado(compara_CLUS, as.integer(res_HCPC$data.clust$clust),
"HCPC")
compara_CLUS <- add_resultado(compara_CLUS, res_PAM$clustering, "PAM")
###Determinar la clase para SOM
som_cluster <- cutree(hclust(object.distances(res_SOM, "codes")), 3)
res_SOM_pred <- predict(res_SOM) #Predecir con el SOM
res_SOM_cluster <- som_cluster[res_SOM_pred$unit.classif] #Asigna la clase asociada
a la celda
compara_CLUS <- add_resultado(compara_CLUS, res_SOM_cluster, "SOM")

formattable(compara_CLUS[order(compara_CLUS[,1], decreasing = TRUE),], digits = 2,
format = "f", row.names=TRUE, list(SIL = color_tile("red", "green"),DUNN = color_tile("red", "green"), SEP = color_tile("red", "green"), WB = color_tile("green", "red") ))

```

	SIL	DUNN	SEP	WB
HC	0.455	0.111	2.0	0.52
HC_corr	0.439	0.108	1.7	0.54
K-means	0.430	0.138	2.6	0.49
SOM	0.430	0.138	2.6	0.49
PAM	0.398	0.090	1.6	0.57
FCM	0.386	0.094	1.6	0.58
HCPC	0.260	0.068	1.3	0.59
EM	0.237	0.068	1.3	0.63
DBSCAN	0.096	0.087	2.5	0.82

Fig. 7.36. Comparación de las diferentes técnicas de agrupamiento según varios índices de desempeño.

Los resultados están ordenados según el índice Silhouette. En una escala de color desde el mejor valor, en verde intenso, hasta el peor, en rojo intenso. El mejor resultado lo proporciona el agrupamiento realizado con SOM y los peores serían DBSCAN, HCPC y EM. También la modificación que se hizo en el agrupamiento jerárquico reasignando algunas instancias ha empeorado el resultado en lugar de mejorarlo. Estos resultados se muestran con carácter didáctico, utilizar 365 instancias con 17 atributos es muy inadecuado para abordar un problema de agrupamiento, lo razonable sería tener varios cientos de miles de instancias.

7.2 Clasificación

El problema de clasificación es uno de los grandes bloques que trata la estadística y el aprendizaje automático. Es un tipo de tarea encuadrada dentro de la categoría del aprendizaje supervisado. Las redes de neuronas artificiales, clasificadores bayesianos, estimadores de *kernel* o *random forest* son ejemplos de técnicas, muy distintas, pero todas ellas pertenecientes a la categoría de aprendizaje supervisado, con aplicaciones en problemas de clasificación. En el capítulo 5 se presentaron las técnicas clásicas para afrontar este problema, y aquí se van a revisar las técnicas basadas en el aprendizaje automático, limitándonos a las comúnmente más usadas.

Como se indicó en el capítulo 5, se va a utilizar el nuevo atributo nominal, O3_Nivel, a partir de la concentración máxima de O3 que divide ésta en tres niveles de peligrosidad: verde, amarillo y rojo. Estos niveles se han tomado arbitrariamente y no tienen rigor en el sentido de las ciencias ambientales, únicamente tienen interés didáctico.

```
dfMedidas$O3_Nivel <- cut(dfMedidas$O3_MAX, c(min(dfMedidas$O3_MAX), 75, 100,
max(dfMedidas$O3_MAX)), include.lowest = TRUE, right = FALSE, labels = c("V", "A", "R"))
summary(dfMedidas$O3_Nivel)
```

```
V    A    R
208 131  26
```

El objetivo va a ser encontrar un modelo que permita etiquetar de forma correcta el nivel de alerta de O3_MAX para nuevas situaciones futuras.

7.2.1 Selección de atributos

Un paso previo a aplicar las técnicas de aprendizaje consiste en seleccionar los atributos más relevantes para realizar la clasificación. Cuando se abordó la realización de modelos de regresión ya se trató en ese contexto, ahora se van a aplicar otras técnicas complementarias para determinar qué atributos pueden descartarse.

Las técnicas de selección de atributos pueden agruparse en dos aproximaciones principales. Una que se basa en realizar una medida sobre cada atributo y establecer un *ranking* a partir de ella, descartando los atributos peor situados en el mismo. La otra se basa en buscar el subconjunto de atributos que maximiza la predicción de un modelo de clasificación. Medidas para establecer un *ranking* hay muchas: basadas en la correlación, chi-cuadrado, entropía, ganancia de información. Cada una de ellas puede ordenar de forma distinta los atributos, de modo que puede tomarse un criterio de decisión mayoritario.

Se añade un atributo que se obtiene al convertir el día de la semana a un valor discreto numérico y se seleccionan los atributos de partida. La librería *Fselector*, orientada a la selección de atributos, dispone de numerosas funciones para evaluar la importancia de los atributos.

Funciones que permiten hacer *rankings*:

```
library(FSelector)

set.seed(10)

dfMedidas$Dia_sem_num <- as.numeric(Dia_sem)

atributos <- c("T_MAX", "T_MIN", "Lluvia", "Viento_MAX", "Viento_MED", "Dia_sem_num",
"Dia_mes",
"SO2", "CO", "NO", "NO2", "PM2.5", "PM10", "TOL", "BEN", "EBE", "TCH", "NMCH",
"SO2_MAX", "CO_MAX", "NO_MAX", "NO2_MAX", "PM2.5_MAX", "PM10_MAX",
"TOL_MAX")

formula <- as.simple.formula(atributos, "O3_Nivel")

importancia_SU <- symmetrical.uncertainty(formula, dfMedidas) #medidas para ranking
importancia_IG <- information.gain(formula, dfMedidas)
importancia_GR <- gain.ratio(formula, dfMedidas)

resultado_ranking <- data.frame(importancia_SU, importancia_IG, importancia_GR)
names(resultado_ranking) <- c("Symmetrical Uncertainty", "Information Gain", "Gain
Ratio")

round(resultado_ranking, 2)

plot(resultado_ranking$`Symmetrical Uncertainty`, resultado_ranking$`Information Gain`)
text(x = resultado_ranking$`Symmetrical Uncertainty`, y = resultado_ranking$`Infor-
mation Gain` + 0.0075, cex = 0.75, labels = row.names(resultado_ranking), col="red")
plot(resultado_ranking$`Symmetrical Uncertainty`, resultado_ranking$`Gain Ratio`)
text(x = resultado_ranking$`Symmetrical Uncertainty`, y = resultado_ranking$`Gain Ra-
tio` + 0.0075, cex = 0.75, labels = row.names(resultado_ranking), col="red")
plot(resultado_ranking$`Information Gain`, resultado_ranking$`Gain Ratio`)
text(x = resultado_ranking$`Information Gain`, y = resultado_ranking$`Gain Ratio` +
0.0075, cex = 0.75, labels = row.names(resultado_ranking), col="red")
```

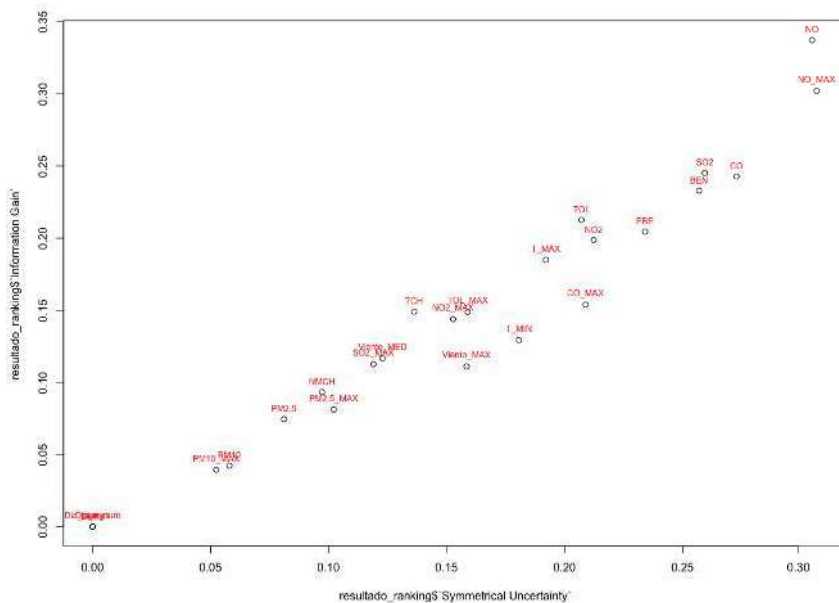


Fig. 7.37. Ranking de la importancia de los atributos. En los ejes, *Symmetrical Uncertainty* frente a *Information Gain*.

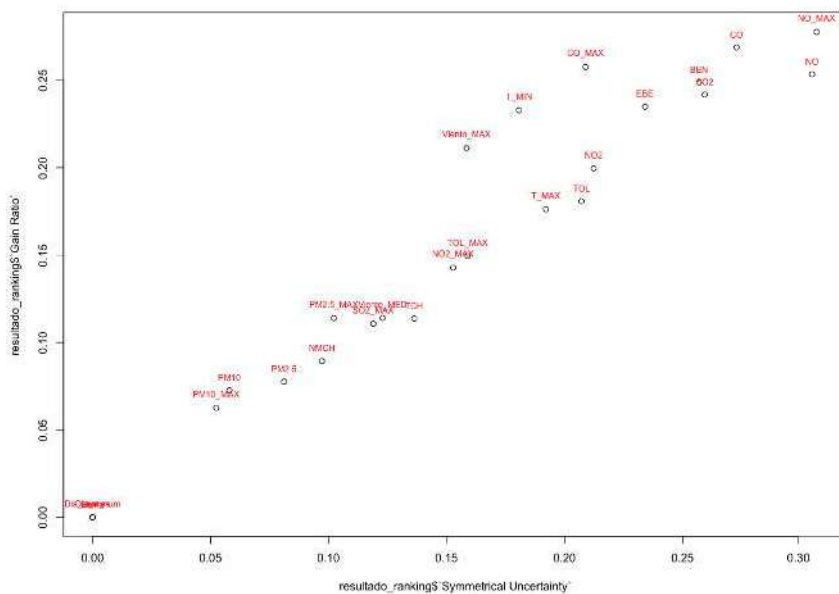


Fig. 7.38. Ranking de la importancia de los atributos. En los ejes, *Symmetrical Uncertainty* frente a *Gain Ratio*.

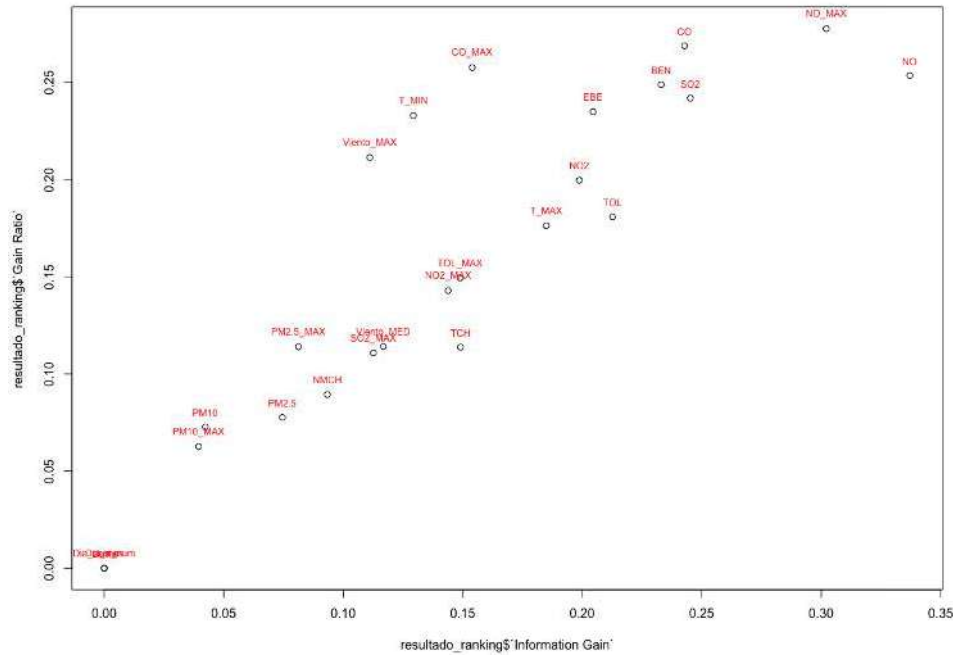


Fig. 7.39. Ranking de la importancia de los atributos. En los ejes, *Information Gain* frente a *Gain Ratio*.

Se muestra en la tabla el resultado de evaluar la importancia de los atributos siguiendo diferentes medidas. Se aprecia que algunos atributos, por ejemplo, *Dia_mes*, son muy malos para todas las medidas, pero otros, *T_MIN*, son muy buenos en una medida y normal en otras. Se presenta ahora otro problema, cómo elegir los atributos cuando se tienen varias medidas para ellos. Puede hacerse la media si las medidas se realizan sobre la misma magnitud, si no, habría que normalizarlas, o voto ponderado o establecer el frente de Pareto. Por ejemplo, en la gráfica que representa *Symmetrical Uncertainty* frente a *Information Gain*, la medida en *Information Gain* para *NO* es mejor que *NO_MAX*, pero se invierte si miramos *Symmetrical Uncertainty*. No puede determinarse, atendiendo a estos criterios, qué atributo es mejor, pero ambos son mejores que el resto de atributos. No hay ningún atributo que mejore el valor de ninguna de las medidas, ambos formarían parte del frente de Pareto. Este análisis se puede hacer con el resto de atributos y medidas para seleccionar los atributos más importantes.

Otra forma de establecer el *ranking* es tomando la importancia de los atributos que proveen algunos métodos de clasificación, por ejemplo, *random forest*.

```
rf_imp <- random.forest.importance(formula, dfMedidas, importance.type = 1)
print(rf_imp)
```

	attr_importance
T_MAX	21.6689159
T_MIN	16.4400705
Lluvia	-0.4465499
Viento_MAX	8.1592336
Viento_MED	2.9693372
Dia_sem_num	2.1263141
Dia_mes	5.5156866
SO2	21.6849125
CO	16.5523292
NO	32.9324832
NO2	14.1619586
PM2.5	13.7781355
PM10	16.2668820
TOL	12.7122937
BEN	17.6458139
EBE	11.0626806
TCH	15.2719689
NMCH	11.5821065
SO2_MAX	10.3424960
CO_MAX	9.8521816
NO_MAX	24.6237146
NO2_MAX	11.9327528
PM2.5_MAX	11.0795941
PM10_MAX	14.2583091
TOL_MAX	16.6283979

La librería *Fselector* también incorpora funciones que realizan selección de un subconjunto de atributos. La idea de este tipo de algoritmo es la de incorporar un método de clasificación y un algoritmo de búsqueda para encontrar la combinación de atributos que obtienen un mejor modelo de clasificación. Por ejemplo, CFS y CON usan un algoritmo de búsqueda primero para encontrar la mejor combinación de atributos.

```
subset_CFS <- cfs(formula, dfMedidas)
subset_CON <- consistency(formula, dfMedidas)

print("CFS")
subset_CFS
print("CON")
subset_CON
```

```
[1] "CFS"
[1] "T_MAX"      "T_MIN"      "Viento_MED" "SO2"        "CO"         "NO"
[7] "BEN"        "EBE"        "TCH"        "NMCH"       "NO_MAX"
[1] "CON"
[1] "T_MAX"      "Viento_MED" "SO2"        "CO"         "NO"         "NO2"
[7] "PM2.5"      "PM10"       "TOL"        "TCH"        "NMCH"       "NO_MAX"
[13] "NO2_MAX"    "PM2.5_MAX"  "PM10_MAX"   "TOL_MAX"
```

Con CFS se obtiene un subconjunto de once atributos y un subconjunto de dieciséis con CON.

También hay funciones con mayor flexibilidad, permitiendo elegir tanto el algoritmo de clasificación como el de búsqueda. Eso es, al método de búsqueda hay que incorporar la llamada al método de clasificación. En el ejemplo se ha realizado una función que aplica una validación cruzada de cuatro particiones usando el método de clasificación *rpart*. En *rpart* se crea un árbol de decisión para clasificar las instancias en las diferentes clases del atributo O3_Nivel.

```
library(rpart)

evaluador <- function(subset) {
  k <- 4 #k-fold cross validation
  splits <- runif(nrow(dfMedidas))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- dfMedidas[test.idx, , drop=FALSE]
    train <- dfMedidas[train.idx, , drop=FALSE]
    tree <- rpart(as.simple.formula(subset, "O3_Nivel"), train)
    error.rate = sum(test$O3_Nivel != predict(tree, test, type="c")) / nrow(test)
    return(1 - error.rate)
  })
}
```

```

    })
  return(mean(results))
}

subset <- hill.climbing.search(atributos, evaluador)
f <- as.simple.formula(subset, "O3_Nivel")
print("Hill Climbing:")
print(f)

subset <- forward.search(atributos, evaluador)
f <- as.simple.formula(subset, "O3_Nivel")
print("Forward Search:")
print(f)

subset <- backward.search(atributos, evaluador)
f <- as.simple.formula(subset, "O3_Nivel")
print("Backward Search:")
print(f)

subset <- best.first.search(atributos, evaluador)
f <- as.simple.formula(subset, "O3_Nivel")
print("Best First Search:")
print(f)

[1] "Hill Climbing:"
O3_Nivel ~ Lluvia + Viento_MAX + Viento_MED + SO2 + CO + PM2.5 +
TOL + TCH + NO_MAX + PM2.5_MAX + TOL_MAX

[1] "Forward Search:"
O3_Nivel ~ SO2 + SO2_MAX + NO_MAX

[1] "Backward Search:"
O3_Nivel ~ T_MAX + T_MIN + Lluvia + Viento_MAX + Viento_MED +
      Dia_sem_num + Dia_mes + CO + NO + NO2 + PM2.5 + PM10 + TOL +
BEN + EBE + TCH + NMCH + SO2_MAX + CO_MAX + NO_MAX + NO2_MAX +
      PM2.5_MAX + PM10_MAX + TOL_MAX

[1] "Best First Search:"
O3_Nivel ~ T_MAX + SO2 + NO + TOL + CO_MAX

```

Se aprecia que usando el mismo método de clasificación el resultado difiere en función del algoritmo de búsqueda empleado.

Se muestra con otras librerías, *caret* y *mlr*, cómo se implementan los métodos de obtención de la importancia y subconjuntos de atributos.

```
library(caret)

control <- trainControl(method="repeatedcv", number = 4, repeats = 3)

# Entrenar el modelo

modelo <- train(formula, data = dfMedidas, method = "lvq", preProcess = "scale", tr-
Control = control)

# estimar la importancia

importancia <- varImp(modelo, scale = FALSE)

print(importancia)

plot(importancia)
```

ROC curve variable importance

variables are sorted by maximum importance across the classes
only 20 most important variables shown (out of 25)

	V	A	R
NO	0.9534	0.8999	0.9534
T_MAX	0.9530	0.8517	0.9530
T_MIN	0.9405	0.8491	0.9405
NO_MAX	0.9395	0.8916	0.9395
SO2	0.9002	0.8420	0.9002
EBE	0.8724	0.8253	0.8724
BEN	0.8710	0.8626	0.8710
CO	0.8665	0.8264	0.8665
CO_MAX	0.8551	0.7624	0.8551
NO2	0.8435	0.8435	0.8111
TOL	0.8309	0.8309	0.7700
TOL_MAX	0.7932	0.7932	0.6967
NO2_MAX	0.7901	0.7901	0.6934
PM2.5	0.6916	0.7757	0.7757
PM10	0.7347	0.7663	0.7663
PM2.5_MAX	0.7092	0.7623	0.7623
Viento_MAX	0.7598	0.7598	0.6894
Viento_MED	0.7526	0.7526	0.6936
PM10_MAX	0.6904	0.7381	0.7381
SO2_MAX	0.7194	0.7194	0.7025

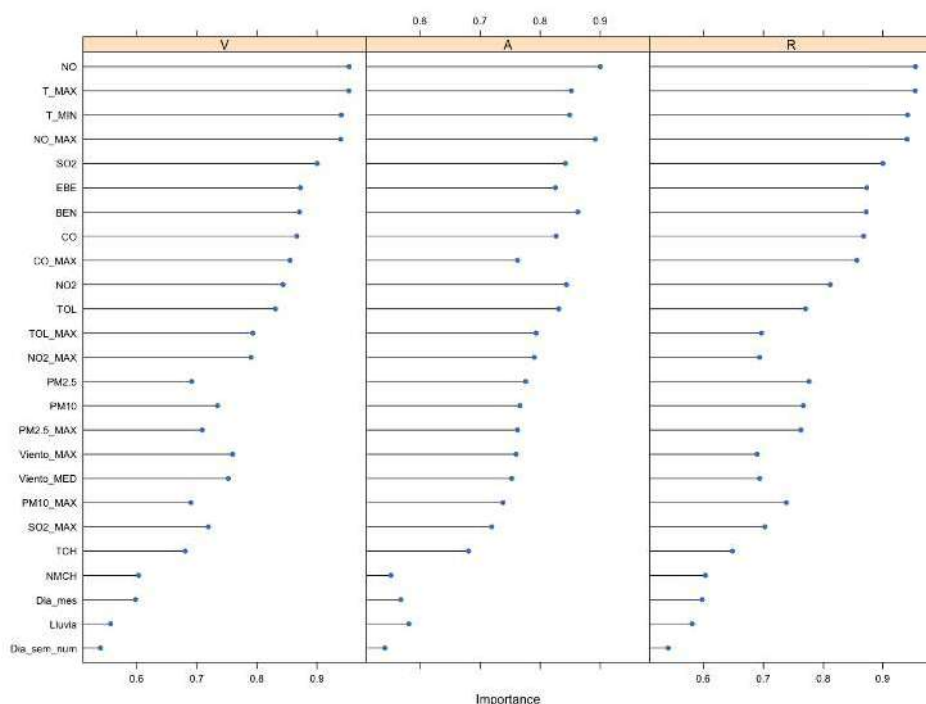


Fig. 7.40. Medida de la importancia de los atributos para predecir cada clase.

Es interesante mostrar el resultado que arroja las funciones de *caret*. Se desglosa por cada clase la importancia de cada atributo. Esto es interesante cuando la importancia de predecir cada una de las clases es distinta, la elección de un atributo u otro puede inclinar la capacidad de clasificación hacia una clase determinada.

En el gráfico de la figura 7.41 se muestra la ganancia en precisión del modelo según aumenta el número de atributos. El máximo ocurre en 22, pero la ganancia de 6 atributos a 22 no es lo suficientemente grande, pudiendo ser éste un buen valor de corte para el número de atributos a utilizar.

```
control <- rfeControl(functions=rfFuncs, method="cv", number=10)

resultados <- rfe(dfMedidas[,atributos], O3_Nivel, sizes=c(1:length(atributos)),
rfeControl=control)

print(resultados)

predictors(resultados)

plot(resultados, type=c("g", "o"))
```

The top 5 variables (out of 21):

NO, NO_MAX, T_MAX, SO2, BEN

[1]	"NO"	"NO_MAX"	"T_MAX"	"SO2"	"BEN"	"PM10"	"T_MIN"
[8]	"CO"	"PM2.5"	"PM10_MAX"	"NMCH"	"EBE"	"NO2"	"TOL"
[15]	"TOL_MAX"	"PM2.5_MAX"	"TCH"	"CO_MAX"	"SO2_MAX"	"NO2_MAX"	"Dia_mes"

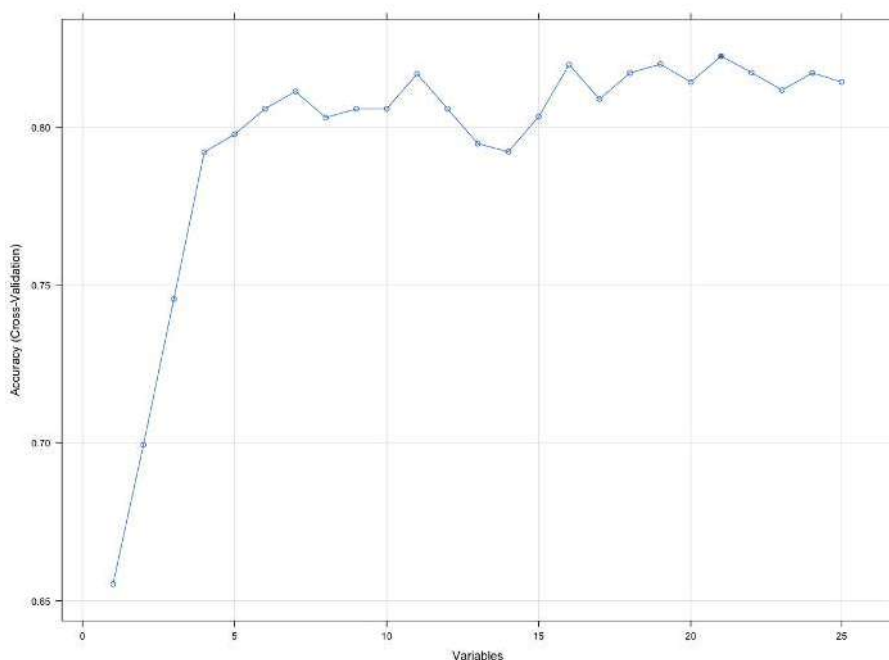


Fig. 7.41. Ganancia en precisión con la cantidad de atributos utilizados.

Una librería muy popular para aplicar técnicas de aprendizaje automático es *mlr*. Esta librería es un *wrapper* a las funciones de otras librerías, con la ventaja de que uniformiza las estructuras de datos de entrada/salida y las llamadas a las funciones.

```
library(mlr)

df <- dfMedidas[, names(dfMedidas) %in% c(atributos, "O3_Nivel")]

classif.task = makeClassifTask(data = df, target = "O3_Nivel")

fv <- generateFilterValuesData(classif.task, method = c("information.gain", "chi.squared"))

fv$data[order(fv$data[,4], decreasing = TRUE),]

plotFilterValues(fv)
```

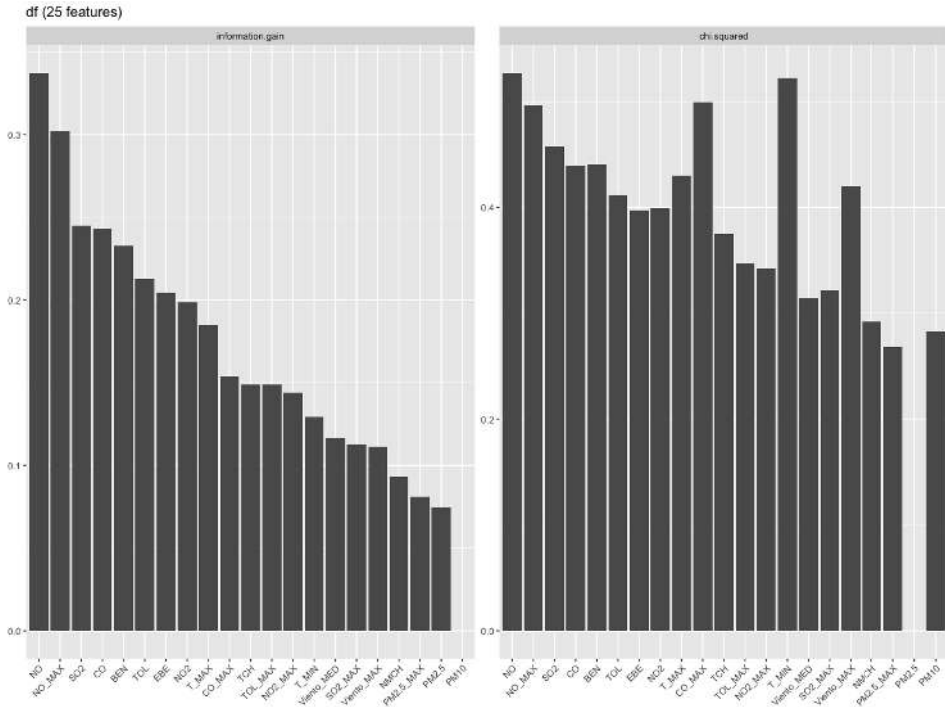


Fig. 7.42. Importancia de los atributos según dos figuras de mérito.

Para finalizar esta sección, se muestra el resultado de aplicar otra librería, *Boruta*, que simplifica en una sola línea la llamada a un clasificador (*random forest*) junto con un algoritmo de búsqueda. Se muestra la importancia que *Boruta* asigna a cada atributo.

```
library(Boruta)

boruta_output <- Boruta(formula, data=dfMedidas, doTrace=2)
attStats(boruta_output)
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")
```

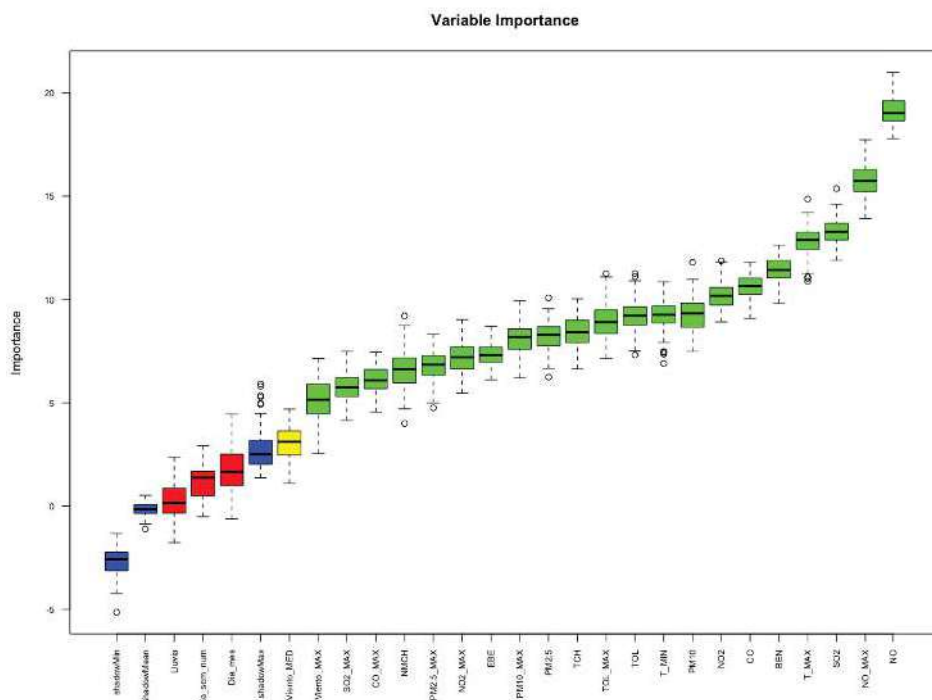


Fig. 7.43. Importancia de los atributos según *Boruta*.

7.2.2 Reducción de la dimensionalidad

Se han visto en la sección anterior técnicas para seleccionar y descartar atributos. Pero puede suceder que el número de atributos aún sea demasiado grande. Una estrategia muy utilizada es la de combinar atributos, mediante combinaciones lineales, sustitu-

yendo los originales por los nuevos y manteniendo gran parte de la información. El análisis de componentes principales (PCA) se realiza sobre la matriz de covarianza de los atributos normalizados de media cero y desviación estándar uno. El primer componente principal captura la máxima varianza de los datos y determina la dirección, a lo largo de la cual se produce la mayor variabilidad. El segundo componente principal es ortogonal al primero, es decir, no está correlado con él, etc. Los componentes principales se obtienen a partir de los autovalores y autovectores de las matrices de covarianza.

Se pueden utilizar las librerías *factoextra* o *FactoMineR* para calcular y representar los resultados del PCA.

```
library(factoextra)
library(FactoMineR)
df <- dfMedidas[ , names(dfMedidas) %in% c(atributos, "O3_Nivel")]
res.pca <- prcomp(df[, -25], scale = TRUE) #no se utiliza O3_nivel porque no es numérico
summary(res.pca)
fviz_screplot(res.pca, addlabels = TRUE)
```

Importance of components:

		PC1	PC2	PC3	PC4	PC5	PC6	PC7
PC8	PC9							
Standard deviation		3.6277	1.8219	1.2510	1.13554	1.04791	0.97611	0.9578
0.73250								0.88793
Proportion of Variance		0.5264	0.1328	0.0626	0.05158	0.04392	0.03811	0.0367
0.02146								0.03154
Cumulative Proportion		0.5264	0.6592	0.7218	0.77336	0.81729	0.85540	0.8921
0.94509								0.92363
		PC10	PC11	PC12	PC13	PC14	PC15	PC16
PC17								
Standard deviation		0.60543	0.50671	0.4124	0.37334	0.31189	0.27597	0.25245
								0.21622
Proportion of Variance		0.01466	0.01027	0.0068	0.00558	0.00389	0.00305	0.00255
								0.00187
Cumulative Proportion		0.95976	0.97003	0.9768	0.98240	0.98630	0.98934	0.99189
								0.99376
		PC18	PC19	PC20	PC21	PC22	PC23	PC24
PC25								
Standard deviation		0.18843	0.17368	0.15563	0.14441	0.1227	0.11483	0.09527
								0.08894
Proportion of Variance		0.00142	0.00121	0.00097	0.00083	0.0006	0.00053	0.00036
								0.00032
Cumulative Proportion		0.99518	0.99639	0.99736	0.99819	0.9988	0.99932	0.99968
								1.00000

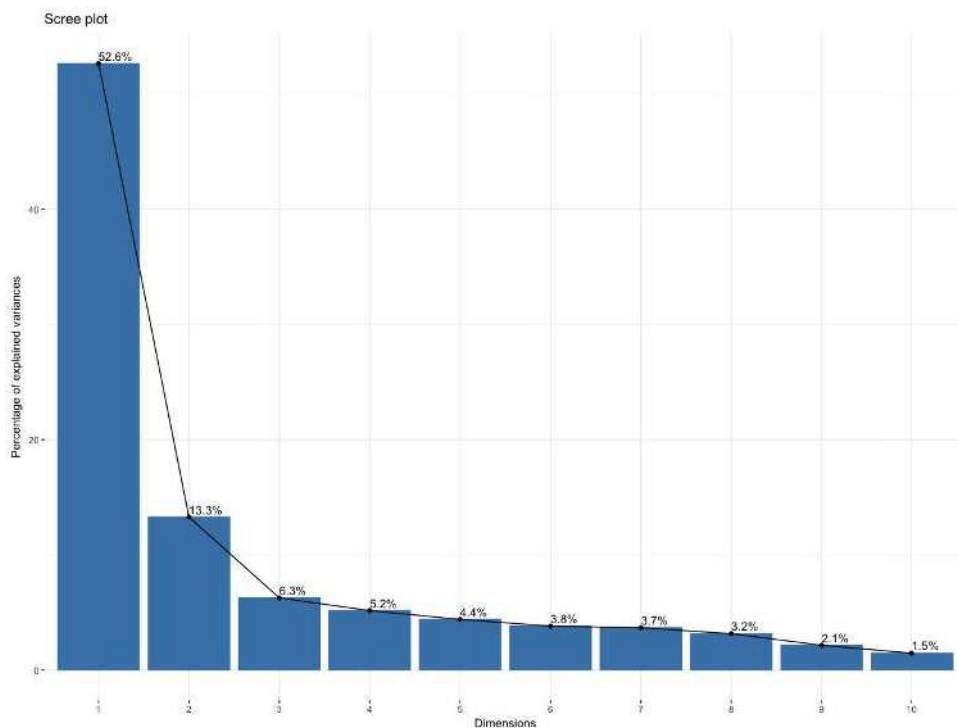


Fig. 7.44. Contribución de los componentes principales.

El primer componente principal puede explicar el 52 % de la variación de los datos, el segundo el 13 %, ambos pueden explicar (el acumulado) el 66 % de la variación. Aparece ahora una cuestión, ¿cuántos componentes principales deben mantenerse?

La librería *nFactors* dispone de diferentes medidas que ayudan a resolver esta cuestión.

```
library(nFactors)
ev <- get_eig(res.pca)[1] #los autovalores
ap <- parallel(subject=nrow(df),var=ncol(df[, -25]), rep=100, cent=.05)
nS <- nScree(x=ev$eigenvalue, aparallel=ap$eigen$qevpea)
plotnScree(nS)
```

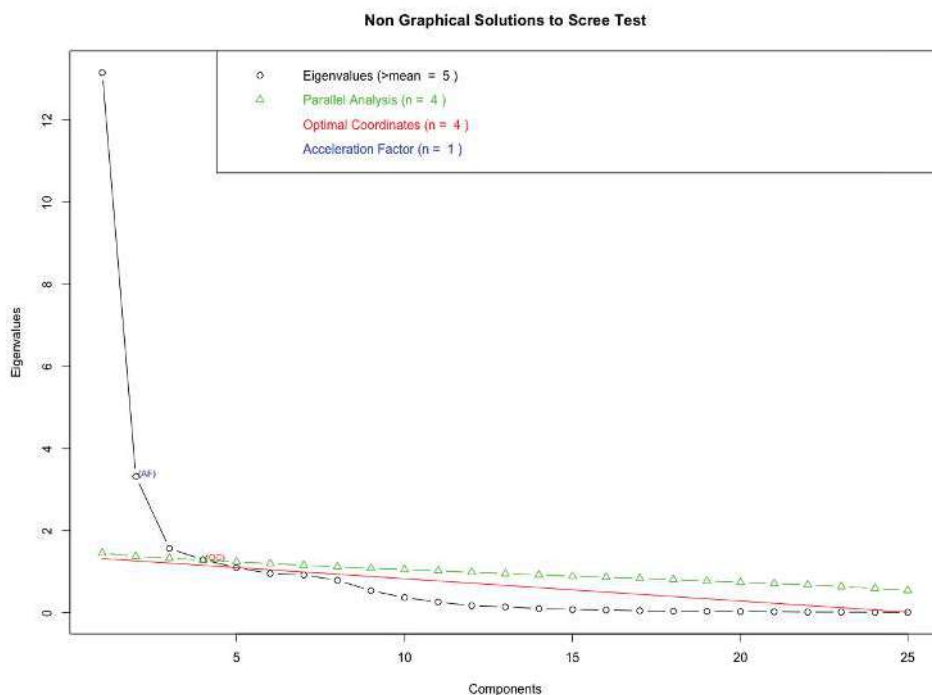


Fig. 7.45. Número óptimo de componentes principales.

Diferentes medidas arrojan diferentes resultados acerca del número óptimo de componentes principales a retener. En el caso más conservador habría que tomar cinco y con el más arriesgado sería suficiente el primer componente principal. Se reduce de veinticinco a cinco dimensiones. Estas nuevas surgen realizando combinaciones lineales de los anteriores atributos.

Se pueden obtener la correlación y la calidad de representación que existen entre los atributos y los componentes principales, esto es, examinar los atributos con su signo, distancia al origen y ángulo con ejes de los componentes principales. La calidad de representación se puede obtener a través del cálculo del cuadrado del coseno del ángulo que forma un atributo con el eje que está representando un componente principal, su contribución por la distancia al origen. A la representación de los atributos y las instancias en los ejes que forman los componentes principales se le denomina *gráfico biplot*.

```
fviz_pca_var(res.pca)
```

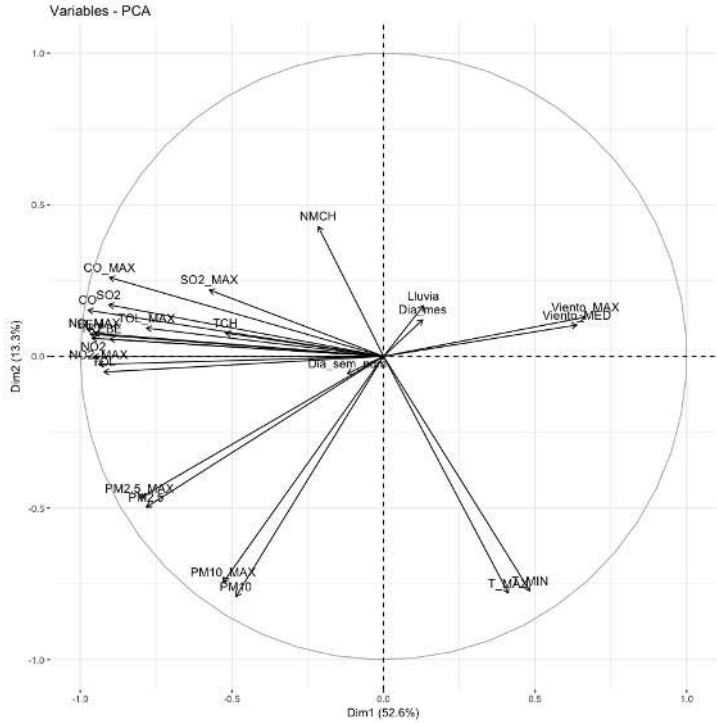


Fig. 7.46. Biplot de los atributos.

En el *biplot* se han representado sólo los atributos y se aprecia los que están correlados entre sí (se encuentran próximos en ángulo) y la calidad de representación, su correlación y contribución con cada dimensión (están representados los dos primeros componentes principales). Así, las concentraciones de CO, NO, benceno, SO2, etc. (segundo cuadrante) están correladas inversamente con el primer componente principal con el viento. Para el segundo componente principal, las variables que más contribuyen son las concentraciones de partículas en suspensión y las temperaturas.

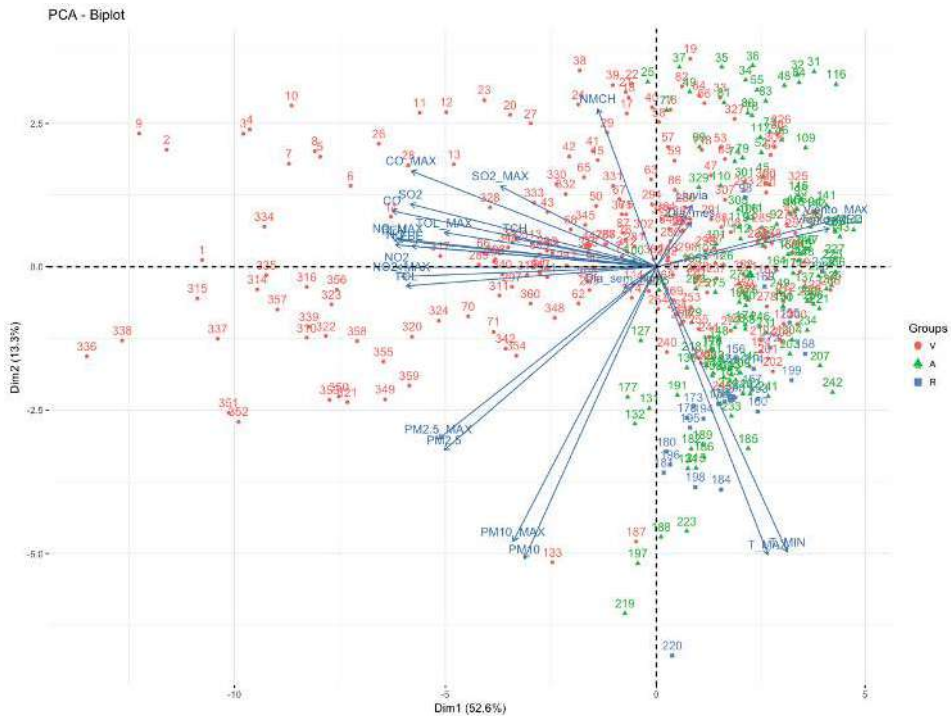


Fig. 7.48. Biplot con atributos, instancias y en código de color las clases.

Se aprecia un gran solapamiento entre las clases, mucho entre las clases A y R y en menor medida con la clase V.

Representando elipses que agrupan las diferentes clases se aprecia el gran solapamiento entre las clases A y R. Pero, efectivamente, cuando la separación entre clases es grande, el análisis de componentes principales puede servir para obtener modelos de clasificación y análisis de valores atípicos.

```
fviz_pca_ind(res.pca, label="none", habillage = df$O3_Nivel, addEllipses=TRUE, ellipse.level=0.90)
```

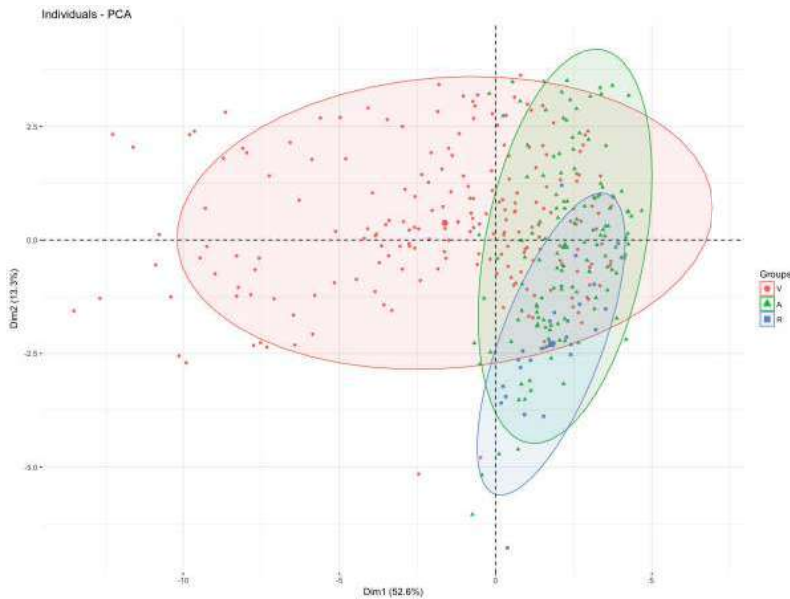


Fig. 7.49. Biplot con el agrupamiento de las clases representadas en componentes principales.

Es posible que, al aumentar el número de dimensiones, añadiendo los siguientes componentes principales, las clases puedan estar mejor separadas y tener un modelo efectivo de clasificación con los nuevos atributos sintetizados.

7.2.3 Árboles de decisión

Los árboles de decisión permiten realizar la clasificación de un conjunto de instancias siguiendo un conjunto de reglas situadas en los nodos de una estructura de datos de tipo árbol. Existen muchos algoritmos de clasificación basados en árboles de decisión, dependiendo del tipo de métrica que usan para la creación de los nodos. Así mismo hay una gran variedad de técnicas de ajuste para controlar la profundidad, podar, refinar los árboles. Se van a mostrar algunos de los algoritmos de esta clase más utilizados.

7.2.3.1 RPART (*Recursive Partitioning and Regression Trees*)

El algoritmo RPART divide de forma recursiva el conjunto de entrenamiento siguiendo una regla que involucra un atributo independiente, este proceso continúa con los conjuntos resultantes hasta que un criterio de terminación se cumple. La regla de división elegida es aquella que sigue el criterio de reducir, en mayor medida, la heterogeneidad del atributo dependiente del conjunto original. Este criterio toma el nombre de *impureza (impurity)*.

```
library(rpart)

modelo.rpart <- rpart(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX +
Viento_MED, data = setEntrena, method="class")

par(mar=c(0,0,0,0))

plot(modelo.rpart)

text(modelo.rpart, cex = 0.75)
```

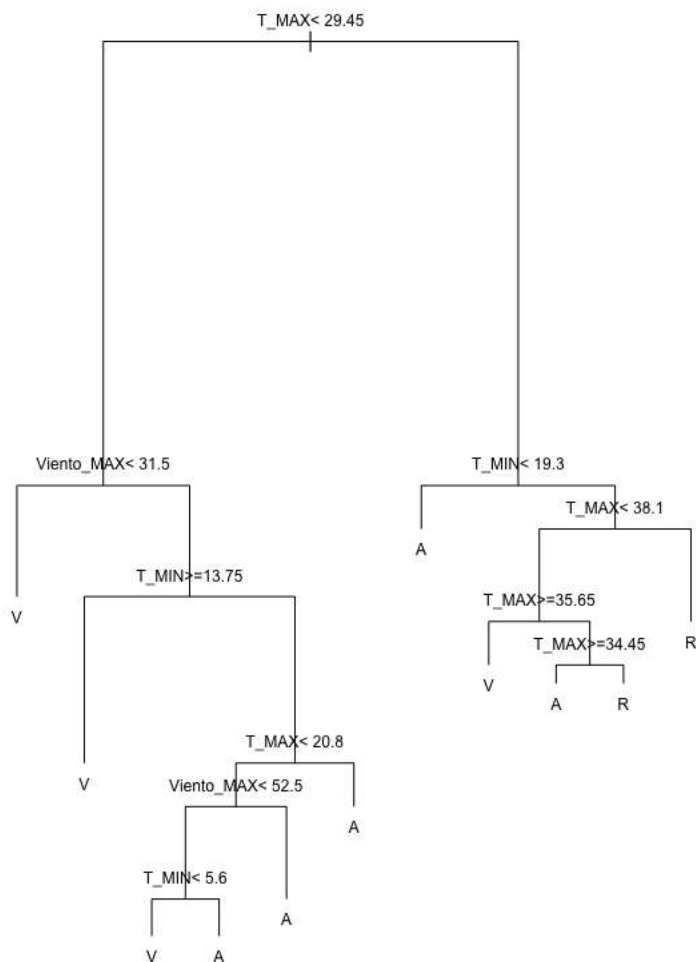


Fig. 7.50. Árbol de decisión obtenido con RPART.

En la figura 7.50 podemos ver el árbol de decisión obtenido. La rama de la derecha se sigue cuando la condición del nodo se cumple, en caso contrario, se sigue la rama izquierda. La ventaja de este tipo de clasificación es que es fácilmente interpretable, se pueden determinar qué reglas son las asociadas a cada clase.

De los algoritmos de clasificación basados en árboles de decisión también se extrae información adicional muy interesante. Por ejemplo, una medida de la importancia de los atributos.

```
modelo.rpart$variable.importance
```

T_MAX	T_MIN	Viento_MAX	Viento_MED	Lluvia	Lluvia_SN
38.335003	35.480712	19.558571	16.099196	4.031800	3.208305

Se aprecia que los atributos de lluvia son muy inferiores al resto. Efectivamente, si repasamos el árbol de decisión, estos dos atributos, junto con Viento_Med, no aparecen como regla en ningún nodo. Lo que sugiere que debería reentrenarse el modelo sin la presencia de estos atributos, dado que no se utilizan.

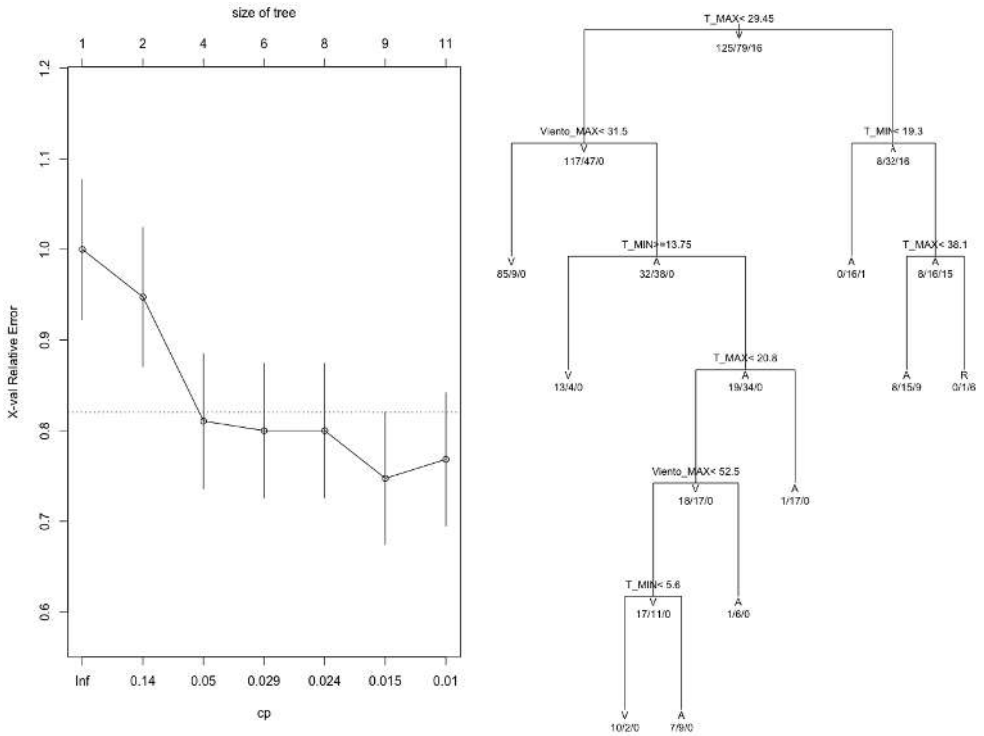


Fig. 7.51. Árbol de decisión optimizado.

Es conveniente hacer la poda del árbol si no se realiza de forma automática. En este caso, se va a seleccionar el árbol que minimiza el error mediante *cross validation*.

```
plotcp(modelo.rpart)

poda.modelo.rpart <- prune(modelo.rpart, cp = modelo.rpart$cptable[ which.min( modelo.
rpart$cptable[, "xerror" ] ), "CP" ])

par(mar=c(0,0,0,0))

plot(poda.modelo.rpart, uniform=TRUE)

text(poda.modelo.rpart, use.n=TRUE, all=TRUE, cex=.8)
```

Comparado con el primer árbol, ahora el podado tiene dos nodos finales menos. Guardamos el valor de *accuracy* para compararlo con el resto de clasificadores.

```
predice.modelo.rpart <- predict(poda.modelo.rpart, setValida, type="class")

c <- confusionMatrix(predice.modelo.rpart, setValida$O3_Nivel)

compara_ACC <- rbind(compara_ACC, c$overall[1])

row.names(compara_ACC)[nrow(compara_ACC)] <- "RPART"
```

7.2.3.2 Árboles de inferencia condicional, CTREE

La diferencia que existe entre CTREE y RPART es la forma de determinar el conjunto de atributos para realizar la división de un nodo. Mientras que RPART toma una medida de impureza, CTREE aplica un test de permutaciones para tomar la decisión.

```
library(party)

modelo.ctree <- ctree(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX +
Viento_MED, data = setEntrena)

plot(modelo.ctree)
```

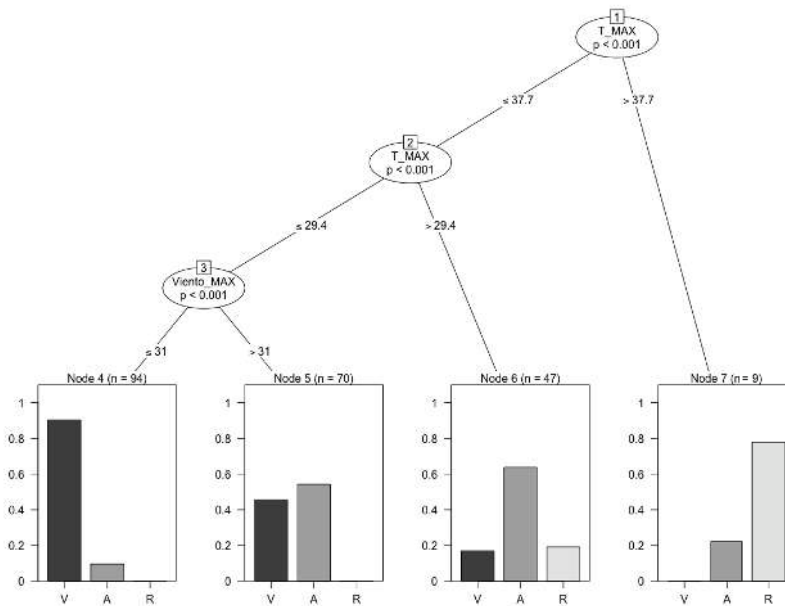


Fig. 7.52. Árbol de decisión obtenido con CTREE.

El árbol de decisión obtenido con CTREE es mucho más compacto que con RPART, sólo cuatro nodos finales. En la representación gráfica se aprecia que en los nodos finales se representa la probabilidad de pertenencia a cada clase. RPART también provee esta salida, pero en la representación gráfica se muestra la clase con la mayor probabilidad.

```
predice.modelo.ctree <- predict(modelo.ctree, setValida)

c <- confusionMatrix(predice.modelo.ctree, setValida$O3_Nivel)
compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "CTREE"
```

7.2.3.3 C5.0

Es una extensión del algoritmo C4.5, uno de los algoritmos más conocidos y usados en aprendizaje automático. La diferencia con los anteriores reside en que utiliza la medida de ganancia de información para decidir el atributo que divide el conjunto de instancias de entrenamiento.

```
library(C50)

modelo.C50 <- C5.0(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Vien-
to_MED, data = setEntrena)

library(partykit)

plot(modelo.C50)

predice.modelo.C50 <- predict(modelo.C50, setValida)

c <- confusionMatrix(predice.modelo.C50, setValida$O3_Nivel)

compara_ACC <- rbind(compara_ACC, c$overall[1])

row.names(compara_ACC)[nrow(compara_ACC)] <- "C50"
```

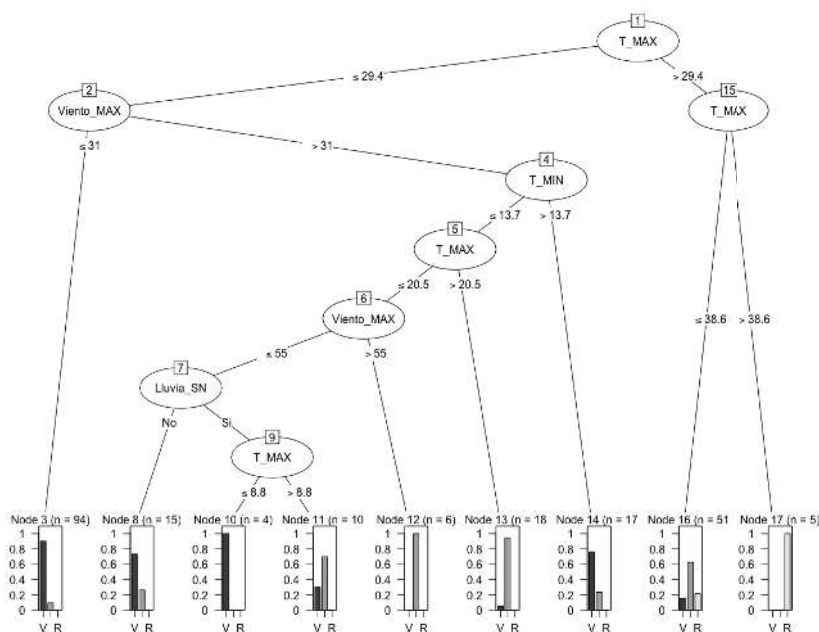


Fig. 7.53. Árbol de decisión obtenido con C5.0.

La librería C5.0 tiene unas cuantas modificaciones interesantes. Por ejemplo, se pueden obtener, en lugar de un árbol de decisión, reglas.

```
summary(C5.0(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED,
data = setEntrena, rules = TRUE))
```

```
Class specified by attribute 'outcome'
Read 220 cases (7 attributes) from undefined.data
Rules:
```

```

Rule 1: (69/6, lift 1.6)
  Lluvia_SN = No
  T_MAX <= 20.5
  Viento_MAX <= 55
  -> class V [0.901]

Rule 2: (94/9, lift 1.6)
  T_MAX <= 29.4
  Viento_MAX <= 31
  -> class V [0.896]

Rule 3: (5, lift 1.5)
  Lluvia_SN = Si
  T_MAX <= 8.8
  -> class V [0.857]

Rule 4: (31/6, lift 1.4)
  T_MAX <= 29.4
  T_MIN > 13.7
  -> class V [0.788]

Rule 5: (19/1, lift 2.5)
  T_MAX > 20.5
  T_MIN <= 13.7
  Viento_MAX > 31
  -> class A [0.905]

Rule 6: (6, lift 2.4)
  T_MIN <= 13.7
  Viento_MAX > 55
  -> class A [0.875]

Rule 7: (15/3, lift 2.1)
  Lluvia_SN = Si
  T_MAX > 8.8
  T_MIN <= 13.7
  Viento_MAX > 31
  -> class A [0.765]

Rule 8: (51/19, lift 1.7)
  T_MAX > 29.4
  T_MAX <= 38.6
  -> class A [0.623]

Rule 9: (5, lift 11.8)
  T_MAX > 38.6
  -> class R [0.857]

Default class: V

```

La clasificación que arroja el sistema de reglas es exactamente la misma que la obtenida con el árbol. Siempre puede convertirse un árbol en un sistema de reglas. La elección de uno u otro formato debe realizarse por cuestiones de claridad en la interpretación de los resultados o por facilidad en la implementación del modelo.

La otra extensión consiste en incorporar el mecanismo de *boosting* para crear el clasificador.

El *boosting* consiste en generar un conjunto de árboles de decisión en lugar de uno solo para realizar la clasificación. La elección de la clase se realiza por votación (pueden aplicarse diferentes criterios de votación). En la sección dedicada a esta técnica se detallará la forma de calcular el conjunto de árboles.

```
modelo.C50boost <- C5.0(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX +
Viento_MED, data = setEntrena, trials = 10) #Se especifican 10 iteraciones de "boosting"

predice.modelo.C50boost <- predict(modelo.C50boost, setValida)

c <- confusionMatrix(predice.modelo.C50boost, setValida$O3_Nivel)

compara_ACC <- rbind(compara_ACC, c$overall[1])

row.names(compara_ACC)[nrow(compara_ACC)] <- "C50 boost"
```

Ahora se crean diez árboles (*trials*) y se establece con ellos una votación para determinar la clase a la que pertenece una instancia.

7.2.4 Metaalgoritmos

Con C5.0 se ha introducido la técnica de *boosting*; ésta junto con *bagging* y *stacking* conforman un conjunto de metaalgoritmos. Un metaalgoritmo es un algoritmo que aprovecha el resultado conjunto de otros algoritmos para mejorar la capacidad que tendrían estos últimos de forma individual. Todos se aplican en dos etapas, en la primera se obtiene un conjunto de modelos de predicción sobre el conjunto de datos de entrenamiento y una segunda etapa de agregación en la que se combinan para obtener un modelo. En el *boosting* clásico, se crea un modelo de predicción sobre todo el conjunto de entrenamiento, las instancias mal clasificadas se toman como conjunto de entrenamiento para crear el siguiente modelo de predicción (siguiente *trial*). De esta forma se crea un conjunto de modelos que especializan su comportamiento sobre un subconjunto específico de los datos. Al final, además se obtiene la importancia relativa de los atributos independientes y la posibilidad de determinar datos atípicos que escapan a cualquier modelización. Una variante del *boosting* consiste en elegir aleatoriamente el conjunto de instancias sobre las que especializar un modelo, curiosamente, esta estrategia funciona igual o mejor que la variante clásica.

Mientras que en *boosting* se busca mejorar la capacidad predictiva, en *bagging* (*Bootstrap AGGREGatING*) se busca reducir la varianza con el fin de evitar el sobreaprendizaje. Consiste en generar nuevos conjuntos de entrenamiento por muestreo uniforme con reemplazo y construir un modelo de predicción para cada uno. El caso de más éxito de

este tipo de algoritmo es el *random forest*.

El menos usado de los tres metaalgoritmos es el *stacking*. Si bien *bagging* y *boosting* se aplican para un conjunto homogéneo de modelos de clasificación, *stacking* se aplica principalmente sobre conjuntos heterogéneos. Las predicciones que realizan los clasificadores de la primera etapa se usan como valores de entrada para el metaalgoritmo. Éste deberá aprender a determinar, para un dato o conjunto de datos, qué clasificadores son los que tendrán mejor comportamiento. Un algoritmo de *stacking* podría tener como clasificadores de la primera etapa algoritmos de *bagging* y *boosting*.

El comportamiento de estos metaalgoritmos ha sido tan exitoso que actualmente se han convertido en el estándar para abordar problemas de clasificación. Muestran mejor comportamiento cuando el conjunto de datos es muy grande. Uno de los inconvenientes que presentan es que la interpretación del modelo resultante es extremadamente difícil.

7.2.4.1 AdaBoost (ADAPtative BOOSTing)

Tiene el mérito de ser el primer algoritmo de *boosting* con capacidad para adaptarse y que obtuvo buenos resultados, dando lugar posteriormente a toda una familia de algoritmos basados en él.

Para que la visualización de resultados no sea muy grande, se va a elegir una cantidad pequeña de *trials*.

```
library(adabag)

modelo.adaboost <- boosting(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX +
  T_MIN + Viento_MAX + Viento_MED, data = setEntrena,
  boos = TRUE, mfinal = 10, #10 árboles
  coeflearn = "Breiman")

modelo.adaboost$weights      #el peso de cada clasificador
modelo.adaboost$importance   #la importancia de cada atributo
plot(errorrevol(modelo.adaboost, setEntrena)$error, ylab = "Error")
library(tree)
plot(modelo.adaboost$trees[[1]])
text(modelo.adaboost$trees[[1]], pretty=0)

predice.modelo.adaboost <- predict(modelo.adaboost, setValida)
c <- confusionMatrix(predice.modelo.adaboost$class, as.character(setValida$O3_Nivel))
compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "AdaBoost"
```

```
[1] 0.5863601 0.4447875 0.5277013 0.2819733 0.3684966 0.3702397 0.3555507 0.3590060
[9] 0.3238859 0.4462252

Lluvia  Lluvia_SN      T_MAX      T_MIN Viento_MAX Viento_MED
1.3753436 0.8789779 46.5511357 21.1124428 22.7149328 7.3671671
```

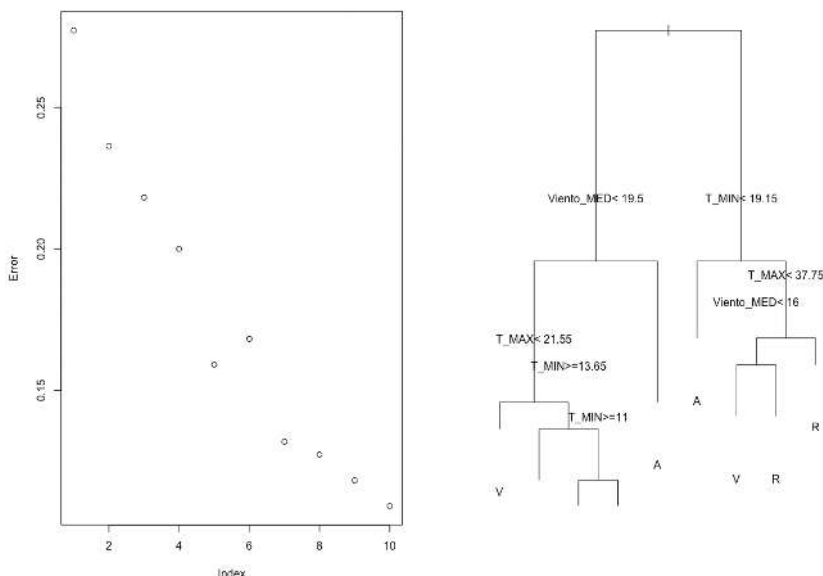


Fig. 7.54. Evolución del error con los *trials* y representación del árbol del primer *trial*.

Se muestra el peso que tiene cada clasificador en el conjunto final y la importancia de los atributos para realizar la clasificación. En las gráficas se ha representado la evolución del error de clasificación con el número de *trials* y el árbol del primer *trial*. De la gráfica de la evolución del error se puede apreciar que sería conveniente aumentar el número de *trials* hasta asegurar que el error converge.

7.2.4.2 GBM (*Gradient Boosting Machine*)

GBM construye el conjunto de predictores tratando de minimizar una función de pérdida, para lo cual realiza un descenso de gradiente.

Uno de los problemas importantes de todas las técnicas que utilizan la agregación de modelos es el sobreajuste. Es decir, el aprendizaje debe pararse en el momento en que ya no puede generalizarse la información y lo que se produce es aprendizaje de los datos concretos con los que se entrena. Para evitar esta situación es conveniente utilizar validación cruzada o utilizar la medida del error de entrenamiento y validación con el fin de parar el aprendizaje cuando el error de entrenamiento es inferior al de

validación o el error de validación alcanza un mínimo y comienza a aumentar. En este caso se va a utilizar una fracción del 70 % del conjunto de entrenamiento y el restante 30 % para validación. Se representarán los errores para determinar cuándo debe pararse el entrenamiento.

```
library(gbm)
set.seed(123)

modelo.gbm <- gbm(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX +
  Viento_MED, data = setEntrena, distribution = "multinomial", interaction.depth = 3,
  shrinkage = 0.01, n.trees = 500, train.fraction = 0.7)

n <- gbm.perf(modelo.gbm, plot.it = TRUE, oobag.curve = FALSE, overlay = TRUE,
  method="test")

predice.modelo.gbm <- predict(modelo.gbm, setValida, n.trees = n)#devuelve la proba-
  bilidad de pertenecer a cada clase

p <- apply(predice.modelo.gbm, 1, which.max)
c <- confusionMatrix(p, as.integer(setValida$O3_Nivel))

compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "GBM"
```

```
[1] 203
```

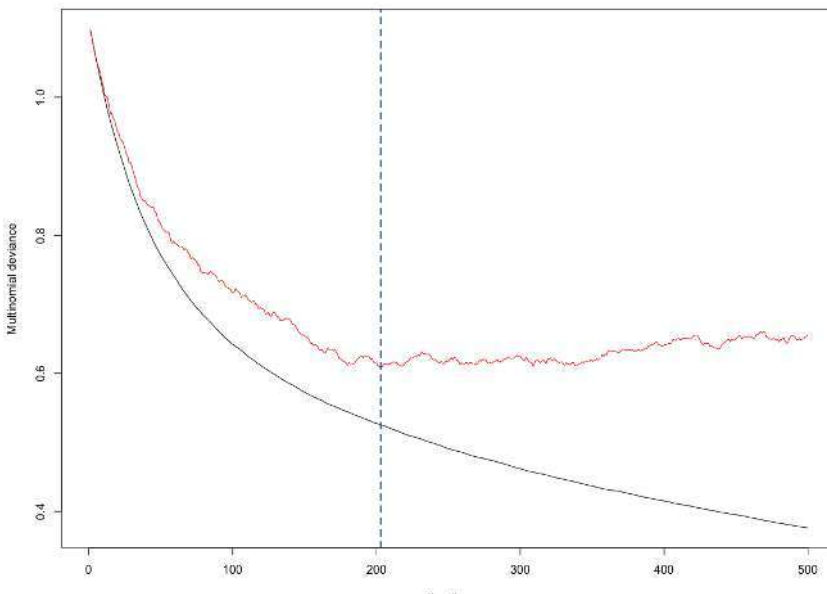


Fig. 7.55. Evolución del error de entrenamiento y validación con las iteraciones. Corte en azul del punto óptimo.

En la iteración 203 el error de validación tiene tendencia a crecer, por tanto, la predicción se realizará con el modelo obtenido en ese momento.

7.2.4.3 Random forest

Es la técnica más popular de algoritmos de ensamblado de modelos. Utiliza el procedimiento de *bagging* para construir el modelo final y puede usarse tanto para realizar regresión de valores continuos como para clasificación. De nuevo, es muy interesante el resto de información que puede extraerse, así como la importancia de los atributos.

```
library(randomForest)

set.seed(11)

modelo.RF <- randomForest(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED,
  data = setEntrena, ntree=200, importance=T)

plot(modelo.RF)

modelo.RF.leyenda <- if (is.null(modelo.RF$test$err.rate)) {colnames(modelo.RF$err.rate)} else {colnames(modelo.RF$test$err.rate)}

legend("top", cex =0.5, legend = modelo.RF.leyenda, lty=c(1,2,3,4), col=c(1,2,3,4), horiz=T)

varImpPlot(modelo.RF, sort = T, main="Importancia de los Atributos", n.var = 6)

predice.modelo.RF <- predict(modelo.RF, setValida)
c <- confusionMatrix(predice.modelo.RF, setValida$O3_Nivel)
compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "Random Forest"
```

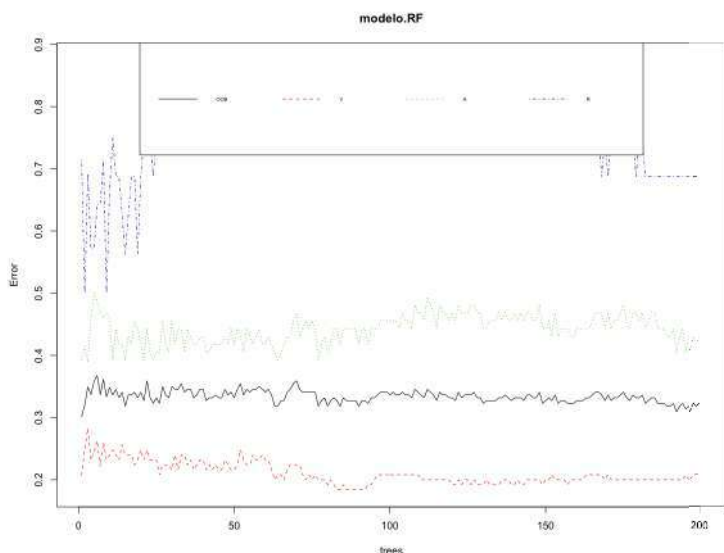


Fig. 7.56. Evolución del error para cada una de las clases aplicando *random forest*.

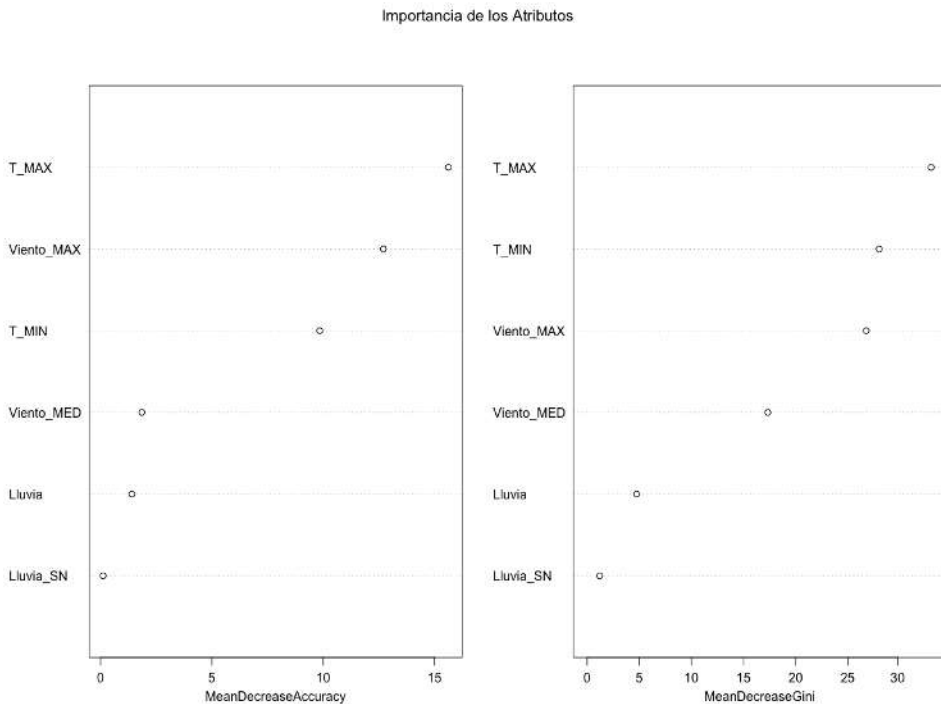


Fig. 7.57. Importancia de los atributos obtenida con *random forest*.

En la gráfica de la evolución del error (figura 7.56), en negro se muestra el error OOB (*Out-of-bag error*), es decir, el error cometido sobre las instancias que no han sido utilizadas para generar el árbol. En código de color, el error cometido sobre cada una de las clases. En el gráfico de la importancia de los atributos (figura 7.57) se muestra para dos medidas de mérito la importancia de cada atributo. Ambas medidas coinciden en la poca importancia para los atributos de lluvia y en asignar a T_MAX la máxima importancia. Como se ha visto previamente, esta información es útil para refinar los modelos reduciendo la dimensionalidad al descartar los atributos menos importantes.

7.2.5 SVM, máquinas de vectores de soporte

Terminada una panorámica por los métodos de clasificación por agregación de modelos, se van a presentar para finalizar este capítulo dos técnicas bastante populares y con un planteamiento ligeramente distinto de las anteriores.

Las máquinas de soporte vectorial (*support vector machine*) buscan encontrar el hiperplano que hace separable dos clases. Para ello, se “proyectan” las instancias de entrenamiento en un espacio de mayor dimensión en el que existe una separación lineal entre las clases. La función que proyecta las instancias es el *kernel*.

De las técnicas tratadas en este capítulo, la más parecida conceptualmente a SVM es la regresión logística, presentándose como una alternativa a ésta cuando las clases no son linealmente separables. También SVM se muestra eficiente cuando se tratan problemas de muy alta dimensionalidad, como por ejemplo la clasificación de textos. El problema que tiene es que su entrenamiento es muy ineficiente y su aplicación debe descartarse cuando el conjunto de entrenamiento es muy grande.

Al igual que la regresión logística, SVM está diseñado para problemas de clasificación binaria. Sin embargo, se pueden aplicar dos estrategias para tratar el caso de clasificaciones con más de dos clases.

- **Todos contra todos.** Para cada par de clases se entrena un SVM. En el ejemplo que estamos tratando, significa que habría que construir un clasificador para discriminar la clase V de A, otro para V de R y un tercero para A de R.
- **Uno contra todos.** Se entrena un clasificador que discrimine entre una clase y el resto. Para el problema práctico que se está tratando implica hacer un clasificador para discriminar V de A+R, A de V+R y R de V+A.

Se va a utilizar la librería *liquidSVM* que permite la clasificación multiclase con SVM y atributos categóricos. Las funciones de la librería realizan internamente las transformaciones ahorrando mucho código al usuario.

```
library(liquidSVM)

set.seed(1)

modelo.SVM <- mcSVM(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX +
Viento_MED, setEntrena, mc_type="AvA_ls", max_gamma = 625)

predice.modelo.SVM <- predict(modelo.SVM, setValida)
c <- confusionMatrix(predice.modelo.SVM, setValida$O3_Nivel)
compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "SVM"
```

En este caso se ha aplicado una estrategia de todos contra todos (AvA).

Para finalizar se presenta un algoritmo con un planteamiento completamente diferente a los vistos hasta ahora.

7.2.6 K vecinos próximos. k-NN (*k-Nearest Neighbors*)

Hasta ahora, todos los métodos de clasificación tratados son paramétricos, es decir, se crea un modelo donde un conjunto de atributos independientes pesados se combinan por parámetros para obtener el atributo dependiente. En el caso anterior del SVM, los parámetros son los vectores de soporte y la distancia (márgenes) al hiperplano que separa las clases. El algoritmo k-NN pertenece a la categoría de técnicas de clasificación no-paramétrica. La identificación de una instancia a una clase se realiza determinando qué clase es la mayoritaria entre sus vecinos más próximos. Generalmente, para calcular los vecinos más cercanos se utiliza la distancia euclídea, aunque pueden usarse otras métricas o pesar de forma distinta los atributos en función de su importancia. Otro requisito es que todos los atributos (menos la clase) tienen que ser numéricos o atributos ordinales. Por tanto, vamos a convertir el atributo de Lluvia_SN a numérico.

```
setEntrenaNUM <- setEntrena
setEntrenaNUM$Lluvia_SN <- as.integer(setEntrenaNUM$Lluvia_SN) #de factores a números

setValidaNUM <- setValida
setValidaNUM$Lluvia_SN <- as.integer(setValidaNUM$Lluvia_SN)
```

Se va a utilizar la implementación de k-NN de la librería *caret*.

```
library(caret)
set.seed(10)

modelo.KNN <- train(O3_Nivel ~ Lluvia_SN + Lluvia + T_MAX + T_MIN + Viento_MAX + Viento_MED,
  data = setEntrenaNUM, method = "knn",
  trControl=trainControl(method = "repeatedcv", number = 8, repeats
= 4),
  preProcess = c("center", "scale"), tuneLength = 10)

plot(modelo.KNN)
```

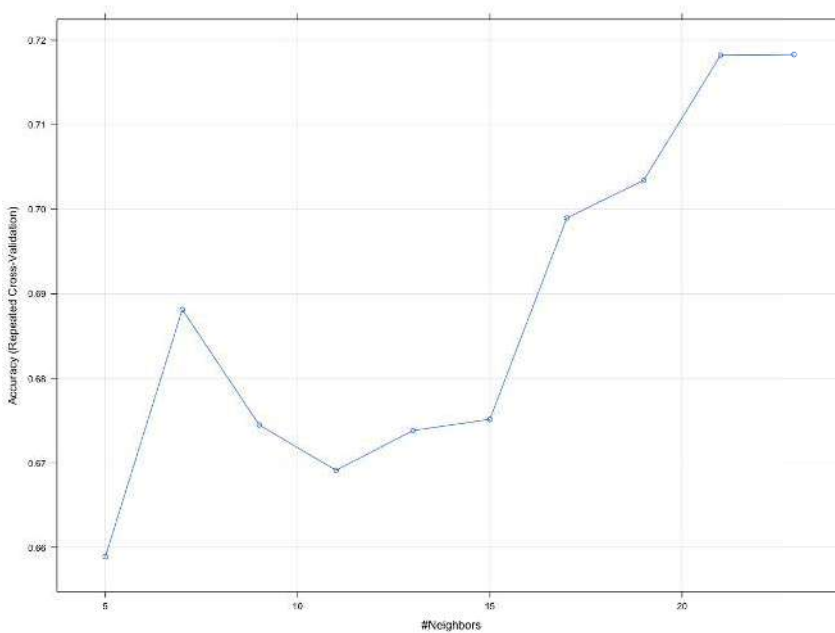


Fig. 7.58. Evolución de la precisión según el número de vecinos considerados.

Se elige aplicar validación cruzada de ocho partes. Se preprocesan las instancias para centrarlas y escalarlas. El motivo de este preprocesado es tener todos los atributos en la misma escala de valores, con valor medio cero, y puedan así ser comparables. Esta transformación se denomina *estandarización*. La gráfica se aprecia que la mejor precisión se obtiene con veintitún vecinos.

```
predice.modelo.KNN <- predict(modelo.KNN, setValidaNUM)
c <- confusionMatrix(predice.modelo.KNN, setValidaNUM$O3_Nivel)
compara_ACC <- rbind(compara_ACC, c$overall[1])
row.names(compara_ACC)[nrow(compara_ACC)] <- "KNN"
```

La ventaja de k-NN es que es sencillo tanto conceptualmente como en cuanto a implementación. El modelo no tiene que ser reentrenado al incorporar nuevas instancias, mientras que para las técnicas de clasificación paramétrica sí es necesario. Pero tiene el inconveniente de que la respuesta para determinar la clase a la que pertenece una instancia depende del tamaño del conjunto de entrenamiento, hay que calcular la distancia entre la nueva instancia y todas las presentes en el conjunto de entrenamiento. Por este motivo, en el contexto de *big data* se han desarrollado implementaciones paralelas de este algoritmo.

7.2.7 Redes de neuronas artificiales

En esta sección nos centraremos en las capacidades de clasificación de las redes de neuronas artificiales (RNA) dentro de la plataforma R. La topología y algoritmos de aprendizaje relacionados con las RNA es muy amplio y nos daría para escribir un libro sólo con ellas, sin embargo, para completar las técnicas de clasificación nos centraremos en las RNA de varias capas con propagación hacia delante (*feedforward neural networks*). El objetivo de nuestro diseño es encontrar las funciones que relacionen las variables de nuestro espacio de entrada con las de la salida. Con las RNA vamos a poder conseguir modelos de regresión no lineales, conociéndose estos modelos como los *aproximadores universales*.

Desde R tenemos distintas opciones para poder diseñar RNA multicapa. Algunas de ellas son las siguientes:

- **nnet.** Se trata de una librería que permite diseñar y validar modelos de RNA multicapa. Posee ciertas desventajas en cuanto a las capacidades de representación, aunque es sencillo de trabajar.
- **neuralnet.** Es otra librería que implementa el algoritmo de retropropagación del gradiente para el diseño de RNA multicapa.
- **RSNNS.** Se trata de la librería que implementa la interfaz del famoso paquete SNNS (*Stuttgart Neural Network Simulator*) que contempla una amplia variedad de modelos de RNA.

Por otro lado, han ido surgiendo nuevos algoritmos de aprendizaje y topologías de RNA basados en un gran número de capas ocultas que permiten crear distintos grados de abstracción respecto a las variables del espacio de entrada. A estas redes se las conoce como *aprendizaje profundo* (*deep learning*). Estos nuevos modelos se han visto favorecidos por las mejoras en la capacidad de cómputo, ya que el proceso de aprendizaje de estas redes necesita de importantes recursos computacionales. Igualmente, existen distintas topologías de redes profundas las cuales están siendo usadas para diferentes propósitos (por ejemplo, redes de neuronas convolucionales para el análisis de imágenes, o las redes de neuronas recurrentes, de uso en el análisis de series temporales y modelado de lenguaje natural). En esta sección nos centraremos en la resolución de problemas de clasificación con RNA multicapa, y para ello, introducimos una librería que permitirá trabajar con RNA profundas y con muestras de datos de gran tamaño (a tenor de la corriente *big data* a las que se enfrentan estas tecnologías).

La librería seleccionada es *h2o* (*fast scalable machine learning*, “aprendizaje automático fácilmente escalable”). *h2o* permite diseñar modelos de aprendizaje automático basándose en cómputo paralelo y almacenamiento eficiente, que nos permitirá manipular conjuntos de datos de tal tamaño que R no podría soportar. *h2o* se ejecuta en un servidor compuesto por el número de clústeres que definamos y, aunque tengamos la

sensación de estar trabajando desde R, tanto el cómputo como el almacenamiento se llevan a cabo en la instancia *h2o*. De esta manera, tenemos que elegir un servidor desde donde queremos que *h2o* se ejecute y nuestra memoria estará limitada a la memoria que posea dicha instancia (obviamente, *h2o* lo podremos instalar en local en nuestra máquina, ateniéndonos a los recursos que ella tenga).

El primer paso, será inicializar la instancia *h2o*.

```
## Instrucciones de intalación http://h2o.ai/download
library(h2o)

# Podemos personalizar parámetros como los cores y la memoria asignada
# En este caso, permitimos el uso de todos los cores del clúster y una memoria de
2 GigaBytes
h2o.init(nthreads=-1, max_mem_size="2G")
h2o.removeAll() ## se elimina cualquier fichero que pudiera existir
```

La función que implementa el algoritmo de aprendizaje en *h2o* posee un gran número de parámetros. Por suerte, el método fue diseñado para que fuera sencillo el modelado de RNA. *h2o* aporta mecanismos que permiten la parada temprana (se comienza a entrenar con pesos pequeños y se para antes de sobreaprender), técnicas de regularización (se penalizan pesos grandes en función de sus valores al cuadrado —L2— o absolutos —L1—), la gestión de variables categóricas y variables en las que no se posee valor para todas las instancias, así como tasas de aprendizaje adaptativas. Por una parte, es interesante que se puedan manejar este tipo de funcionalidades y, por otra, permite definir un modelo de RNA indicando sólo unos pocos parámetros (número y tamaño de las capas ocultas, número máximo de ciclos de aprendizaje, la función de activación o algunos métodos de regularización).

Siguiendo el ejemplo de estimación de nivel de ozono (*O3_nivel*), vamos a construir una RNA sencilla mediante:

```
## Lo primero es enviar los datos a la instancia h2o, para ello usamos las
# funciones de conversión
dfMedidas_CLUS.hex <- as.h2o(dfMedidas_CLUS)

response <- "O3_Nivel" #Es la variable a estimar que debe ser categórica

predictors <- c("Lluvia_SN", "Lluvia", "T_MAX", "T_MIN", "Viento_MAX", "Viento_MED")
# Son las variables de entrada al modelo (poseen formato numérico)
```

```

predictors

# Separamos el conjunto de datos en entrenamiento y validación
splits <- h2o.splitFrame(data=dfMedidas_CLUS.hex, ratios=0.66)
train <- h2o.assign(splits[[1]], "train.hex") # 66%
valid <- h2o.assign(splits[[2]], "valid.hex") # 44%

# En resumen, tenemos:
# *Una variable de salida (O3_nivel). Debe ser categórica
# * 6 variables de entrada. Deben ser numéricas
# * Hemos dividido el conjunto de datos en entrenamiento y validación (66-44)

# Invocamos el procedimiento de aprendizaje
m1 <- h2o.deeplearning(
  model_id="O3_1",
  training_frame=train,
  validation_frame=valid,
  x=predictors,
  y=response,
  hidden=c(50,50,50), ## Red de neuronas con tres capas de 50,50,50 neuronas
  epochs=1000000, ## Máximo de ciclos, que esperemos no llegar
  score_validation_samples=80, ## Para acelerar el aprendizaje, se escogen aleatoria-
mente 80 datos del conjunto de validación
  stopping_rounds=1, #Si durante 1 o más ciclos no se mejora el aprendizaje, se para
stopping_metric="misclassification", ## puede ser "MSE","logloss","r2"
stopping_tolerance=0.01
)
summary(m1)
plot(m1)

```

En este ejemplo, hemos usado la técnica de parada temprana (*early stopping*) para prevenir sobreaprendizaje. El algoritmo de aprendizaje se parará automáticamente si el índice de clasificación errónea (*misclassification*) se estabiliza (es decir, que la media móvil del error de longitud 2 no mejora al menos un 1 %).

Es posible descubrir el modelo aprendido mediante el comando "summary(m1)". *h2o* posee una interfaz web que nos permitirá tener una versión gráfica del modelo entrenado. Esa información se puede consultar en el ordenador que posee la instancia de *h2o* (si es el mismo con el que estamos trabajando, pondremos en el navegador "http://localhost:54321/").

h2o nos indica la siguiente información relevante, entre otras:

- La importancia de las variables en el modelo de clasificación.

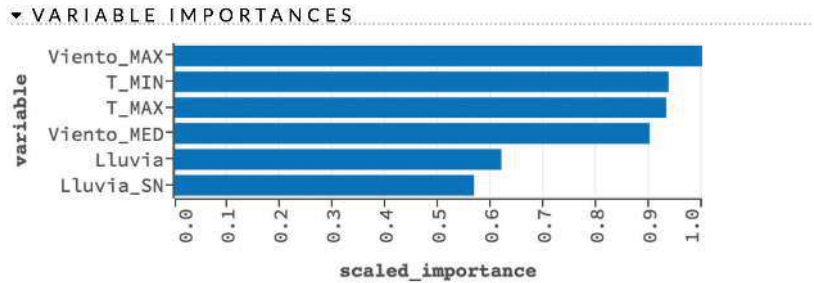


Fig. 7.59. Importancia de las variables en el modelo de clasificación.

- Matriz de confusión con los datos de validación.

▼ VALIDATION METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS

	A	R	V	Error	Rate
A	24	4	8	0.3333	12 / 36
R	2	0	2	1.0	4 / 4
V	10	0	32	0.2381	10 / 42
Total	36	4	42	0.3171	26 / 82

Fig. 7.60. Matriz de confusión con los datos de validación.

- Métricas de la predicción con datos de validación.

▼ OUTPUT - VALIDATION_METRICS

model	03_1
model_checksum	5493350785660442624
frame	•
frame_checksum	0
description	Metrics reported on temporary validation frame with 82 samples
model_category	Multinomial
scoring_time	1520935761902
predictions	•
MSE	0.303252
RMSE	0.550683
nobs	82
custom_metric_name	•
custom_metric_value	0
r2	0.679392
logloss	3.453797
mean_per_class_error	0.523810

Fig. 7.61. Métricas de la predicción con datos de validación.

Un tema interesante que se puede trabajar para mejorar las predicciones con estos modelos es el análisis de la salida de la RNA. En nuestro caso, la red tiene como salida los valores categóricos de la variable O3_nivel. Vamos a obtener los valores de predicción con muestras con las que no ha aprendido.

```
## Instrucciones de intalación http://h2o.ai/download
pred <- h2o.predict(m1, valid)
pred
```

Tendremos el resultado de la predicción para cada uno de los valores de las variables categóricas:

predict	A	R	V
1	V 6.055265e-06	2.749277e-10	0.9999939
2	A 5.341888e-01	7.177879e-06	0.46580400

En la primera muestra, claramente se trata de una predicción de nivel verde (V), ya que se encuentra cercano a 1 (0.9999939). En la segunda se establece nivel amarillo (A) porque es la respuesta mayor, pero tampoco está claro (5.341888e-01). De todas formas, se puede comprobar en la matriz de confusión que el valor de desempeño del modelo es de un 0,776.

El mayor problema que poseen las RNA es encontrar los parámetros de configuración y aprendizaje que se adapten al problema que se quiere resolver. En este caso, nosotros lo hemos realizado manualmente. *h2o* incorpora funcionalidades para hacer una búsqueda de estos parámetros mediante una búsqueda *grid*. Esta funcionalidad se escapa del objeto de este libro y se invita al lector a explorar esta funcionalidad en la documentación de *h2o*. Igualmente, se ha modelizado el problema de estimación de niveles de ozono para seguir una línea argumental con los distintos clasificadores. Para explotar las capacidades de las RNA con *h2o* se anima al lector a modelizar problemas de clasificación con un número de muestras mayor (aproximándose al concepto de *big data*).

Para finalizar, sólo resta comparar los resultados de los algoritmos. Es una tarea muy difícil determinar cuál de ellos es el que tiene mejor desempeño, se tienen varias medidas y la situación normal es que no halla un tipo de clasificador que posea el mejor comportamiento en todas ellas. Para el ejemplo que estamos desarrollando, sólo hemos guardado el valor de precisión, la suma de todos los aciertos divididos entre todos los casos, pero, dependiendo del problema, podría ser más importante disminuir el error de falso negativo o la sensibilidad o aumentar la precisión para una determinada clase en detrimento de las otras, etc.

En el gráfico de la figura 7.62 se muestra el valor ordenado del desempeño de los clasificadores, se aprecia que los basados en *boosting* y *bagging* están en las primeras posiciones.

Estos resultados son de carácter didáctico, hay que considerar que no se ha realizado un ajuste minucioso de los parámetros de los clasificadores que podrían mejorar el desempeño y, por tanto, modificar su posición en el *ranking*.

```
barplot(sort(compara_ACC[,1]), names.arg = rownames(compara_ACC) [order (compara_
ACC[,1])], las=2, ylim = c(0,1))

text(x=(0:11)*1.2+0.6, y = 0.02 + compara_ACC[order(compara_ACC[,1]),],round(compara_
ACC[order(compara_ACC[,1]),],2))
```

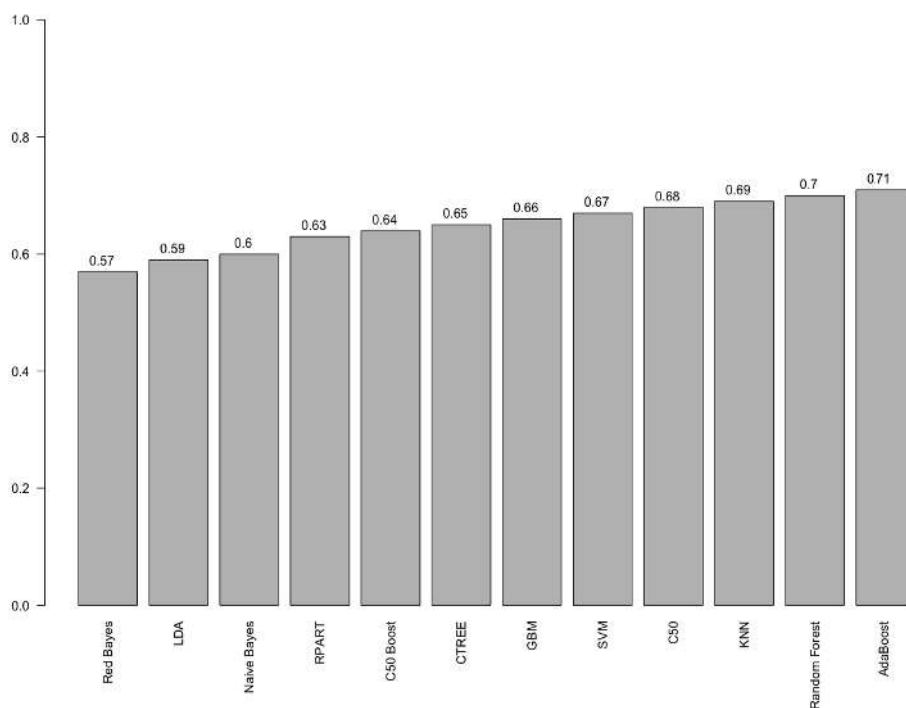


Fig. 7.62. Comparación de las diferentes técnicas de clasificación aplicadas.

Internet de las cosas y análisis de series temporales

CAPÍTULO 8

8.1 Internet de las cosas

El Internet de las cosas (IoT, *Internet of Things*) está generalmente conceptualizado como el hecho de conectar cosas a Internet y utilizar esta conexión para proporcionar algún tipo de utilidad, como monitorización remota o control sobre los dispositivos (Weber, 2010). Se trata básicamente de un cambio de nombre del mercado M2M (*Machine to Machine*) existente [KIM14], o inclusive una evolución de las redes de sensores y actuadores inalámbricos (WSAN) [ALEM10]. Aunque la idea subyacente consiste en permitir la creación de una red inteligente, invisible, que puede ser monitorizada, controlada y programada. Las tecnologías y los productos IoT, por tanto, tienen que permitir la comunicación entre ellos e Internet, ya sea directa o indirectamente [GUBB13].

En la década de los noventa, la conectividad a Internet comenzó a proliferar en los entornos profesionales y del consumidor, pero estaba aún limitada por el poco rendimiento de las redes de datos. En el 2000, la conexión a Internet se convirtió en la norma para multitud de aplicaciones, y hoy en día forma parte de multitud de empresas, industrias y productos de consumo para proporcionar acceso a la información. Sin embargo, estos dispositivos aún requieren de interacción humana y monitorización a través de aplicaciones e interfaces. La verdadera idea del IoT consiste en que la tecnología invisible trabaje en un segundo plano con el fin de responder dinámicamente para que las “cosas” actúen como nosotros queremos.

Hasta la fecha, en el mundo podemos encontrar alrededor de cinco mil millones de dispositivos inteligentes conectados, mientras que las predicciones hablan de cincuenta mil millones conectados para 2020. Estos números a gran escala introducen numerosos retos tecnológicos a nivel de infraestructura que necesitan ser desarrollados [ATZO10].

Dentro de estos retos podemos encontrarnos problemas diferentes:

- **Monitorización de un entorno complejo.** La monitorización de nuestro entorno supone un gran reto dada la cantidad de variables que podemos llegar a monitorizar. Entre ellas podríamos destacar la temperatura, humedad, presencia, localización de las personas, incidencia de rayos uva, calidad del aire, detección de gases peligrosos, presión, proximidad, aceleración, etc. [SURY12]. Esto supone tener que seleccionar correctamente los sensores que son necesarios para llevar a cabo las diferentes tareas inteligentes, así como su integración en el entorno de forma transparente para el usuario (computación ubicua). En algunos casos, como en los dispositivos *wearables* [PATE12], estos sensores tienen que ser lo suficientemente pequeños y quedar bien integrados para que las dimensiones no supongan un impedimento.
- **Conectividad.** Hoy día podemos encontrar numerosas formas de conexión a Internet, como las conocidas redes inalámbricas (WiFi, Bluetooth 4.0, 6LoWPAN), redes móviles (4G LTE), redes cableadas (Ethernet, fibra óptica), etc. (Florez, Jorge *et al.* 2012). Esto nos posibilita la conexión a Internet de los sensores inteligentes de múltiples formas, pero a su vez plantea retos tecnológicos en las infraestructuras que las soportan, como los sistemas de computación en la nube [GUBB13]. Esto supone trabajar con la optimización de los protocolos de red para trabajar con sensores con recursos limitados, enfrentarse a la escalabilidad de las arquitecturas de computación para soportar la conexión concurrente de millones de dispositivos y la disponibilidad de acceso permanente. Además, es necesario pensar y estandarizar nuevas formas de autodescubrimiento de los sensores dentro de nuestro entorno en caso de que no necesitemos dependencia con la nube.
- **Autonomía.** Si bien contamos con numerosos sensores y actuadores que se pueden integrar en un entorno IoT inteligente, aún se plantean numerosos retos en cuanto a la autonomía de los sensores en determinados entornos, como los *wearables*. Los dispositivos que se integran en las personas o que tienen cierta capacidad de movilidad normalmente se alimentarán a través de baterías, lo que les confiere un limitado tiempo de uso. De modo que es necesario también a este nivel de la arquitectura ser capaces de optimizar todo lo relacionado con el consumo de los dispositivos, lo que vendrá determinado principalmente por la cantidad de información que tienen que procesar o fuentes de datos que es necesario censar, así como la cantidad de información que tienen que transmitir [PANT13]. Aquí tiene cabida la optimización de los protocolos de comunicación, ajuste de los periodos de monitorización, configuración de las capacidades de ahorro de energía de los sensores, etc. [SUDE11].

- **Seguridad.** Los sensores desplegados por el entorno normalmente podrán estar monitorizando información de carácter personal, e inclusive gestionar automáticamente nuestros recursos y elementos cercanos, como los que podemos encontrar en el hogar o en la industria [GARC13]. Es de vital importancia que toda la infraestructura IoT sea segura en términos de transferencia de la información, y que además asegure la anonimidad de los datos transferidos a la nube en caso de producirse una brecha de seguridad [HUI13].

Actualmente existen numerosas plataformas que tratan de satisfacer la gran demanda de dispositivos que requieren estar conectados, así como intentar solucionar los retos anteriormente mencionados. Son muchos los fabricantes globales que están apostando por sus propias plataformas de integración de entornos IoT. Desde Facebook con su plataforma Parse, Samsung con su tecnología ARTIK, ARM con ARM Mbed, a otras plataformas incipientes como Golgi, Temboo, Xively, Carriots, etc. Todas estas plataformas son de origen propietario y cerrado, es decir, será difícil contar con ellas fuera de un entorno empresarial y de pago, y con dispositivos y sensores no homologados para su plataforma. En este aspecto podemos encontrar otras iniciativas de código fuente abierto como Thinger.io, Kaa, ThingSpeak, Particle y OpenRemote, que dada su naturaleza son más propensas para su utilización y adaptación a entornos académicos y científicos. Por tanto, en el desarrollo de nuevos proyectos IoT habrá que tener en cuenta estas plataformas, o bien plantear alternativas más eficientes y compatibles con un mayor número de sensores y dispositivos.

Las posibilidades que brindan los sistemas IoT son numerosas, y se aplican en múltiples campos, desde las *smart homes* (casas conectadas), *smart cities* (ciudades conectadas), *connected car*, monitorización ambiental y, en general, cualquier sistema que sea necesario monitorizar, como puede ser la monitorización del consumo de combustible, procesos de cultivo, de condiciones de almacenamiento, de polución, sistemas de regadío, etc. También tienen relevancia los recientes desarrollos relacionados con la monitorización de pacientes crónicos, personas dependientes, etc. (*healthcare*), la agricultura, los sistemas de generación de energía, etc.

8.2 Thinger.io IoT

El objetivo de esta sección será proporcionar un ejemplo práctico de adquisición de determinada información meteorológica a través de una solución de *Internet of Things*. La solución escogida para este caso práctico será la plataforma Thinger.io, ya que proporciona tanto el entorno tecnológico de conectividad de los sensores, así como determinados dispositivos *hardware* que nos permiten simplificar la tarea de integración y conectividad de los sensores utilizando entornos de desarrollo conocidos como Arduino.

8.2.1 Hardware

Para el desarrollo de este caso práctico, se utilizará un dispositivo *hardware* de la plataforma Thinger.io, llamado ClimaStick, que integra tanto los sensores como la conectividad a Internet, lo que nos permitirá recolectar información meteorológica a lo largo del tiempo sin tener que tener conocimientos de electrónica, buses de datos, sensores, etc. El dispositivo está basado en el microcontrolador Espressif ESP8266, que cuenta con un Cortex-M0 a 80 Mhz, 4 MB de *flash*, y 160 KB de RAM. Esta placa también cuenta con conectividad wifi 802.11 b/g/n para su conexión a Internet, así como interfaces de puerto serie a través de USB que permite su programación a través del entorno de desarrollo de Arduino. La placa de desarrollo tiene unas pequeñas dimensiones de 37x17x5 mm. A continuación, se proporcionan ilustraciones del dispositivo *hardware* a emplear:

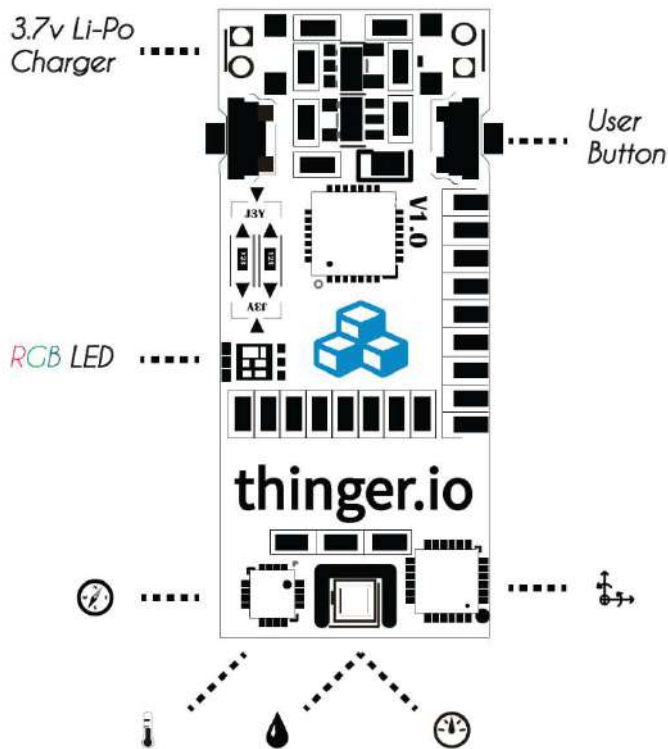


Fig. 8.1. Interfaz Thinger.io. Interfaces con sensores y alimentación

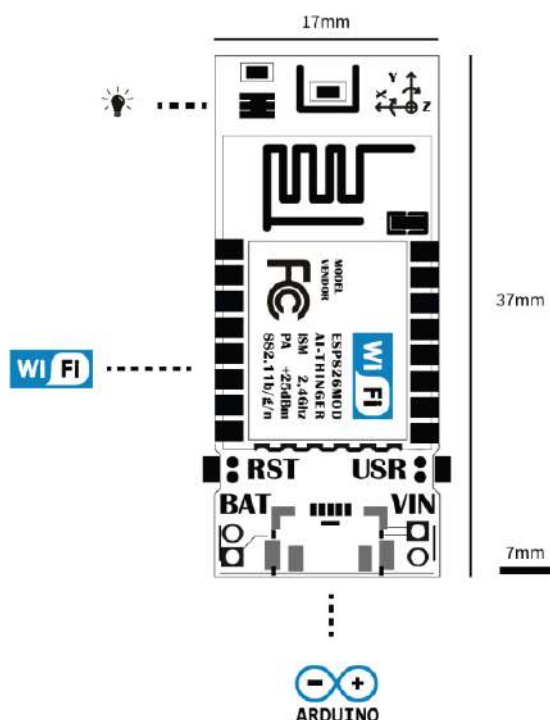


Fig. 8.2. Interfaz Thingier.io. Conectividad y extensiones

Esta placa integra sensores que permiten medir la temperatura, la humedad relativa, la presión barométrica o la luminosidad. También integra otros sensores, como un magnetómetro, un acelerómetro y un giróscopo, que pueden ser útiles para otros casos prácticos donde sea necesario contar con sensores inerciales. Los sensores empleados por este dispositivo, así como sus rangos operativos, resolución y precisión, se especifican en la siguiente figura:

Feature	Spec					
	Sensor	Variable	Range	Resolution	Accuracy	Condition
Embedded Sensors (I2C)	BME280	Temperature	(-40, 85) °C	0.01 °C	(±0.5, ±1) °C	(0, 60) °C
		Relative humidity	(0, 100) %Rh	0.01 %Rh	(±1, ±3) %Rh	
		Barometric pressure	(300, 1100) hPa	0.18 hPa	(±0.1, ±1) hPa	
	TSL2561	Light	(0, 65535) Lux	0.01 Lux	±3.5 lux	(-30, 70) °C (0, 95) %Rh
	HMC5883	Magnetometer	(-8, 8) gauss	(0.73, 4.35) milli-gauss	(230, 1370) LSB/gauss	
Voltage sensor	MPU6050	Accelerometer	±16 g	2048 LSB/g	15 LSB/g	
		gyroscope	±2000°/sec	16.4°/sec	0.005dps/√Hz	
Buttons	ADC	VIN / BAT voltage	(0, 5) Vdc	0.01 Vdc (10 bit ADC)	0.1 Vdc	
Other	1 Reset button		Pulled-up to 3.3Vdc, ground when is push			
	1 Flash / User button (GPIO-0)		Pulled-up to 3.3Vdc, ground when is push			
Other	Full RGB programmable LED		Up to 1024 analogic stepping per color			
	Serial Port Tx LED					

Fig. 8.3. Sensores integrados en Thingier.io.

Nota: Además del *hardware* mencionado, es posible emplear cualquier otro microcontrolador con conectividad a Internet (NodeMCU, Wemos, Arduino MKR1000, Arduino+Ethernet, etc.), utilizando sensores similares a los utilizados por este dispositivo (BME280 y TSL2561), que pueden ser fácilmente conectados por I²C (*Inter-Integrated Circuit*) al microcontrolador, ya que la plataforma está pensada para permitir la conexión de prácticamente cualquier microcontrolador.

8.2.2 Configuración de la plataforma

Tras registrarnos en la plataforma Thinger.io, será necesario dar de alta dos recursos dentro del panel de administración. En este caso habrá que dar de alta un dispositivo, que será el que se conecte a la plataforma para proporcionar la información de los sensores (en nuestro caso el ClimaStick), y también habrá que dar de alta un almacén de datos que permita guardar la información a lo largo del tiempo. La gestión de estos recursos se encuentra documentada con mayor detalle en "<http://docs.thinger.io/console/>", aunque aquí se proporcionan los pasos básicos necesarios para el desarrollo del caso práctico.

Para crear el dispositivo, accederemos a la sección de "Devices" del menú lateral de la plataforma, desde la que podremos gestionar los dispositivos asociados a nuestra cuenta de usuario.



Fig. 8.4. Integración de dispositivos en Thinger.io.

Una vez estamos en esta sección, será necesario dar de alta el dispositivo pulsando el botón de "Add Device" de la lista de dispositivos, que nos abrirá una sección para registrar información básica del dispositivo, como su identificador, una descripción y sus credenciales de acceso. Es importante quedarse con la información del identificador y de las credenciales de acceso, porque luego será necesario para utilizarlo en el código que se programará en el dispositivo. Para este ejemplo práctico, el identificador del dispositivo es "climastick", y sus credenciales son "h4Pxg!a1aDEK", tal y como se puede observar en la siguiente captura de pantalla:

Device details

Device Id ⓘ

climastick

Device description ⓘ

Dispositivo para registrar información meteorológica

Device credentials ⓘ

h4Pzgl1a1aDEK

Generate Random Credential

✓ Add Device

Fig. 8.5. Registro de dispositivos conectados a Thingiverse.io.

Para crear el almacén de datos, de forma análoga, debemos ir a la sección correspondiente de la consola de administración, denominada “Data Buckets” en este caso.



Fig. 8.6. Almacén de datos en Thingiverse.io.

Una vez estamos en esta sección, será necesario dar de alta el almacén pulsando el botón de “Add Bucket” de la lista de almacenes, que nos abrirá una sección para configurar el almacén de información. Del mismo modo que con el dispositivo, será necesario asignar un identificador del almacén de datos, un nombre, una descripción, si lo queremos activar o la fuente de datos de donde cogerá la información. En este caso, vamos a crear un almacén de datos con el identificador “MeteoBucket”, que va a estar activado, y la fuente de datos va a ser “From Write Call”, es decir, el dispositivo va a iniciar la escritura de la información cuando lo considere necesario. A continuación, se puede ver un ejemplo de la configuración del almacén mencionado:



The image shows a web form titled "Bucket details" for configuring a data bucket in Thingiverse. The form contains the following fields and controls:

- Bucket id**: A text input field containing "MeteoBucket".
- Bucket name**: A text input field containing "Datos Meteorológicos".
- Bucket description**: A text input field containing "Almacén de datos para guardar la información meteorológica".
- Enabled**: A toggle switch that is currently turned on (green).
- Data Source**: A dropdown menu with "From Write Call" selected.
- Add Bucket**: A green button with a checkmark icon.

Fig. 8.7. Configuración del almacén de datos en Thingier.io.

Con estos dos recursos creados, ya podemos preparar nuestro dispositivo para que, por un lado, se conecte a la plataforma utilizando las credenciales generadas y, por otro, envíe información meteorológica al almacén de datos.

8.2.3 *Software* del dispositivo

Una vez se ha configurado la plataforma para soportar la conexión de un nuevo dispositivo, así como creado un almacén de información para almacenar los datos, es necesario programar el dispositivo para que se conecte a la plataforma, lea la información de los sensores y escriba en el almacén de información. La programación del dispositivo se hará a través del entorno Arduino, para el que es necesario llevar a cabo algunos ajustes con el fin de utilizar las librerías del ClimaStick, y permitir la programación del microcontrolador ESP8266 en el que está basado. Este proceso inicial está documentado de forma detallada dentro de la documentación de la plataforma en la dirección "<http://docs.thingier.io/hardware/climaStick/>" (en la sección *configure environment*), y es similar al proceso de añadir cualquier otra placa dentro del entorno de Arduino.

Una vez tenemos configurado el entorno para permitir programar el dispositivo ClimaStick, es necesario definir el código fuente basado en las librerías proporcionadas por la plataforma. Para ello, podemos partir de un código como el que se detalla a continuación.

```
#include <ClimaStick.h>

#define USERNAME "alvarolb"
#define DEVICE_ID "climastick"
#define DEVICE_CREDENTIAL "h4Pxg!alaDEK"
#define SSID "Meeble"
#define SSID_PASSWORD "MeebleLabs"

ClimaStick thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
    // configure board wifi
    thing.add_wifi(SSID, SSID_PASSWORD);

    // initialize board sensors
    thing.init_sensors();

    // define Environment resource
    thing["Environment"] >> [] (pson& out){
        out["temperatura"] = thing.get_temperature();
        out["humedad"] = thing.get_humidity();
        out["altitud"] = thing.get_altitude();
        out["presion"] = thing.get_pressure();
        out["lux"] = thing.get_lux();
    };
}

void loop() {
    // handle internet connectivity
    thing.handle();

    // write Environment resource to MeteoBucket
    thing.write_bucket("MeteoBucket", "Environment");

    // sleep the device 60 seconds
    thing.sleep(60);
}
```

Fig. 8.8. Configuración de dispositivos conectados a Thingier.io.

En este código se están definiendo los siguientes parámetros:

- **USERNAME.** Es el nombre de usuario que hemos escogido en el proceso de registro de nuestra cuenta.
- **DEVICE_ID.** Es el identificador del dispositivo que utilizamos en el proceso de registro, en nuestro ejemplo “climastick”.
- **DEVICE_CREDENTIAL.** Es la credencial del dispositivo que utilizamos en el proceso de alta del dispositivo, en nuestro ejemplo “h4Pxg!a1aDEKa”.
- **SSID.** Es el nombre de la red wifi a la que se va a conectar el dispositivo, la utilizada en este ejemplo sería “Meeble”.
- **SSID_PASSWORD.** Es la contraseña de la red wifi a la que se va a conectar el dispositivo, que en el ejemplo sería “MeebleLabs”.

Con los parámetros de USERNAME, DEVICE_ID, y DEVICE_CREDENTIAL, se está inicializando una variable llamada *thing* de tipo *ClimaStick*, que representa nuestro dispositivo, sobre el que podremos configurar la conexión wifi, así como leer la información de los sensores.

Posteriormente, dentro del método *setup* (que se ejecuta una única vez al comienzo del programa), se realizará:

- Inicializar la conexión wifi del dispositivo con los parámetros SSID y SSID_PASSWORD a través del método “*thing.add_wifi*”.
- Inicializar los sensores del dispositivo a través de la llamada “*init_sensors()*”.
- Definir un recurso dentro del dispositivo llamado “Environment”, que va a contener la información meteorológica que queremos almacenar, que en este caso es temperatura, humedad, altitud, presión y luminosidad.

Nota: para más detalle acerca de la programación de dispositivos Arduino para la plataforma Thingier.io, puede consultarse la documentación disponible en: “<http://docs.thingier.io/arduino/#coding>”.

Por último, dentro del método *loop*, que se ejecuta de forma indefinida, después del *setup*, se realiza:

- La gestión de la conexión del dispositivo a la plataforma a través de la llamada “*thing.handle()*”, donde se comprueba si el dispositivo está conectado a Internet, autentica el dispositivo y lo mantiene conectado en caso necesario.
- La escritura del recurso “Environment” definido en el *setup* al almacén de datos “MeteoBucket” registrado previamente en la plataforma. Para ello se utiliza el método “*thing.write_bucket()*”.

- La pausa del dispositivo durante un periodo de tiempo determinado (en este caso un minuto, pero puede ajustarse según sea necesario) a través del método “thing.sleep()”. Este método no retrasa la ejecución de la próxima iteración, básicamente duerme el dispositivo, los sensores, la radio de la conexión inalámbrica y reinicia el dispositivo después del tiempo especificado. De este modo, el dispositivo puede estar alimentado a través de una batería.

Una vez programado el dispositivo, éste se conectará a la red wifi establecida, utilizará las credenciales especificadas para conectarse a la plataforma, leerá la información de los sensores escribiéndola en el almacén de información de la plataforma, y, por último, entrará en un modo *sleep* de bajo consumo hasta la próxima lectura. A partir de este punto, podremos acceder a la información registrada a través de la plataforma.

8.2.4 Visualización y exportación de la información

Tras haber configurado la plataforma y programado el dispositivo, se puede verificar que el proceso está funcionando adecuadamente a través de la consola de administración de la plataforma. Para ello, accederemos al almacén de datos “MeteoBucket” creado previamente, donde podemos observar si los datos están siendo registrados. En este caso debería aparecer una tabla dentro de la sección “Bucket Explorer” que contiene las últimas medidas almacenadas en la plataforma. Como se puede observar en la figura 8.9, aparece una tabla con la marca de tiempo de la medida (registrada por la plataforma), así como los valores definidos dentro del recurso “Environment” establecido en el dispositivo, que son altitud, humedad, lux, presión y temperatura.

Date	altitud	humedad	lux	presion	temperatura
2017-07-31T16:40:55.126+0200	980.379	16.2197	64512	90090.5	35.9
2017-07-31T16:39:50.918+0200	979.533	14.21	64512	90099.8	35.84
2017-07-31T16:38:46.697+0200	979.162	14.1621	64512	90103.8	35.85
2017-07-31T16:37:42.367+0200	979.224	14.2168	64512	90103.1	35.67
2017-07-31T16:36:37.609+0200	979.182	14.5791	64512	90103.6	35.65
2017-07-31T16:35:33.398+0200	979.589	16.9258	64512	90099.2	35.37
2017-07-31T16:34:29.223+0200	979.226	14.8154	64512	90103.1	35.48
2017-07-31T16:33:24.771+0200	979.812	14.6445	64512	90096.7	35.42
2017-07-31T16:32:24.500+0200	979.262	15.3018	64512	90104.1	35.21
2017-07-31T16:31:19.959+0200	978.892	15.2256	64512	90106.8	35.08

Refresh

Viewing 0 to 38 items

Fig. 8.9. Configuración de ejemplo de datos meteorológicos con Thinger.io.

Una vez verificado el funcionamiento del proceso de registro de datos, y tras haber almacenado suficiente información para su análisis, es posible exportar esta serie temporal para su tratamiento con otras herramientas externas. Para ello, dentro del almacén de datos, hay una sección llamada “Bucket Data Export” que permite definir el formato de los datos a exportar, especificar el intervalo de datos a exportar o configurar el formato del *timestamp*. Una vez configurados estos parámetros, y pulsando sobre “Export Data”, se iniciará el proceso de exportación. A continuación, se muestra un ejemplo de la interfaz de exportación.

Fig. 8.10. Configuración de captura de datos mediante almacén (*bucket export*).

Cuando este proceso haya concluido, llegará un correo electrónico a la dirección proporcionada en el proceso de registro de la cuenta en la plataforma, en el que se proporciona un enlace de descarga al fichero de datos, tal y como se refleja en la figura 8.11. En este caso, el fichero se encontraría en formato CSV (*Comma-Separated Values*), que puede ser abierto directamente por numerosas herramientas de análisis.



Thinger.io Bucket Export

The data extraction from your bucket **MeteoBucket** has been completed.

Download the data here: <http://thinger.io.bucket.s3-eu-west-1.amazonaws.com/20170731T145910.alvarolb.MeteoBucket.7WYF045A.csv>

The download link will be available for 3 months.

Fig. 8.11. Configuración de descarga de datos.

8.3 Series temporales

En este capítulo hemos introducido las series temporales para la representación y tratamiento de datos de sensores, como consecuencia de la disponibilidad de plataformas de tipo IoT. Hemos comenzado con un ejemplo de datos meteorológicos entre la multitud de aplicaciones posibles. Por ejemplo, en el área de IoT para teleasistencia, aparecen en multitud de medidas fisiológicas corporales: ritmo cardíaco, respiración, sudoración, etc.

Sin embargo, la formalización de las series temporales ha aparecido anteriormente en otras áreas, tradicionalmente en ciencias sociales, como estadísticas sobre índices económicos, tasa de interés, desempleo, crimen, natalidad y mortalidad, etc. Una serie temporal puede contener ciclos, como el ciclo diario de sensores de temperatura y luminosidad, ciclo semanal de tráfico viario, ciclo anual de ocupaciones hoteleras, magnitudes atmosféricas, etc. En la figura 8.12 se muestra un ejemplo de serie temporal que representa la actividad solar.

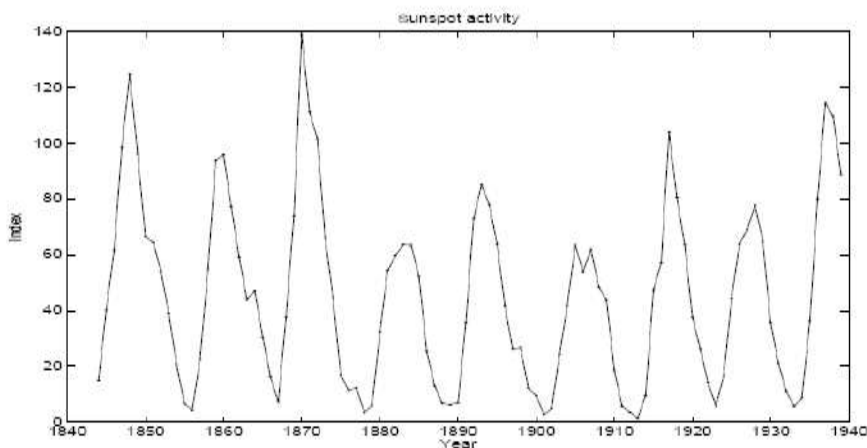


Fig. 8.12. Ejemplo de serie temporal de actividad solar.

En todo caso, el aspecto fundamental que distingue las series temporales de otros problemas de análisis de datos es que en la serie los datos no son independientes, sino que tiene significado su secuencia y orden temporal. Por esta razón, el análisis de series de tiempo es totalmente distinto al análisis de datos convencional, no pueden tratarse como conjuntos independientes de datos. Nos concentraremos principalmente en el objetivo de poder predecir valores futuros de una serie temporal utilizando valores previos, que pueden ser sólo de esta serie o bien utilizar también otras variables que estén disponibles también como series temporales. Otro objetivo frecuente es estudiar relaciones de influencia entre variables que forman series temporales, un ejemplo clásico es analizar si los cambios en el desempleo están relacionados con cambios en la delincuencia a partir de la búsqueda de correlaciones entre las series.

Un problema típico con el estudio de los efectos de una variable sobre otra en formato de serie es que las observaciones dentro de cada serie no son independientes entre sí, de modo que la probabilidad de encontrar una alta correlación entre las dos series puede ser mayor que la proporcionada por las fórmulas estándar para datos independientes. Otro problema habitual es de alineamiento, puesto que rara vez puede asumirse que la secuencia de tiempo de los patrones causales coincide con los periodos de tiempo de las series. Siguiendo con el ejemplo anterior, si el aumento del desempleo produce un aumento en la delincuencia exactamente seis meses más tarde, pero no cinco meses después, sería bastante fácil descubrir esa relación correlacionando los cambios mensuales en el desempleo con los cambios mensuales en el crimen seis meses después. Sin embargo, si el aumento del desempleo en un mes produce un leve aumento de la delincuencia durante varios meses puede ser más difícil de detectar.

A veces se puede estudiar la influencia de factores externos sobre el comportamiento de una serie, como por ejemplo el impacto de un evento político sobre la volatilidad (medida a través de la varianza) en una serie de cotizaciones bursátiles, en este último caso se habla de un problema de segmentación de la serie a partir de una variable externa. Este último ejemplo es un problema importante en la predicción que surge en muchas áreas de pronóstico. El conocimiento de un evento puede modificar el comportamiento de la serie y no es predecible a partir de los datos pasados.

8.3.1 Predicción con series temporales

La clave para plantear la predicción con series de tiempo es el conocimiento de la distribución con la que se está trabajando, lo que raramente se conoce *a priori*. Además, otro aspecto que habitualmente se desconoce es la estacionariedad de la serie, saber si es válida la suposición de parámetros constantes en el tiempo, lo que puede cambiar de forma no predecible a partir de los datos históricos de la serie. Ejemplos típicos de no estacionariedad son las variaciones bruscas de índices económicas en una situación de cambio, algo imposible de predecir a partir de los datos pasados históricos de la serie.

8.3.1.1 Predicción lineal (autorregresión)

El objetivo básico de la autorregresión con series temporales es encontrar una fórmula que pronostique con precisión cada entrada de la serie a partir de las entradas anteriores. Entonces, puede ser razonable esperar que la misma fórmula se pueda usar para pronosticar entradas futuras en la serie de las últimas entradas conocidas actualmente (predicción).

Lo habitual con series temporales es tomar X como la variable dependiente, que se traza como una función de tiempo. Otra forma de calcular la predicción es tomar una

serie de puntos anteriores que implícitamente tienen una relación temporal, habitualmente un número fijo de periodos de tiempo. En este punto conviene definir los diferentes problemas que pueden plantearse sobre una secuencia de datos, ilustrados en la Figura 8.13.

- **Predictor fwd/bkd.** Predice valores de la serie hacia delante o hacia atrás a partir de las observaciones actuales y pasadas, o de las observaciones actuales y futuras respectivamente.
- **Suavizado (*smoothing*).** Estima valores de parámetros utilizando medidas futuras, actuales y pasadas.

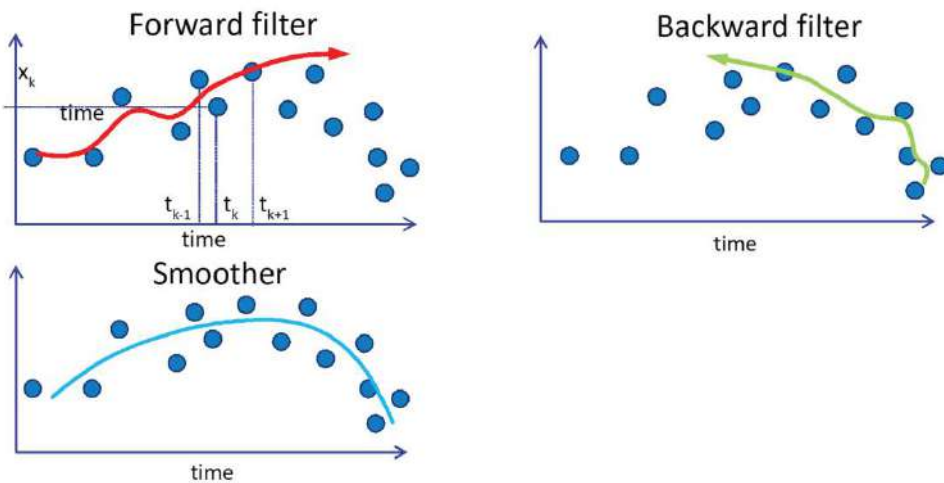


Fig. 8.13. Representación gráfica de operaciones de predicción o suavizado en series temporales.

Una forma simple de hacer la predicción es utilizar las últimas n observaciones anteriores a cada punto x_i para hacer una estimación de x_i . Podrían plantearse diferentes alternativas, como por ejemplo el promedio, mantener el último valor, una recta con las dos últimas para extrapolar su valor a x_i , etc. Cuando la predicción es una función lineal de las observaciones que preceden a x_i , hablamos de regresión lineal, y el procedimiento habitual consiste en seleccionar el tamaño de la ventana (número de observaciones precedentes a incluir en la función lineal), y determinar los coeficientes óptimos para la serie en un proceso de entrenamiento, siendo el número de observaciones utilizadas para realizar cada predicción lo que constituye el orden de la autorregresión. Por tanto, puede verse como un caso particular de regresión mediante mínimos cuadrados, con una formulación apropiada, como se ilustra en la Figura 8.14.

$$\hat{x}_k = a_0 + a_1 x_{k-1} + a_2 x_{k-2} + \dots + a_m x_{k-m}$$

Ec. 9.1

$$\hat{x}_k = a_0 + a_1 x_{k-1} + a_2 x_{k-2} + \dots + a_m x_{k-m}$$

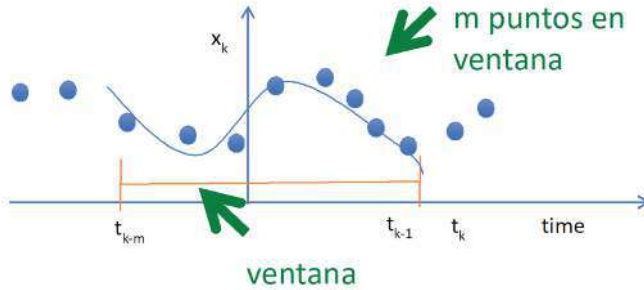


Fig. 8.14. Predicción mediante ventana de m últimos datos de la serie.

Antes de construir el modelo para la predicción de la serie temporal, el primer paso para la regresión es formular el modelo con pares de entrada-salida. Si los datos disponibles son $\{x_i\}_{i=1}^N$, el modelo de predicción lineal con ventana de tamaño m puede formularse así:

$$\begin{bmatrix} \hat{x}_N \\ \hat{x}_{N-1} \\ \vdots \\ \hat{x}_{m+1} \end{bmatrix} = \begin{bmatrix} 1 & x_{N-1} & x_{N-2} & \cdots & x_{N-1-m} \\ 1 & x_{N-2} & x_{N-3} & \cdots & x_{N-2-m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_{m-1} & \cdots & x_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = H \bar{a} \quad \text{Ec. 9.2}$$

Y con esta formulación tenemos de nuevo el problema clásico de minimización de función cuadrática cuya solución global viene dada por la fórmula de mínimos cuadrados:

$$\bar{a} = [H^t H]^{-1} H^t \bar{x} \quad \text{Ec. 9.3}$$

Y así puede obtenerse con el procedimiento explicado en el capítulo 4.

8.3.1.2 Error de predicción

Los resultados de la predicción mediante la autorregresión tienen asociado un error, tal como se presentó en el capítulo 4 en la teoría de regresión ordinaria:

$$C_{\bar{a}} = \begin{bmatrix} \sigma_{a_0}^2 & \cdots & \cdots \\ \vdots & \sigma_{a_1}^2 & \vdots \\ \vdots & \vdots & \ddots \\ \vdots & \cdots & \cdots & \sigma_{a_m}^2 \end{bmatrix} = \sigma_x^2 [H^t H]^{-1} \quad \text{Ec. 9.4}$$

Si los errores de pronóstico son aproximadamente normales, el error cuadrático medio (MSE) de la predicción de series de tiempo puede aproximarse como ya vimos en el capítulo 4 de regresión lineal, en el apartado de error de predicción. Su valor vendrá dado por la suma de errores cuadrados ponderado por la matriz H , y ajustado al número de parámetros (N , número de parámetros utilizados para ajustar el modelo). De esta manera, los límites de confianza del 95 % se obtienen multiplicando la raíz cuadrada del MSE por 1.96, o el valor apropiado de la distribución t -Student si el número de datos es moderado (no es superior a 100).

Sin embargo, los detalles del modelo estadístico son sólo una componente de la incertidumbre que aparece al hacer una predicción a futuro. Como se mencionó anteriormente, la clave es si se puede asumir estacionariedad y, por tanto, que el periodo predicho sería simplemente otro caso extraído de la población empleada para construir la fórmula de regresión (entrenamiento). En este sentido, únicamente cabe analizar si la fórmula que mejor ha funcionado para pronosticar las entradas de series pasadas es razonable para pronosticar los próximos valores de la serie a partir de los últimos datos. Una posibilidad para aumentar la confianza en la predicción puede ser considerar el error de la fórmula de predicción y analizar si ha sido uniforme a lo largo del tiempo, especialmente si hay tramos de la serie en los que el error es significativamente distinto de cero (sesgo), y si hay tramos con variaciones significativas de la varianza. La presencia de estas variaciones en media y varianza del error pueden sugerir que en todo caso el modelo funciona mejor algunas veces que otras y, por lo tanto, habría que tener precaución por si los límites de confianza que proporciona la regresión subestiman el error de predicción real.

8.3.1.3 Predicción no lineal

La generalización del problema de predicción con métodos no lineales puede llevarse a cabo de manera análoga a como se ha presentado en el caso general. Tomando la formulación anterior como una predicción de cada valor de la serie a partir de las m muestras anteriores, podemos aplicar métodos no lineales, como por ejemplo los método de inducción de árboles de regresión M5, regresión mediante vectores de soporte, redes neuronales o métodos *kernel* presentados en el capítulo 6. Además de las m últimas muestras, los modelos pueden incorporar la variable temporal (con un escalado apropiado para evitar problemas numéricos). En este planteamiento generalizado es interesante además considerar la posibilidad de incorporar variables adicionales que puedan mejorar la capacidad predictiva del modelo, incluso puedan prepararse variables adicionales de periodicidad derivadas del tiempo, como pueden ser mes del año, cuatrimestre, día de la semana, franja horaria, etc. Estas variables discretas pueden ser de gran ayuda para construir una función no lineal que optimice las predicciones en función de su valor.

8.3.2 Análisis y descomposición de series

Además de plantear modelos precisos para realizar predicciones, en ocasiones se puede llevar a cabo un análisis de los datos de la serie completa para plantear un modelo estacionario de la serie. Esto puede ayudar a comprender el fenómeno y mejorar el modelo de predicción, en lugar de utilizar una ventana con las últimas medidas limitada a predicciones locales. Por ejemplo, el conocimiento de ciclos en los datos y su periodicidad puede ayudar a descomponer la serie en diferentes factores. Si tenemos una serie de temperaturas medidas periódicamente (cada hora, por ejemplo), tendríamos una componente de periodicidad diaria, con un máximo y mínimo, y una tendencia a más largo plazo en función de las variaciones estacionales.

8.3.2.1 Tendencia y estacionariedad

A continuación, se muestra un ejemplo clásico en el modelado de series temporales, el número de pasajeros aéreos en una aerolínea de Estados Unidos a lo largo de diez años.

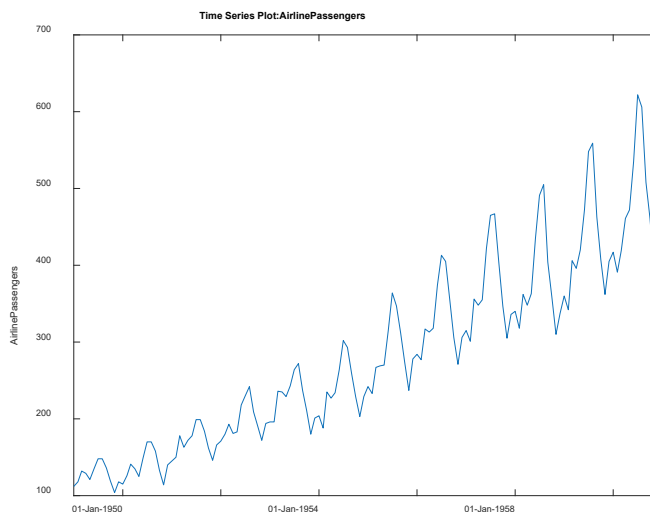


Fig. 8.15. Serie de datos de pasajeros de una línea aérea.

Puede apreciarse una clara tendencia creciente en la magnitud considerada a la que se superpone una fuerte variación periódica anual. Si tomamos el logaritmo de los valores, la serie puede verse que el crecimiento es de tipo exponencial, de modo que esta serie admitiría un modelo de tendencia lineal y periodo constante sobre el logaritmo de la variable observada:

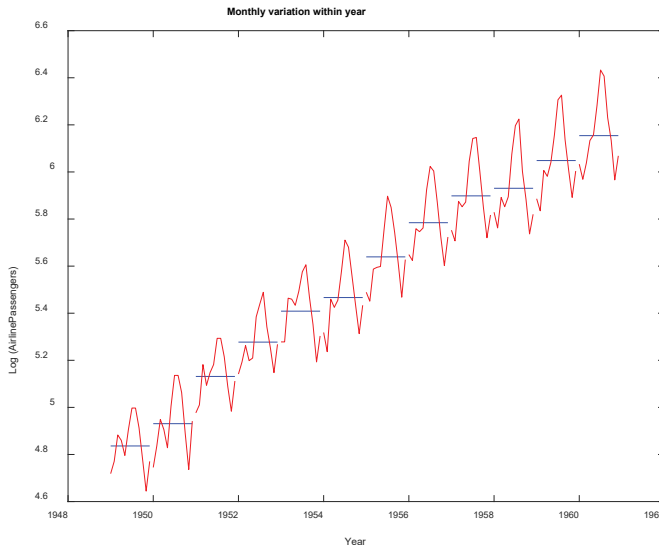


Fig. 8.16. Variaciones y promedios mensuales de la serie (logaritmo).

Para calcular de manera directa los términos de la tendencia y los valores periódicos mensuales puede utilizarse de nuevo un método de regresión lineal, aplicado ahora a la serie completa en función del tiempo, y con una extensión apropiada que tenga en cuenta los doce valores mensuales en la estimación. Se ha construido el modelo extendiendo la regresión a factores periódicos con una función mensual constante más una tendencia lineal sobre el logaritmo de los datos:

$$\hat{x}_k = \sum_{m=1}^{12} a_{0m} \delta_m(t_k) + a_1 t_k \quad \text{Ec. 9.5}$$

Donde a_{0m} es el valor constante que se corresponde a cada mes, y $\delta_m(t_k)$ es un operador binario que denota el mes al que le corresponde la fecha representada en t_k :

$$\delta_m(t_k) = \begin{cases} 1, & \text{si la fecha } t_k \text{ está en el mes } m \\ 0, & \text{en caso contrario} \end{cases} \quad \text{Ec. 9.6}$$

Como tenemos un dato por cada mes, puede plantearse la estimación de los coeficientes con la siguiente formalización del modelo anterior:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & t_1 \\ 0 & 1 & \cdots & 0 & t_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & t_{12} \\ 1 & 0 & \cdots & 0 & t_{13} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & t_n \end{bmatrix} \begin{bmatrix} a_{01} \\ \vdots \\ a_{012} \\ a_1 \end{bmatrix} \quad \text{Ec. 9.7}$$

Y de esta manera se pueden obtener los coeficientes $\{a_{0i}\}_{i=1,\dots,12}$, y a_1 con la expresión habitual de mínimos cuadrados presentada anteriormente. Puede verse el ajuste en la figura 8.17, y la descomposición de la serie en la figura 8.18, validando el modelado con coeficientes constantes y un término lineal en el tiempo (recuérdese que el modelo es sobre el logaritmo de la variable de interés).

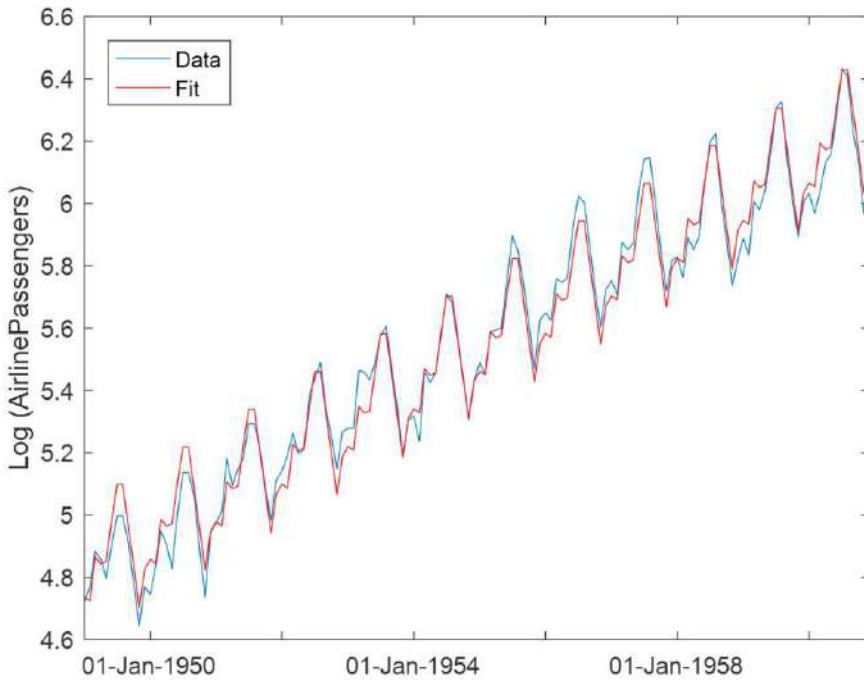


Fig. 8.17. Ajuste de la serie mediante tendencia y estacionariedad (logaritmo).

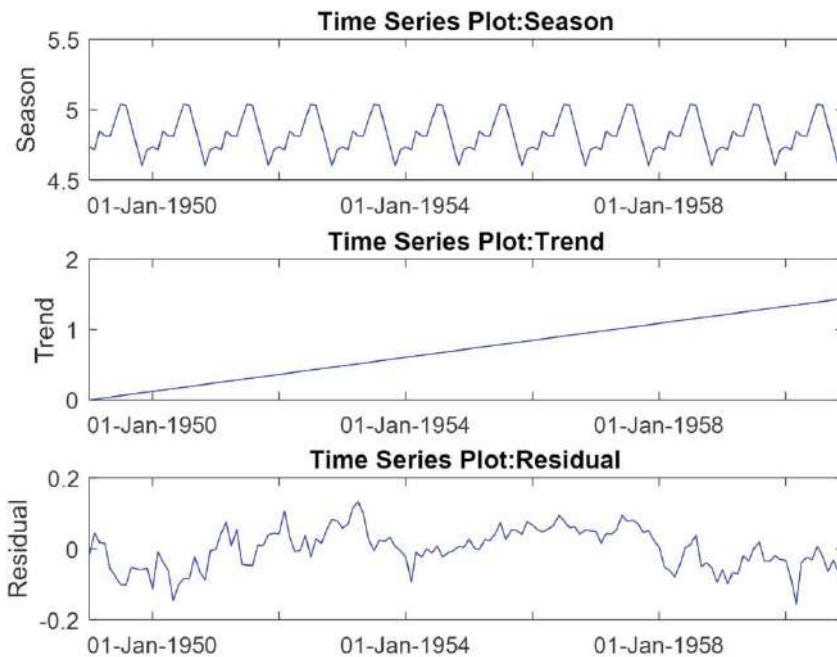


Fig. 8.18. Descomposición de la serie de pasajeros.

8.3.2.2 Modelos ARMA/ARIMA

Finalmente, los modelos ARIMA (*Auto Regressive Integrated Moving Average*), también conocidos como modelo Box-Jenkins, sirven para modelar series estacionarias y para hacer predicciones tras realizar un ajuste que permita calcular las propiedades estadísticas de la serie. Un modelo ARMA viene típicamente dado por la siguiente expresión:

$$\begin{aligned}
 x_k = & \varphi_1 x_{k-1} + \varphi_2 x_{k-2} + \dots + \varphi_p x_{k-p} + \dots \\
 & + n_{k-1} + \theta_1 n_{k-1} + \theta_2 n_{k-2} + \dots + n_q x_{k-q}
 \end{aligned}
 \quad \text{Ec. 9.8}$$

La parte AR (*auto regressive*) modela la dependencia con valores anteriores, mientras que la parte MA (*moving average*) modela la aleatoriedad a partir de q entradas aleatorias, n_k . Se habla de un modelo ARMA (p, q) para modelar un proceso estacionario, y los coeficientes se ajustan a partir de las propiedades de autocorrelación y autocorrelación parcial calculadas para la serie, un procedimiento estadístico estándar que no detallamos aquí.

La limitación del modelo ARMA es que es aplicable a procesos estacionarios, algo que no cumplen la mayoría de las series (por ejemplo, lo más habitual es que la media varíe en el tiempo). Para superar esta limitación, hay familias de procesos que admiten

un modelo estacionario a través del operador de diferenciación, lo que constituye el modelo ARIMA. Un proceso ARIMA (p,d,q) es una extensión del proceso ARMA (p,q), que modela el comportamiento de la d-ésima diferencia de la serie, w_k :

$$\begin{aligned} w_k &= \nabla^d x_k \\ w_k &= \varphi_1 w_{k-1} + \varphi_2 w_{k-2} + \dots + \varphi_p w_{k-p} + \dots \\ &\quad + \eta_{k-1} + \theta_1 \eta_{k-1} + \theta_2 \eta_{k-2} + \dots + \eta_q x_{k-q} \end{aligned} \quad \text{Ec. 9.9}$$

Donde $\nabla^d x_k A = \pi r^2 A = \pi r^2$ denota el operador de la d-ésima diferencia de la serie:

$$\begin{aligned} \nabla^0 x_k &= x_k \\ \nabla^1 x_k &= x_k - x_{k-1} \\ \nabla^2 x_k &= \nabla^1 x_k - \nabla^1 x_{k-1} = x_k - x_{k-1} - (x_{k-1} - x_{k-2}) \\ \nabla^3 x_k &= \nabla^2 x_k - \nabla^2 x_{k-1} \\ &\dots \end{aligned} \quad \text{Ec. 9.10}$$

El modelo ARIMA es una técnica con mayor capacidad representativa, que permite modelar muchas series en la práctica. Por ejemplo, es fácil demostrar que un modelo de orden 1 modela una serie con una media de tendencia lineal, un modelo de orden 2 una media de tendencia cuadrática, etc.

En todo caso, la capacidad predictiva de ARIMA vendrá dada por el correcto ajuste de los coeficientes (con suficiente número de datos), y bajo la hipótesis de que el proceso real tiene las propiedades asumidas, lo que en la práctica suele ser bastante restrictivo. Como en otros problemas de análisis de datos, cuando el modelo estadístico no se cumple para el problema abordado, o bien no hay suficientes datos como para calcular con precisión los parámetros, las técnicas de aprendizaje automático permiten incrementar la precisión y tener en cuenta más elementos con modelos más abiertos y métodos heurísticos de construcción.

8.4 Análisis de series con R

Para ilustrar la utilización del entorno R con series temporales, se seguirán empleando datos correspondientes al registro de calidad del aire y valores meteorológicos de la ciudad de Madrid, como en los ejemplos presentados en capítulos anteriores. Sin embargo, dado que la periodicidad de los eventos meteorológicos es de años, los registros del ejemplo no serían adecuados para realizar análisis de series temporales por no disponer de varios periodos. Por tanto, se va a realizar el análisis sobre la temperatura media mensual de la ciudad de Madrid entre los años 1881 a 2015 tomados

de AEMET. El objetivo es poder predecir la temperatura media mensual del año 2015 a partir de los valores de años anteriores, extrayendo información adicional como su periodicidad, tendencia, etc.

Al cargar los datos de las temperaturas medias mensuales, se observa que están ordenados en filas por años. Se desea tener todas las medias en un vector, para lo cual aplicamos la función *as.vector* sobre la traspuesta.

```
df_2015 <- df_TMED[135,2:13] #El último año lo vamos a usar para comparar
df_TMED <- as.vector(t(df_TMED[1:134,2:13]))
```

Se han desarrollado en R librerías que extienden las estructuras de datos para recoger formatos orientados al tratamiento de series temporales. Por ejemplo, la librería *stats* tiene un formato específico y funciones para extraer información de series temporales.

Transformamos los datos a la estructura de serie temporal con *ts*. Se muestran en los gráficos siguientes todos los valores hasta el año 2014 y el correspondiente al año 2015 (figuras 8.19 y 8.20 respectivamente).

```
library(stats)
ts_TMED <- ts(df_TMED, frequency = 12, start = 1881) #serie temporal
ts_2015 <- ts(t(df_2015), frequency = 12, star = 2015)
autoplot(ts_TMED)
autoplot(ts_2015)
```

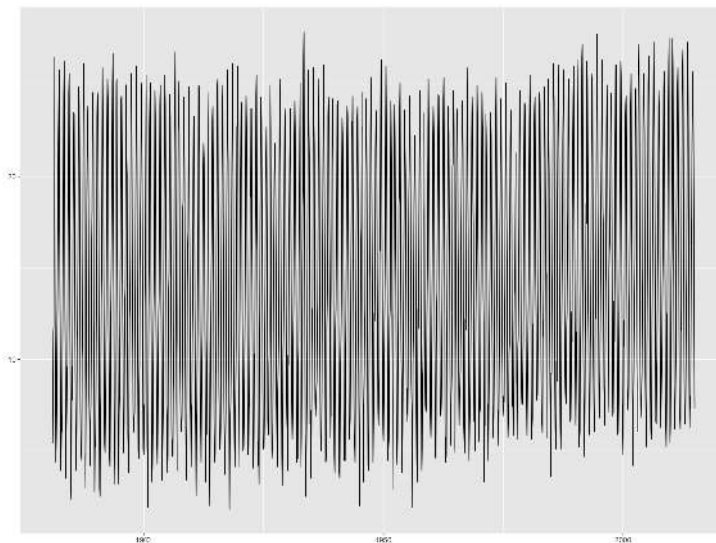


Fig. 8.19. Datos de temperatura media mensual en Madrid desde el año 1881 hasta el 2014.

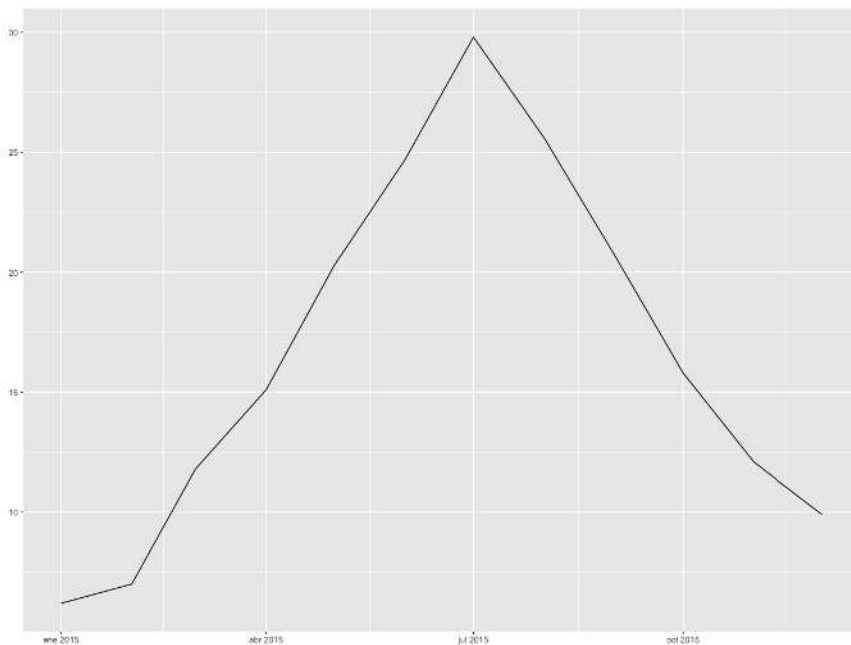


Fig. 8.20. Datos de temperatura media mensual en Madrid del año 2015.

Con todos los datos del gráfico es muy difícil determinar tendencias y regularidades.

8.4.1 Componentes de la serie temporal

La librería *forecast* está construida específicamente para realizar modelos de predicción para modelos lineales y series temporales.

Se muestran en los siguientes gráficos (figura 8.21) los valores representados por frecuencia, es decir, cada año es una serie, y en código de color, los más antiguos en color más oscuro y los más modernos con color más claro. También se representan en una coordenada polar (figura 8.22).

```
library(forecast)
ggseasonplot(ts_TMED, continuous = TRUE)
ggseasonplot(ts_TMED, continuous = TRUE, polar = TRUE)
```

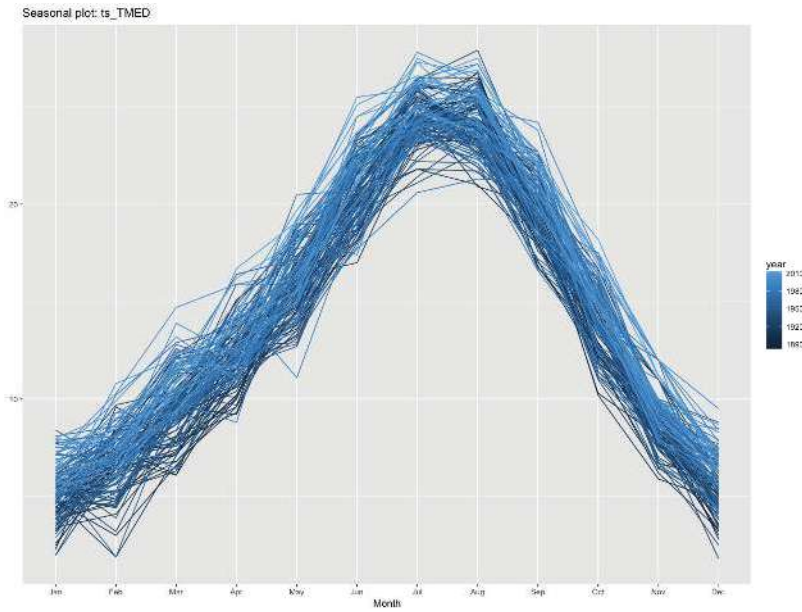


Fig. 8.21. Datos de temperatura media mensual en Madrid por meses.

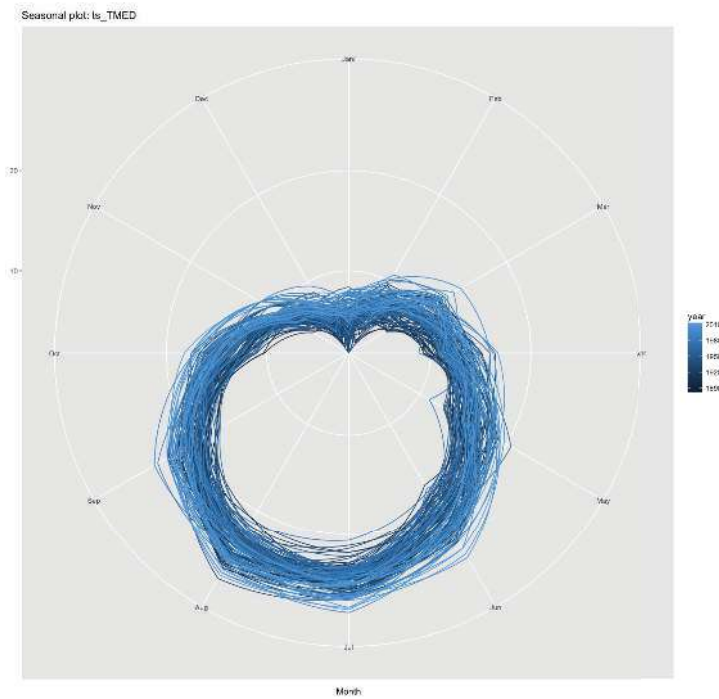


Fig. 8.22. Datos de temperatura media mensual en Madrid por meses en coordenadas polares.

En estos gráficos ya sí se aprecian claramente regularidades, hay una periodicidad anual en el comportamiento de la temperatura. Un primer modelo podría consistir en tomar el valor medio mensual de toda la serie y con éste predecir el de cualquier año siguiente.

Un gráfico muy interesante es el que muestra todos los valores por subseries mensuales (figura 8.23).

```
ggsubseriesplot(ts_TMED)
```

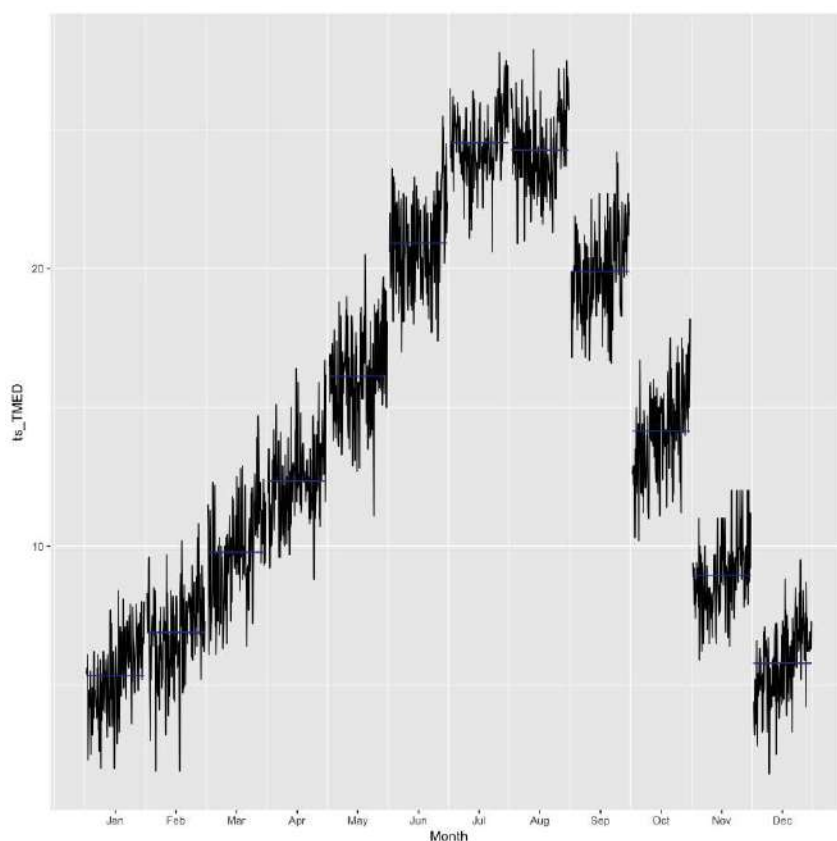


Fig. 8.23. Datos de temperatura media mensual con los años agrupados en series y mostrando el promedio.

Con un segmento azul se muestra el valor promedio en toda la serie, por mes. Se observa un hecho notable y es la tendencia a que la temperatura media por mes es, en general, ascendente. Por tanto, la serie temporal tiene, por un lado, una componente periódica mensual y, por otro, una tendencia lineal ascendente con los años. Podría existir una tercera componente cíclica que es cuando los datos presentan una fluctuación variable

sin tener una frecuencia fija. Es difícil determinar si existe esta última componente a partir de los gráficos.

En el siguiente gráfico de la figura 8.24 se muestra una representación del gráfico de dispersión del dato en un instante t frente al dato retrasado en un instante $t-lag(n)$. En código de color están representados los meses.

```
gglagplot(ts_TMED, do.lines = FALSE, lags = 12) + geom_point(size = 0.1, stroke = 0)
```

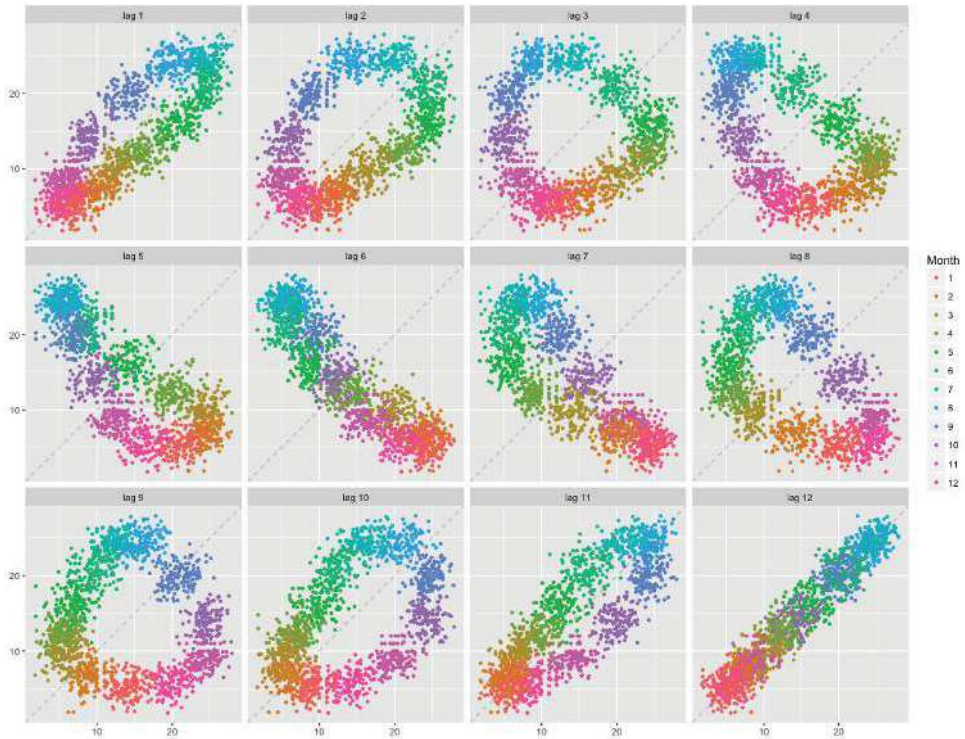


Fig. 8.24. Gráficos de dispersión que muestran un valor y el valor retrasado un instante $lag(n)$.

Este tipo de representación es útil cuando la frecuencia es desconocida, permite estimarla a partir del valor de retraso (lag) en el que la correlación es máxima. En este caso, como ya conocíamos, la correlación mayor positiva ocurre con un retraso de doce. Se está estimando la autocorrelación de la serie.

Para realizar un cálculo de autocorrelación se puede usar la función `acf` que permite también calcular la autocovarianza y la autocorrelación parcial.

```
ggAcf(ts_TMED)
```

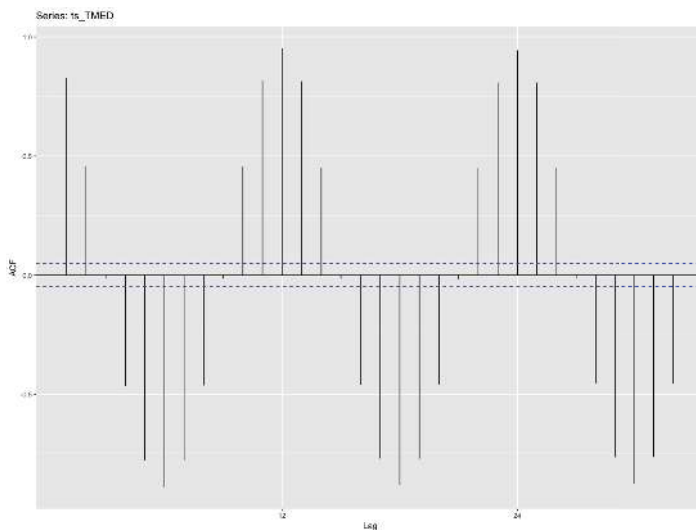


Fig. 8.25. Gráficos de autocorrelación de la serie.

Efectivamente, la correlación es máxima para doce, veinticuatro..., múltiplos enteros de la frecuencia de la serie.

Representando la serie en el dominio de las frecuencias también puede determinarse la frecuencia dominante.

```
spectrum(ts_TMED)
```

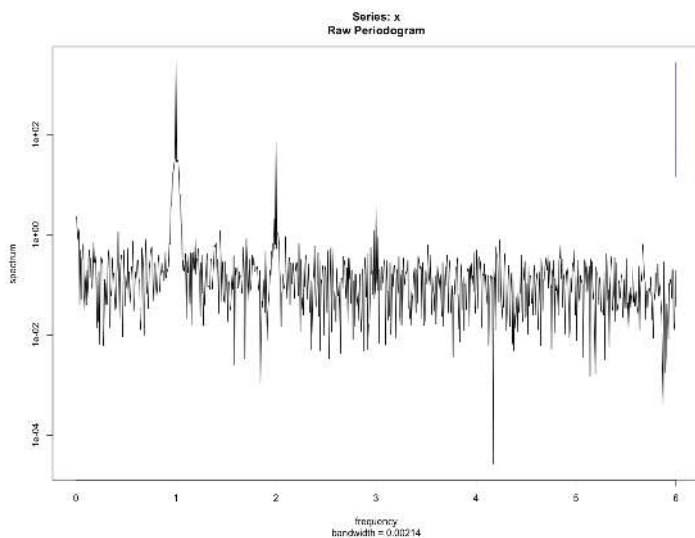


Fig. 8.26. Gráfico de frecuencias.

Efectivamente, el mayor pico ocurre al periodo definido en la serie.

Volviendo a la autocorrelación, pero utilizando valores mayores de *lag*:

```
ggAcf(ts_TMED, lag = 300)
```

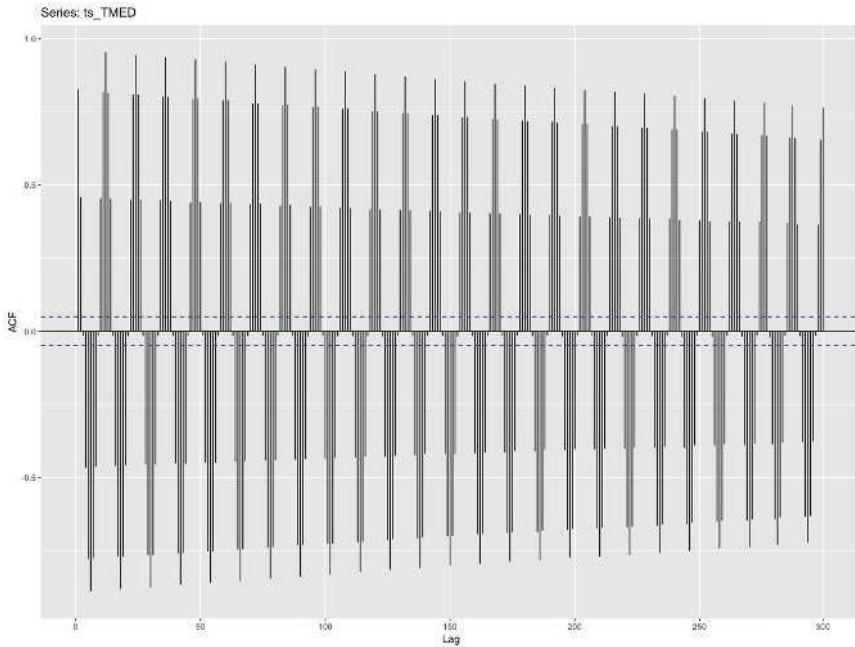


Fig. 8.27. Gráfico de autocorrelación ampliado para apreciar la tendencia.

Se aprecia claramente que los máximos son para múltiplos de doce, pero éstos tienen una tendencia decreciente. Para separar las diferentes tendencias presentes en la serie temporal se puede utilizar *decompose*, que usa medias móviles para crear un modelo aditivo o multiplicativo. El modelo resultante es una función de la tendencia, la parte periódica y un error aleatorio: $Y(t)=T(t)+P(t)+e(t)$ aditivo, $Y(t)=T(t)*P(t)*e(t)$ multiplicativo.

En la gráfica de la figura 8.28 se muestra el resultado de la descomposición de las componentes de la serie temporal en un modelo aditivo.

```
ts_TMED_desc <- decompose(ts_TMED, type = "additive")
autoplot(ts_TMED_desc)
```

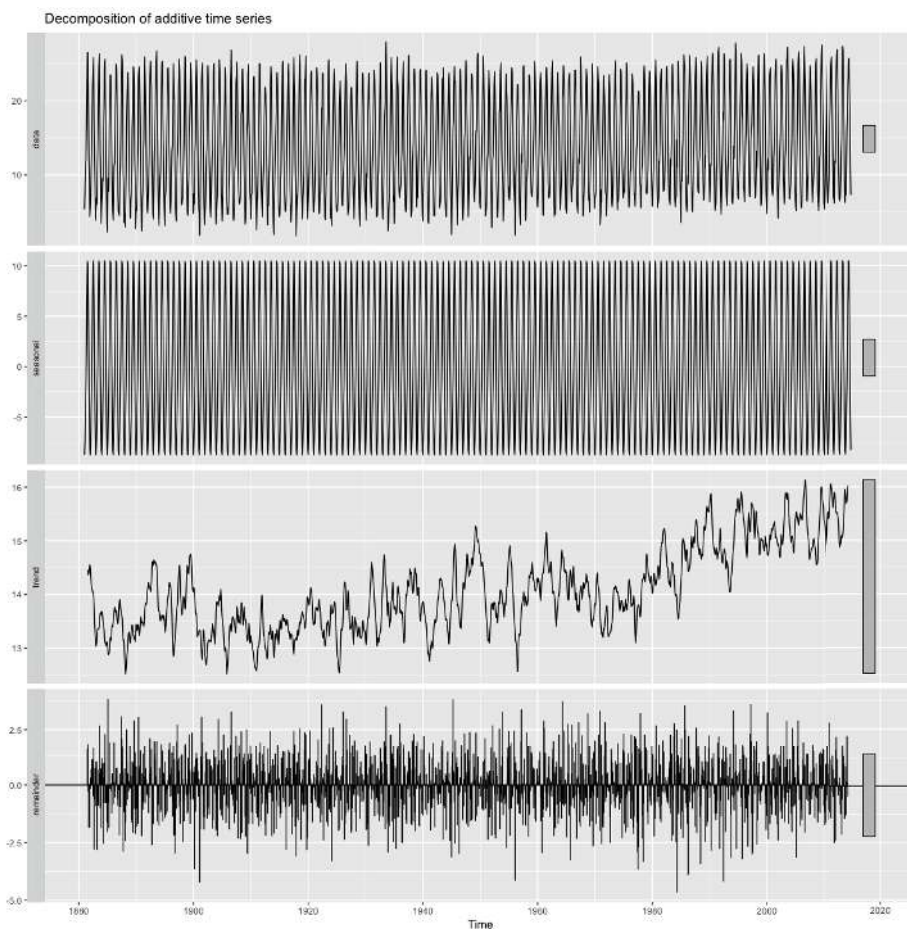


Fig. 8.28. Gráficos que muestran la descomposición de la serie en diferentes componentes.

En lugar de medias móviles se puede usar un método más avanzado como la regresión local (LOESS).

```
ts_TMED_stl <- stl(ts_TMED, s.window = "periodic")
autoplot(ts_TMED_stl)
```

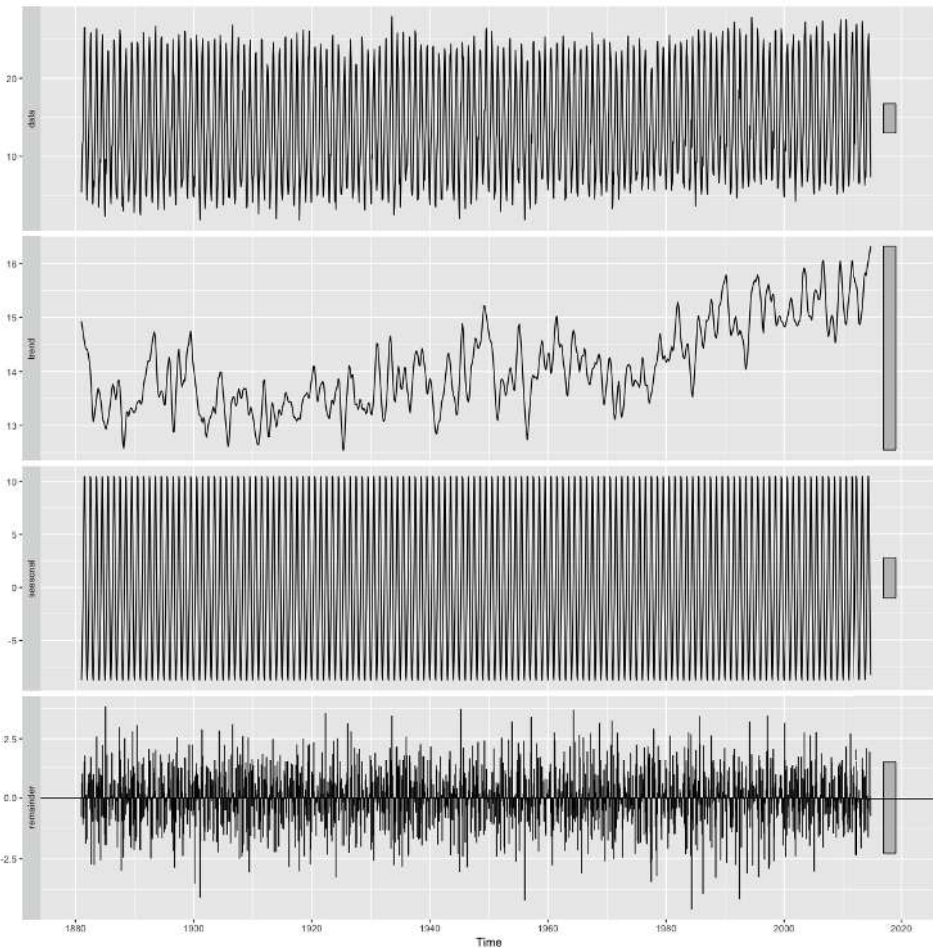


Fig. 8.29. Descomposición de la serie utilizando regresión lineal local.

Ahora puede establecerse una regresión para la tendencia. Se observa que la predicción de futuro no es alentadora, en algo más de un siglo la tendencia lineal de la temperatura media mensual es a aumentar dos grados sobre la componente estacional, aún mayor si se utiliza una regresión local.

```
autoplot(ts_TMED_stl$time.series[,2]) +
  stat_smooth(method = "lm", aes(color = "Lineal"), se = TRUE, level = 0.95) +
  stat_smooth(method = "loess", aes(color = "LOESS"), se = TRUE, level = 0.95)
```

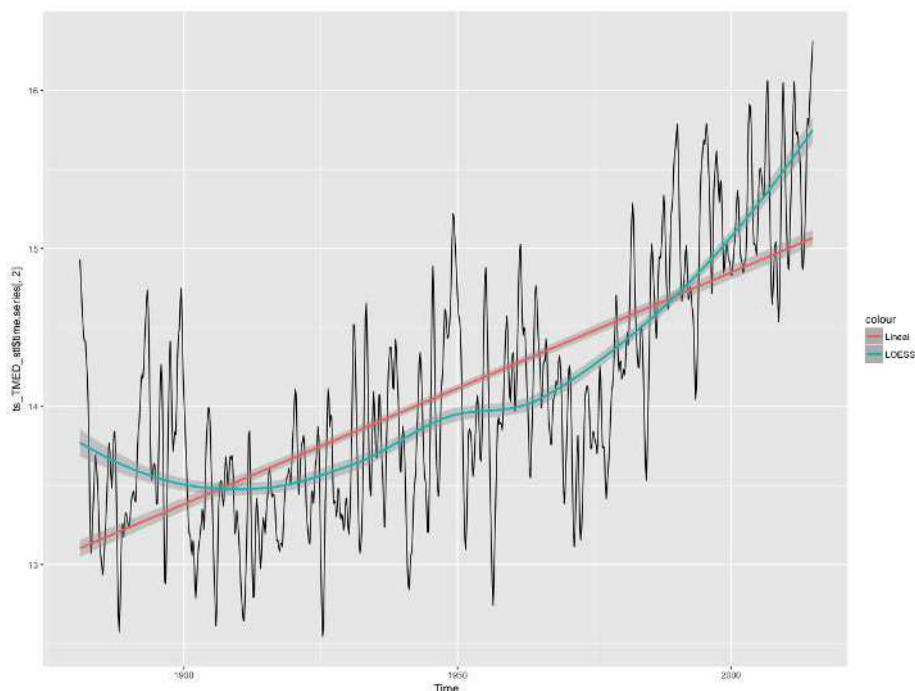


Fig. 8.30. Regresión de la tendencia de la serie utilizando un modelo lineal y LOESS.

8.4.2 Modelos de predicción

Se van a construir diferentes modelos para la serie que permitan predecir las temperaturas de 2015. Una forma muy habitual de proceder consiste en predecir de forma independiente cada una de las componentes de la serie, la parte estacionaria, la parte de tendencia y la aleatoria. Después se unen las tres para dar una predicción para una ventana temporal fijada.

El primero es un método *naive*, se predice la parte de tendencia para posteriormente aplicarla a los datos desestacionalizados. En el gráfico de la figura 8.31 se muestra la serie con la predicción. Para poder visualizar bien los datos predichos se muestra sólo la serie a partir de 2010.

```
fit_naive_ts <- stlf(ts_TMED, method = "naive", h = 12, s.window = 12, robust = TRUE)
autoplot(fit_naive_ts, xlim = c(2010, 2016))
```

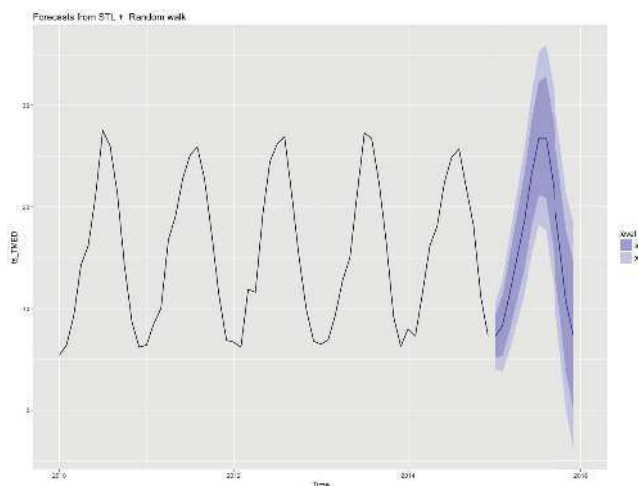


Fig. 8.31. Predicción del año 2015 aplicando STL.

En la predicción *naive*, todos los valores previos usados en la predicción tienen el mismo peso. Pero puede establecerse un peso relativo a la antigüedad del dato, cuanto más antiguo, el peso es menor. Este tipo de estrategia se denomina *suavizado exponencial*.

El método Holt-Winters y el Box-Jenkins son dos de las principales técnicas de suavizado exponencial que se aplican a la predicción de series temporales.

```
fit_HW_ts <- hw(ts_TMED, h = 12)
autoplot(fit_HW_ts, xlim = c(2010, 2016))
```

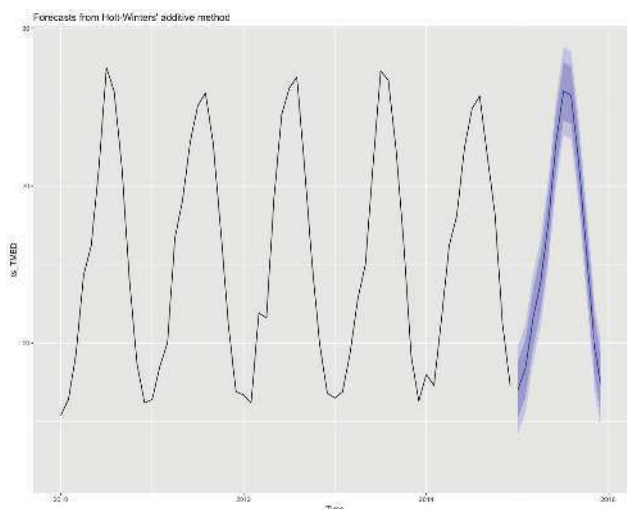


Fig. 8.32. Predicción del año 2015 aplicando Holt-Winters.

Los métodos de suavizado exponencial requieren que los errores, en el intervalo de predicción, estén distribuidos según una normal de media cero. Por tanto, los errores no están correlados y la varianza es constante. Si se quiere utilizar la correlación del error para mejorar la estimación del modelo de predicción, entonces hay que usar el modelo autorregresivo integrado de promedio móvil, ARIMA. Sin embargo, la utilización de ARIMA impone que la serie debe ser estacionaria. Por tanto, los datos de la serie se deben preparar restando la componente de tendencia, y se consigue así que el error sea cero. Después se resta del valor del año anterior para desestacionalizar la serie.

En la gráfica se muestra la serie preparada para aplicar ARIMA.

```
ts_TMED_ARIMA <- diff(ts_TMED, differences = 12)
autoplot(ts_TMED_ARIMA)
ggAcf(ts_TMED_ARIMA)
```

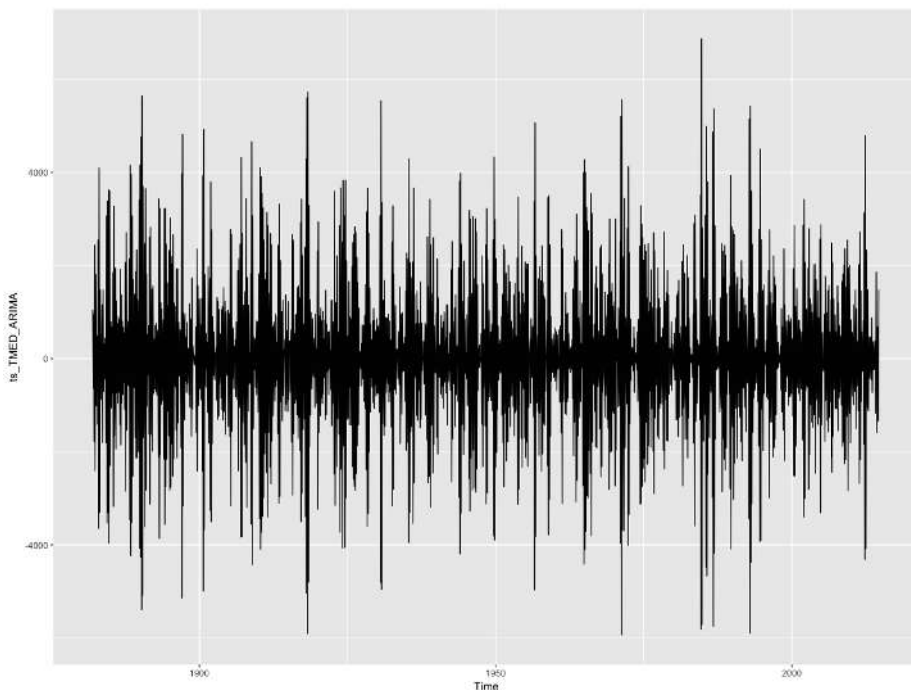


Fig. 8.33. Transformación de la serie para poder aplicar ARIMA.

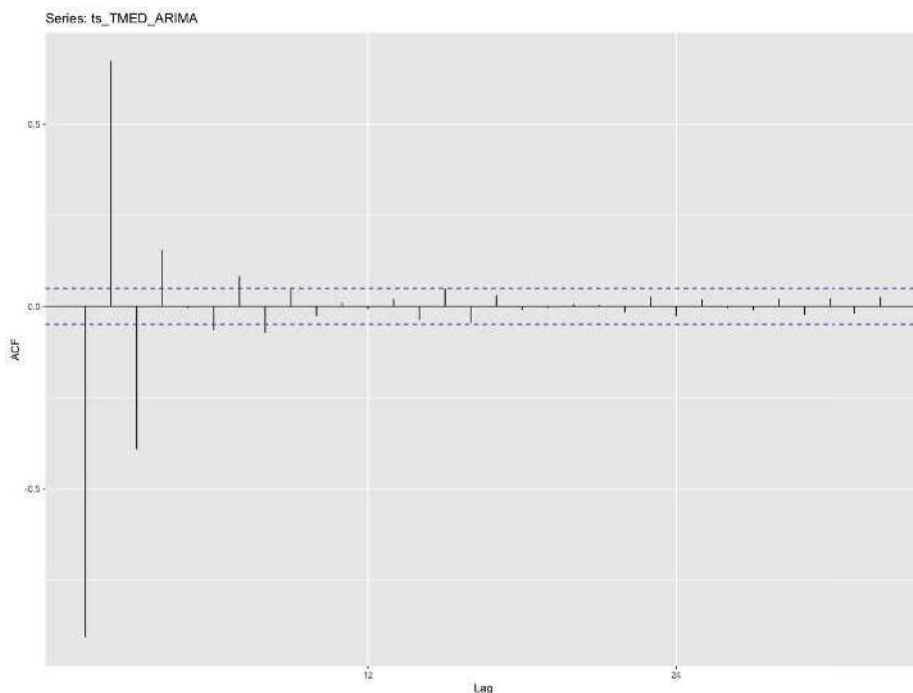


Fig. 8.34. Autocorrelación de la serie transformada para poder aplicar ARIMA.

Comparando el gráfico de autocorrelación de la serie transformada con la original se aprecia que ha desaparecido en gran parte la dependencia del error de valores previos de la serie.

Ahora se puede ajustar un modelo ARIMA. El siguiente paso es determinar los parámetros del modelo (p, d, q). Esta tarea se puede simplificar extraordinariamente utilizando una función que busca el mejor modelo basándose en los valores de AIC, AICc y BIC. Así mismo, no requiere desestacionalizar la serie, dado que también ajusta las transformaciones necesarias.

```
fit_ARIMA_model <- auto.arima(ts_TMED, stepwise = FALSE, approximation = FALSE)
fit_ARIMA_ts <- forecast(fit_ARIMA_model, h = 12)
autoplot(fit_ARIMA_ts, xlim = c(2010, 2016))
```

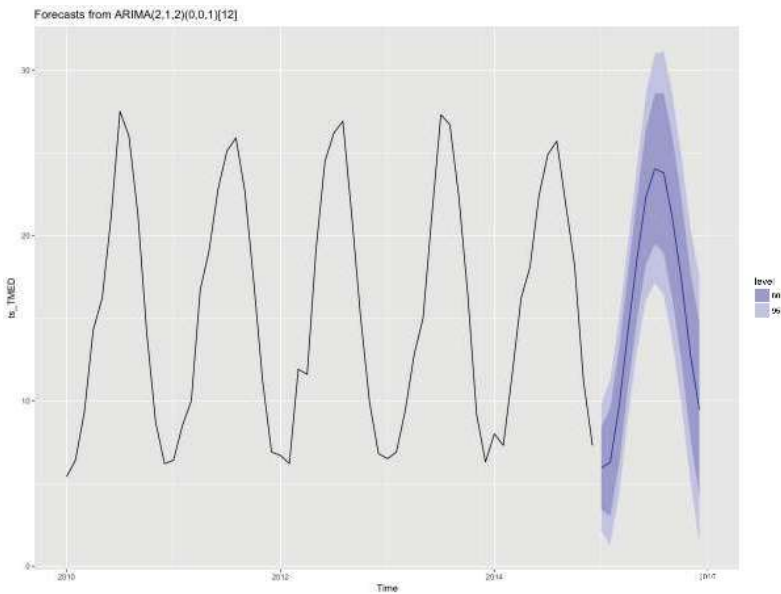


Fig. 8.35 Predicción del año 2015 aplicando auto-ARIMA.

Finalmente, se va a aplicar un modelo TBATS (*Exponential Smoothing State Space Model with Box-Cox Transformation, ARMA Errors, Trend and Seasonal Components*). Es una extensión del modelo de suavizado exponencial.

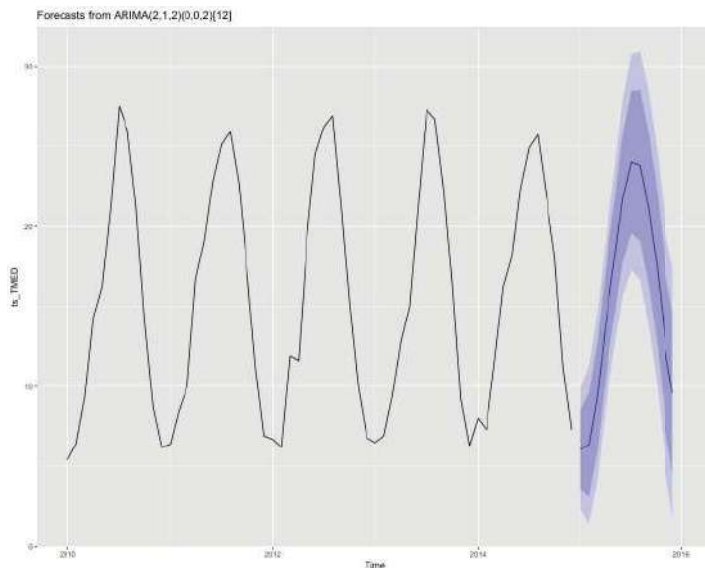


Fig. 8.36. Predicción del año 2015 aplicando TBATS.

```
fit_TBATS_model <- auto.arima(ts_TMED)
fit_TBATS_ts <-forecast(fit_TBATS_model, h = 12)
autoplot(fit_TBATS_ts, xlim = c(2010, 2016))
```

Nótese que TBATS también ha obtenido un modelo ARIMA, con diferentes parámetros que los obtenidos con *auto.arima*.

Podemos representar en una gráfica los valores reales y predichos por cada uno de los diferentes métodos, en ella se aprecia un ajuste bastante aceptable (figura 8.37).

```
ts.plot( fit_naive_ts$mean, fit_naive_ts$lower, fit_naive_ts$upper,
         fit_HW_ts$mean, fit_HW_ts$lower, fit_HW_ts$upper,
         fit_ARIMA_ts$mean, fit_ARIMA_ts$lower, fit_ARIMA_ts$upper,
         fit_TBATS_ts$mean, fit_TBATS_ts$lower, fit_TBATS_ts$upper,
         ts_2015,
         col = c(2,2,0,2,0,3,3, 0, 3, 0, 4, 4, 0, 4, 0, 5, 5, 0, 5, 0, 1),
         lty = c(1,2,2,2,2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1),
         lwd = c(2,0.5,0.5,0.5, 0.5, 2, 0.5, 0.5, 0.5, 0.5, 2, 0.5, 0.5, 0.5,
                 0.5, 2, 0.5, 0.5, 0.5, 0.5, 2),
         gpars=list(xaxt="n"),
         xlab = "Mes", ylab = "Temperatura media °C")
points(seq(2015,2016-(1/12),by=1/12),fit_naive_ts$mean[1:12],pch=19, col = 2)
points(seq(2015,2016-(1/12),by=1/12),fit_HW_ts$mean[1:12],pch= 19, col = 3)
points(seq(2015,2016-(1/12),by=1/12),fit_ARIMA_ts$mean[1:12],pch= 19, col = 4)
points(seq(2015,2016-(1/12), by=1/12),fit_TBATS_ts$mean[1:12],pch= 19,col = 5)
points(seq(2015,2016-(1/12), by=1/12), ts_2015[1:12], pch = 19, col = 1)
axis(1 ,at=seq(2015,2016-(1/12), by=1/12), labels=c("ENE", "FEB", "MAR", "ABR", "MAY",
"JUN", "JUL", "AGO", "SEP", "OCT", "NOV", "DEC"))
legend("topleft", legend = c("Naive","Holt-Winters","ARIMA", "TBATS", "Real"),
      col = c(2, 3, 4, 5, 1), lty = 1, text.width = 0.1, cex = 0.5)
title("Predicción temperatura media 2015", cex.main = 1, sub = "Intervalo 95%", cex.
sub = 1)
```

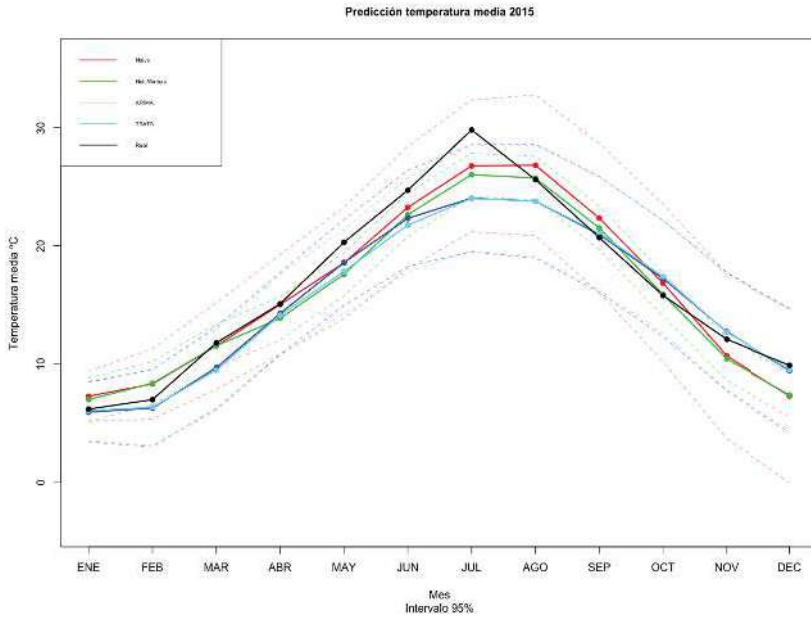


Fig. 8.37. Comparación de la predicción de los diferentes algoritmos aplicados para predecir la temperatura media mensual en el año 2015.

Comparemos la precisión, tanto en entrenamiento como en validación, de las diferentes técnicas utilizadas. En la predicción de las temperaturas medias mensuales, el modelo que mejor realiza la predicción, en términos de error cuadrático medio, es el *naive*, a pesar de que el error en entrenamiento es menor para Holt-Winters. Mientras que TBATS es el método que ha obtenido peores resultados tanto en entrenamiento como en validación. Las predicciones están, en general, bastante ajustadas a los datos reales, en especial para los meses de las estaciones de invierno y otoño. Es para los meses de mayo, junio y julio cuando las predicciones cometen mayor error, posiblemente, fueron meses mucho más calurosos de lo normal. Si se examinan los valores de toda la serie para el mes de julio se verá que es el mayor de toda la serie, 5.2°C superior a la media.

Puede ocurrir que en la predicción de otro año o para un periodo mayor, los errores de los modelos sean diferentes.

```

rmse_nv <- forecast::accuracy(fit_naive_ts)[2]
rmse_hw <- forecast::accuracy(fit_HW_ts)[2]
rmse_ar <- forecast::accuracy(fit_ARIMA_model)[2]
rmse_tb <- forecast::accuracy(fit_TBATS_model)[2]

par(mfrow=c(2,1))

barplot(c(rmse_nv, rmse_hw, rmse_ar, rmse_tb), names.arg = c("Naive", "HW", "Arima",
"TBATS"), ylim = c(0, 3), main = "RMSE entrenamiento 1881-2014")

text(c(0.7,1.9,3.1,4.3), 0.3 + round(c(rmse_nv, rmse_hw, rmse_ar, rmse_tb), 1), labels
= round(c(rmse_nv, rmse_hw, rmse_ar, rmse_tb), 1))

library(Metrics)

barplot(c(rmse(ts_2015, fit_naive_ts$mean), rmse(ts_2015, fit_HW_ts$mean), rmse(ts_2015,
fit_ARIMA_ts$mean), rmse(ts_2015, fit_TBATS_ts$mean)), names.arg = c("Naive", "HW",
"Arima", "TBATS"), ylim = c(0, 3), main = "RMSE año 2015")

text(c(0.7,1.9,3.1,4.3), 0.3 + round(c(rmse(ts_2015, fit_naive_ts$mean), rmse(ts_2015,
fit_HW_ts$mean), rmse(ts_2015, fit_ARIMA_ts$mean), rmse(ts_2015, fit_TBATS_ts$mean)),
1), labels = round(c( rmse(ts_2015, fit_naive_ts$mean), rmse(ts_2015, fit_HW_ts$mean),
rmse(ts_2015, fit_ARIMA_ts$mean), rmse(ts_2015, fit_TBATS_ts$mean)), 1))

```

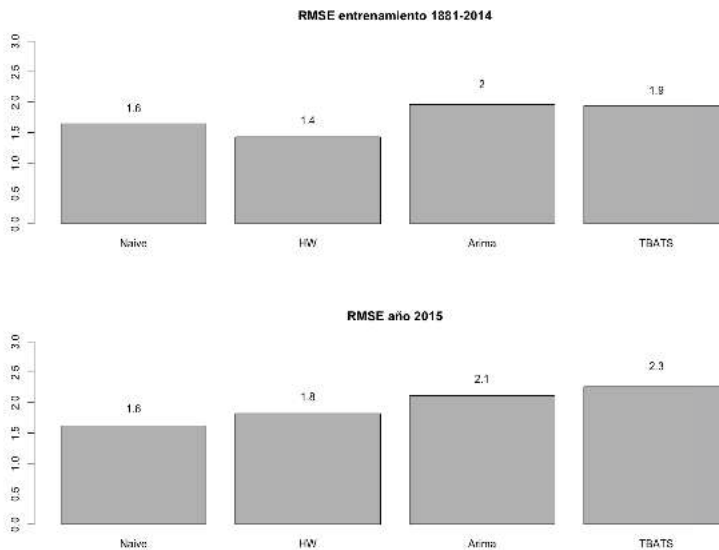


Fig. 8.38. Comparación del error cometido en entrenamiento y validación por los diferentes algoritmos aplicados para predecir la temperatura media mensual en el año 2015.

8.4.3 Detección de anomalías

Ya se trató en el capítulo dedicado a la clasificación la detección de datos anómalos. Los criterios que se seguían estaban relacionados con la densidad o con valores superiores a diferentes estadísticos. En el contexto de las series temporales, un dato anómalo es también el que no sigue, dentro de un umbral, las regularidades del resto de la serie. Por tanto, para la detección de anomalías en series temporales será necesaria la creación de un modelo predictivo y considerar si los datos se alejan de forma excesiva de la predicción.

De nuevo, hay bastantes librerías en R que incorporan diferentes algoritmos para la detección de anomalías en series temporales, aquí se van a mostrar dos de ellos.

El primero, es de la librería *tsoutliers* y utiliza un ARIMA en el que los parámetros son los que fueron obtenidos en *auto.arima*.

```
library(tsoutliers)

outliers <- tso(ts_TMED, tsmethod = "arima", args.tsmethod = list(order = c(2, 1, 2)))
outliers
plot(outliers)
```

Call:

```
structure(list(method = NULL), .Names = "method")
```

Coefficients:

	ar1	ar2	ma1	ma2	A01241
	1.6963	-0.9622	-1.8501	0.8677	-6.8006
s.e.	0.0069	0.0068	0.0100	0.0110	1.4425

sigma^2 estimated as 4.001: log likelihood = -3398.09, aic = 6808.19

Outliers:

	type	ind	time	coefhat	tstat
1	AO	1241	1984:05	-6.801	-4.714

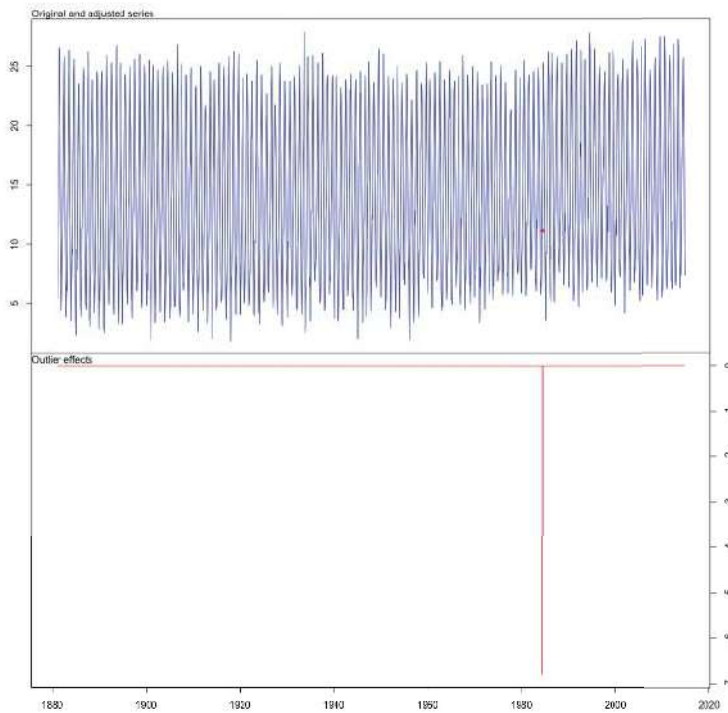


Fig. 8.39. Detección de anomalías en la serie utilizando el modelo ARIMA.

El algoritmo marca el dato correspondiente a mayo de 1984 como anómalo. Efectivamente, repasando la serie, ese valor se corresponde con el mínimo de la serie correspondiente a la temperatura media del mes de mayo, 11.1°C, 5°C inferior a la media para ese mes en la serie.

Otra librería orientada a la detección de anomalías en series es *bfast*. En este caso, realiza la descomposición de la serie en la parte de tendencia, periódica y error con el fin de detectar cambios abruptos en alguna de las partes.

```
library(bfast)
outliers <- bfast(ts_TMED, h=0.15, max.iter=1)
outliers
plot(outliers)

TREND BREAKPOINTS
Confidence intervals for breakpoints
of optimal 2-segment partition:
```

```

Call:
confint.breakpointsfull(object = bp.Vt, het.err = FALSE)

Breakpoints at observation number:

  2.5 % breakpoints 97.5 %
1  1147           1171   1196

Corresponding to breakdates:

  2.5 %   breakpoints 97.5 %
1 1976(7) 1978(7)     1980(8)

SEASONAL BREAKPOINTS: None

```

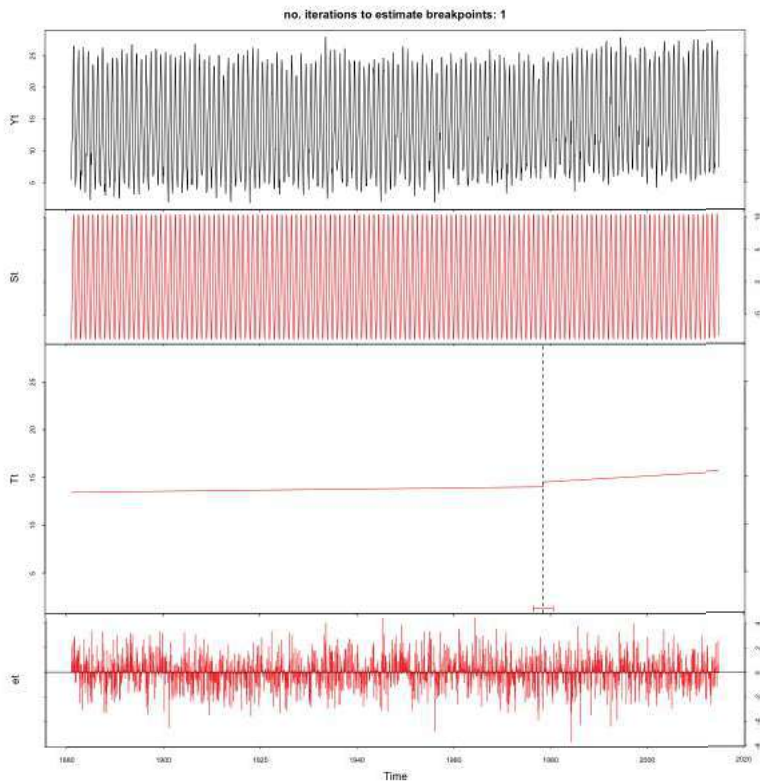


Fig. 8.40. Detección de anomalías en la serie utilizando BFAST.

En este caso, el algoritmo no detecta ningún dato anómalo en la parte periódica, pero encuentra una discontinuidad en la tendencia en torno a 1978, a partir de entonces el ritmo de crecimiento de la temperatura media aumenta.

Análisis de datos espaciales

CAPÍTULO 9

9.1 Introducción

Este capítulo realiza un acercamiento práctico a la representación geográfica que permite identificar un punto de manera única y ubicarlo físicamente en el globo terráqueo, utilizando los conceptos de coordenadas geográficas, latitud y longitud y la utilización de la herramienta RStudio con su conjunto de librerías [R17]:

```
# Librería utilizadas R Studio
library(sp)
library(ggmap)
library(tmap)
library(ggplot2)
library(GADMTTools)
library(rgeos)
library(gdalUtils)
library(gstat)
library(geoR)
library(proj4)
library(crs)
library(raster)
library(maps)
library(readr)
```

Los conceptos empleados tanto en la conversión de datos en puntos de tipo espacial, la determinación de una malla o *grid* y las estimaciones a futuro se realizan con los datos disponibles en AENET [AEMET18]. El sistema geodésico mundial creado a partir de 1984 (WGS 84) representa a la Tierra en una proyección de un elipsoide [GDAL17] [PROJ 17], además nos permite identificar un punto sobre la superficie del planeta tal como se visualiza en la figura 9.1.

```
# Visualizar el mapa del mundo
map("world")
map.axes(cex.axis=0.8)
title(main="Proyección WGS 84", axes= TRUE, xlab="Meridianos", ylab= "Paralelos")
abline(v=seq(-150,150,50))
abline(h=seq(-50,50,50))
abline(h=0,v=0, col="blue")
```

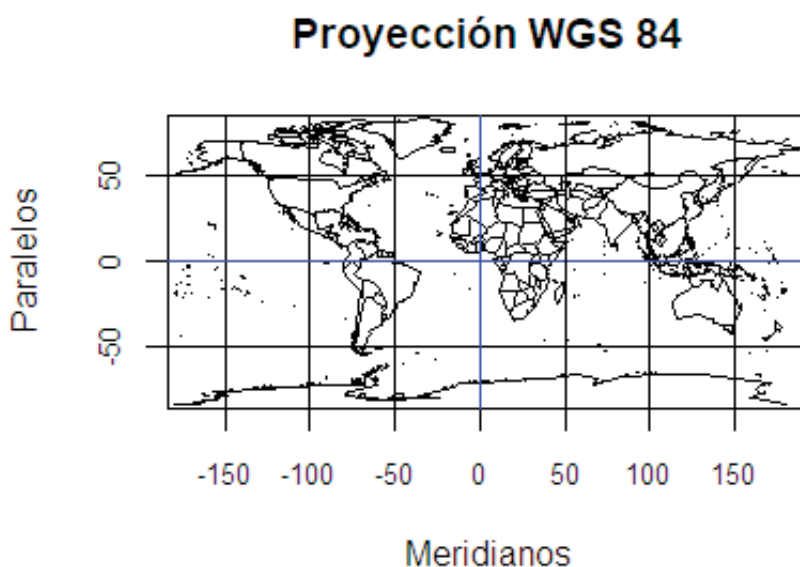


Fig. 9.1. Sistema geodésico Mundial 84.

9.1.1 Datos de tipo espacial

Cuando el atributo de un registro se refiere a su localización estamos hablando que posee una característica de ubicación dentro de la superficie terrestre. La manera más común de representar esta característica es por latitud y longitud, que se asocian a líneas imaginarias trazadas de forma horizontal (este a oeste), denominadas *paralelos*, y de forma vertical (norte a sur), denominadas *meridianos* [ABEL14].

9.1.2 Latitud, longitud

El paralelo central de referencia 0° se denomina *ecuador* y divide al planeta en hemisferio norte y hemisferio sur. La distancia que existe desde cualquier punto al paralelo 0 se denomina *latitud* y su valor oscila entre $0 + 90$ norte y $0 - 90$ sur.

El meridiano central de referencia 0° se denomina *Greenwich* y su valor oscila entre $0 + 180$ este y $0 - 180$ oeste, y divide al planeta entre oriente y occidente. La distancia que existe al meridiano central se denomina *longitud*.

Por ejemplo, conociendo los valores de longitud -4.00 y latitud 40 encontramos el mapa de España, ver figura 9.2.

```
# Coordenadas geográficas de España
España <- c(lon = -4.00, lat = 40.0)

# 1. Obtener el mapa
España_map <- get_map(location = España, zoom = 6)

# 2. Dibujar el mapa
ggmap(España_map)
```

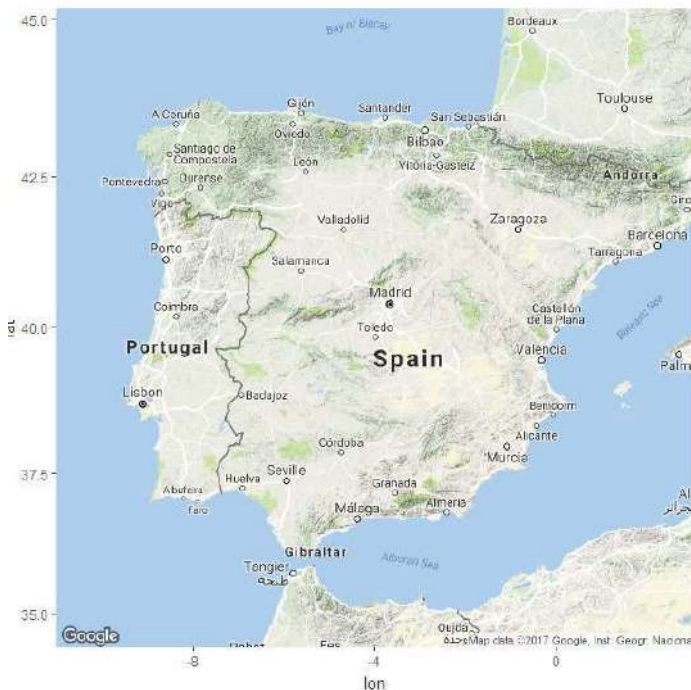


Fig. 9.2. Mapa de España.

9.1.3 La clase de datos Spatial en RStudio

La estructura básica de un objeto de la clase Spatial está dado por dos *slots* o componentes:

- *Bounding box*, que permite conocer el cuadro delimitador de la ubicación espacial (@bbox).
- *CRC (Coordinate Reference System)*, que indica la proyección utilizada, por ejemplo, WGS 84 (@proj4string).

```
getClass("Spatial")
```

```
> getClass("Spatial")
Class "Spatial" [package "sp"]

Slots:

Name:      bbox proj4string
Class:     matrix          CRS

Known Subclasses:
Class "SpatialPoints", directly
Class "SpatialMultiPoints", directly
Class "SpatialGrid", directly
Class "SpatialLines", directly
Class "SpatialPolygons", directly
Class "SpatialPointsDataFrame", by class "SpatialPoints", distance 2
Class "SpatialPixels", by class "SpatialPoints", distance 2
Class "SpatialMultiPointsDataFrame", by class "SpatialMultiPoints", distance 2
Class "SpatialGridDataFrame", by class "SpatialGrid", distance 2
Class "SpatialLinesDataFrame", by class "SpatialLines", distance 2
Class "SpatialPixelsDataFrame", by class "SpatialPoints", distance 3
Class "SpatialPolygonsDataFrame", by class "SpatialPolygons", distance 2
```

Fig. 9.3. Subclases del tipo Spatial.

En este ejemplo de la figura 9.3 se utilizan las subclases SpatialPoints, SpatialGrid y SpatialPolygons, entre otras.

9.1.4 Datos

Partiendo de un conjunto de datos geográficos mostrados en la tabla de la figura 9.4, se realiza el proceso para su transformación en puntos de tipo espacial.

Datos de puntos geográficos

X	y
-3.289711	40.52836
-4.017505	40.659725
-3.546025	40.06714
-3.497784	40.31167
-3.613809	41.006985
-3.765135	40.696068
-3.555593	40.46656
-3.72417	40.45167
-3.711944	40.561386
-3.678095	40.411804
-3.813369	40.448437
-4.141883	40.70634
-4.010556	40.793056
-3.888285	40.889687
-4.249927	40.42775
-4.487227	40.31305
-3.580467	41.135685
-3.311671	40.247227
-4.061107	40.496662
-4.729864	41.071484
-4.679167	40.65889
-5.519077	40.355385
-5.311326	40.1392
-4.742033	40.828964
-5.044091	40.543312
-5.142986	40.350468
-5.012748	40.339664
-5.018689	40.88213
-4.590932	40.288563

Fig. 9.4. Datos de puntos geográficos.

9.2 Tipos de datos

9.2.1 Creación de objetos SpatialPoints

```
#-----Convertir a datos de tipo espacial
puntosxy <- read_csv("C:/doctorado/libro/puntosxy.csv")
View(puntosxy)
class(puntosxy)
plot(puntosxy)
coordinates(puntosxy) <- ~x+y
class(puntosxy)
str(puntosxy)
plot(puntosxy, pch=20, axes=TRUE, main="Ubicación de Puntos")
```

Como se observa en la figura 9.5, la estructura de puntos de tipo espacial consta de tres elementos, se agrega @coords a la clase Spatial, y en la figura 9.6 se aprecia la ubicación de puntos en forma de coordenadas.

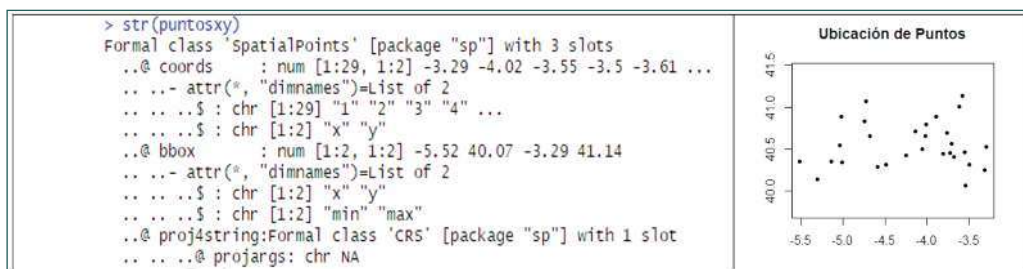


Fig. 9.5. Estructura SpatialPoints.

Fig. 9.6. Ubicación de puntos.

Los *slots* @bbox y @proj4string se mantienen en las estructuras de tipo Spatial, mientras que el *slot* @coordenadas va sufriendo transformaciones y adaptaciones a los tipos línea, polígono y *grid*.

En la figura 9.7 vemos el detalle de la lista de coordenadas geográficas.

```
> puntosxy@coords
      x      y
1 -3.289711 40.52836
2 -4.017505 40.65973
3 -3.546025 40.06714
4 -3.497784 40.31167
5 -3.613809 41.00699
6 -3.765135 40.69607
7 -3.555593 40.46656
8 -3.724170 40.45167
9 -3.711944 40.56139
10 -3.678095 40.41180
11 -3.813369 40.44844
12 -4.141883 40.70634
13 -4.010556 40.79306
14 -3.888285 40.88969
15 -4.249927 40.42775
16 -4.487227 40.31305
17 -3.580467 41.13569
18 -3.311671 40.24723
19 -4.061107 40.49666
20 -4.729864 41.07148
21 -4.679167 40.65889
22 -5.519077 40.35538
23 -5.311326 40.13920
24 -4.742033 40.82896
25 -5.044091 40.54331
26 -5.142986 40.35047
27 -5.012748 40.33966
28 -5.018689 40.88213
29 -4.590932 40.28856
```

Fig. 9.7. Coordenadas geográficas.

Slot Proj4string

Hace referencia a la proyección CRS (*Coordinate Reference System*) que se utiliza, sólo acepta “+ proj = longlat + ellps = WGS84” para coordenadas geográficas.

```
p4s<-CRS(projargs = "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0")
puntosxy@proj4string =p4s
str(puntosxy)

Formal class 'SpatialPoints' [package "sp"] with 3 slots
  ..@ coords      : num [1:29, 1:2] -3.29 -4.02 -3.55 -3.5 -3.61 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:29] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "x" "y"
  ..@ bbox        : num [1:2, 1:2] -5.52 40.07 -3.29 41.14
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2] "x" "y"
  .. .. ..$ : chr [1:2] "min" "max"
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
  .. .. ..@ projargs: chr "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
+towgs84=0,0,0"

plot(puntosxy, pch=20, axes=TRUE, main="Ubicación de Puntos Geograficos", col="blue")
```

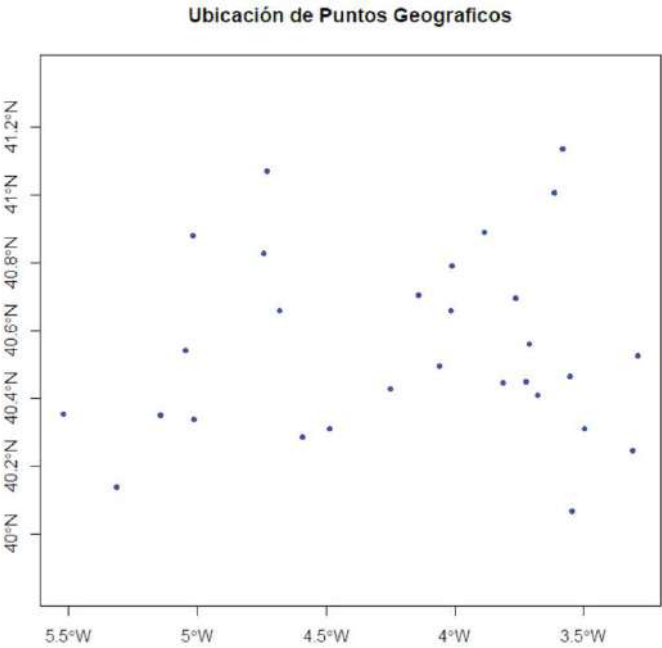


Fig. 9.8. Ubicación de datos tipo Spatial.

Los puntos ahora se encuentran en un formato de tipo espacial, como se constata obteniendo los valores para los tres *slots*.

```
#Estructura final puntosxy
coordinates (puntosxy)

1 -3.289711 40.52836
2 -4.017505 40.65973
3 -3.546025 40.06714
4 -3.497784 40.31167
5 -3.613809 41.00699
6 -3.765135 40.69607
7 -3.555593 40.46656
8 -3.724170 40.45167
9 -3.711944 40.56139
10 -3.678095 40.41180
11 -3.813369 40.44844
```

```

12 -4.141883 40.70634
13 -4.010556 40.79306
14 -3.888285 40.88969
15 -4.249927 40.42775
16 -4.487227 40.31305
17 -3.580467 41.13569
18 -3.311671 40.24723
19 -4.061107 40.49666
20 -4.729864 41.07148
21 -4.679167 40.65889
22 -5.519077 40.35538
23 -5.311326 40.13920
24 -4.742033 40.82896
25 -5.044091 40.54331
26 -5.142986 40.35047
27 -5.012748 40.33966
28 -5.018689 40.88213
29 -4.590932 40.28856

```

```

#Estructura final puntosxy
coordinates(puntosxy)
bbox(puntosxy)
proj4string(puntosxy)

```

```

          min          max
x -5.519077 -3.289711
y 40.067140 41.135685
"+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"

```

En la figura 9.9, después de realizar las transformaciones del sistema de coordenadas y de su proyección, se puede apreciar el mapa de la región donde se encuentran los puntos geográficos, en el fondo se aprecia el mapa proporcionado por Google.

```

# Fondo de Google
merc = CRS("+init=epsg:3857")
WGS84 = CRS("+init=epsg:4326")
puntosxyg = spTransform(puntosxy, WGS84)
bgMap = get_map(as.vector(bbox(puntosxyg)), source = "google", zoom = 8)
# plot con ggmap-google bg:
plot(spTransform(puntosxyg, merc), bgMap = bgMap, pch = 2, cex = .8, col="red")

```



Fig. 9.9. Puntos utilizando fondo del satélite.

Una descripción detallada de los sistemas de coordenadas y proyecciones se encuentra en European Petroleum Survey Group [EPSG 18].

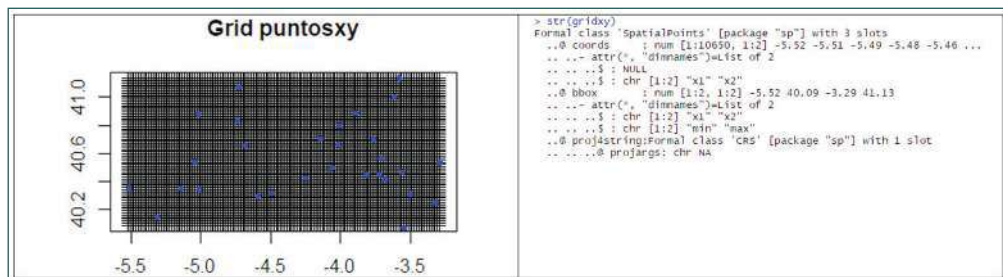
9.2.2 Creación de objetos SpatialGrid

Los objetos de tipo SpatialGrid utilizan tres componentes o *slots*: *grid*, *bbox* y *proj4string*; esta estructura permite representar áreas compuestas por un conjunto de celdas semejanado a la estructura de una malla, que es el principio básico de un ráster.

El *slot* `@grid` tiene tres componentes, *cellcenter*, que indica la esquina inferior izquierda donde comienza el *grid*, *cellsize*, que indica las dimensiones de la celda en alto y ancho, y la tercera componente nos indica el número de celdas tanto en filas como columnas del *grid*.

A continuación se crea una estructura SpatialGrid o también conocida como *malla* sobre los puntos definidos en la sección anterior.

```
# crear grid sobre puntosxy
bbxy<-bbox(puntosxy)
gridxy<-makegrid(puntosxy, cellsize = 0.1, offset = rep(0.5, nrow(bbxy)))
plot(gridxy)
class(gridxy)
coordinates(gridxy) <- c("x1", "x2")
plot(gridxy, main="Grid puntosxy" )
plot(puntosxy, pch=20, col="blue",add=TRUE)
```

Fig. 9.10. Creación de un *grid* sobre puntos xy.

Utilizando GridTopology podemos encontrar un *grid* que se ajuste a nuestro requerimiento en una estructura SpatialGrid.

```
bbxy<-bbox(puntosxy)
csxy <- c(0.1, 0.1)
ccxy <- bbxy[, 1] + (csxy/2)
cdxy <- ceiling(diff(t(bbxy))/csxy)
xy_grd <- GridTopology(cellcentre.offset = ccxy, cellsize = csxy, cells.dim = cdxy)
xy_SG <- SpatialGrid(xy_grd, proj4string = p4s)
str(xy_SG)
plot(xy_SG, main="Grid puntosxy continuo", axes=TRUE)
plot(puntosxy, pch=20, col="blue", axes= TRUE, add=TRUE)
```

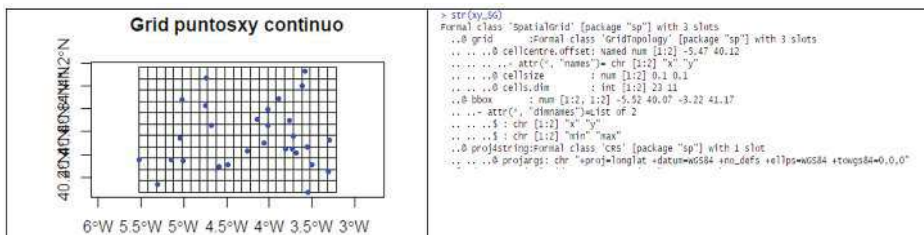


Fig. 9.11. Grid de 22x11 celdas.

9.3 Visualización de datos espaciales

Una tercera estructura que hasta el momento no la hemos analizado pero que es de gran ayuda para este trabajo es la clase SpatialPolygonsDataFrame. Bajo esta estructura se pueden encontrar mapas de todos los países del mundo [GAA17]. Los códigos del país que se desea encontrar corresponden a la codificación ISO 3166 [ISO 17] o a sus siglas.

```
#---- Mapa de España

mapesp = gadm.loadCountries(c("ESP"), level=2, simplify=0.01, basefile="C:/doctorado/
libro/", baseurl="http://biogeo.ucdavis.edu/data/gadm2.8/rds/")

str(mapesp, max.level = 1)

str(mapesp$spdf, max.level = 2)

plotmap(mapesp, title="Mapa de España Level = 2")

listNames(mapesp, level = 2)
```

```
List of 4
$ basename: chr "C:/doctorado/libro/"
$ spdf      :Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
$ level     : num 2
$ stripped: logi FALSE
- attr(*, "class")= chr "GADMWrapper"
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data      :'data.frame': 56 obs. of  15 variables:
..@ polygons  :List of 56
..@ plotOrder : int [1:56] 39 40 15 12 16 21 9 18 10 20 ...
..@ bbox      : num [1:2, 1:2] -18.16 27.64 4.33 43.79
.. -- attr(*, "dimnames")=List of 2
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

En la figura 9.12 se observa la extracción del mapa de España y su división en nivel 2.

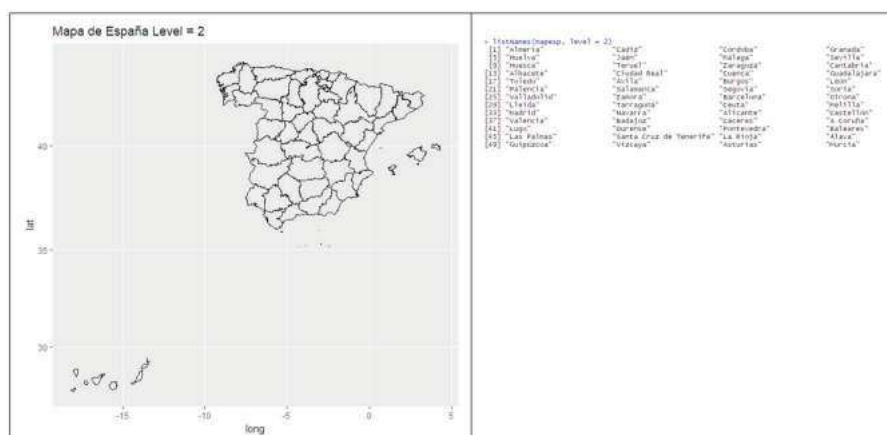


Fig. 9.12. Mapa España en formato spdf.

Se realiza la extracción del área de Ávila y Madrid; sobre esta región, figura 9.13, se llevarán a cabo los distintos procesos de predicción que se presentan en el resto de este capítulo.

```
AvilaMadrid = subset(mapesp, regions=c("Ávila", "Madrid"), level = 2)
plotmap(AvilaMadrid, title = "Ávila y Madrid")
```

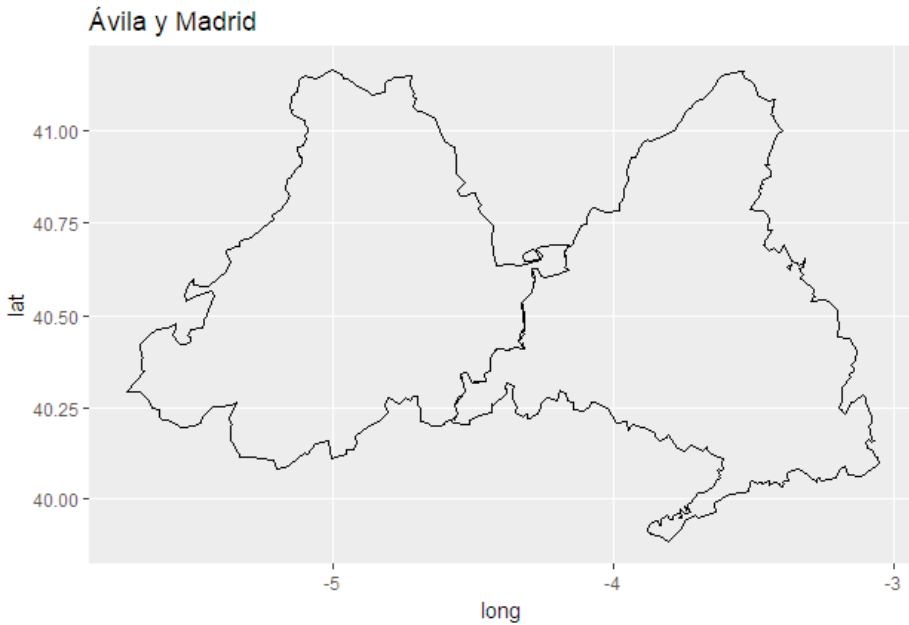


Fig. 9.13. Área de Ávila y Madrid.

9.4 Análisis estadístico (interpolación)

Los datos a ser analizados, figura 9.14, son recogidos de la Agencia Estatal de Meteorología [AEMET 18] en formato json y transformados a formato CSV, la información corresponde a las ciudades de Ávila y Madrid de fecha 6 de diciembre de 2017.

La metodología para desarrollar las estimaciones a futuro es la siguiente:

- Obtención de datos y área de aplicación.
- Crear *grid* ajustado al área de estudio.
- Realizar la estimación de comportamiento a futuro.
 - Utilizando ISW (*Inverse Distance Weighted*).
 - ◊ Presentación de resultados.
 - Utilizando correlación espacial (variograma).
 - ◊ Presentación de resultados.

9.4.1 Análisis exploratorio de datos

La muestra final consta de veintinueve datos, los registros que se han considerado en este ejemplo son solamente la longitud y latitud de las estaciones receptoras y la velocidad del viento.

idestma	lon	lat	prec	alt	vmxa	vv	dy	est	dmxa	ubi	pr	lamin	ta
3170V	-3.289711	40.52880	0	609	1.3	0.3	07	40.52880	182	ALCALA DE HENARES ENC	56	3.2	3.2
3168C	-4.017501	40.658725	0	824	1.6	0.4	342	40.658725	332	AL PEDRETE	61	3.5	3.5
3100B	-3.549025	40.06714	0	510	1.5	0	70	40.06714	343	ARANJUEZ	61	2	2
3102V	-3.427784	40.31167	0	533	1.3	0.4	53	40.31167	147	ARGANDA DEL REY	63	3.7	3.7
3110C	-3.018602	41.009995	0	1020	1.0	0.2	2	41.009995	22	BUITRAGO			
3151E	-3.703132	40.980038	0	1064	2.1	1	170	40.980038	178	COLVINCAR VIEJO/FAME	45	5.9	5.9
3151E	-3.555993	43.46056	0	608	3.1	1.4	190	43.46056	190	MADRID/BARAJAS	50	5	5.1
3154U	-3.72417	40.45167	0	661	2	1	181	40.45167	183	MADRID C. UNIVERSITAR	44	3.9	3.9
3126Y	-3.712444	40.581395	0	740	1	0.2	201	40.581395	202	EL GOUZO (AUTOMATIC	62	1.5	1.5
3135	3.073093	40.411804	0	667	0.8	0	0	40.411804	179	MADRID RETIRO	53	5.3	5.3
3184V	-3.813308	40.448437	0	665				40.448437		POZUELO DE ALARCON (A	63	1	1
3166A	-4.141385	40.70534	0	1532	5.6		257	40.70534	263	ALTO DE LOS LEONES	35	1.4	1.4
2452	-4.012556	40.793056	0	1894	4.8	2.8	14	40.793056	347	NAVACERRADA PUERTO	26	1.7	1.9
3104Y	-3.895285	40.899027	0	1139	1.3	0.5	230	40.899027	227	EL PAULAR. KASCARRIA (71	-2	-2
3385	-4.248927	40.427725	0	790	0.4	0.1		40.427725	21	ROBLEDO DE CHAVELA	51	1.7	1.7
3140V	-4.487257	40.31165	0	846				40.31165		ROZAS DE DUFRITO REAL	63	5.7	5.7
3111D	-3.580467	41.135685	0	1150	1.5	0	41	41.135685	29	SOMOSIERMA	37	0.9	1.1
3123V	-3.911673	40.347222	0	594				40.347222		TELVAES	39	4.4	4.4
3343Y	-4.051107	40.499002	0	884	1.3	0.1	84	40.499002	70	VALDEAVORILLO	57	3.7	3.7
2456B	-4.728804	41.071434	0	820	1.5	0.5	21	41.071434	335	AREVALO	50	-3	-3
2444	-4.679167	40.45585	0	1130	1.4	0.5	17	40.45585	53	AVILA	70	2.6	2.6
2828Y	-5.510077	40.355385	0	1012	2	0.8	245	40.355385	280	BARCO DE AVILA	54	3.5	3.5
3422U	-5.812426	40.1894	0	750	1.7	0.9	399	40.1894	347	CANDELEDA (AUTOMATI	72	6	6
2431E	-4.742038	40.828904	0	920	0.6	0.4	204	40.828904	200	GOTARENCUERA (AUTOM	81	0.4	0.4
2436V	-5.044091	40.543512	0	1178	5.7	0.9	205	40.543512	187	MILLOUTRILLO	57	4.8	4.8
2611A	-5.142098	40.350485	0	1535	2.2	1.3	11	40.350485	28	NAVARREDONDA DE GON	33	1.4	1.4
3119D	-5.012748	40.339584	0	1205	1.7	0.8	175	40.339584	205	PUERTO EL PICO	72	-2.6	-2.6
2332Y	-3.918889	40.88228	0	925	0.8	0.4	252	40.88228	247	RIVILLA DE BARAJAS	89	0.2	0.2

Fig. 9.14. Información obtenida y procesada.

9.4.1.1 Crear *grid* Ávila-Madrid

Con el área de Ávila y Madrid obtenida en la sección anterior se procede a crear el *grid* o malla de la superficie a investigar.

```
#-----Crear Grid Madrid Avila

class(AvilaMadrid)

madavila<-AvilaMadrid$spdf

str(madavila, max.level = 3)gridma<-makegrid(madavila)

bbox(madavila)

bb <- bbox(madavila)

class(gridma)

str(gridma, level=1)

coordinates(gridma) <- c("x1", "x2")

cs <- c(0.1, 0.1)

cc <- bb[, 1] + (cs/2)

cd <- ceiling(diff(t(bb))/cs)

ma_grd <- GridTopology(cellcentre.offset = cc,cellsize = cs, cells.dim = cd)

p4s<-proj4string(AvilaMadrid$spdf)

ma_SG <- SpatialGrid(ma_grd, proj4string = p4s)

plot(ma_SG, main="Grid Avila/Madrid", axes=TRUE)

plot(madavila, add=TRUE)
```

```

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data      :'data.frame': 2 obs. of  15 variables:
.. ..$ OBJECTID : int [1:2] 19 34
.. ..$ ID_0      : int [1:2] 215 215
.. ..$ ISO       : chr [1:2] "ESP" "ESP"
.. ..$ NAME_0    : chr [1:2] "Spain" "Spain"
.. ..$ ID_1      : int [1:2] 5 8
.. ..$ NAME_1    : chr [1:2] "Castilla y León" "Comunidad de Madrid"
.. ..$ ID_2      : int [1:2] 18 33
.. ..$ NAME_2    : chr [1:2] "Ávila" "Madrid"
.. ..$ HASC_2    : chr [1:2] "ES.CL.AV" "ES.MD.MD"
.. ..$ CCN_2     : int [1:2] NA NA
.. ..$ CCA_2     : chr [1:2] "05" "28"
.. ..$ TYPE_2    : chr [1:2] "Provincia" "Provincia"
.. ..$ ENGTYP_2  : chr [1:2] "Province" "Province"
.. ..$ NL_NAME_2 : chr [1:2] "" ""
.. ..$ VARNAME_2 : chr [1:2] "" ""
..@ polygons    :List of 2
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
..@ plotOrder   : int [1:2] 1 2
..@ bbox        : num [1:2, 1:2] -5.74 39.88 -3.05 41.17
.. ..- attr(*, "dimnames")=List of 2
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot

```

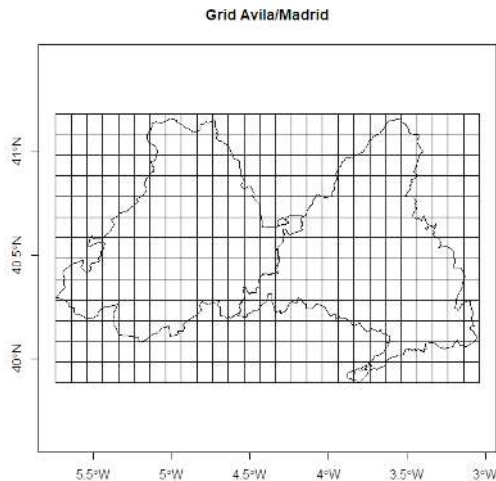


Fig. 9.15. *Grid total Ávila-Madrid.*

```

class(ma_SG)
class(ma_SG)
attr("package")
class(madavila)
attr("package")

# Ajustar el grid
pmadav<-SpatialPolygons(madavila@polygons)
gridpointsma <- SpatialPoints(ma_SG, proj4string = CRS(projection(pmadav)))
cropped_gridpointsmadav <- crop(gridpointsma, pmadav,drop=TRUE)
spgridma <- SpatialPixels(cropped_gridpointsmadav)
str(spgridma)
coordnames(spgridma) <- c("x", "y")
plot(spgridma, main="Grid Ajustado Límites Ávila/Madrid")

[1] "SpatialGrid"
[1] "sp"
[1] "SpatialPolygonsDataFrame"
[1] "sp"

```

Es necesario ajustar el *grid* al área de estudio:

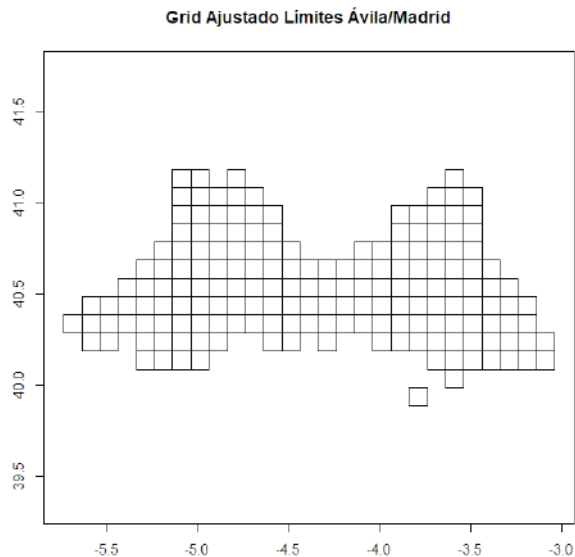


Fig. 9.16. *Grid* ajustado Ávila-Madrid.

Desde este punto es necesario trabajar con el *grid* ajustado. En las secciones siguientes se encuentran procesos de interpolación utilizando dos procesos distintos con un mismo fin: tratar de establecer la velocidad del viento para lugares donde no se tienen datos reales.

9.4.2 Interpolación IDW (*Inverse Distance Weighted*)

El archivo presentado muestra, en base al campo *vmax*, la velocidad máxima del viento en lugares donde no se posee un valor real tomado, utilizando el proceso de *Inverse Distance Weighted*, IDW [BPG13].

IDW se basa en determinar los valores de celda a través de una combinación ponderada linealmente de un conjunto de puntos de la muestra.

$$\hat{Z}(s_0) = \frac{\sum_{i=1}^n w(s_i) Z(s_i)}{\sum_{i=1}^n w(s_i)}$$

Donde $w(s_i) = ||s_i - s_0||^{-p}$, que indica la distancia euclidiana y p una potencia de ponderación de distancia inversa, por defecto es igual a dos.

```
completona <- read_csv("C:/doctorado/libro/completona.csv")
coordinates(completona) <- c("x", "y")
idwma <- idw(vmax) ~ 1, completona, spgridma)
as.data.frame(idwma)[1:5, ]
as.data.frame(idwma$var1.pred)[1:10,]
spplot(idwma["var1.pred"], main="IDW Avila Madrid", panel = panel.gridplot, xlab =
TRUE, ylab = TRUE)
```

En la figura 9.17 se observa el resultado de la estimación utilizando IDW.

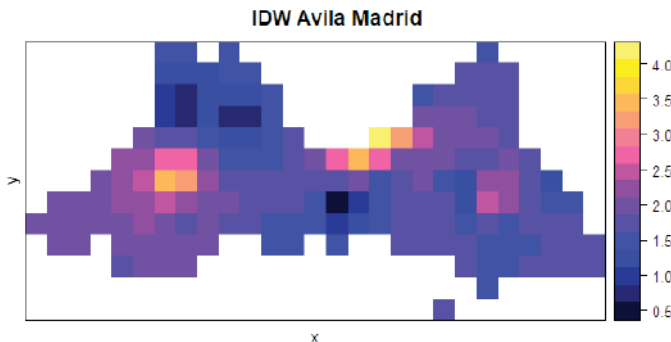


Fig. 9.17. *Inverse Distance Weighted* Ávila-Madrid.

9.4.2.1 Presentación de resultados IDW

Vemos en la figura 9.18 la representación gráfica cartográfica de los resultados obtenidos.

```
image(idwma)
contour(idwma, add=TRUE, nlevels=25)
plot(madavila, add=TRUE)
title(main = "Predicción Velocidad del viento Avila/Madrid IDW")
```

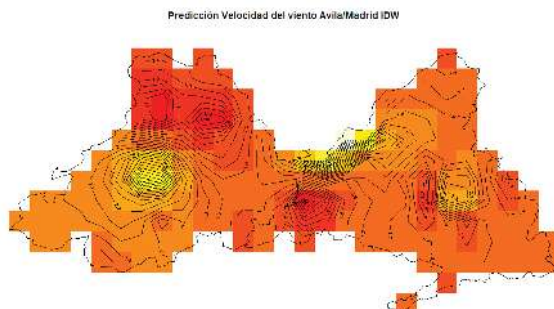


Fig. 9.18. Predicción velocidad del viento utilizando IDW.

9.4.3 Correlación espacial (variograma)

La geoestadística utiliza el concepto de función aleatoria para encontrar los valores no determinísticos sobre una región D . Si x recorre la región, se obtiene una serie de variables aleatorias, $Z = \{Z(x), x \in D\}$, que constituyen una función aleatoria, $Z(s) = m + e(s)$, y por tanto su valor medio es $E(Z(s)) = m$.

El variograma (semivariograma) viene definido por la función $\gamma(h) = \frac{1}{2}E\{[Z(x+h) - Z(x)]^2\}$ donde h es la distancia de separación a un punto.

Para construir la función de predicción se debe establecer la correlación espacial, y para ello se debe comenzar por establecer el variograma siguiendo los pasos indicados en las siguientes secciones.

9.4.3.1 Selección modelo de variograma

La figura 9.19 indica el modelo de variograma a utilizar, la librería de utilidad para los siguientes cálculos se encuentra en *Gstat* [Gstat17].

```
v.eye <- eyeFit(variog(as.geodata(completona["vmax"]), max.dist = 1))
ve.fit <- as.vgm.variomodel(v.eye[[1]])
```

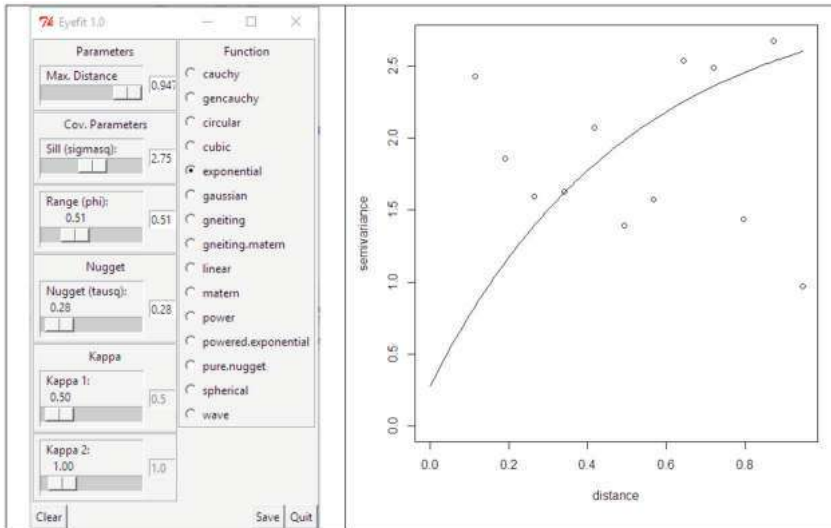


Fig. 9.19. Modelo de variograma utilizado.

Utilizando un modelo exponencial con los parámetros de *distance* 0.947, *sill* 2.75, *range* 0.51 y *nugget* 0.28 se procede a realizar la estimación utilizando la función *krige*, figura 9.20.

```
mma<-vgm(2.75, "Exp",0.51, 0.28)
pma <- krige((vmax)~1, completona, spgridma, model = mma)
spplot(pma["var1.pred"], main = "Predicción kriging ordinario")
spplot(pma["var1.var"], main = "Varianza kriging ordinario")
```

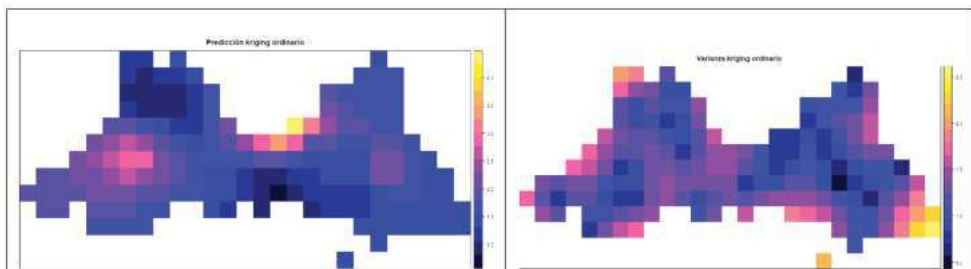
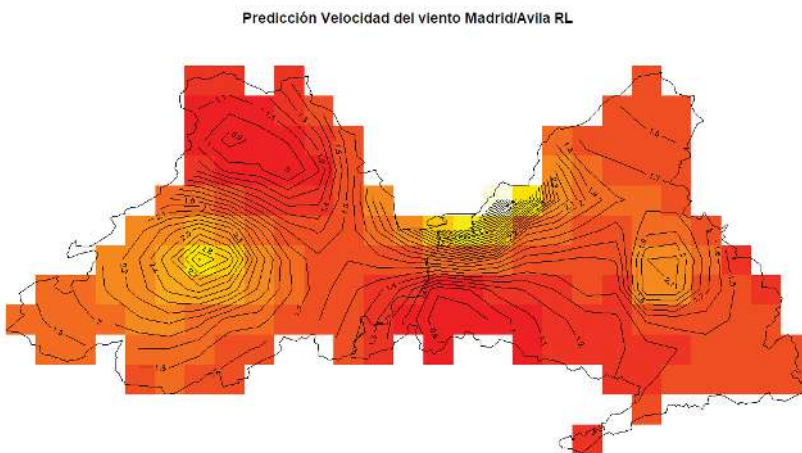


Fig. 9.20. Predicción utilizando correlación espacial.

9.4.3.2 Presentación de resultados

Para finalizar, en la figura 9.21 se presenta el mismo ejemplo de predicción del apartado anterior, el cálculo de la velocidad del viento, utilizando ahora la correlación espacial basada en el modelo de variograma presentado anteriormente.

```
image(pma["var1.pred"])  
contour(pma, add=TRUE, nlev=25)  
plot(madavila, add=TRUE)  
title(main = "Predicción Velocidad del viento Madrid/Avila RL")
```



Bibliografía

- [ABEL14] Abella, I. “Meridianos y paralelos. Latitud y longitud”, *Aprende geografía, historia, arte, TIC y metodología de enseñanza-aprendizaje*. 10-oct.-2014.
- [ACM90] *Special Issue on Heterogeneous Database Systems*. ACM Computing Surveys, septiembre 1990.
- [ACM91] *Special Issue on Next Generation Database Systems*. Communications of the ACM, octubre, 1991.
- [ACM95] *Special Issue on Digital Libraries*. Communications of the ACM, mayo, 1995.
- [ACM96a] *Special Issue on Data Mining*. Communications of the ACM, noviembre, 1996.
- [ACM96b] *Special Issue on Electronics Commerce*. Communications of the ACM, junio, 1996.
- [ADRI96] Adriaans, P. y Zantinge, D. *Data Mining*. Addison Wesley, Massachusetts, 1996.
- [AEMET 18] Agencia Estatal de Meteorología (AEMET). Gobierno de España. Disponible en: <http://www.aemet.es/es/portada>. [Acceso: 30-enero-2018].
- [AFCE97] *Proceedings of the First Federal Data Mining Symposium*, Washington D.C., diciembre, 1997.
- [AFSB83] *Air Force Summer Study Board Report on Multilevel Secure Database Systems*. Department of Defense Document, 1983.
- [AGRA93] Agrawal, A. et al., *Database Mining a Performance Perspective*. IEEE Transactions on Knowledge and Data Engineering, vol. 5, diciembre, 1993.
- [ALBA13] Alba Castro, J. L. “Máquinas de vectores soporte (SVM)”. Consultado el 14 de junio de 2013. Disponible en: <https://web.archive.org/web/20140801145654/http://www.gts.tsc.uvigo.es/~jalba/doctorado/SVM.pdf>.
- [ALEM10] Alemdar H. y Ersoy C. *Wireless sensor networks for healthcare: a survey*. Computer Networks, vol. 54, Issue 15:2688–2710, 2010.
- [ATZO10] Atzori, L., Iera, A. y Morabito, G. *The Internet of Things: A survey*. Computer Networks, vol. 54, Issue 15, 28 octubre 2010, pp. 2787-2805.
- [BANE87] Banerjee, J. et al. *A Data Model for Object-Oriented Applications*, ACM Transactions on Office Information Systems, vol. 5, 1987.

- [BELL92] Bell D. y Grimson, J. *Distributed Database Systems*. Addison Wesley, Massachusetts, 1992.
- [BENS95] Bensley, E. et al. *Evolvable Systems Initiative for Realtime Command and Control Systems*. Proceedings of the first IEEE Complex Systems Conference, Orlando, Florida, noviembre, 1995.
- [BERN87] Bernstein, P. et al., *Concurrency Control and Recovery in Database Systems*. Addison Wesley, Massachusetts, 1987.
- [BERR97] Berry, M. y Linoff, G. *Data Mining Techniques for Marketing, Sales, and Customer Support*. John Wiley, Nueva York, 1997.
- [BGV92] Boser B., Guyon, I. y Vapnik, V. *A training algorithm for optimal margin classifiers*. Proceedings of the fifth annual Workshop on Computational Learning Theory. Proceeding COLT '92, 1992, pp. 144-152.
- [BPG13] Bivand, R. S., Pebesma, E. y Gómez-Rubio, V. *Applied Spatial Data Analysis with R*. Springer New York, Nueva York, 2013.
- [BRIS96] Briscoe, G. y Caelli, T. *A Compendium of Machine Learning. Vol. 1: Symbolic Machine Learning*. Ablex Publishing Corporation, Nueva Jersey, 1996.
- [BROD84] Brodie, M. et al. *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer Verlag, Nueva York, 1984.
- [BROD86] Brodie, M. y Mylopoulos, J. *On Knowledge Base Management Systems*. Springer Verlag, Nueva York, 1986.
- [BROD88] Brodie, M. et al. *Readings in Artificial Intelligence and Databases*. Morgan Kaufmann, California, 1988.
- [BROD95] Brodie M. y Stonebraker, M. *Migrating Legacy Databases*, Morgan Kaufmann, California, 1995.
- [BUNE82] Buneman, P. *Functional Data Model*. ACM Transactions on Database Systems, 1983.
- [CARB98] Carbone, P. *Data Mining*. Handbook of Data Management, Auerbach Publications, (ed.: B. Thuraisingham), Nueva York, 1998.
- [CERI84] Ceri, S. y Pelagatti, G. *Distributed Databases, Principles and Systems*. McGraw Hill, Nueva York, 1984.
- [CHAN73] Chang C., y Lee R. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Nueva York, 1973.
- [CHAU95] Chauvin, Y. y Rumelhart, D. *Backpropagation: Theory, Architectures, and Applications*. Hillsdale, Lawrence Erlbaum Assoc., Nueva Jersey, 1995.

- [CHEN76] Chen, P. *The Entity Relationship Model - Toward a Unified View of Data*. ACM Transactions on Database Systems, vol. 1, 1976.
- [CHOR94] Chorafas, D. *Intelligent Multimedia Databases*. Prentice Hall, Nueva Jersey, 1994.
- [CHY96] Chen, M. S., Han, J. y Yu, P. S. *Data mining: An overview from a database perspective*. IEEE Trans. Knowledge and Data Engineering, 8:866-883, 1996.
- [CLAR89] Clark, P. y Niblett, T. *The CN2 Induction Algorithm*. *Machine Learning*. 3:261-283, 1989.
- [CLEA95] Cleary, J. G. y Trigg, L. E. *K*: An Instance-based Learner Using an Entropic Distance Measure*. Proceedings of the 12th International Conference on Machine learning, 1995, pp. 108-114.
- [CLIF96a] Clifton, C. y Morey, D. *Data Mining Technology Survey*. Private Communication, Bedford, Massachusetts, diciembre, 1996.
- [CLIF96b] Clifton, C. y Marks, D. *Security and Privacy Issues for Data Mining*. Proceedings of the ACM SIGMOD Conference Workshop on Data Mining, Montreal, Canadá, junio, 1996.
- [CLIF98a] Clifton, C. *Image Mining*. Private Communication, Bedford, Massachusetts, julio, 1998.
- [CLIF98b] Clifton C. *Privacy Issues for Data Mining*. Private Communication, Bedford, Massachusetts, abril, 1998.
- [CODD70] Codd, E. F. *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, vol. 13, 1970.
- [COOL98] Cooley, R. *Taxonomy for Web Mining*. Private Communication, Bedford, Massachusetts, agosto, 1998.
- [DARP98] *Workshop on Knowledge Discovery in Databases*. Defense Advanced Research Projects Agency, Pittsburgh, Pensilvania, junio, 1998.
- [DAS92] Das, S. *Deductive Databases and Logic Programming*. Addison Wesley, Massachusetts, 1992.
- [DATE90] Date, C. J. *An Introduction to Database Management Systems*. Addison Wesley, Massachusetts, 1990 (sexta edición publicada en 1995 por Addison Wesley).
- [DCI96] Proceedings of the DCI Conference on Databases and Client Server Computing, Boston, Massachusetts, marzo, 1996.
- [DE98] Proceedings of the 1998 Data Engineering Conference, Orlando, Florida, febrero, 1998.

- [DEAT97] Deaton, A. *The Analysis of Household Surveys. A Microeconomic Approach to Development Policy*. The World Bank. The Johns Hopkins University Press, 1997.
- [DECI] Decision Support Journal, Elsevier/North Holland Publications.
- [DEGR86] DeGroot, T. *Probability and Statistics*. Addison Wesley, Massachusetts, 1986.
- [DEVL88] Devlin, B. y Murphy, P.T. *An Architecture for a Bussiness and Information System*. IBM Sys, J 27, n.º 1, 1988.
- [DEVO95] Devore, J. L. *Probability and Statistics for Engineering and the Sciences*. Cuarta ed. Duxbury Press, Nueva York, 1995.
- [DFL96] DiNardo, J., Fortin, N. y Lemieux, T. *Labor Market Institutions and the Distribution of Wages, 1973-1992: a Semiparametric Approach*. *Econometrica*, vol. 64, n.º 5, septiembre, 1996.
- [DIGI95] Proceedings of the Advances in Digital Libraries Conference, McLean, (ed.: N. Adam *et al.*), Virginia, mayo, 1995.
- [DIST98] Workshop on Distributed and Parallel Data Mining, Melbourne, Australia, abril, 1998.
- [DMH94] Data Management Handbook, Auerbach Publications, (ed.: B. von Halle and D. Kull), Nueva York, 1994.
- [DMH95] Data Management Handbook Supplement, Auerbach Publications, (ed.: B. von Halle and D. Kull), Nueva York, 1995.
- [DMH98] Data Management Handbook Supplement, Auerbach Publications, (ed.: B. von Halle and D. Kull), Nueva York, 1998.
- [DOBS90] Dobson, A. J. *An Introduction to Generalized Linear Models*. Chapman and Hall, Nueva York, 1990.
- [DOD94] Proceedings of the 1994 DoD Database Colloquium, San Diego, California, agosto, 1994.
- [DOD95] Proceedings of the 1994 DoD Database Colloquium, San Diego, California, agosto, 1995.
- [DSV98] DSV Laboratory, *Inductive Logic Programming*. Private Communication, Estocolmo, Suecia, junio, 1998.
- [DUDA73] Duda, R. y Hart, P. *Pattern Classification and Scene Analysis*. John Wiley & Sons, Nueva York, 1973.

- [EEE89] *Parallel Architectures for Databases*. IEEE Tutorial, (ed.: A. Hurson *et al.*), 1989.
- [ELP97] Summer School on Inductive Logic Programming, Praga, República Checa, septiembre, 1998.
- [EPSG 18] Códigos EPSG de Sistemas de Referencia. Red de Información Ambiental de Andalucía, Consejería de Medio Ambiente y Ordenación del Territorio, Junta de Andalucía. Disponible en:
- http://www.juntadeandalucia.es/medioambiente/site/rediam/menuitem.04dc44281e5d53cf8ca78ca731525ea0?vgnextoid=2a412abcb86a2210VgnVCM1000001325e50aRCRD&lr=lang_es. [Acceso: 27-enero-2018].
- [FAYY96] Fayyad, U. *et al.* *Advanced in Knowledge Discovery and Data Mining*. MIT Press, Massachusetts, 1996.
- [FELD95] Feldman, R. y Dagan, I. *Knowledge Discovery in Textual Databases (KDT)*. Proceedings of the 1995 Knowledge Discovery in Databases Conference, Montreal, Canadá, agosto, 1995.
- [FISH87] D. Fisher, *Improving inference through conceptual clustering*. In Proc. 1987 AAAI Conference, Seattle, Washington, julio, 1987, pp. 461-465.
- [FOWL97] Fowler, M. *et al.*, *UML Distilled: Applying the Standard Object Modeling Language*. Addison Wesley, Massachusetts, 1997.
- [FRAN98] Frank, E. y Witten, I. H. *Generating Accurate Rule Sets Without Global Optimization*. In Shavlik, J., ed., *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, California, 1998.
- [FROS86] Frost, R. *On Knowledge Base Management Systems*. Collins Publishers, Reino Unido, 1986.
- [FU94] Fu, L. *Neural Networks in Computer Intelligence*. McGraw Hill, Nueva York, 1994.
- [FURN87] Furnas, G. W. *et al.* *The vocabulary problem in human system communication*. Communications of the ACM, 30, n.º 11, noviembre, 1987.
- [GAA17] *Global Administrative Areas. Boundaries without limits*. Disponible en: <http://www.gadm.org/>. [Acceso: 20-dic.-2017].
- [GALL78] Gallaire, H. y Minker, J. *Logic and Databases*. Plenum Press, Nueva York, 1978.
- [GARC13] García-Morchon, Ó., Loong, S., Kumar, S., Moreno-Sánchez, P., Vidal-Meca, F. y Ziegeldorf, J. H. *Securing the IP-based Internet of Things with HIP and*

DTLS. WiSec '13, Proceedings of the sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2013.

- [GDAL 17] GDAL (Geospatial Data Abstraction Library). Disponible en: <http://www.gdal.org/>. [Acceso: 09-dic.-2017].
- [GOLD89] Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Addison-Wesley, Massachusetts, 1989.
- [GRIN95] Grinstein, G. y Thuraisingham, B. *Data Mining and Visualization: A Position Paper*. Proceedings of the Workshop on Databases in Visualization, Atlanta, Georgia, octubre, 1995.
- [GROO86] DeGroot, T. *Probability and Statistics*. Addison Wesley, Massachusetts, 1986.
- [GRUP98] Grupe F. y Owrang, M. *Database Mining Tools*. Handbook of Data Management Supplement, Auerbach Publications (ed.: B.Thuraisingham), Nueva York, 1998.
- [GSTA17] <https://cran.r-project.org/web/packages/gstat/gstat.pdf> (acceso: 11-nov.-2017).
- [GUBB13] Gubbi, J., Buyya, R., Marusic, S. y Palaniswami, M. *Internet of Things (IoT): A vision, architectural elements, and future directions*. Future Generation Computer Systems, vol. 29, Issue 7, septiembre, 2013, pp. 1645-1660.
- [HAN98] Han, J. y Kamber, M. *Data Mining: Concepts and Techniques*. Academic Press, 2001.
- [HAN98] Han, J. *Data Mining*. Keynote Address, Second Pacific Asia Conference on Data Mining, Melbourne, Australia, abril, 1998.
- [HARD94] Härdle, W. y Linton, O. *Applied Nonparametric Methods*. Handbook of Econometrics, vol. IV, capítulo 38, Elsevier Science, 1994.
- [HEAR96] Hearst, M. y Hirsh, H. (eds.). *Papers from the AAAI Spring Symposium on Machine Learning in information Access*, Stanford, marzo 25-27, 1996. Disponible en: <http://www.parc.xerox.com/istl/projects/mlia>.
- [HINK88] Hinke, T. *Inference and Aggregation Detection in Database Management Systems*. Proceedings of the 1988 Conference on Security and Privacy, Oakland, California, abril, 1988.
- [HMM86] Hong, J., Mozetic, I. y Michalski, R. S. *AQ15: Incremental learning of attribute-based descriptions from examples, the method and user's guide*. In Report ISG 85-5, UIUCDCS-F-86-949, Department of Computer Science, University of Illinois at Urbana-Champaign, 1986.

- [HOLT93] Holte, R. C. *Very simple classification rules perform well on most commonly used datasets*. Machine Learning, vol. 11, 1993, pp. 63-91.
- [HUI13] Hui, S., Zhuohua, L., Jiafu, W. y Keliang Z. *Security and Privacy in Mobile Cloud Computing Wireless*. Communications and Mobile Computing Conference, 2013.
- [IBM16], <http://www.ibm.com/analytics/us/en/technology/data-science/>, 2016.
- [IBM17] Web IBM, *Data Science*. Acceso: enero, 2017.
- [ICTA97] Panel on Web Mining, International Conference on Tools for Artificial Intelligence, Newport Beach, California, noviembre, 1997.
- [IEEE89] *Parallel Architectures for Databases*. IEEE Tutorial (ed.: A. Hurson *et al.*), 1989.
- [IEEE91] *Special Issue in Multidatabase Systems*. IEEE Computer, diciembre, 1991.
- [IEEE98] IEEE Data Engineering Bulletin, junio, 1998.
- [IFIP] Proceedings of the IDFIP Conference Series in Database Security, Holanda del Norte.
- [IFIP97] *Web Mining*. Proceedings of the 1997 IFIP Conference in Database Security, Lake Tahoe, California, agosto, 1997.
- [INMO88] Inmon, W. *Data Architecture: The Information Paradigm*. Wellesley, QED Information Sciences, Massachusetts, 1988.
- [INMO93] Inmon, W. *Building the Data Warehouse*. John Wiley and Sons, Nueva York, 1993.
- [ISO17] *ISO 3166 Country Codes*. Disponible en: <https://www.iso.org/iso-3166-country-codes.html>. [Acceso: 23-dic.-2017].
- [JAME] James, M. *Classification Algorithms*. John Wiley & Sons, Nueva York, 1985.
- [JOHN97] John, G. H. *Enhancements to the Data Mining Process*. Ph. D. Thesis, Computer Science Dept., Stanford University, 1997.
- [JUNG98] Junglee Corporation, *Virtual Database Technology, XML, and the Evolution of the Web*. IEEE Data Engineering Bulletin, junio, 1998 (autores: Prasad, K. y Rajaraman, R.).
- [KDD95] Proceedings of the First Knowledge Discovery in Databases Conference, Montreal, Canadá, agosto, 1995.
- [KDD96] Proceedings of the Second Knowledge Discovery in Databases Conference, Portland, Oregón, agosto, 1996.

- [KDD97] Proceedings of the Third Knowledge Discovery in Databases Conference, Newport Beach, California, agosto, 1997.
- [KDD98] Proceedings of the Fourth Knowledge Discovery in Databases Conference, Nueva York, agosto, 1998.
- [KDP98] *Panel on Privacy Issues for Data Mining*. Knowledge Discovery in Databases Conference, Nueva York, agosto, 1998.
- [KDT98] *Tutorial on Commercial Data Mining Tools*. Knowledge Discovery in Databases Conference, agosto, 1998 (moderadores: J. Elder y D. Abbott).
- [KIM85] Kim, W. *et al. Query Processing in Database Systems*. Springer Verlag, Nueva York, 1985.
- [KIM14] Jaewoo, K. *M2M Service Platforms: Survey, Issues, and Enabling Technologies*. IEEE Communications Surveys & Tutorials, vol. 16, issue: 1, primer cuarto, 2014.
- [KODR90] Kodratoff, Y. y Michalski, R. S. *Machine Learning and Artificial Intelligence Approach*. Vol. 3, Morgan Kaufmann, San Mateo, California, 1990.
- [KORT86] Korth, H. y Silberschatz, A. *Database System Concepts*. McGraw Hill, Nueva York, 1986.
- [KOSA00] Kosala, R. y Blockeel, H. *Web Mining Research: A Survey*. ACM SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, vol. 2, n.º 1, junio, 2000, pp. 1-15.
- [KOWA74] Kowalski, R. A. *Predicate Logic as a Programming Language*. Information Processing 74, North Holland Publications, Estocolmo (Suecia), 1974.
- [LANG96] Langley, P. *Elements of Machine Learning*. Morgan Kaufmann, San Francisco, 1996.
- [LIN97] Lin, T. Y. (editor). *Rough Sets and Data Mining*. Kluwer Publishers, Massachusetts, 1997.
- [LLOY87] Lloyd, J. *Foundations of Logic Programming*. Springer Verlag, Alemania, 1987.
- [LOOM95] Loomis, M. *Object Databases*. Addison Wesley, Massachusetts, 1995.
- [LOPE01] López Cilleros, A. *Modelización del comportamiento humano para un agente de la Robocup mediante aprendizaje automático*. Proyecto fin de carrera, Universidad Carlos III de Madrid, 2001.
- [MACQ67] MacQueen, J. *Some methods for classification and analysis of multivariate observations*. Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1:281-297, 1967.

- [MAIE83] Maier, D. *Theory of Relational Databases*. Computer Science Press, Maryland, 1983.
- [MAJO95] Major, J. y Mangano, J. *Selecting among rules induced from a hurricane database*. Journal of Intelligent Information Systems, 4:39-52, 1995.
- [MATTO98] Mattox, D. *et al. Software Agents for Data Management*. Handbook of Data Management. Auerbach Publications (ed.: B. Thuraisingham), Nueva York, 1998.
- [MBK98] Michalski, R. S., Brakto, I. y Kubat, M. *Machine Learning and Data Mining. Methods and Applications*. John Wiley & Sons, Nueva York, 1998.
- [MDDS94] *Proceedings of the Massive Digital Data Systems Workshop*. published by the Community Management Staff, Washington, 1994.
- [MERL97] Merlino, A. *et al. Broadcast News Navigation using Story Segments*. Proceedings of the 1997 ACM Multimedia Conference, Seattle, Washington, noviembre, 1998.
- [META96] Proceedings of the 1st IEEE Metadata Conference, Silver Spring, Maryland, abril, 1996 (publicado originariamente en Internet, editores: R. Musick, Lawrence Livermore y National Laboratory).
- [MICH83] Michalski, R. S. y Stepp, R. E. *Learning from observation: Conceptual clustering*. Editores: Michalski, R. S., Carbonell, J. G. y Mitchell, T. M. Machine Learning: An Artificial Intelligence Approach, vol. 1. Morgan Kaufmann, San Mateo, California, 1983.
- [MICH92] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolutions Programs*. Springer-Verlag, Nueva York, 1992.
- [MINK88] Minker, J. (editor). *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, California, 1988.
- [MIT] *Technical Reports on Data Quality*. Sloan School, Massachusetts Institute of Technology, Cambridge, Massachusetts.
- [MIT96] Mitchell, M. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, Massachusetts, 1996.
- [MITC97] Mitchell, T. *Machine Learning*. McGraw Hill, Nueva York, 1997.
- [MORE98] Morey, D. *Knowledge Management Architecture*. Handbook of Data Management, Auerbach Publications (ed.: B. Thuraisingham), Nueva York, 1998.
- [MORE98a] Morey, D. *Knowledge Management Architecture*. Handbook of Data Management, Auerbach Publications (ed.: B. Thuraisingham), Nueva York, 1998.

- [MORE98b] Morey, D. *Web Mining*. Private Communication, Bedford, Massachusetts, junio, 1998.
- [MORG88] Morgenstern, M. *Security and Inference in Multilevel Database and Knowledge Base Systems*. Proceedings of the 1987 ACM SIGMOD Conference, San Francisco, California, junio, 1987.
- [NG97] Ng, R. *Image Mining*. Private Communication, British Columbia, Vancouver, diciembre, 1997.
- [NISS96] *Panel on Data Warehousing, Data Mining, and Security*. Proceedings of the 1996 National Information Systems Security Conference, Baltimore, Maryland, octubre, 1996.
- [NISS97] *Papers on Internet Security*, Proceedings of the 1997 National Information Systems Conference, Baltimore, Maryland, octubre, 1997.
- [NIST15] NIST Big Data Public Working Group Definitions and Taxonomies Subgroup, *NIST Big Data Interoperability Framework: Volume 1, Definitions*, National Institute of Standards and Technology, NIST SP 1500-1, octubre, 2015.
- [NSF90] Proceedings of the Database Systems Workshop. Informe publicado por National Science Foundation, 1990 (también en ACM SIGMOD Record, diciembre 1990).
- [NSF95] Proceedings of the Database Systems Workshop. Informe publicado por National Science Foundation, 1995 (también en ACM SIGMOD Record, marzo, 1996).
- [NWOS96] Nwosu, K. et al. (editores). *Multimedia Database Systems, Design and Implementation Strategies*. Kluwer Publications, Maryland, 1996.
- [ODMG93] *Object Database Standard: ODMB 93*. Object Database Management Group, Morgan Kaufmann, California, 1993.
- [OMG95] *Common Object Request Broker Architecture and Specification*. OMG Publications, John Wiley, Nueva York, 1995.
- [ORFA94] Orfali, R. et al. *Essential, Client Server Survival Guide*. John Wiley, Nueva York, 1994.
- [ORFA96] Orfali, R. et al. *The Essential, Distributed Objects Survival Guide*. John Wiley, Nueva York, 1996.
- [PAKDD97] Proceedings of the Knowledge Discovery in Databases Conference, Singapur, febrero, 1997.
- [PAKDD98] Proceedings of the Second Knowledge Discovery in Databases Conference, Melbourne, Australia, abril, 1998.

- [PANT13] Pantazis, N. A., Nikolidakis, S. A. y Vergados, D. D. *Energy-Efficient Routing Protocols in Wireless Sensor Networks: A Survey*. IEEE Communications Surveys & Tutorials, vol. 15, issue: 2, segundo cuarto de 2013.
- [PATE12] Patel, S., Park, H., Bonato, P., Chan, L. y Rodgers, M. *A review of wearable sensors and systems with application in rehabilitation*. Journal of NeuroEngineering and Rehabilitation, 9:21, 2012.
- [PAW91] Pawlak, Z. *Rough Sets, Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Massachusetts, 1991.
- [PRAB97] Prabhakaran, B. *Multimedia Database Systems*, Kluwer Publications, Massachusetts, 1997.
- [PROJ 17] Using PROJ (PROJ.4 5.0.0 documentation). Disponible en: <http://proj4.org/usage/index.html>. [Acceso: 08-dic.-2017].
- [PSF91] Piatetsky-Shapiro, G. y Frawley, W. J. *Knowledge Discovery in Databases*. MA:AAA/MIT Press, Cambridge, 1991.
- [PTVF96] Press, W. H., Teukolosky, S. A., Vetterling, W. T. y Flannery B. P. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1996.
- [QUIN79] Quinlan, J. R. *Discovering Rules from Large Collections of Examples*. Expert Systems in the Microelectronic Age, Michie, D. (ed.), Edimburgo University Press, Edimburgo, 1979.
- [QUIN86] Quinlan, J. R. *Induction of Decision Trees (ID3 algorithm)*. Machine Learning J, vol. 1, n.º 1, 1986, pp. 81-106.
- [QUIN87] Quinlan, J. R. *Simplifying decision trees*, International Journal of Man-Machine Studies, n.º 27, 1987, pp. 221-234.
- [QUIN88] Quinlan, J. R. *Decision trees and multivalued attributes*, Machine Intelligence, n.º 11, 1988, pp. 305-318.
- [QUIN89] Quinlan, J. R. *Unknown attribute values in induction*. In Proc. 6th Int. Workshop on Machine Intelligence, Ithaca, Nueva York, junio, 1989, pp. 164-168.
- [QUIN90] Quinlan, J. R. *Learning logic definitions from relations*. Machine Learning, 5:139-166, 1990.
- [QUIN93] Quinlan, J. R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, California, 1993.
- [QUIN96] Quinlan, J. R. *Bagging boosting, and C4.5*. In Proc. 13th Natl. Conf Artificial Intelligence (AAAI'96), Portland, Oregón, agosto, 1996, pp. 725-730.

- [R 17] *RStudio – Open source and enterprise-ready professional software for R*. Disponible en: <https://www.rstudio.com/>. [Acceso: 08-dic.-2017].
- [RAMA94] Ramakrishnan, R. (editor). *Applications of Deductive Databases*, Kluwer Publications, Massachusetts, 1994.
- [RHW86] Rumelhart, D. E., Hinton, G. E. y Williams, R. J. *Learning internal representations by error propagation*, (eds. D. E. Rumelhart y J. L. McClelland). Parallel Distributed Processing. MIT Press, Massachusetts, 1986.
- [RIOS02] Ríos Montaña, P. M. *Sistemas clasificadores extendidos (XCS). Desarrollo de una librería y comparación CS versus XCS*. Proyecto fin de carrera, Universidad Carlos III de Madrid, 2002.
- [RMS98] Ramaswamy, S., Mahajan, S. y Silberschatz, A. *On the discovery of interesting patterns in association rules*. In Proc. 1998 Int. Conf Very Large Data Bases (VLDB'98), Nueva York, agosto, 1998, pp. 368-379.
- [ROSE98] Rosenthal, A. *Multi-Tier Architecture*, Private Communication, Bedford, Massachusetts, agosto, 1998.
- [RUME86] Rumelhart, D. E. y McClelland, J. L. *Parallel Distributed Processing*. Cambridge, MIT Press, Massachusetts, 1986.
- [RUME86] Rumelhart, D. E., Hinton, G. E. y Williams, R. J. *Learning internal representations by error propagation*, (eds. D. E. Rumelhart y J. L. MacClelland). Parallel Distributed Processing. Cambridge, MIT Press, Massachusetts, 1986.
- [SAIT88] Saito, K. y Nakano, R. *Medical diagnostic expert system based on PDP model*. Proceedings of the IEEE International Conference on Neural Networks, vol. 1, San Mateo, California, 1988, pp. 225-262.
- [SIGM96] Proceedings of the ACM SIGMOD Workshop on Data Mining, Montreal, Canadá, mayo, 1996.
- [SIGM98] Proceedings of the 1998 ACM SIGMOD Conference, Seattle, Washington, junio, 1998.
- [SILV86] Silverman, B. W. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, Londres y Nueva York, 1986.
- [SIMO95] Simoudis, E. *et al. Recon Data Mining System*. Technical Report, Lockheed Martin Corporation, 1995.
- [SLK96] Simoudis, E., Livezey, B. y Kerber, R. *Integrating Inductive and Deductive Reasoning for Data Mining*, (eds. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smiyth y R. Uthurusamy). Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, Massachusetts, 1996, pp. 353-373.

- [SPEN16], Spencer, G. A. *Big Data: More than Just Big and More than Just Data*, Frontiers of Health Services Management, 32, 4 (verano, 2016), pp. 27-33.
- [SQL3] SQL3. American National Standards Institute, Draft, 1992, (una versión también presentada por J. Melton del departamento de Navy's DISWG NGCR meeting, Salt Lake City, Utah, noviembre, 1994).
- [STAN98] Stanford Database Group Workshop, Jungalee Virtual Relational Database, septiembre, 1998 (también aparece en el IEEE Data Engineering Bulletin, junio, 1998).
- [SUDE11] Sudevalayam, S. y Kulkarni, P. *Energy harvesting sensor nodes: Survey and Implications*. IEEE Communications Surveys & Tutorials, vol. 13, n.º 3, 2011, pp. 443-461.
- [SURY12] Suryadevara, N. K. y Mukhopadhyay, S. C. *Wireless sensor network based home monitoring system for wellness determination of elderly*. IEEE Sensors Journal 12 (6), pp. 1965-1972.
- [THUR87] Thuraisingham, B. *Security Checking in Relational Database Systems Augmented by an Inference Engine*. Computers and Security, vol. 6, 1987.
- [THUR90a] Thuraisingham, B. *Nonmonotonic Typed Multilevel Logic for Multilevel Secure Database Systems*. Informe MITRE, junio, 1990 (también publicado en los Proceedings of the 1992 Computer Security Foundations Workshop, Franconia, Nuevo Hampshire, junio, 1991).
- [THUR90b] Thuraisingham, B. *Recursion Theoretic Properties of the Inference Problem*. Informe MITRE, junio, 1990 (también presentado en el 1990 Computer Security Foundations Workshop, Franconia, Nuevo Hampshire, junio, 1990).
- [THUR90c] Thuraisingham, B. *Novel Approaches to Handle the Inference Problem*. Proceedings of the 1990 RADC Workshop in Database Security, Castile, Nueva York, junio, 1990.
- [THUR91] Thuraisingham, B. *On the Use of Conceptual Structures to Handle the Inference Problem*. Proceedings of the 1991 IFIP Database Security Conference, Shepherdstown, Virginia Occidental, noviembre, 1991.
- [THUR93] Thuraisingham, B. *et al. Design and Implementation of a Database Inference Controller*. Data and Knowledge Engineering Journal, vol. 8, Holanda Septentrional, diciembre, 1993.
- [THUR95] Thuraisingham, B. y Ford, W. *Security Constraint Processing in a Multilevel Secure Distributed Database Management System*. IEEE Transactions on Knowledge and Data Engineering, vol. 7, 1995.

- [THUR96a] Thuraizingham, B. *Data Warehousing, Data Mining, and Security (version 1)*. Proceedings of the 10th IFIP Database Security Conference, Como, Italia, 1996.
- [THUR96b] Thuraizingham, B. *Internet Database Management*. Auerbach Publications, Nueva York, 1996.
- [THUR96c] Thuraizingham, B. *Interactive Data Mining and the World Wide Web*. Proceedings of Compugraphics Conference, París, Francia, diciembre, 1996.
- [THUR97] Thuraizingham, B. *Data Management Systems Evolution and Interoperation*. CRC Press, Florida, mayo, 1997.
- [THUR98] Thuraizingham, B. *Data Warehousing, Data Mining, and Security (version 2)*. Keynote Address at Second Pacific Asia Conference on Data Mining, Melbourne, Australia, abril, 1998.
- [THUR99] Thuraizingham, B. *Data Mining: Technologies, Techniques, Tools and Trends*. CRC Press, 1999.
- [TKDE93] Special Issue on Data Mining, IEEE Transactions on Knowledge and Data Engineering, diciembre, 1993.
- [TKDE96] Special Issue on Data Mining, IEEE Transactions on Knowledge and Data Engineering, diciembre, 1996.
- [TRUE89] Trueblood, R. y Potter, W. *Hyper-Semantic Data Modeling*. Data and Knowledge Engineering Journal, vol. 4, Holanda Septentrional, 1989.
- [TSIC82] Tsichritzis, D. y Lochovsky, F. *Data Models*. Prentice Hall, Nueva Jersey, 1982.
- [TSUR98] Tsur, D. et al. *Query Flocks: A Generalization of Association Rule Mining*. Proceedings of the 1998 ACM SIGMOD Conference, Seattle, Washington, junio, 1998.
- [ULLM88] Ullman, J. D. *Principles of Database and Knowledge Base Management Systems*. Vol. 1 y 11, Computer Science Press, Maryland, 1988.
- [UTG88] Utgoff, P. E. *An Incremental ID3*. In Proc. Fifth Int. Conf. Machine Learning, San Mateo, California, 1988, pp. 107-120.
- [VAPN74] Vapnik, V. N. y Chernovenkis, A. YA. *Theory of Pattern Recognition*. Nauka, Moscú, 1974.
- [VAPN79] Vapnik, V. N. *Estimation of Dependences Based on Empirical Data*. Nauka, Moscú, 1979.
- [VIS95] Proceedings of the 1995 Workshop on Visualization and Databases, (ed.: G. Grinstein), Atlanta, Georgia, octubre, 1995.

- [VIS97] Proceedings of the 1997 Workshop on Visualization and Data Mining, (ed.: G. Grinstein), Phoenix, Arizona, octubre, 1997.
- [VLDB98] Proceedings of the Very Large Database Conference, ciudad de Nueva York City, Nueva York, agosto, 1998.
- [WEIS91] Weiss, S. M. y Kulikowski, C. A. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, San Mateo, California, 1991.
- [WEIS98] Weiss, S. M. y Indurkha, N. *Predictive Data Mining*. Morgan Kaufmann, San Francisco, 1998.
- [WIED92] Wiederhold, G. *Mediators in the Architecture of Future Information Systems*. IEEE Computer, marzo, 1992.
- [WITT00] Witten, H. y Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, California, 2000.
- [WOEL86] Woelk, D. et al. *An Object-Oriented Approach to Multimedia Databases*. Proceedings of the ACM SIGMOD Conference, Washington D. C., junio, 1986.
- [WVD97] Proceedings of the 1995 Workshop on Visualization and Databases, (ed.: G. Grinstein), Atlanta, Georgia, octubre, 1997.
- [WVDM97] Proceedings of the 1997 Workshop on Visualization and Data Mining, (ed.: G. Grinstein), Phoenix, Arizona, octubre, 1997.
- [XML98] *Extended Markup Language*. Documento publicado por el World Wide Web Consortium, Cambridge, Massachusetts, febrero, 1998.