Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

Asignatura: APRENDIZAJE AUTOMÁTICO I
Grado en Ciencia e Ingeniería de Datos

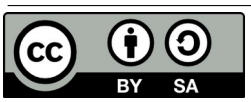
Apuntes de la asignatura

(Fecha del material: Diciembre 2023)

Curso académico 2023-2024

Material docente en abierto de la Universidad Rey Juan Carlos

Autores: Isaac Martín de Diego, Carmen Lancho



En el siguiente enlace se puede encontrar la versión html actualizada de los presentes apuntes:

https://urjcdslab.github.io/AprendizajeAutomaticoI/

Table of contents

Prefacio

Bienvenidos al emocionante mundo del Aprendizaje Automático en el ámbito de la Ciencia e Ingeniería de Datos. Este curso representa un apasionante viaje hacia el dominio de la Inteligencia Artificial y la toma de decisiones basada en datos, donde exploraremos las herramientas y técnicas que están revolucionando la forma en que comprendemos y utilizamos la información.

Como bien sabes, en la actualidad, los datos están en todas partes. Desde la información generada en redes sociales hasta la recopilación de datos en sensores y dispositivos, estamos rodeados de una abundancia de información que puede revelar conocimientos valiosos. Sin embargo, la capacidad para extraer ese conocimiento y tomar decisiones informadas a partir de datos complejos es lo que distingue a los profesionales en **Ciencia e Ingeniería de Datos**.

En este curso, profundizaremos en el Aprendizaje Automático, una rama de la Inteligencia Artificial que se centra en la creación de sistemas que pueden aprender y mejorar a partir de datos. Exploraremos cómo los algoritmos de Aprendizaje Automático pueden detectar patrones en datos, predecir resultados futuros, automatizar tareas y tomar decisiones inteligentes.

Nuestra misión es equiparos con las habilidades y el conocimiento necesarios para abordar problemas del mundo real y aprovechar el potencial de los datos en vuestro grado en Ciencia e Ingeniería de Datos. A lo largo del curso, trabajaremos en proyectos prácticos que os permitirán aplicar los conceptos y técnicas que vais a aprender.

Como profesores de esta asignatura, estamos entusiasmados de embarcarnos en este viaje de descubrimiento y aprendizaje con vosotros. La Inteligencia Artificial y el Aprendizaje Automático están transformando industrias y sociedades en todo el mundo, y os queremos ayudar para enfrentar los desafíos y aprovechar las oportunidades que estos campos ofrecen.

¡Aquí y ahora, comienza este apasionante viaje hacia el Aprendizaje Automático en Ciencia e Ingeniería de Datos!

Resultados de aprendizaje.

1. Conocer y aplicar adecuadamente la metodología para diseñar y evaluar modelos dirigidos por datos, sabiendo aplicar mecanismos para evitar el sobreajuste.
2. Conocer y saber aplicar técnicas de pre-procesamiento de datos, incluyendo las de reducción de dimensionalidad y tratamiento de valores ausentes.

3. Conocer y saber aplicar las técnicas de Aprendizaje Automático supervisado y no supervisado más convencionales.

4. Diseñar las etapas de un proceso completo de análisis de datos utilizando las técnicas de Aprendizaje Automático más adecuadas en función de la tarea a resolver.

! Grado en Ciencia e Ingeniería de Datos

Este libro presenta el material de la asignatura de Aprendizaje Automático I del grado en Ciencia e Ingeniería de Datos de la Universidad Rey Juan Carlos. Os recordamos que en próximos cursos os encontraréis con Aprendizaje Automático II y Aprendizaje Automático III.

! Conocimientos previos

Conveniente haber superado con éxito las asignaturas Fundamentos de Programación, Probabilidad y Simulación, así como haber cursado Inferencia Estadística y Optimización I.

i Sobre los autores

Carmen Lancho Martín es graduada en Matemáticas y Estadística por la Universidad Complutense de Madrid, doctora en Tecnologías de la Información y las Comunicaciones por la Universidad Rey Juan Carlos y profesora del departamento de Informática y Estadística de la Universidad Rey Juan Carlos. Miembro del grupo de investigación de alto rendimiento en Fundamentos y Aplicaciones de la Ciencia de Datos, DSLAB, de la URJC. Pertenece al grupo de innovación docente, DSLAB-TI.
Isaac Martín de Diego es diplomado en Estadística por la Universidad de Valladolid, licenciado en Ciencias y Técnicas Estadísticas por la Universidad Carlos III de Madrid (UC3M), doctor en Ingeniería Matemática por la UC3M, profesor titular del departamento de Informática y Estadística de la URJC. Es fundador y coordinador del DSLAB y del DSLAB-TI.

1 Introducción

En el enorme campo de la tecnología, una revolución ¡¡¡en forma de tsunami!!! está transformando la manera en que las máquinas interpretan y toman decisiones. A lo largo de este curso llevaremos a cabo un apasionante recorrido por el universo del Aprendizaje Automático (**ML**, de "Machine Learning" en inglés). Te invitamos a sumergirte en un campo que ha revolucionado la Inteligencia Artificial (**IA**, o AI, de "Artificial Intelligence" en inglés), dando forma al futuro de la tecnología y la ciencia de datos. Desde sus cimientos más elementales hasta algunas de las técnicas más avanzadas, exploraremos cómo las máquinas pueden aprender de los datos y mejorar su rendimiento en tareas que abarcan desde el reconocimiento de patrones hasta la toma de decisiones complejas. A lo largo de varios temas, desvelaremos los secretos que se esconden en los algoritmos que impulsan la IA moderna, permitiéndote aprender cómo aplicar estas técnicas para resolver desafíos del mundo real.

1.1 Revoluciones industriales

En la encrucijada de la historia y la tecnología, hemos sido testigos de cuatro revoluciones industriales que han moldeado radicalmente la forma en que vivimos, trabajamos y nos relacionamos con el mundo que nos rodea. Estas transformaciones, conocidas como las "Cuatro Revoluciones Industriales", han sido impulsadas por avances en la automatización, la maquinaria, la informática y la conectividad, y han desencadenado un cambio radical en la industria y la sociedad en su conjunto.

Despliega los siguientes paneles para averiguar más sobre estas cuatro revoluciones:

i Industria 1.0

La Primera Revolución Industrial marcó un punto de inflexión en la historia de la humanidad al introducir la **mecanización** en la producción y transformar la economía global. Esta revolución tuvo su inicio en el siglo XVIII, concretamente en 1760 en Gran Bretaña, y se caracterizó por la adopción masiva de la máquina de vapor como elemento central del desarrollo industrial. La invención de la máquina de vapor, principalmente por James Watt, desencadenó un cambio sin precedentes en la producción y la industria. La máquina de vapor permitía la generación de energía a partir del vapor de agua, lo que revolucionó los procesos de fabricación al reemplazar la dependencia de la fuerza humana y animal en la producción. Una de las consecuencias más notables fue la creación de

fábricas. Es importante resaltar que antes de este período, la producción era en su mayoría artesanal y descentralizada, pero con la mecanización impulsada por la máquina de vapor, se pudo concentrar la producción en instalaciones específicas, las **fábricas**. Esto dio lugar a una mayor eficiencia en la producción y la capacidad de fabricar productos a mayor escala. Esta revolución trajo consigo avances en la industria textil, la minería, el transporte y la agricultura. La producción de bienes se volvió más rápida y económica, lo que llevó a un aumento en la disponibilidad de productos manufacturados para la población en general.

i Industria 2.0

La Segunda Revolución Industrial se desarrolló a partir de la década de 1860 y marcó una evolución significativa en la forma en que se llevaba a cabo la producción industrial. Uno de los elementos más destacados de esta revolución fue la adopción generalizada de la **electricidad** como fuente de energía, lo que permitió la creación de nuevas industrias y la producción en masa de bienes en una escala antes nunca vista. La electricidad revolucionó la forma en que se generaba energía, reemplazando gradualmente la dependencia de la máquina de vapor. Esto permitió un mayor control y eficiencia en los procesos industriales, lo que a su vez impulsó el crecimiento de industrias clave como la siderurgia (producción de acero) y la industria petrolera. La disponibilidad de energía eléctrica también condujo al desarrollo de maquinaria eléctrica y electrónica, lo que mejoró aún más la producción y la automatización de las tareas. Uno de los conceptos más emblemáticos fue la introducción de líneas de montaje, un enfoque de producción en serie que revolucionó la fabricación. *Henry Ford* es conocido por popularizar este concepto en la industria del automovil, lo que permitió la producción eficiente y económica de automóviles en masa. Este enfoque también se aplicó a otras industrias, lo que hizo que la producción en serie fuera una práctica común.

i Industria 3.0

La década de 1960 marcó el comienzo de una de las revoluciones más impactantes en la historia de la tecnología y la informática: la Revolución **Digital**. Este período de transformación fue impulsado por avances significativos en la electrónica y la informática, y se caracterizó por la creación de la computadora personal, el desarrollo de semiconductores y el surgimiento de Internet.

-La Computadora Personal: A fines de la década de 1960, surgieron las primeras computadoras personales. Aunque eran dispositivos rudimentarios en comparación con las computadoras modernas, estas máquinas permitieron que las personas tuvieran acceso a la informática en sus hogares y oficinas, lo que allanó el camino para la revolución digital al **democratizar el acceso a la tecnología**.

-Semiconductores: La Revolución Digital fue impulsada en gran medida por avances en la

tecnología de semiconductores. La miniaturización de componentes electrónicos permitió la creación de dispositivos más pequeños, eficientes y asequibles. La invención del circuito integrado (o microchip) en 1958 fue un hito clave que facilitó el desarrollo de dispositivos electrónicos más poderosos y compactos.
-Internet: Aunque la gestación de Internet comenzó en la década de 1960 con la creación de ARPANET (una red de comunicación militar), fue en esta década cuando Internet comenzó a tomar forma como una red global. La creación del Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP) en la década de 1970 allanó el camino para la expansión de la red. A lo largo de la década de 1960, los investigadores avanzaron en la comunicación por computadora y desarrollaron los primeros esbozos de lo que se convertiría en la World Wide Web.
-Revolución Electrónica: La Revolución Digital también vio avances significativos en la electrónica, como la creación de circuitos integrados, transistores y microprocesadores. Estos componentes permitieron la creación de dispositivos electrónicos más poderosos y eficientes, lo que impulsó la innovación en áreas como la informática, las comunicaciones y la electrónica de consumo.

i Industria 4.0

La Cuarta Revolución Industrial, representa un capítulo fundamental en la historia de la tecnología y la industria. Emergió en torno a 2016 y ha sido impulsada por una serie de tecnologías avanzadas que han transformado radicalmente la forma en que operan las empresas y la sociedad en general.
Aquí se detallan algunos de los aspectos clave de la Industria 4.0:
-**Digitalización**: La digitalización es la piedra angular de la Industria 4.0. Se refiere a la conversión de procesos, datos y objetos en formatos digitales. Esto ha permitido una mayor eficiencia, precisión y velocidad en la toma de decisiones y en la ejecución de tareas.
-**Inteligencia Artificial**: En la Industria 4.0, la IA se utiliza para automatizar procesos, optimizar la producción y prever problemas antes de que ocurran.
-**Robótica**: Los robots desempeñan un papel fundamental en la fabricación y la logística en la Industria 4.0. Estos robots son cada vez más autónomos y colaborativos, trabajando junto a los humanos en tareas de producción y manipulación.
-**Internet de las cosas (IoT)**: El IoT implica la conexión de objetos y dispositivos a través de Internet. En la Industria 4.0, los sensores y dispositivos IoT recopilan datos en tiempo real, permitiendo un control y una toma de decisiones más precisos.
-**Aprendizaje Automático**: En la Industria 4.0, el **ML** se aplica para mejorar la calidad y la eficiencia de la producción, así como para la detección temprana de problemas.
-**Big Data**: La recopilación y el análisis de grandes volúmenes de datos son esenciales en la Industria 4.0. El Big Data permite identificar patrones, tendencias y oportunidades que antes eran difíciles de detectar.

-**Computación en la Nube (Cloud Computing)**: La infraestructura en la nube proporciona almacenamiento y procesamiento escalables, lo que facilita el acceso y el uso de datos y aplicaciones desde cualquier lugar.
-**Hiperconectividad**: La hiperconectividad es la interconexión de dispositivos, sistemas y personas a través de Internet. Esto crea una red global que permite la comunicación y la colaboración en tiempo real.
La Industria 4.0 ha revolucionado la forma en que las empresas operan y compiten. Ha impulsado la eficiencia, la productividad y la personalización en la producción y los servicios. Al mismo tiempo, presenta desafíos en términos de seguridad cibernética, privacidad de datos y adaptación de la fuerza laboral. A medida que la Industria 4.0 continúa evolucionando, se espera que tenga un impacto aún mayor en la economía global y la sociedad en general.

1.2 La Inteligencia Artificial

Existen varias definiciones de IA, todas ellas relevantes:

- La automatización de actividades que asociamos con la forma de pensar del ser humano (R. E. Bellman 1978)
- El estudio de los mecanismos que hacen posible percivir, razonar, y actuar (Winston 1992)
- El arte de crear máquinas que realicen tareas que requieren de inteligencia cuando son realizadas por seres humanos (Kurzweil et al. 1990)
- Estudio del diseño de agentes inteligentes (Poole, Goebel, and Mackworth 1998)

Pueses buscar otras posibles definiciones en este libro (Russell 2010). Nosotros proponemos la siguiente:

Inteligencia Artificial

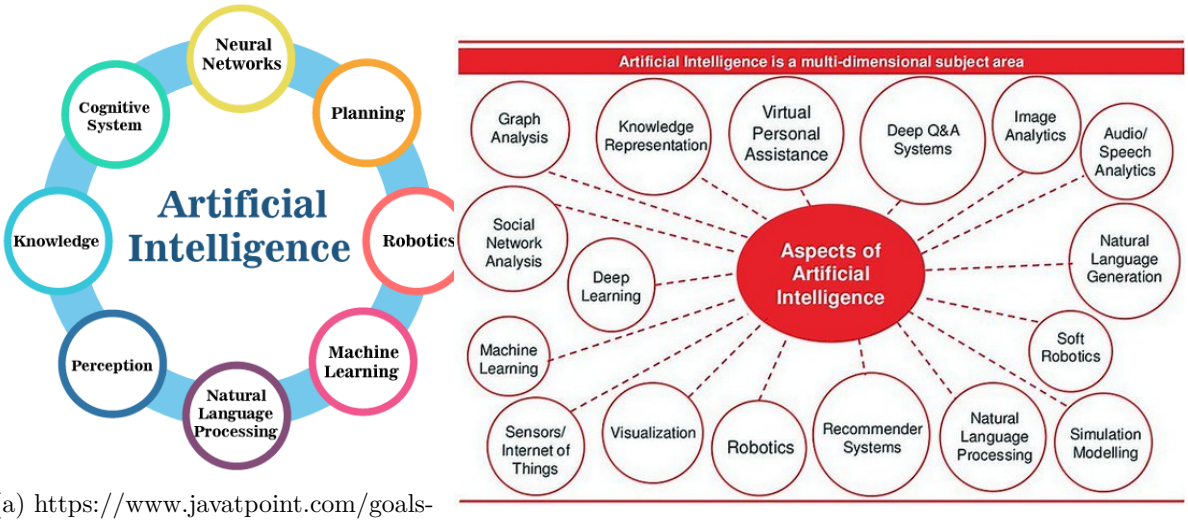
La Inteligencia Artificial es un campo de la informática que se enfoca en el desarrollo de sistemas y programas informáticos capaces de realizar tareas que, cuando se ejecutan por parte de seres humanos, requieren inteligencia y aprendizaje. Estos sistemas de IA pueden aprender de datos, adaptarse a nuevas situaciones, tomar decisiones, resolver problemas y realizar tareas específicas sin intervención humana directa. La IA busca imitar y replicar procesos cognitivos y de toma de decisiones humanas, permitiendo a las máquinas realizar actividades que normalmente requerirían la inteligencia humana.

Para saber más sobre IA

Human Cognition and Artificial Intelligence — A Plea for Science
How AI could become an extension of your mind

1.2.1 Organización de la IA

Exiten varios modos de abordar una organización de la IA. Podemos hablar de diferentes áreas en las que organizar la IA:



(a) https://www.javatpoint.com/goals-of-artificial-intelligence

(b) https://twitter.com/essec/status/989442209484328960

Figure 1.1: Organización de la IA

Atención

A día de hoy, no hay una organización exclusiva, aceptada unánimemente para la IA. Os animamos a que busques otras, alternativas a las anteriores, en Internet.

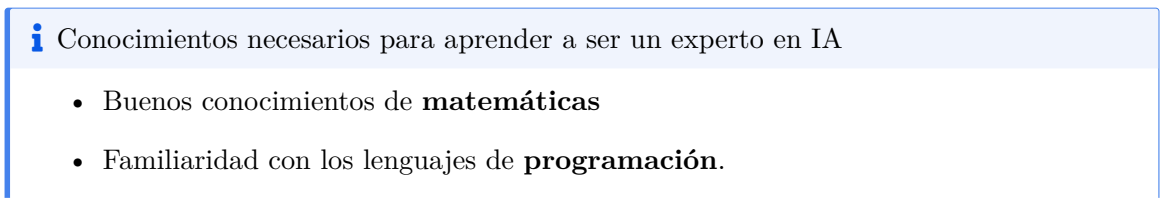
Como puedes ver, la IA es un campo diverso que se beneficia de una variedad de disciplinas interconectadas. Estas áreas complementarias se combinan para dar forma al espectro completo de la IA, cada una con su enfoque y aplicaciones únicas:

1. **Robótica**: La robótica se centra en la creación y programación de máquinas capaces de interactuar con su entorno de manera autónoma. La IA desempeña un papel crucial en la toma de decisiones de los robots y su capacidad para aprender y adaptarse.

2. **Aprendizaje Automático**: El Aprendizaje Automático (objeto fundamental de estos apuntes) es una subdisciplina de la IA que se centra en el desarrollo de algoritmos y modelos que pueden aprender de datos y realizar tareas sin estar explícitamente programados. Esto es fundamental en multitud de aplicaciones, muchas de las cuales trataremos a lo largo de este curso.
3. **Procesamiento de Lenguaje Natural (NLP)**: El NLP se ocupa de la interacción entre las computadoras y el lenguaje humano. Se utiliza en la traducción automática, chatbots, análisis de sentimientos y otras aplicaciones relacionadas con el lenguaje.
4. **Visión por Computador**: La visión por computador se centra en la capacidad de las máquinas para comprender y procesar imágenes y videos. Esto se aplica en reconocimiento facial, conducción autónoma y diagnóstico médico, entre otros.
5. **Lógica Difusa**: La lógica difusa permite manejar la incertidumbre y la imprecisión en la toma de decisiones. Se utiliza en sistemas de control automático y en aplicaciones donde la información es vaga o ambigua.
6. **Redes Neuronales Artificiales**: Inspiradas en el funcionamiento del cerebro humano, las redes neuronales artificiales son modelos de aprendizaje profundo que se utilizan en una variedad de aplicaciones, desde el reconocimiento de voz hasta el procesamiento de imágenes. De ellas surge el llamado Deep Learning.
7. **Reconocimiento de Patrones**: El reconocimiento de patrones implica la identificación de estructuras y tendencias en datos. Esto es fundamental en aplicaciones como la bioinformática, la seguridad y la análisis financiero.
8. **Computación Evolutiva**: La computación evolutiva se basa en principios biológicos como la selección natural y la evolución para optimizar algoritmos y resolver problemas complejos.

Estas áreas interconectadas colaboran para impulsar el avance de la IA en diversas aplicaciones, desde la automatización industrial y la atención médica hasta la conducción autónoma y la atención al cliente. A medida que la IA continúa evolucionando, la integración de estas disciplinas se vuelve cada vez más importante para desarrollar sistemas inteligentes y capaces de adaptarse al mundo real. En este curso iniciamos un camino en una de estas áreas, el Aprendizaje Automático.

Despliega el siguiente panel para averiguar si tienes los cimientos necesarios (o aún no) para convertirte en un experto en IA.



- Capacidad para escribir algoritmos de búsqueda de **patrones** y aprendizaje.
- Buena capacidad de **análisis**.
- Buenos conocimientos de **Estadística** y modelización.
- Capacidad para aprender nuevos algoritmos de **Machine Learning** y Deep Learning.
- Competencias **no tecnológicas**.

1.3 Aprendizaje Automático

Es el momento de definir, de manera formal, esa parte de la IA que ha aparecido en varias ocasiones. El Aprendizaje Automático o Machine Learning. Es el eje central de todo el libro que vamos a construir.

> Aprendizaje Automático o Aprendizaje Máquina
>
> Podemos definir el Aprendizaje Automático, Aprendizaje Máquina o Machine Learning (ML) como una rama de la IA que se centra en el uso de datos y algoritmos para replicar la forma en la que aprenden los seres humanos, con una mejora gradual de su rendimiento.

El ML, con su capacidad para extraer conocimiento y tomar decisiones a partir de datos, se ha convertido en la herramienta esencial que impulsa la innovación y la eficiencia en la industria y la economía. A medida que avanzamos dentro de la cuarta revolución industrial, en la que la convergencia de lo físico, lo digital y lo biológico es inevitable, el ML desempeña, cada vez más, un papel central en la creación de sistemas más inteligentes, la automatización de tareas complejas y la toma de decisiones informadas. A continuación vamos a averiguar el papel fundamental que juega el ML dentro de la ciencia de datos.

1.3.1 Ciencia de datos

La última revolución asociada a la IA ha estado enmarcada por el crecimiento en el uso del ML dentro del contexto de la ciencia de datos (Kelleher, Mac Namee, and D'arcy 2020).

> Ciencia de datos
>
> La ciencia de datos es un área interdisciplinar que abarca un conjunto de principios, problemas, definiciones, algoritmos y procesos cuyo objetivo es extraer conocimiento no

obvio y útil a partir de un conjunto de datos.

Pero, ¿qué áreas, métodos y técnicas están implicados en la ciencia de datos?. En primer lugar, presentemos los aspectos teórico y prácticos que sustentan un proyecto real de ciencia de datos. Para ellos recurrimos a la Figura Figure 1.2a que representa el clásico diagrama de la ciencia de datos, como una disciplina en la intersección de tres aspectos fundamentales. Para saber más sobre estos aspectos, desplegad los paneles siguientes:

Matemáticas y Estadística

Conocimientos de Matemáticas, y más concretamente de Estadística, son necesarios para analizar correctamente los datos disponibles. Conceptos como intervalo de confianza, histograma de frecuencias, contraste de hipótesis, espacio de características, métrica, hiperplano separador, error de clasificación, p_valor, etc. han de formar parte del conocimiento de todo científico de datos. Un equipo de ciencia de datos ha de contar con uno o varios expertos en Matemáticas y Estadística. Un buen libro de referencia para dominar los conceptos fundamentales en el ámbito matemático y estadístico que son necesarios en ciencia de datos es (Hastie et al. 2009). Además disponéis de esta versión online (James et al. 2013), similar pero más enfocada al análisis de datos. ¿Lo conocías ya?

Ciencias de la Computación

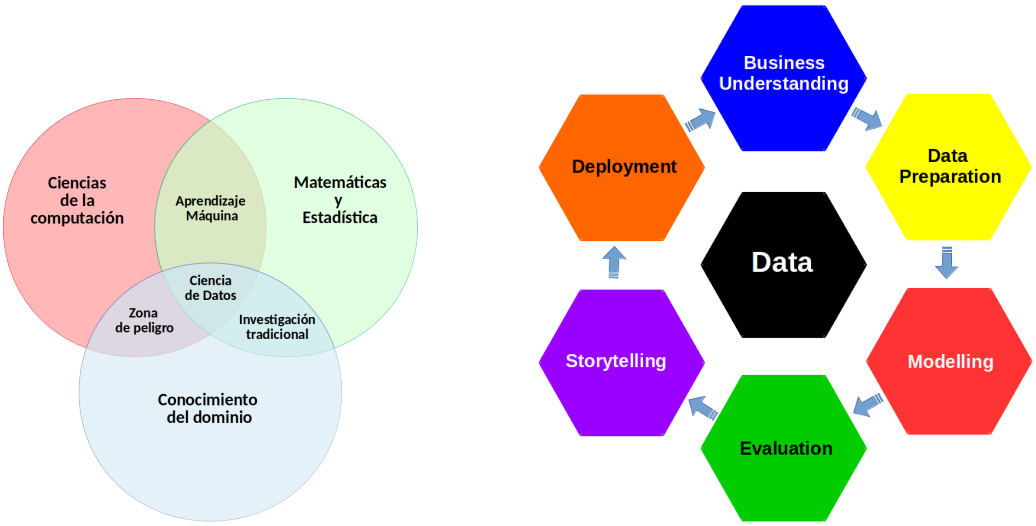
Estudio del diseño y la arquitectura de los ordenadores y su aplicación en el campo de la ciencia y la tecnología, incluyendo el hardware, el software y las redes de comunicación. Un experto en ciencias de la computación ha de dominar lenguajes de programación como Python, JavaScript, C++, así como los elementos fundamentales que hacen que estos lenguajes funcionen. Algunas referencias útiles para estudiar estos lenguajes son (Hao and Ho 2019), (Osmani 2012) y (Oualline 2003). De igual modo, el científico de datos, ha de conocer ámbitos como los diferentes sistemas operativos, redes, seguridad, algoritmos y arquitectura de ordenadores. Un equipo de ciencia de datos ha de contar con uno o varios expertos en Ciencias de la Computación.

Conocimiento del Dominio

Representa el problema que deseamos estudiar, la organización que lo proporciona y su dominio de aplicación. Existen casos de éxito de la ciencia de datos en prácticamente todos los dominios de interés que podamos mencionar: medicina, ciudades inteligentes, energía, telecomunicaciones, finanzas, seguros, ganadería, agricultura, ciencias sociales, ciberseguridad, etc. Un equipo de ciencia de datos ha de contar con uno o varios expertos en el dominio de aplicación. Estos expertos han de implicarse, fuertemente, en el problema que se quiere resolver. En (Kelleher, Mac Namee, and D'arcy 2020) podéis

encontrar ejemplos muy interesantes de cómo la ciencia de datos es aplicada en diferentes dominios.

En muchas ocasiones se asocia el término ML con el término ciencia de datos. Sin embargo, hay una gran diferencia marcada por el conocimiento del dominio. Digamos que un experto en ML conocerá una gran variedad de algoritmos de aprendizaje, será experto en limpieza y depuración de datos, tendrá amplios conocimientos de programación, etc. Sin embargo, para conseguir modelos útiles dentro de un dominio de aplicación ha de añadir a sus habilidades un conocimiento del dominio. Y esto no es inmediato. El ML está relacionado con **usar las características adecuadas para construir los modelos adecuados que realicen las tareas adecuadas dentro del dominio adecuado** (Flach 2012).



(a) Fundamentos (b) Aplicaciones

Figure 1.2: Ciencia de datos

Por su parte, la Figura Figure 1.2b basada en CRISP-DM: "*Cross Industry Standard Process for Data Mining*" (Wirth and Hipp 2000) presenta el ciclo de vida que todo proyecto de ciencia de datos debería seguir. El inicio del proyecto viene dado por la definición de los objetivos de la organización. A continuación, se recogen y gestionan los datos. Como siguiente paso, se desarrollan y evalúan algoritmos matemáticos sobre los datos. Los resultados de estos modelos se presentan a los expertos en el dominio de aplicación para su posterior integración dentro de la organización. Nótese que el proyecto puede tener varias iteraciones, volviendo a alguna de las etapas anteriores siempre que una etapa posterior así lo requiera. Para saber más detalles sobre estas etapas investigad los paneles siguientes:

Definir objetivos

Entender el negocio es el primer paso en el proceso. Con ayuda de expertos en el dominio, se definen las preguntas a responder con el proyecto (definición de objetivos de la entidad responsable del proyecto). Una vez comprendido el negocio se designa una solución analítica para abordar el problema. En esta etapa las reuniones entre los matemáticos, informáticos y los expertos del dominio (habitualmente trabajadores con grandes conocimiento del problema que se trata de abordar) son frecuentes, necesarias y (casi) nunca, suficientes.

Obtener, preparar y gestionar los datos

Mediante técnicas informáticas, se recopilan y se preparan los datos para su posterior análisis. Podremos hablar de Big Data si los datos se caracterizan por su **volumen, variedad o velocidad** de procesamiento. Todo proceso de Ciencia de Datos es un proceso de aprendizaje en torno al dato. Las preguntas no surgen de los datos, pero se necesitan datos para responderlas. Es esta la etapa en la que trataremos una parte fundamental de todo el proceso: el análisis exploratorio de datos. Podrás estudiar más sobre cómo preparar los datos para etapas posteriores en el tema 3 de este libro.

Construir un modelo

A través de métodos matemáticos y estadísticos se estudian y analizan los datos, se construyen algoritmos y se aplican modelos. Es la etapa asociada a los modelos de ML que trataremos en los temas 5,7 y 8.

Evaluar y criticar el modelo

Se definen medidas de rendimiento de los modelos que permitan su evaluación tanto por parte del desarrollador como por parte del cliente. Trataremos estas medidas en el tema 6 de nuestro curso.

Visualización y presentación

Se presentan los resultados buscando la comprensión por parte del cliente. No se trata únicamente de aplicar modelos complejos que nadie, más allá del desarrollador o experto matemático, pueda comprender. Muy al contrario, existe en la actualidad una corriente de investigación orientada a construir métodos capaces de ser explicables, junto con otras técnicas para convertir en entendibles los resultados obtenidos por los más complejos algoritmos matemáticos. Hablaremos de explicabilidad de modelos en el último tema del curso.

Despliegue en producción

La solución final se convierte en un producto que podrá ser comercializado. Los modelos de ML se construyen para un propósito dentro de una organización. La integración de este modelo dentro del proceso de la organización debería de ser el último paso del proyecto de ciencia de datos.

1.4 Técnicas y métodos de ML

Tal y como hemos indicado, la etapa de aprendizaje mediante modelos de ML cobra todo su interés una vez los datos han sido adquiridos, almacenados y preparados de modo correcto. Es el momento de aplicar técnicas y algoritmos de ML, en búsqueda de los objetivos fijados en etapas anteriores del proyecto. Como ya indicamos, el ML se refiere al uso de las características correctas para construir los modelos correctos que desarrollen las tareas correctas.

Para recordar

El ML es un subcampo interdisciplinar de la IA que desarrolla tanto el fundamento matemático como las aplicaciones prácticas de los sistemas que aprenden de los datos con alguna clase de tarea y alguna medida de rendimiento.

En general, dispondremos de un conjunto de observaciones con un número de variables o características, medidas sobre dichas observaciones. Estos atributos, que caracterizan a las observaciones, serán usados por los modelos de ML para aprender a realizar las tareas que les sean encomendadas. Podemos dividir los algoritmos de ML en dos grandes grupos según la existencia o no de una variable que corresponda a una clase etiquetada:

- **Datos etiquetados**: una de las variables recoge la etiqueta de la observación. Esta etiqueta refleja el valor de la variable objetivo y es el atributo que normalmente deseamos ser capaces de predecir mediante el modelo. Dicha variable recibe el nombre de variable objetivo, o variable respuesta. El ML con datos etiquetados se denomina **Aprendizaje Supervisado (Supervised Learning)** y lo trataremos en el tema 7 de este curso. Si la etiqueta es cualitativa la tarea se denomina clasificación. Si la etiqueta es cuantitativa la tarea se llama regresión.
- **Datos no etiquetados**: no se recoge ninguna etiqueta. El ML utilizando datos no etiquetados se llama **Aprendizaje No Supervisado** y será tratado en el tema 5 de este curso. Los algoritmos de agrupamiento o Clustering examinan los datos para encontrar grupos de observaciones que son más similares entre sí respecto a las observaciones de otros grupos.

Para concluir esta introducción, párate un momento en la siguiente tabla que presenta algunos ejemplos de aplicación de ML junto con los algoritmos empleados.

Ejemplo	**Aprendizaje Máquina**	**Algoritmos**
Identificación de spam	Clasificación: decidir si un correo es spam o no	Árboles de decisión, Naïve Bayes, Regresión Logística, Máquinas de Vectores Soporte
Predecir cuánto incrementará las ventas de una compañía una campaña de marketing	Regresión: predecir valores futuros de una variable cualitativa	Regresión Lineal
Identificar páginas web que son visitadas a menudo durante la misma sesión	Reglas de asociación: encontrar correlaciones o causas potenciales de los efectos vistos en los datos, encontrar objetos que tienden a aparecer en el conjunto de los datos	Reglas de asociación
Identificar usuarios con el mismo comportamiento	Clustering: encontrar objetos semejantes entre sí y diferentes de los objetos de otros grupos	K-medias
Hacer recomendaciones de productos para un cliente basadas en las compras de otros clientes similares.	Sistemas de Recomendación: predicción de la propiedad de un dato basado en el dato más similar	Vecinos más cercanos, Filtrado colaborativo

2 Datos

Este tema se centra en la importancia de los datos en el contexto del ML, explorando cómo su almacenamiento y procesamiento permiten a los sistemas informáticos y a los usuarios extraer información valiosa. Si reflexionamos sobre el ciclo de vida de un proyecto de ciencia de datos (recuerda la Figura Figure 1.2b), podemos observar cómo los datos ocupan el centro de todo el proceso, mientras que las demás tareas y etapas se desarrollan en torno a ellos.

En el contexto del ML, los datos representan el recurso fundamental. Desde el inicio de un proyecto en ciencia de datos, se habrá dedicado tiempo y esfuerzo a comprender el problema y definir los objetivos a alcanzar. Estas etapas se han planificado meticulosamente con un enfoque centrado en los datos, contemplando aspectos como la naturaleza de los datos a recopilar, las estrategias para su adquisición, así como las mejores prácticas para su almacenamiento y gestión, entre otros factores cruciales. Piensa que sin una adecuada recolección y mantenimiento de los datos, ningún proyecto de datos puede prosperar.

En el transcurso de este tema, exploraremos brevemente temas de importancia crítica, como la salvaguardia de los datos, cuestiones éticas y la preservación de la privacidad. Estos aspectos son fundamentales en cualquier proyecto vinculado a este campo, ya que garantizar la integridad y la seguridad de los datos, así como respetar los principios éticos y legales, son pilares esenciales en la construcción de soluciones de aprendizaje automático que sean responsables y confiables.

2.1 Tipos de datos

En primer lugar, tratamos los tipos de datos más comunes organizándose mediante dos clasificaciones distintas.

> **i** Atención
>
> Según su **estructura**, los datos pueden dividirse en datos estructurados y datos no estructurados.
> Según su comportamiento a lo largo del **tiempo**, los datos pueden calificarse como datos estáticos o datos dinámicos.

2.1.1 Datos Estructurados frente a Datos No Estructurados

Esta organización se enfoca en determinar si los datos exhiben una estructura definida en el momento de su obtención o si, por el contrario, carecen de un esquema determinista. A continuación, describiremos las distintas categorías de datos de acuerdo con esta clasificación:

> **Datos estructurados**
>
> Los datos estructurados son aquellos que poseen una longitud, un tipo, un formato y un tamaño definidos. Por lo general, estos datos se organizan en forma de tabla o se almacenan en una base de datos relacional (tablas de columnas y filas), que establece relaciones entre las diversas propiedades de las observaciones.

> **Datos no estructurados**
>
> Los datos no estructurados, en cambio, carecen de un formato específico. Suelen extraerse de documentos de texto, videos u otras fuentes similares. Como ejemplo, la oración que estás leyendo en este momento contiene datos no estructurados, ya que no sigue un formato particular, y los usuarios no pueden anticipar si se organiza de acuerdo con algún esquema predefinido.

2.1.2 Datos estáticos frente a datos dinámicos

Esta organización atiende al comportamiento de los datos a lo largo del tiempo. Los datos estáticos son los datos que no cambian a lo largo del tiempo. Por el contrario, los datos pueden evolucionar en el tiempo, llegando nuevas observaciones, o nuevos valores de las variables de interés, que obliguen a los modelos construidos sobre ellos a realizar ajustes de manera constante. En estas situaciones hablaremos de datos dinámicos.

Los datos dinámicos son los que presentan un nivel de complejidad más alto en términos de su tratamiento. Es habitual crear esquemas con el fin de simplificar las posibles operaciones que se llevarán a cabo con los datos dinámicos. Estos esquemas son aplicables principalmente a datos estructurados. Sin embargo, en el caso de los datos dinámicos no estructurados, la tarea radica en procesar estos datos de manera dinámica a medida que evolucionan, sin la capacidad de anticipar de antemano la estructura o dimensión que puedan adquirir.

Para más información sobre este tema, consultar (Weiss 2012).

2.2 Obtención de datos

El proceso de obtención de los datos consiste en la recopilación de información en un dominio específico. Una vez obtenidos los datos, estos deben ser procesados con el objetivo de generar conocimiento a partir de ellos. Despliega los siguientes paneles para conocer algunas de las técnicas más comunes en el proceso de obtención de datos.

Encuestas y entrevistas

Estas técnicas involucran la formulación de preguntas a expertos dentro de un área de interés. En ocasiones, se utilizan para etiquetar los datos, es decir, para asignar categorías o etiquetas a los datos en función de las respuestas proporcionadas. Las preguntas en una encuesta pueden ser abiertas, lo que significa que no tienen restricciones en la respuesta, o cerradas, donde se ofrecen opciones predefinidas para elegir. Por ejemplo, en el campo de la ciberseguridad, los sistemas de detección de virus a menudo utilizan etiquetas para identificar y clasificar virus. Estos sistemas aprovechan el conocimiento de los expertos para tomar decisiones en ese dominio.

Observación

Esta técnica implica observar y verificar lo que está sucediendo con el conjunto de datos de interés desde una distancia. La observación puede ser directa o indirecta. En el primer caso, un observador se limita a verificar los datos directamente. En el segundo caso, el observador se enfoca en los resultados de los análisis realizados en los datos. Un ejemplo de observación directa son los filtros antispam en correos electrónicos, que observan y toman medidas en función de las observaciones realizadas en los mensajes. Los usuarios que revisan la carpeta de *spam* y ven el número de mensajes rechazados son observadores indirectos en este contexto.

Toma de muestras

Esta técnica consiste en obtener subconjuntos representativos de los datos en lugar de analizar todos los datos disponibles. A menudo, no es necesario analizar todos los datos, especialmente cuando se trata de conjuntos de datos extensos. En su lugar, se toma una muestra lo suficientemente grande y diversa que sea representativa. Por ejemplo, para determinar si ha ocurrido un ciberataque y se ha producido un robo de información, puede ser suficiente analizar un porcentaje de los datos de red de los últimos días u horas en lugar de examinar todos los datos durante períodos más largos, como meses o años.

Web Scraping

Esta técnica implica la extracción de datos directamente desde sitios web. Los *web scrapers* automatizan el proceso de navegación web y recuperación de información de páginas web, lo que permite obtener datos de fuentes públicas en línea.

Sensores y Dispositivos IoT

Los sensores y dispositivos de Internet de las cosas (IoT) generan datos en tiempo real, que pueden incluir información sobre temperatura, humedad, ubicación geográfica, entre otros. Estos datos pueden ser recopilados y utilizados para análisis.

Registros y Bitácoras (Logs)

Muchos sistemas y aplicaciones generan registros (logs) que registran eventos y actividades. Estos registros pueden ser valiosos para el análisis de problemas, seguimiento de comportamientos y seguridad.

Extracción de Texto de Documentos

Para datos no estructurados, como texto en documentos, se pueden utilizar técnicas de extracción de texto para convertir la información en un formato procesable. Esto es útil en áreas como el procesamiento de documentos legales o médicos.

Captura de Imágenes y Video

Las cámaras y sistemas de captura de imágenes y video generan grandes cantidades de datos visuales que pueden ser analizados para reconocimiento de patrones, detección de objetos, análisis de imágenes médicas y más.

APIs (Interfaces de Programación de Aplicaciones)

Muchas aplicaciones y servicios ofrecen APIs que permiten acceder a sus datos de manera estructurada y programática. Esto es común en aplicaciones de redes sociales, servicios de mapas, pronósticos del tiempo y más.

Datos de Redes Sociales

Las plataformas de redes sociales generan una gran cantidad de datos en forma de publicaciones, comentarios y actividad de usuarios. Estos datos pueden ser recopilados y analizados para comprender tendencias, opiniones del público y más.

Datos de Dispositivos Móviles

Las aplicaciones móviles pueden recopilar datos del dispositivo, como la ubicación GPS, el uso de la aplicación y la interacción del usuario. Estos datos pueden utilizarse para análisis de comportamiento del usuario y personalización.

Datos de Sensores Biométricos

En aplicaciones de salud y bienestar, se pueden utilizar sensores biométricos, como los dispositivos de seguimiento de actividad física y las pulseras de ritmo cardíaco, para recopilar datos sobre la salud y el estado físico de los individuos.

2.2.1 Fuentes de datos para las prácticas

Para las prácticas que se van a llevar a cabo a lo largo de este curso, vamos a necesitar datos. En un entorno educativo como el nuestro, es habitual recurrir a los llamados "*benchmarks*". Un *benchmark* se refiere a un conjunto de datos de referencia o estándar que se utiliza como punto de comparación para evaluar y medir el rendimiento de algoritmos, modelos y técnicas de análisis de datos. Los *benchmarks* son esenciales en la ciencia de datos porque proporcionan un punto de referencia común y objetivo para evaluar la calidad y eficacia de diferentes enfoques. Puedes encontrarlos en diferentes portales y su uso es habitual en artículos de investigación.

Despliega el siguiente panel para averiguar dónde localizar numerosos conjuntos de datos de referencia.

Fuentes de datos

Existen varias fuentes de conjuntos de datos de *benchmark* que puedes utilizar en proyectos de ciencia de datos. Algunas de las fuentes más populares y confiables incluyen:

1. **Kaggle:** Kaggle es una plataforma en línea que alberga competiciones de ciencia de datos y proporciona una amplia variedad de conjuntos de datos para prácticas y competiciones. Puedes explorar conjuntos de datos en su sección de datasets. Cuidado con los análisis de estos datos en la plataforma Kaggle, pues no siempre son correctos o precisos. ¡Ojo!

2. **UCI Machine Learning Repository:** La Universidad de California, Irvine (UCI), ofrece un amplio repositorio de conjuntos de datos de *benchmark* para ML. Puedes encontrarlos en su página web.

3. **OpenML:** OpenML es una plataforma en línea que recopila y comparte conjuntos de datos, así como flujos de trabajo y resultados de la comunidad de ciencia de datos. Puedes explorar conjuntos de datos en su sitio web.

4. **GitHub:** GitHub es una plataforma de desarrollo de software que también alberga una gran cantidad de repositorios públicos que contienen conjuntos de datos. Puedes utilizar la función de búsqueda de GitHub para encontrar conjuntos de datos relevantes.

5. **Data.gov:** Data.gov es el portal de datos abiertos del gobierno de los Estados Unidos. Ofrece una amplia variedad de conjuntos de datos gubernamentales en áreas como salud, medio ambiente, educación y más.

6. **Google Dataset Search:** Google Dataset Search es una herramienta de búsqueda específica para conjuntos de datos. Te permite buscar conjuntos de datos en línea utilizando palabras clave relevantes.

7. **Awesome Public Datasets:** Este repositorio de GitHub contiene una lista de conjuntos de datos públicos en diversas categorías. Puedes encontrarlo aquí.

8. **Datasets Subreddit:** La comunidad de Reddit tiene un subreddit llamado "*datasets*" donde los usuarios comparten y discuten conjuntos de datos interesantes. Puedes explorarlo en r/datasets.

9. **World Bank Data:** El Banco Mundial ofrece una gran cantidad de datos relacionados con el desarrollo económico y social de países de todo el mundo. Puedes acceder a estos datos en su sitio web.

10. **Eurostat:** Eurostat es la oficina de estadísticas de la Unión Europea y proporciona una amplia gama de datos relacionados con la UE y sus países miembros. Puedes encontrar datos en su sitio web.

Recuerda que, antes de utilizar cualquier conjunto de datos, es importante verificar los términos de uso y asegurarte de tener permiso para utilizarlos según tus necesidades. Además, ten en cuenta las consideraciones éticas y de privacidad al trabajar con datos, especialmente si contienen información personal o sensible.

2.2.2 Datos en R

R incluye en sus librerías numerosos conjuntos de datos. Para acceder a ellos, basta con cargar la librería correspondiente.

```
library(tidyverse)
library(palmerpenguins)

summary(penguins)
```

```
     species          island    bill_length_mm  bill_depth_mm
 Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
 Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
 Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
                                 Mean   :43.92   Mean   :17.15
                                 3rd Qu.:48.50   3rd Qu.:18.70
                                 Max.   :59.60   Max.   :21.50
                                 NA's   :2       NA's   :2
 flipper_length_mm  body_mass_g       sex           year
 Min.   :172.0     Min.   :2700   female:165   Min.   :2007
 1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
 Median :197.0     Median :4050   NA's  : 11   Median :2008
 Mean   :200.9     Mean   :4202                Mean   :2008
 3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
 Max.   :231.0     Max.   :6300                Max.   :2009
 NA's   :2         NA's   :2
```

Es posible leer datos desde una cuenta de git.

```
library (readr)

urlfile="https://raw.githubusercontent.com/IsaacMartindeDiego/IA/master/datasets/californi

mydata<-read_csv(url(urlfile))

head(mydata)
```

```
# A tibble: 6 x 10
  longitude latitude housing_median_age total_rooms total_bedrooms population
      <dbl>    <dbl>              <dbl>       <dbl>          <dbl>      <dbl>
```

```
1     -122.     37.9                41         880            129        322
2     -122.     37.9                21        7099           1106       2401
3     -122.     37.8                52        1467            190        496
4     -122.     37.8                52        1274            235        558
5     -122.     37.8                52        1627            280        565
6     -122.     37.8                52         919            213        413
# i 4 more variables: households <dbl>, median_income <dbl>,
#   median_house_value <dbl>, ocean_proximity <chr>
```

```
dim(mydata)
```

```
[1] 20640    10
```

```
summary(mydata)
```

```
   longitude         latitude     housing_median_age  total_rooms
 Min.   :-124.3   Min.   :32.54   Min.   : 1.00      Min.   :    2
 1st Qu.:-121.8   1st Qu.:33.93   1st Qu.:18.00      1st Qu.: 1448
 Median :-118.5   Median :34.26   Median :29.00      Median : 2127
 Mean   :-119.6   Mean   :35.63   Mean   :28.64      Mean   : 2636
 3rd Qu.:-118.0   3rd Qu.:37.71   3rd Qu.:37.00      3rd Qu.: 3148
 Max.   :-114.3   Max.   :41.95   Max.   :52.00      Max.   :39320

 total_bedrooms     population      households     median_income
 Min.   :   1.0   Min.   :    3   Min.   :   1.0   Min.   : 0.4999
 1st Qu.: 296.0   1st Qu.:  787   1st Qu.: 280.0   1st Qu.: 2.5634
 Median : 435.0   Median : 1166   Median : 409.0   Median : 3.5348
 Mean   : 537.9   Mean   : 1425   Mean   : 499.5   Mean   : 3.8707
 3rd Qu.: 647.0   3rd Qu.: 1725   3rd Qu.: 605.0   3rd Qu.: 4.7432
 Max.   :6445.0   Max.   :35682   Max.   :6082.0   Max.   :15.0001
 NA's   :207
 median_house_value ocean_proximity
 Min.   : 14999     Length:20640
 1st Qu.:119600     Class :character
 Median :179700     Mode  :character
 Mean   :206856
 3rd Qu.:264725
 Max.   :500001
```

La estructura de datos más habitual para realizar análisis de datos es el **data frame**. ¿Has estudiado este concepto en cursos anteriores?. Los data frame son estructuras de datos de dos dimensiones (como una matriz) que pueden contener datos de diferentes tipos. Normalmente nos referimos a las filas de un data frame como observaciones o registros, mientras que las columnas reciben el nombre de campos, variables, o características.

```
L3 <- LETTERS[1:3]
char <- sample(L3, 10, replace = TRUE)
datos <- data.frame(x = 1, y = 1:10, char = char)
datos
```

```
   x  y char
1  1  1    C
2  1  2    B
3  1  3    C
4  1  4    B
5  1  5    A
6  1  6    A
7  1  7    C
8  1  8    A
9  1  9    C
10 1 10    C
```

```
is.data.frame(datos)
```

```
[1] TRUE
```

Un **tibble**, o tbl_df, es una actualización del concepto del data frame. Los tibbles son data.frames perezosos. Esto le obliga a enfrentarse a los problemas antes, lo que normalmente conduce a un código más limpio y expresivo. Los tibbles también tienen un método print() mejorado que facilita su uso con grandes conjuntos de datos que contienen objetos complejos.

```
library(tidyverse)
as.tibble(iris)
```

```
# A tibble: 150 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
```

```
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
10          4.9         3.1          1.5         0.1 setosa
# i 140 more rows
```

> **Práctica**
>
> Es importante que realices algún ejercicio en R, leyendo datos de diferentes fuentes y familiarizándote con las diferentes estructuras de datos que R proporciona.

2.2.3 Particiones de los datos

El científico de datos normalmente se enfrenta a una base de datos sobre la que trabajar. Puede ser la base de datos completa o una muestra de esta que no incluya el total de las observaciones. Quizás se ha tomado una muestra de datos empleando técnicas de muestreo. El muestreo es un concepto de gran interés que deberías de estudiar en algún curso básico de estadística. En cualquier caso, una de las primeras tareas que vamos a realizar, siempre, es la de dividir la muestra en varias particiones. En este punto es necesario comentar que una de las principales diferencias entre el ML y la estadística clásica es su propósito. Por un lado, los modelos de ML están diseñados para hacer las predicciones más exactas posibles. Por otro lado, los modelos estadísticos están diseñados para inferir las relaciones entre las variables.

Para llevar a cabo una correcta tarea de predicción será necesario contar con las siguientes particiones de los datos:

> **Entrenamiento**
>
> La partición de entrenamiento estará compuesta por las observaciones sobre las que vamos a entrenar nuestro modelo. En esta partición podemos entrenar alternativas a nuestro modelo cuantas veces sea necesario. Estas alternativas vendrán dadas por diferentes elecciones de los parámetros del modelo. ¡Sobre la partición de entrenamiento puedes probar todos los modelos que se te ocurran!

Prueba

La partición de prueba estará compuesta por las observaciones sobre las que vamos a probar nuestro modelo entrenado. Si los resultados del modelo no son los deseados podremos volver a entrenar el modelo con otros parámetros o bien cambiar de modelo. Usa esta partición para probar tus modelos y ver cómo de buenos son al enfrentarse a nuevas observaciones, diferentes a las empleadas para crear el modelo.

Validación

La partición de validación estará compuesta por las observaciones sobre las que vamos a validar nuestro modelo. Esta partición se emplea una vez seleccionado el mejor modelo. Simula la situación en la que unos nuevos datos llegan al modelo y sobre ellos se emplea el modelo para realizar una predicción. Esta partición trata de reflejar el comportamiento del modelo de ML en un entorno real. Si la base de datos original es dividida aleatoriamente en las tres particiones mencionadas, la distribución de las variables será similar en todas ellas. Es decir, el modelo entrenado en la partición de entrenamiento y probado en la partición de prueba, debe de presentar un rendimiento similar en la partición de validación. A veces nos referimos a esta partición como los **datos en producción**.

La partición de validación no será empleada sino al final del proceso. Cuando se pruebe el modelo de aprendizaje sobre ese conjunto de datos ya no podremos volver atrás. Cualquier tarea de entrenamiento del modelo será realizada única y exclusivamente sobre las particiones de entrenamiento y prueba. Es necesario indicar que algunos autores cambian el nombre de las particiones, llamando prueba a validación y validación a prueba. Tienes que comprender que es una mera cuestión de nomenclatura, podrían llamarse particiones 1, 2 y 3, (o A, B, C) siempre que su uso sea el adecuado. En cada una de estas particiones estimaremos el error (y el acierto) de los métodos de ML que consideremos. Este error será evaluado mediante métricas adecuadas que serán presentadas en el tema 6 de este curso.

Para recordar

Emplear particiones de los datos pretende minimizar el llamado **sobreajuste** del modelo que aparece cuando un modelo tiene un funcionamiento altamente dependiente de los datos con los que ha sido entrenado. Estudiaremos este problema más adelante.

En caso de sobreajuste, cuando el modelo se enfrenta a datos nuevos su rendimiento se ve deteriorado.

Existen varias técnicas estadísticas para la construcción de particiones. La primera de ellas es la más sencilla y consiste, simplemente, en reservar un conjunto (fijo) de observaciones para probar los modelos. Podemos aconsejar reservar en torno al 60% de las observaciones para la

muestra de entrenamiento, en torno al 20% para la muestra de prueba y el restante 20% para validación. Sin embargo, estos valores son muy dependientes del volumen de datos y de los objetivos particulares del problema. La principal desventaja de este método es que la estimación del error en la partición de prueba puede ser muy variable, dependiendo precisamente de las observaciones que se incluyan en el conjunto de entrenamiento y de las que se incluyan en el conjunto de prueba. Además, si te fijas bien, solo un conjunto de observaciones, siempre el mismo, se emplea para entrenar el modelo. Este hecho sugiere que esta aproximación tiende a sobreestimar el modelo. Una técnica ampliamente utilizada es la conocida como "*k-folds cross validation*". La idea es dividir el conjunto de entrenamiento en k partes iguales. Para cada de esas partes se ajusta el modelo de ML en los datos de las otras *k-1* partes (combinadas), y se obtiene el error de predicción en la k- ésima parte. De este modo se tienen k errores de los que podremos obtener la media y desviación típica. En la práctica, es muy común elegir k igual a 5 o 10. En el caso extremo de k igual número de observaciones en la muestra de entrenamiento y test, la técnica se conoce con el nombre de "*leave-one out cross validation*", aunque este método podría no mezclar los datos lo suficiente. Es decir, los datos de cada partición están altamente correlacionados y por lo tanto la varianza del error suele ser muy elevada.

2.2.3.1 Particiones en R

Hay varias formas comunes de dividir los datos en conjuntos de entrenamiento y de prueba en R:

- Base R
- caTools
- dplyr

En primer lugar os presentamos una forma muy manual de hacerlo. A veces es la más sencilla de entender y la más inmediata de razonar.

```
# cargamos los datos
data(penguins)
dim(penguins)
```

```
[1] 344   8
```

```
# mediante una semilla conseguimos que el ejercicio sea reproducible
set.seed(1)

# creamos índices
ntotal <- dim(penguins)[1]
indices <- 1:ntotal
```

```r
ntrain <- ntotal * .7
indices.train <- sample(indices, ntrain, replace = FALSE)
indices.test <- indices[-indices.train]

# Usamos el 70% de la base de datos como conjunto de entrenamiento y el resto como conjunt
train  <- penguins[indices.train, ]
test   <- penguins[indices.test, ]

dim(train)
```

```
[1] 240   8
```

```r
dim(test)
```

```
[1] 104   8
```

Otra forma alternativa.

```r
# cargamos los datos
data(penguins)
dim(penguins)
```

```
[1] 344   8
```

```r
# mediante una semilla conseguimos que el ejercicio sea reproducible
set.seed(1)

# Usamos el 70% de la base de datos como conjunto de entrenamiento y el resto como conjunt
sample <- sample(c(TRUE, FALSE), nrow(penguins), replace=TRUE, prob=c(0.7,0.3))
train  <- penguins[sample, ]
test   <- penguins[!sample, ]

dim(train)
```

```
[1] 246   8
```

```r
dim(test)
```

```
[1] 98  8
```

Usando el paquete `caTools` indicamos directamente el ratio de entrenamiento.

```r
# cargamos el paquete
library(caTools)

# cargamos los datos
data(penguins)
dim(penguins)
```

```
[1] 344   8
```

```r
# mediante una semilla conseguimos que el ejercicio sea reproducible
set.seed(1)

# Usamos el 70% de la base de datos como conjunto de entrenamiento y el resto como conjunt
sample <- sample.split(penguins$species, SplitRatio=0.7)
train  <- subset(penguins, sample==TRUE)
test   <- subset(penguins, sample==FALSE)

dim(train)
```

```
[1] 241   8
```

```r
dim(test)
```

```
[1] 103   8
```

Podemos emplear el paquete dplyr tal y como sigue:

```r
# cargamos el paquete
library(dplyr)
```

```r
# cargamos los datos
data(penguins)
dim(penguins)
```

```
[1] 344   8
```

```r
# mediante una semilla conseguimos que el ejercicio sea reproducible
set.seed(1)

# creamos una variable índice
penguins$id <- 1:nrow(penguins)

# Usamos el 70% de la base de datos como conjunto de entrenamiento y el resto como conjunt
train  <- penguins %>% dplyr::sample_frac(0.7)
test   <- dplyr::anti_join(penguins, train, by = 'id')

dim(train)
```

```
[1] 241   9
```

```r
dim(test)
```

```
[1] 103   9
```

2.3 Mantenimiento de los datos

Una vez que los datos han sido adquiridos, es esencial asegurar su mantenimiento y almacenamiento en entornos adecuados. Para lograr este objetivo, se han desarrollado múltiples sistemas de gestión de bases de datos, diseñados para satisfacer las diversas necesidades de usuarios y organizaciones. Estos sistemas de gestión de bases de datos se integran en infraestructuras más amplias y complejas que permiten el manejo eficaz y eficiente de los datos.

2.3.1 Tipos de bases de datos

Las bases de datos se pueden clasificar en relacionales y no relacionales.

Las **bases de datos relacionales** son aquellas que siguen el modelo entidad-relación, también llamado modelo relacional, en donde cada una de las tablas (o entidades) presenta algún tipo de enlace con otras (relaciones).

> **!** Para recordar
>
> Los datos almacenados en bases de datos relacionales suelen ser datos estructurados.

Para el manejo de estas bases de datos, se utiliza un lenguaje ampliamente conocido en el mundo de la Ciencia de la Computación llamado "Structured Query Language" (más conocido por sus siglas, **SQL**). Este lenguaje permite realizar consultas sobre las distintas tablas y sus relaciones, así como la gestión y mantenimiento de los datos.

Las **bases de datos no relacionales**, también llamadas NoSQL, están diseñadas para representar un modelo de datos específicos en donde su estructura y esquema son flexibles (principalmente datos no estructurados). Presentan una gran relajación respecto a las restricciones que deben cumplir por lo que favorecen una gran escalabilidad horizontal (muchos elementos del mismo tipo pueden ser fácilmente almacenados en un único documento que hace las veces de tabla). Las bases de datos no relacionales se pueden organizar como sigue:

> **i** Basadas en documentos
>
> Estas bases de datos permiten el almacenamiento de documentos completos (normalmente textuales) y partes de interés de estos. Son ampliamente utilizadas en los sistemas de gestión documental, o sistemas de recomendación, los cuales son capaces de proporcionar información de interés de acuerdo con una búsqueda determinada. Ejemplo: ElasticSearch (Dixit et al. 2017).

> **i** Gestión mediante clave-valor
>
> Son las bases de datos más comunes y tienen cierta similitud con las bases de datos relacionales. La clave actúa como clave primaria, mientras que el valor puede ser un único elemento o un conjunto de atributos relacionados. Ejemplo: MongoDB (Bradshaw, Brazil, and Chodorow 2019).

Basadas en grafos

Son bases de datos que almacenan su información por medio de nodos y aristas. Los elementos principales son los nodos, que suelen contener una serie de atributos, los cuales se relacionan con otros nodos mediante las aristas. Son muy útiles para organizar datos no estructurados como pueden ser la información que se extrae del contenido textual. Ejemplo: Neo4J (Webber 2012)

Orientadas a objetos

Son bases de datos capaces de emular la abstracción proporcionada por la programación orientada a objetos. Por lo tanto, contienen entidades (clases) que incluyen atributos y métodos de acceso y modificación de estos, produciendo encapsulamiento. Ejemplo: Gemstone (Ra, Kuno, and Rundensteiner 1994)

De trabajo en memoria

Estas bases de datos suelen estar constituidas por estructuras simples con el objetivo de poder ser alojadas en la memoria principal del computador. De esta manera se consigue que el tiempo de respuesta y la gestión y manejo de los datos sea mucho más rápida que en el resto de las bases de datos, que utilizan el disco duro del computador como lugar físico de almacenamiento. Debido a estas características son utilizadas en aplicaciones de alto rendimiento que gestionen volúmenes de datos asequibles. Ejemplo: Redis (Nelson 2016)

Para saber más sobre bases de datos

Las distintas clasificaciones presentadas en este curso son una pequeña aproximación al mundo de las bases de datos. Para más detalles respecto a los conceptos detallados en este apartado ver (Harrington 2016) para las bases de datos relacionales y (Meier and Kaufmann 2019) para las bases de datos no relacionales.

2.3.2 Infraestructuras

La infraestructura para el almacenamiento de datos en el contexto de un proyecto de ciencia de datos desempeña un papel crítico en la capacidad de procesar y analizar información de manera efectiva. Esta infraestructura se compone de sistemas y tecnologías diseñados para almacenar, gestionar y acceder a grandes volúmenes de datos de manera eficiente y segura.

En un proyecto de ciencia de datos, es común utilizar sistemas de bases de datos que pueden ser de diversos tipos, como bases de datos relacionales, bases de datos NoSQL (orientadas a documentos, columnares, gráficas, etc.) o almacenes de datos en la nube. Estos sistemas permiten organizar los datos de manera estructurada o semiestructurada, facilitando su recuperación y análisis.

Además, se pueden implementar tecnologías de almacenamiento distribuido que escalen horizontalmente para manejar conjuntos de datos masivos. Estas tecnologías permiten la redundancia y la disponibilidad continua de los datos, lo que es esencial para proyectos que requieren un alto grado de confiabilidad y rendimiento.

La elección de la infraestructura de almacenamiento adecuada dependerá de la naturaleza de los datos y de los objetivos del proyecto. La capacidad de gestionar y acceder a los datos de manera efectiva es un componente clave en la cadena de valor de la ciencia de datos, ya que permite a los científicos de datos realizar análisis, modelado y generación de conocimiento de manera más eficiente y precisa.

Aquí presentamos algunos ejemplos de infraestructuras de almacenmiento de datos:

Bases de Datos Relacionales

Ejemplos populares incluyen MySQL, PostgreSQL, Microsoft SQL Server y Oracle. Son ideales para datos estructurados y son ampliamente utilizados en aplicaciones empresariales y proyectos de análisis de datos.

Bases de Datos NoSQL

Incluyen bases de datos orientadas a documentos como MongoDB, bases de datos columnares como Apache Cassandra, y bases de datos gráficas como Neo4j. Estas bases de datos son adecuadas para datos semiestructurados y no estructurados, y son especialmente útiles en aplicaciones web y análisis de redes sociales.

Almacenes de Datos en la Nube

Servicios como Amazon S3, Google Cloud Storage y Microsoft Azure Blob Storage ofrecen soluciones de almacenamiento altamente escalables y rentables en la nube para datos estructurados y no estructurados. Son ideales para proyectos que requieren almacenamiento y acceso a gran escala. Los sistemas de computación en la nube son capaces de ofrecer un conjunto de características estandarizadas que se adaptan al consumidor, lo que los hace muy flexibles.

Hadoop Distributed File System (HDFS)

Es una parte integral del ecosistema Hadoop y se utiliza para almacenar y gestionar grandes conjuntos de datos distribuidos en clústeres de servidores. Es comúnmente utilizado en proyectos de big data y análisis de datos masivos.

Sistemas de Almacenamiento en Memoria

Ejemplos incluyen Redis y Apache Kafka. Estos sistemas permiten el almacenamiento y la recuperación ultrarrápidos de datos en memoria, lo que es esencial para aplicaciones en tiempo real y análisis de transmisiones de datos.

Sistemas de Almacenamiento Distribuido

Ejemplos incluyen Apache HBase y Apache CouchDB. Estos sistemas ofrecen alta disponibilidad y escalabilidad, lo que es crucial para aplicaciones que requieren acceso y actualización concurrentes de datos distribuidos.

Sistemas de Almacenamiento de Datos Geoespaciales

Ejemplos incluyen PostGIS y MongoDB con capacidades geoespaciales habilitadas. Estos sistemas son esenciales para proyectos que involucran datos de ubicación y análisis geoespaciales.

Almacenamiento de Datos en Almacenes de Datos Empresariales

Ejemplos incluyen sistemas como Teradata y SAP HANA. Estas soluciones se utilizan en empresas para el almacenamiento y análisis de grandes volúmenes de datos de negocios.

La elección de la infraestructura de almacenamiento dependerá de los requisitos específicos del proyecto, como el volumen de datos, la naturaleza de los datos y las necesidades de rendimiento y escalabilidad. Cada una de estas infraestructuras tiene sus propias ventajas y desafíos, y la elección adecuada contribuirá al éxito del proyecto de ciencia de datos.

2.4 Calidad de los datos

La calidad de los datos es un elemento fundamental en cualquier proyecto de ciencia de datos, ya que influye directamente en la confiabilidad y el valor de los resultados obtenidos. La calidad de los datos se refiere a la precisión, la integridad, la consistencia y la relevancia de los datos que se utilizan en el análisis.

Para recordar

La calidad de los datos es la piedra angular de la confiabilidad y precisión en la ciencia de datos.

Despliega los paneles para averiguar algunas consideraciones clave sobre la calidad de los datos en un proyecto de ciencia de datos.

Precisión

Los datos deben ser precisos y libres de errores. Los errores en los datos pueden surgir de diversas fuentes, como entradas incorrectas, mediciones inexactas o problemas de registro. Es fundamental garantizar la precisión de los datos para evitar interpretaciones erróneas y resultados incorrectos.

Integridad

La integridad de los datos se refiere a su completitud y coherencia. Los datos incompletos o faltantes pueden dificultar el análisis y llevar a conclusiones sesgadas. La integridad también se relaciona con la consistencia de los datos a lo largo del tiempo y entre diferentes fuentes.

Consistencia

Los datos deben ser coherentes en su estructura y formato. Esto implica que las mismas categorías, unidades de medida y convenciones deben aplicarse de manera consistente en todo el conjunto de datos. La falta de consistencia puede dificultar la combinación y el análisis de datos de múltiples fuentes.

Relevancia

Los datos deben ser relevantes para los objetivos del proyecto. Incluir datos irrelevantes puede aumentar la complejidad del análisis sin aportar valor. Es importante definir claramente qué datos son necesarios para abordar las preguntas o problemas específicos que se plantean en el proyecto.

Actualización

Los datos deben mantenerse actualizados. En muchos proyectos de ciencia de datos, la información envejece con el tiempo y puede perder su relevancia o precisión. Es esencial establecer procesos para la actualización y el mantenimiento continuo de los datos.

Limpieza de Datos

La limpieza de datos es el proceso de identificar y corregir errores, duplicados y valores atípicos en el conjunto de datos. Es una etapa crítica para mejorar la calidad de los datos antes de su análisis. Abordaremos esta etapa en el tema 3, cuando tratemos el Análisis Exploratorio de Datos.

Documentación

Es importante documentar el origen de los datos, las transformaciones realizadas, las decisiones tomadas durante la preparación de los datos y cualquier asunción que se haya hecho. La documentación adecuada facilita la comprensión y la replicación de los análisis.

La calidad de los datos influye en todas las etapas de un proyecto de ciencia de datos, desde la recopilación y preparación de datos hasta el modelado y la interpretación de resultados. Los científicos de datos y analistas deben ser conscientes de la calidad de los datos y trabajar activamente para garantizar que los datos sean confiables y adecuados para los objetivos del proyecto. La inversión en la calidad de los datos es esencial para tomar decisiones informadas y obtener conocimientos precisos y significativos.

2.5 Ética, privacidad y seguridad en los datos

La ética, privacidad y seguridad en los datos son aspectos entrelazados y fundamentales para garantizar que la recopilación, el análisis y el uso de datos se realicen de manera responsable y en beneficio de la sociedad. No solo son cuestiones éticas, sino que también tienen implicaciones legales y pueden afectar la reputación de las organizaciones. Los profesionales de datos y las empresas deben adoptar un enfoque proactivo para garantizar que sus prácticas de datos sean éticas y cumplan con las regulaciones aplicables, lo que a su vez contribuye a la construcción de la confianza del público en el uso de la ciencia de datos en la sociedad.

Ética

La **ética** en la ciencia de datos se refiere a la conducta y las decisiones éticas de los profesionales de datos en todas las etapas del proceso, desde la recopilación hasta la interpretación y comunicación de resultados. Algunos puntos clave incluyen:

- **Transparencia**: Debe existir claridad sobre cómo se obtienen y utilizan los datos, así como sobre los métodos de análisis.
- **Consentimiento**: Es fundamental obtener el consentimiento informado de las personas cuyos datos se recopilan, especialmente en el contexto de datos personales.

- **Equidad**: Los análisis deben ser imparciales y justos, evitando la discriminación y el sesgo en los resultados.
- **Responsabilidad**: Los profesionales de datos deben asumir la responsabilidad de los posibles impactos de sus decisiones y comunicar sus hallazgos de manera precisa y ética.

i Privacidad

La **privacidad** de los datos se refiere a la protección de la información personal y sensible. En la ciencia de datos, es crucial proteger la privacidad de las personas y cumplir con las regulaciones de privacidad aplicables. Algunos aspectos importantes son:

- **Anonimización**: La eliminación o la codificación de información identificable en los datos es esencial para proteger la privacidad.
- **Seguridad de los Datos**: Los datos deben ser almacenados y transmitidos de manera segura para evitar la exposición no autorizada.
- **Cumplimiento Normativo**: Es necesario cumplir con leyes y regulaciones de privacidad, como el Reglamento General de Protección de Datos (RGPD) en la Unión Europea.

i Seguridad

La **seguridad** de los datos es la protección contra el acceso no autorizado, la alteración o la destrucción de los datos. Para garantizar la seguridad de los datos, es importante:

- **Control de Acceso**: Limitar el acceso a los datos solo a personas autorizadas mediante autenticación y autorización adecuadas.
- **Cifrado**: Utilizar técnicas de cifrado para proteger los datos tanto en reposo como en tránsito.
- **Monitoreo y Detección de Amenazas**: Implementar sistemas de monitoreo y detección de amenazas para identificar y responder a posibles ataques o violaciones de seguridad.
- **Respaldo de Datos**: Realizar copias de seguridad regulares para evitar la pérdida de datos en caso de incidentes de seguridad.

3 Análisis Exploratorio de Datos

El análisis exploratorio de datos o (EDA, del inglés "Exploratory Data Analysis") representa un conjunto de técnicas que permiten resumir los aspectos más importantes de un conjunto de datos, normalmente con especial énfasis en el uso de métodos de visualización gráfica. El término fue popularizado, entre otros, por el estadístico norteamericano *John W. Tukey* como método para descubrir información importante (y no evidente) contenida en los datos (Tukey et al. 1977). Estas técnicas se emplean habitualmente como paso previo a la inferencia estadística, orientada hacia un análisis confirmatorio. Así, con EDA se estudian los datos, se descubre cómo son y cómo se comportan y con la inferencia estadística se comprueba analíticamente si esos comportamientos y diferencias halladas son realmente significativos (desde un punto de vista estadístico). El EDA es, sin duda, fundamental para adquirir conocimiento de los datos, antes de emplearlos dentro de un modelo de ML como los que vais a aprendener a lo largo del grado en Ciencia e Ingeniería de Datos. Es un error muy típico de algunos analistas de datos aplicar modelos a sus datos en cuanto estos están disponibles sin pasar previamente por el necesario análisis exploratorio de los mismos.

> John Tukey
>
> "El análisis exploratorio de datos es una actitud, un estado de flexibilidad, una voluntad de buscar aquellas cosas que creemos que no están ahí, así como aquellas que creemos que están ahí."

Es decir, el EDA no sigue un proceso formal con normas estrictas. Más bien, es una mentalidad o enfoque. En otras palabras, una forma de hacer las cosas. Cuando lleves a cabo un EDA, debes sentirte libre para explorar todas las ideas que se te ocurran. Algunas de estas ideas serán fructíferas, mientras que otras pueden llevarte a callejones sin salida. No te preocupes, probablemente no hayas roto nada. Simplemente habrás "gastado" tiempo. A medida que sigas explorando, te enfocarás en áreas particularmente prometedoras, las cuales documentarás y compartirás con otros.

A menudo se necesita mucho tiempo para explorar los datos. Se dice que el *80%* del tiempo del proyecto se gasta en EDA. A través del proceso de EDA, podemos pedir que se redefina el enunciado del problema o la definición de nuestro conjunto de datos, lo cual es muy importante.

! Para recordar

Cuando nos enfrentamos a un EDA, lo ideal es contar con un objetivo que se haya definido junto con los datos, indicando qué se quiere conseguir a partir de ellos. Por ejemplo, "*predecir las ventas en los próximos 30 días*", "*estimar el riesgo que tiene un paciente de no superar una determinada operación quirúrgica*", "*clasificar como fraudulenta, o no, una página web*", etc.

EDA es un ciclo iterativo:

1. Genera preguntas sobre los datos.
2. Busca respuestas visualizando, transformando y modelizando los datos.
3. Utiliza lo que hayas aprendido para refinar las preguntas y/o generar otras nuevas.

3.1 Preguntas

Tu objetivo principal durante el EDA es adquirir una comprensión profunda de los datos. La forma más sencilla de hacerlo es utilizar preguntas como herramientas para guiar la investigación. Cuando planteas una pregunta, ésta centra tu atención en una parte específica del conjunto de datos y te ayuda a decidir qué gráficos, modelos o transformaciones realizar.

EDA es un proceso creativo y como tal, la clave para llevarlo a cabo consiste en el planteamiento de preguntas de calidad. ¿Qué preguntas son las correctas? La respuesta es que depende del conjunto de datos con el que se trabaje.

John Tukey

Mucho mejor una respuesta aproximada a la pregunta correcta, que a menudo es vaga, que una respuesta exacta a la pregunta incorrecta, que siempre se puede precisar

Al inicio del análisis, puede resultar todo un desafío formular preguntas reveladoras, ya que aún no se conoce completamente la información contenida en el conjunto de datos. Sin embargo, cada nueva pregunta que plantees te llevará a explorar un nuevo aspecto de tus datos, aumentando así las posibilidades de hacer descubrimientos importantes.

! Para recordar

Durante la preparación y limpieza de los datos acumulamos pistas sobre los modelos más adecuados que podrán ser aplicados en etapas posteriores.

Algunas de las preguntas que, generalmente, deberían de abordarse durante el EDA son:

- ¿Cuál es el tamaño de la base de datos? Es decir:
  - ¿Cuántas observaciones hay?
  - ¿Cuántas variables/características están medidas?
  - ¿Disponemos de capacidad de cómputo en nuestra máquina para procesar la base de datos o necesitamos más recursos?
  - ¿Existen valores faltantes?
- ¿Qué tipo variables aparecen en la base de datos?
  - ¿Qué variables son discretas?
  - ¿Cuáles son continuas?
  - ¿Qué categorías tienen las variables?
  - ¿Hay variables tipo texto?
- Variable objetivo: ¿Existe una variable de "respuesta"?
  - ¿Binaria o multiclase?
- ¿Es posible identificar variables irrelevantes?. Estudiar variables relevantes requiere, habitualmente, métodos estadísticos.
- ¿Es posible identificar la distribución que siguen las variables?
- Calcular estadísticos resumen (media, desviación típica, frecuencia,...) de todas las variables de interés.
- Detección y tratamiento de valores atípicos.
  - ¿Son errores de media?
  - ¿Podemos eliminarlos?
- ¿Existe correlación entre variables?

> **!** Para recordar
>
> Una correcta preparación y limpieza de datos implica, sin duda, un ahorro de tiempo en etapas posteriores del proyecto.

3.2 Entender el negocio

La comprensión del problema que estamos abordando representa una de las primeras etapas en cualquier proyecto de ciencia de datos. En la mayoría de los casos, esta tarea se realiza en estrecha colaboración con expertos en el dominio correspondiente, quienes a menudo son las personas que han solicitado (y a menudo financian) el análisis de datos. Es importante recordar que cualquier estudio que involucre ciencia de datos requiere un conocimiento profundo del dominio, el cual debe ser compartido con el científico de datos. Por lo tanto, el profesional de la ciencia de datos debe poseer un conocimiento suficiente para enfrentar con confianza los diversos desafíos que puedan surgir. Esta comprensión inicial permite establecer los objetivos del proyecto y procesar los datos de manera significativa para obtener información valiosa. A través de esta información, se busca derivar conocimientos aplicables. Este conocimiento puede ser aprendido y almacenado para su uso futuro, lo que lleva a la sabiduría, según la jerarquía de conocimiento presentada en la Figura Figure 3.1.

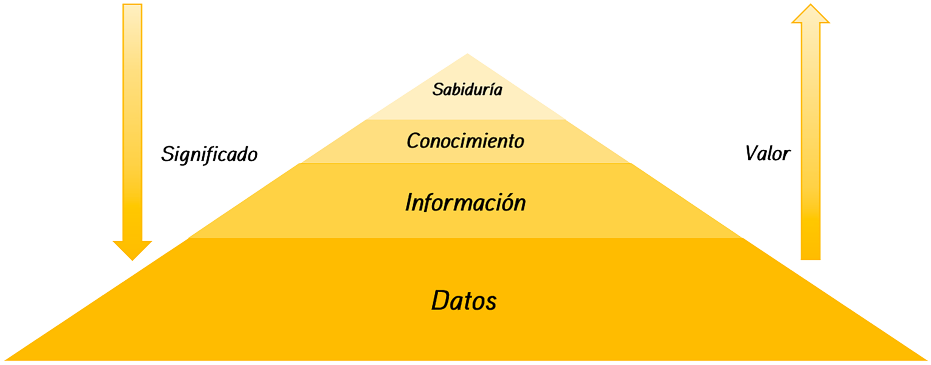


Figure 3.1: Jerarquía de Conocimiento

Claude Lévi-Strauss

"El científico no es una persona que da las respuestas correctas, sino una persona que hace las preguntas correctas."

3.3 Un primer vistazo a los datos

En este tema vamos a trabajar con los datos de **Bank Marketing** del repositorio UCI. En primer lugar debemos comprender el problema. ¿Qué sabes del marketing bancario? En el caso que nos ocupa, los datos están relacionados con campañas de marketing directo (llamadas telefónicas) de una entidad bancaria portuguesa. El objetivo de la clasificación es *predecir si el cliente suscribirá un depósito* a plazo (variable objetivo).

Las variables que debemos estudiar son:

Variables de entrada:

datos del cliente bancario:

1. edad (variable numérica)
2. empleo : tipo de empleo (variable categórica con las siguientes categorías: “admin.”, “desconocido”, “desempleado”, “directivo”, “empleada del hogar”, “empresario”, “estudiante”, “obrero”, “autónomo”, “jubilado”, “técnico”, “servicios”)
3. estado civil : estado civil (variable categórica con categorías: “casado”, “divorciado”, “soltero”; nota: “divorciado” significa divorciado o viudo)
4. educación (variable categórica con categorías: “desconocida”, “secundaria”, “primaria”, “terciaria”)
5. impago: ¿tiene un crédito impagado? (variable binaria con dos posibles valores: “sí”, “no”)
6. saldo: saldo medio anual, en euros (variable numérica)
7. vivienda: ¿tiene préstamo para vivienda? (variable binaria: “sí”, “no”)
8. préstamo: ¿tiene préstamo personal? (variable binaria: “sí”, “no”)

   # relacionado con el último contacto de la campaña actual:

9. contacto: tipo de comunicación del contacto (variable categórica: “desconocido”, “teléfono”, “móvil”)
10. día: día del mes del último contacto (variable numérica)
11. mes: mes del año del último contacto (variable categórica: “ene”, “feb”, “mar”, ..., “nov”, “dic”)
12. duración: duración del último contacto, en segundos (variable numérica)

otros atributos

13. campaña: número de contactos realizados durante esta campaña y para este cliente (variable numérica, incluye el último contacto)
14. pdays: número de días transcurridos desde que el cliente fue contactado por última vez en una campaña anterior (variable numérica, -1 significa que el cliente no fue contactado previamente)
15. previous: número de contactos realizados antes de esta campaña y para este cliente (variable numérica)

16. poutcome: resultado de la campaña de marketing anterior (variable categórica: "desconocido", "otro", "fracaso", "éxito")

Variable de salida (objetivo deseado):

17 - y: ¿ha suscrito el cliente un depósito a plazo? (variable binaria: "sí", "no")

> **!** Para recordar
>
> A veces (muchas veces) la descripción que encontramos en una primera etapa no coincide al completo con los datos que luego nos entrega el cliente.
> En otras ocasiones no se dispone de la descripción de las variables. En ese caso, ¡hay que hacer lo imposible por conseguirla!

Leemos los datos con R.

```
library(tidyverse)
bank = read.csv('https://raw.githubusercontent.com/rafiag/DTI2020/main/data/bank.csv')
dim(bank)
```

```
[1] 11162    17
```

```
bank=as.tibble(bank)
bank
```

```
# A tibble: 11,162 x 17
     age job       marital education default balance housing loan  contact   day
   <int> <chr>     <chr>   <chr>     <chr>     <int> <chr>   <chr> <chr>   <int>
 1    59 admin.    married secondary no         2343 yes     no    unknown     5
 2    56 admin.    married secondary no           45 no      no    unknown     5
 3    41 technici~ married secondary no         1270 yes     no    unknown     5
 4    55 services  married secondary no         2476 yes     no    unknown     5
 5    54 admin.    married tertiary  no          184 no      no    unknown     5
 6    42 manageme~ single  tertiary  no            0 yes     yes   unknown     5
 7    56 manageme~ married tertiary  no          830 yes     yes   unknown     6
 8    60 retired   divorc~ secondary no          545 yes     no    unknown     6
 9    37 technici~ married secondary no            1 yes     no    unknown     6
10    28 services  single  secondary no         5090 yes     no    unknown     6
# i 11,152 more rows
# i 7 more variables: month <chr>, duration <int>, campaign <int>, pdays <int>,
#   previous <int>, poutcome <chr>, deposit <chr>
```

Disponemos de más de 10000 observaciones y un total de 17 variables.

3.4 Tipo de variables

Para averiguar qué tipo de variables manejamos, ejecutar:

```
str(bank)
```

```
tibble [11,162 x 17] (S3: tbl_df/tbl/data.frame)
 $ age      : int [1:11162] 59 56 41 55 54 42 56 60 37 28 ...
 $ job      : chr [1:11162] "admin." "admin." "technician" "services" ...
 $ marital  : chr [1:11162] "married" "married" "married" "married" ...
 $ education: chr [1:11162] "secondary" "secondary" "secondary" "secondary" ...
 $ default  : chr [1:11162] "no" "no" "no" "no" ...
 $ balance  : int [1:11162] 2343 45 1270 2476 184 0 830 545 1 5090 ...
 $ housing  : chr [1:11162] "yes" "no" "yes" "yes" ...
 $ loan     : chr [1:11162] "no" "no" "no" "no" ...
 $ contact  : chr [1:11162] "unknown" "unknown" "unknown" "unknown" ...
 $ day      : int [1:11162] 5 5 5 5 5 5 6 6 6 6 ...
 $ month    : chr [1:11162] "may" "may" "may" "may" ...
 $ duration : int [1:11162] 1042 1467 1389 579 673 562 1201 1030 608 1297 ...
 $ campaign : int [1:11162] 1 1 1 1 2 2 1 1 1 3 ...
 $ pdays    : int [1:11162] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ previous : int [1:11162] 0 0 0 0 0 0 0 0 0 0 ...
 $ poutcome : chr [1:11162] "unknown" "unknown" "unknown" "unknown" ...
 $ deposit  : chr [1:11162] "yes" "yes" "yes" "yes" ...
```

Creamos las particiones sobre los datos y trabajamos sobre la partición de entrenamiento.

```
# Parciticionamos los datos
set.seed(2138)
n=dim(bank)[1]
indices=seq(1:n)
indices.train=sample(indices,size=n*.5,replace=FALSE)
indices.test=sample(indices[-indices.train],size=n*.25,replace=FALSE)
indices.valid=indices[-c(indices.train,indices.test)]

bank.train=bank[indices.train,]
bank.test=bank[indices.test,]
bank.valid=bank[indices.valid,]
```

Atrévete

¿Te has hecho (ya) alguna pregunta sobre los datos? Si es así, no esperes más, busca la respuesta!

Por ejemplo, ¿qué te parecen estas preguntas que nosotros proponemos?

¿Qué día del año se producen más depósitos por parte de los estudiantes?

```
bank.train %>%
  filter(deposit=="yes") %>%
  count(month, day) %>%
  top_n(1,n)
```

```
# A tibble: 1 x 3
  month   day     n
  <chr> <int> <int>
1 apr      30    79
```

¿En qué mes del año realizan más depósitos los estudiantes?

```
bank.train %>%
  filter(deposit=="yes" & job=="student") %>%
  count(month) %>%
  top_n(1,n)
```

```
# A tibble: 1 x 2
  month     n
  <chr> <int>
1 apr      21
```

¿Qué trabajo está asociado con el mayor porcentaje de depósitos?

```
bank.train %>%
  group_by(job) %>%
  mutate(d = n()) %>%
  group_by(job, deposit) %>%
  summarise(Perc = n()/first(d), .groups = "drop") %>%
  pivot_wider(
    id_cols = job,
```

```
    names_from = deposit,
    values_from = Perc
  ) %>%
  top_n(1)
```

```
# A tibble: 1 x 3
  job        no   yes
  <chr>   <dbl> <dbl>
1 student 0.218 0.782
```

> Repaso
>
> `dplyr` es un paquete en R diseñado para facilitar la manipulación y transformación de datos de manera eficiente y estructurada. Fue desarrollado por **Hadley Wickham** y se ha convertido en una de las herramientas más populares en la ciencia de datos y análisis de datos.

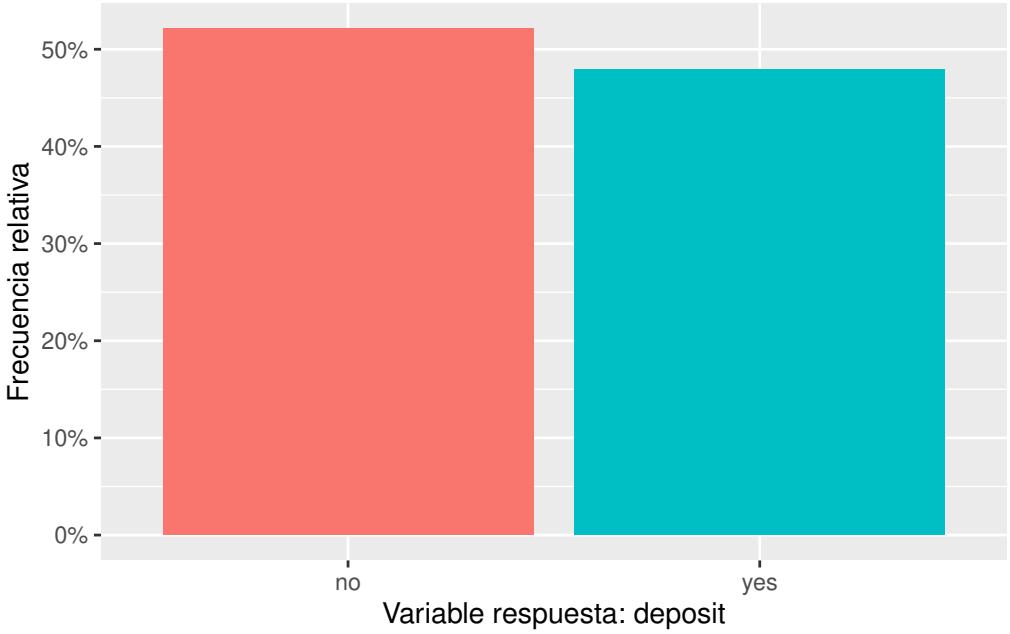
3.5 Variable objetivo

En problemas de clasificación (Aprendizaje Supervisado, ver apartado Section 1.4) existe una variable de interés fundamental, es la variable respuesta o variable objetivo. En el caso que nos ocupa dicha variable es la característica: "deposit". Vamos a estudiar la información que nos proporciona dicha variable.

```
library(ggplot2)
table(bank.train$deposit)
```

```
  no  yes
2908 2673
```

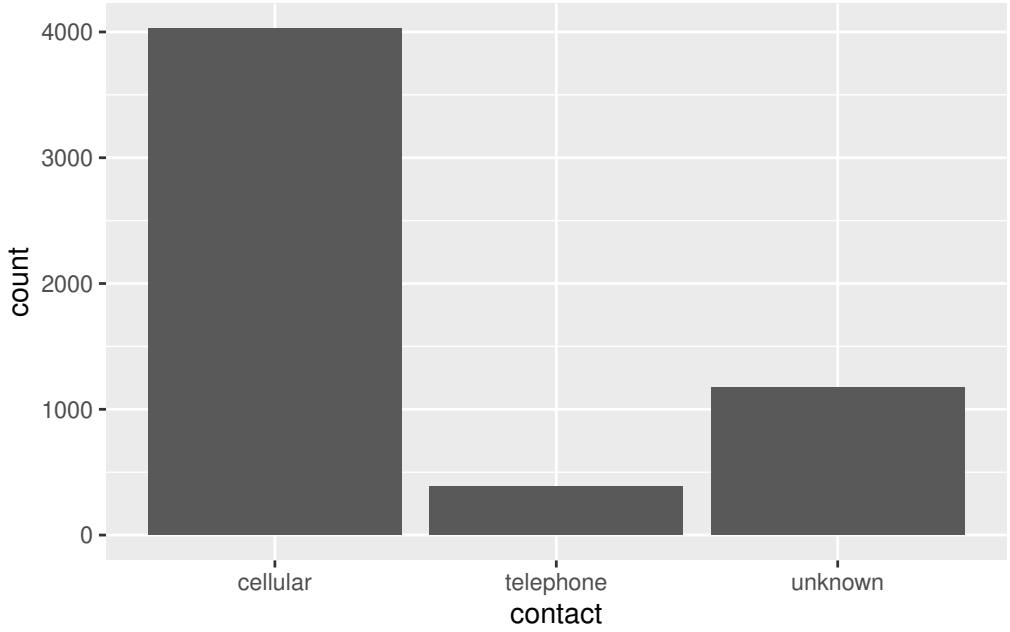
```
ggplot(data=bank.train,aes(x=deposit,fill=deposit)) +
  geom_bar(aes(y=(..count..)/sum(..count..))) +
  scale_y_continuous(labels=scales::percent) +
  theme(legend.position="none") +
  ylab("Frecuencia relativa") +
  xlab("Variable respuesta: deposit")
```



3.6 Visualizar distribuciones

La forma de visualizar la distribución de una variable dependerá de si la variable es categórica o continua. Una variable es categórica si sólo puede tomar uno de un pequeño conjunto de valores. En R, las variables categóricas suelen guardarse como factores o vectores de caracteres. Para examinar la distribución de una variable categórica, utiliza un gráfico de barras:

```
ggplot(data = bank.train) +
  geom_bar(mapping = aes(x = contact))
```



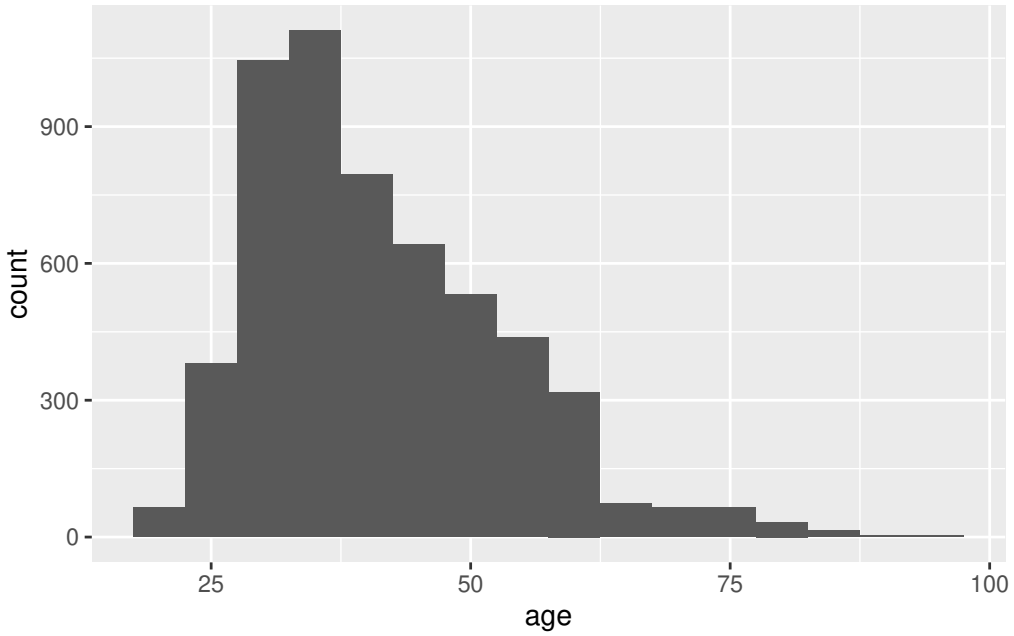
Puedes obtener los valores exactos en cada categoría como sigue:

```
bank.train%>%
  count(contact)
```

```
# A tibble: 3 x 2
  contact       n
  <chr>     <int>
1 cellular   4025
2 telephone   383
3 unknown    1173
```

Una variable es continua si puede tomar cualquiera de un conjunto infinito de valores ordenados. Para examinar la distribución de una variable continua, utiliza un histograma:

```
ggplot(data = bank.train) +
  geom_histogram(mapping = aes(x = age), binwidth = 5)
```

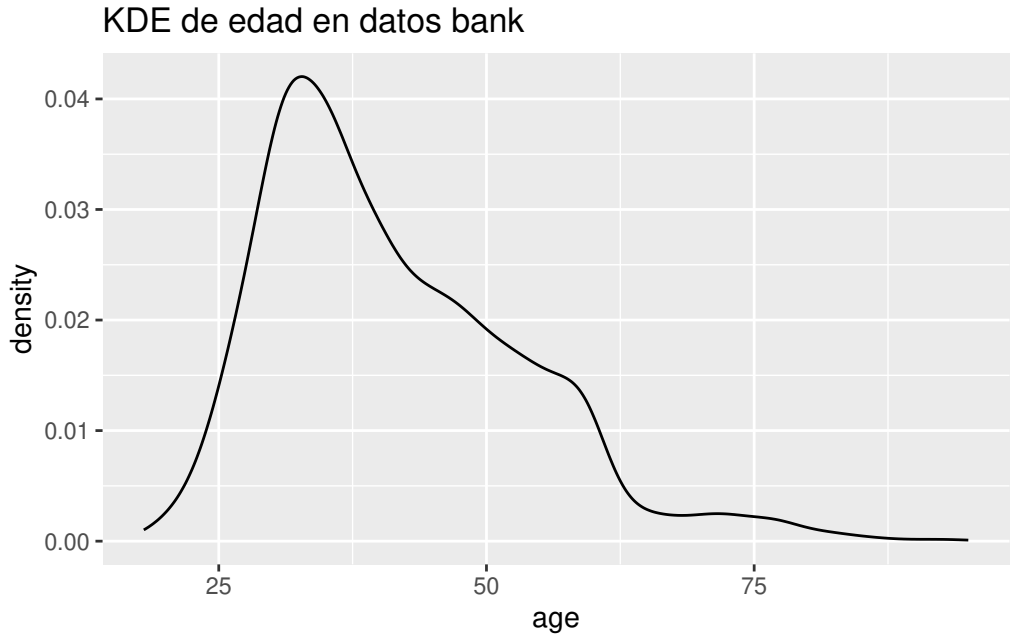


Un histograma divide el eje x en intervalos equidistantes y, a continuación, utiliza la altura de una barra para mostrar el número de observaciones que se encuentran en cada intervalo. En el gráfico anterior, la primera barra muestra unas 100 observaciones (realmente son 119) tienen un valor de edad por debajo de 22.5 años. Puede establecer la anchura de los intervalos en un histograma con el argumento binwidth, que se mide en las unidades de la variable x.

> ❗ Para recordar
>
> Siempre se deben explorar una variedad de anchos de intervalo cuando trabajamos con histogramas, ya que diferentes anchos de intervalo pueden revelar diferentes patrones.

Podemos representar funciones de densidad de probabilidad.

```
ggplot(bank.train, aes(x = age)) +
geom_density() +
ggtitle('KDE de edad en datos bank')
```

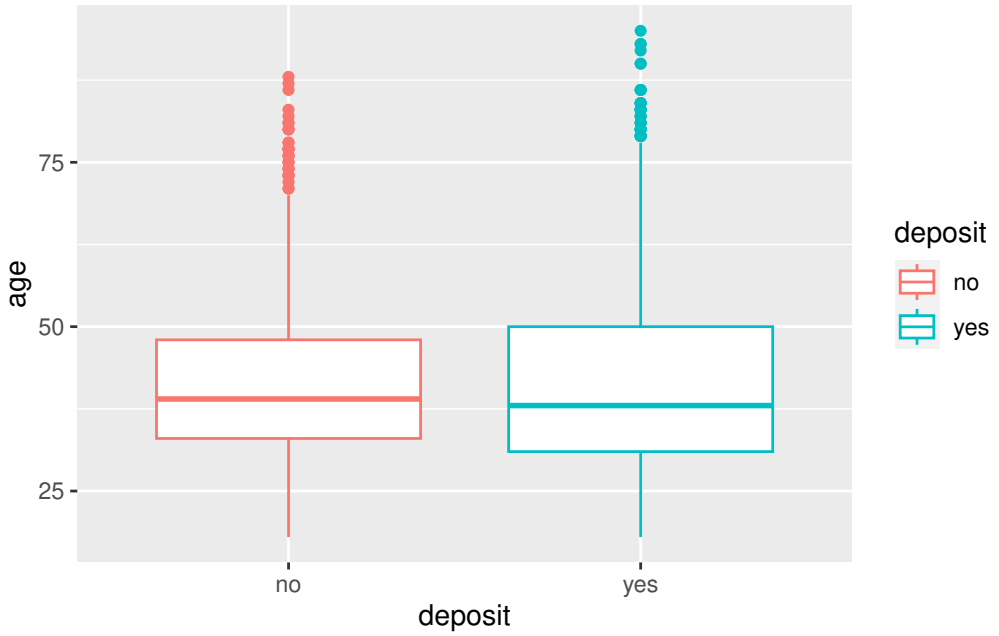


Otro gráfico muy utilizado para variables cuantitativas univariantes es el *boxplot*, también llamado *box-and-whisker plot* (diagrama de caja y bigotes). Es especialmente útil para detectar posibles datos atípicos en los valores de una variable, siempre que su distribución sea parecida a una distribución *Normal*. El gráfico muestra:

- Una caja cuyos límites son el primer y el tercel cuartil de la distribución de valores.
- Una línea central, que marca la mediana.
- Los bigotes, que por defecto (en R) se extienden hasta 1.5 veces el valor del rango intercuartílico (IQR) por encima y por debajo de la caja.
- Puntos individuales, que quedan más allá del límite de los bigotes, marcan posibles datos atípicos.

En distribuciones muy asimétricas o con muchos valores extremos, muy diferentes a una distribución *Normal*, aparecerán demasiados puntos más allá de los bigotes y no se podrán apreciar fácilmente los atípicos (demasiados puntos considerados como tales). En ese caso, es conveniente intentar una transformación de la variable antes de representar el *boxplot*.

```
ggplot(bank.train, aes(x=deposit, y=age, color=deposit)) +
  geom_boxplot()
```

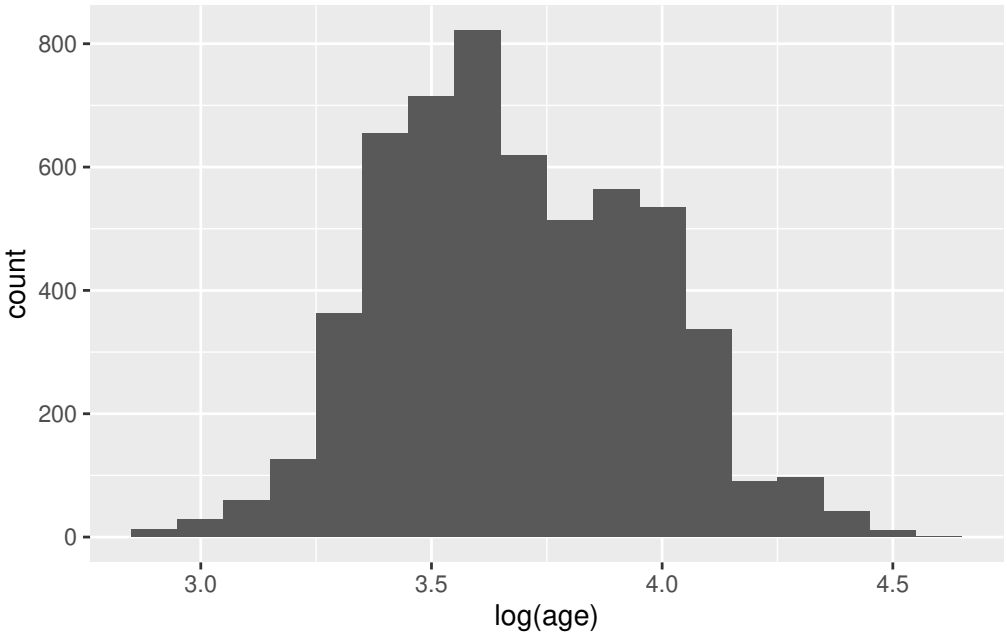


3.7 Transformación de variables

3.7.1 Transformación de variables cuantitativas

En algunos métodos de ML es necesario contar con variables que cumplan requisitos de normalidad. Por ejemplo, si tomamos la transformación *log* sobre la variable `edad` obtenemos una distribución multimodal que, probablemente, corresponda a la combinación de dos (o más) normales.

```
ggplot(data = bank.train) +
  geom_histogram(mapping = aes(x = log(age)), binwidth = .1)
```



! Para recordar

Los modelos de ML serán tan buenos como lo sean las variables de entrada de dichos algoritmos.

3.7.1.1 Transformaciones para igualar dispersión

Con frecuencia, el objetivo de la transformación de variables cuantitativas es obtener una variable cuya distribución de valores sea:

- Más simétrica y con menor dispersión que la original.
- Más semejante a una distribución normal (e.g. para algunos modelos lineales).
- Restringida en un intervalo de valores (e.g. $[0, 1]$).

La forma más sencilla de detectar que alguna de nuestras variables necesita ser transformada es representar un gráfico que muestre la distribución de valores de la variable. Por ejemplo, un histograma o un diagrama de densidad de probabilidad (o ambos).

El uso de los logaritmos tiene su propia recomendación en preparación de datos (Fox and Weisberg 2018):

John Fox

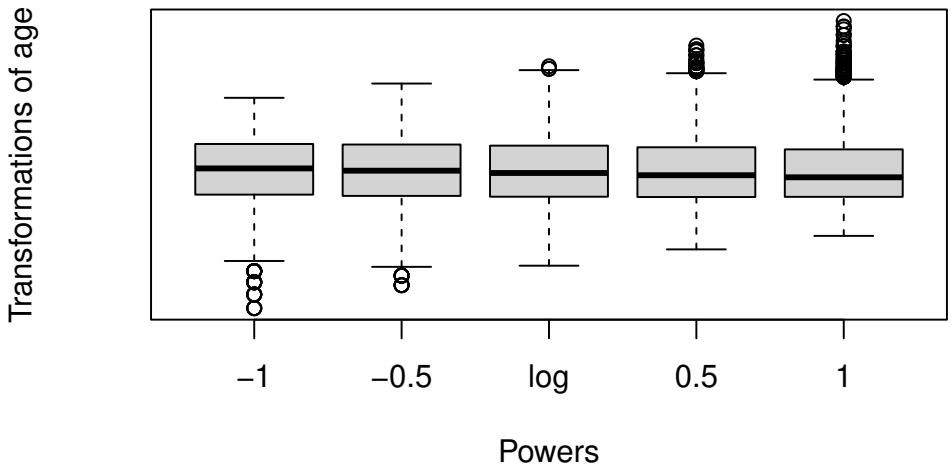
"Si la variable es estrictamente positiva, no tiene un límite superior para sus valores, y su rango abarca dos o más órdenes de magnitud (potencias de 10), entonces la transformación logarítmica suele ser útil. A la inversa, cuando la variable tiene un rango de valores pequeño (menor de un orden de magnitud), el logaritmo o cualquier otra transformación simple no ayudará mucho."

La versión general de esta transformación son las transformaciones de escala-potencia (scaled-power transformations), también denominadas transformaciones de **Box-Cox**.

$$x(\lambda) = \begin{cases} \frac{x^{\lambda}-1}{\lambda}, & \text{cuando } \lambda \neq 0, \\ log_e(x), & \text{cuando } \lambda = 0 \end{cases}$$

La función `car::symbox(...)` permite probar varias combinaciones típicas del parámetro , para comprobar con cuál de ellas obtenemos una distribución más simétrica de valores.

```
library(car)

bank.train %>% symbox(~ age, data = .)
```



3.7.1.2 Transformaciones para igualar dispersión

También es bastante común aplicar transformaciones en datos cuantitativos para igualar las escalas de representación de las variables. En muchos modelos, si una de nuestras variables

tiene una escala mucho mayor que las demás, sus valores tienden a predominar en los resultados, enmascarando la influencia del resto de variables en el modelo.

Por este motivo, en muchos modelos es importante garantizar que todas las variables se representan en escalas comparables, de forma que ninguna predomine sobre el resto. Conviene aclarar un poco algunos términos que se suelen emplear de forma indistinta:

- **Reescalado o cambio de escala**: Consiste en sumar o restar una constante a un vector, y luego multiplicar o dividir por una constante. Por ejemplo, para transformar la unidad de medida de una variable (grados Farenheit → grados Celsius).
- **Normalización**: Consiste en dividir por la norma de un vector, por ejemplo para hacer su distancia euclídea igual a 1.
- **Estandarización**: Consiste en restar a un vector una medida de localización o nivel (e.g. media, mediana) y dividir por una medida de escala (dispersión). Por ejemplo, si restamos la media y dividimos por la desviación típica hacemos que la distribución tenga media 0 y desviación típica 1.

Algunas aternativas comunes son:

$$Estandarizacin \rightarrow Y = \frac{X - \overline{x}}{s_x}$$

$$Escaladomin - max \rightarrow Y = \frac{X - min_x}{max_x - min_x}$$

En R, la función `scale()` se puede utilizar para realizar estas operaciones de estandarización. Automáticamente, puede actuar sobre las columnas de un `data.frame`, aplicando la misma operación a todas ellas (siempre que todas sean cuantitativas).

3.7.2 Transformación de variables cualitativas

A diferencia de las variables cuantitativas, que representan cantidades numéricas, las variables cualitativas, también conocidas como variables categóricas, se utilizan para describir características o cualidades que no tienen un valor numérico intrínseco. Las variables cualitativas son esenciales en la investigación y el análisis de datos, ya que a menudo se utilizan para clasificar, segmentar y comprender información sobre grupos, categorías o características. Algunas técnicas comunes para analizar variables cualitativas incluyen la creación de tablas de frecuencia para contar la ocurrencia de cada categoría y el uso de gráficos como gráficos de barras o diagramas de sectores para visualizar la distribución de categorías. Estos análisis pueden proporcionar información valiosa sobre patrones, tendencias y relaciones en los datos cualitativos, lo que puede ser fundamental para tomar decisiones informadas en una amplia gama de campos, desde marketing hasta investigación social y más.

Las variables cualitativas se dividen en dos categorías principales:

> **Variables Cualitativas Nominales**
>
> Las variables nominales representan categorías o etiquetas que no tienen un orden inherente. Ejemplos comunes incluyen el género (masculino, femenino, otro), el estado civil (soltero, casado, divorciado) o los colores (rojo, azul, verde). No se pueden realizar operaciones matemáticas en variables nominales, como sumar o restar.

> **Variables Cualitativas Ordinales**
>
> Las variables ordinales representan categorías con un orden natural o jerarquía, pero la distancia entre las categorías no es necesariamente uniforme ni conocida. Ejemplos incluyen la calificación de satisfacción del cliente (muy insatisfecho, insatisfecho, neutral, satisfecho, muy satisfecho) o el nivel de educación (primaria, secundaria, universitaria). Aunque se pueden establecer comparaciones de orden (por ejemplo, "mayor que" o "menor que"), no es apropiado realizar operaciones matemáticas en variables ordinales.

En R, las variables categóricas se denominan **factores** (factors) y sus categorías **niveles** (levels). Es importante procesarlos adecuadamente para que los modelos aprovechen la información que contienen estas variables. Por otro lado, si se codifica incorrectamente esta información los modelos pueden estar realizando operaciones absurdas aunque nos devuelvan resultados aparentemente válidos.

> **R**
>
> Por defecto, R transforma columnas tipo string en factores al leer los datos de un archivo. Además, por defecto, R **ordena los niveles de los factores alfabéticamente**, según sus etiquetas. Debemos tener cuidado con esto, puesto que en muchos análisis es muy importante saber qué nivel se está tomando como referencia, de entre los valores posibles de un factor, para comparar con los restantes. En ciertos modelos, la elección como referencia de uno de los valores del factor (típicamente el primero que aparece en la lista de niveles) cambia por completo los resultados, así como la interpretación de los mismos.

En variables ordinales se debe respetar estrictamente el orden preestablecido de los niveles. Por ejemplo, una ordenación ("regular" < "bueno" < "malo") es inaceptable. Para establecer una ordenación explícita entre los niveles hay que especificarla manualmente si no coincide con la alfabética, y además configurar el argumento ordered = TRUE en la función factor():

```
satisfaccion <- rep(c("malo", "bueno", "regular"), c(3,3,3))
satisfaccion <- factor(satisfaccion, ordered = TRUE, levels = c("malo", "regular", "bueno"
satisfaccion
```

```
[1] malo    malo    malo    bueno   bueno   bueno   regular regular regular
Levels: malo < regular < bueno
```

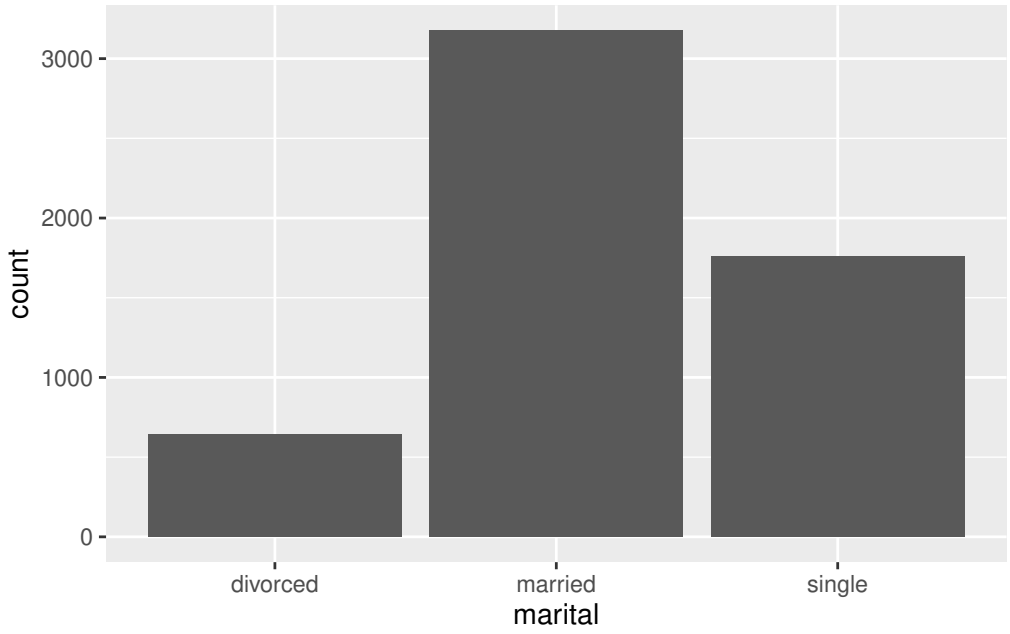
Para comprobar qué nivel se toma como referencia en cada uno de los factores de una base de datos usamos la funión `levels()`:

```
levels(bank.train$marital)
```

```
NULL
```

Y esto, ¿es correcto? Veamos la distribución de las observaciones en las categorías de la variable `marital`:

```
ggplot(data = bank.train) +
  geom_bar(mapping = aes(x = marital))
```



Recomendación

Habitualmente será más recomendable elegir como categoría de referencia para variables categóricas aquella categoría con mayor número de observaciones.

Por tanto, en este caso particular deberíamos modificar la categoría de referencia como sigue:

```
reorder_marital = factor(bank.train$marital, levels=(c(' married', ' single', ' divorced')
levels(reorder_marital)
```

```
[1] " married"  " single"   " divorced"
```

Nótese que la nueva variable aquí creada, `reorder_marital`, no ha sido incluida (aún) en el tibble `bank`. Para ello:

```
bank.train$marital = reorder_marital
```

3.7.2.1 Conversión de variables cuantitativas a variables categóricas

La conversión de variables cuantitativas a variables categóricas es un proceso importante en EDA que implica transformar datos numéricos en categorías. Esto se realiza con el propósito de simplificar el análisis, resaltar patrones específicos y facilitar la interpretación de los resultados. A continuación, se destacan algunas situaciones comunes en las que se realiza esta conversión y cómo se lleva a cabo:

1. **Agrupación de datos numéricos:** En ocasiones, es útil agrupar datos numéricos en intervalos o categorías para resaltar tendencias generales. Por ejemplo, en un estudio de edades de una población, en lugar de analizar cada edad individual, se pueden crear grupos como "menos de 18 años", "18-30 años", "31-45 años" y así sucesivamente.
2. **Creación de variables binarias:** A menudo, se convierten variables numéricas en variables binarias (1 o 0) para simplificar el análisis. Por ejemplo, en un estudio de satisfacción del cliente, se puede crear una variable binaria donde "1" indica clientes satisfechos y "0" indica clientes insatisfechos.
3. **Categorización de variables continuas:** Las variables continuas, como ingresos o puntuaciones, se pueden convertir en categorías para segmentar la población. Esto puede ser útil en análisis demográficos o de segmentación de mercado.
4. **Simplificación de modelos:** Algunos modelos de ML pueden beneficiarse de la conversión de variables cuantitativas a categóricas para mejorar la interpretación y la eficacia del modelo.

> **!** Para recordar
>
> El proceso de conversión de variables cuantitativas a categóricas generalmente implica definir criterios o reglas claras para agrupar los valores numéricos en categorías significativas. Estos criterios pueden basarse en **conocimiento previo del dominio**, EDA o consideraciones específicas del problema. En esta etapa te vendrá genial contar con

la ayuda de un experto en el dominio de aplicación, y puedes llevar a cabo cambios catastróficos en caso de no contar con esa ayuda.

Es importante tener en cuenta que la conversión de variables cuantitativas a categóricas debe realizarse de manera cuidadosa y considerar el impacto en el análisis. La elección de cómo categorizar los datos debe estar respaldada por una **comprensión sólida del problema** y los objetivos del estudio. Además, se debe documentar claramente el proceso de conversión para que otros puedan replicarlo y comprender las categorías resultantes.

A modo de ejemplo, vamos a categorizar la varible `age` en la base de datos `bank`. Para ello elegimos (elegimos!!!) las siguientes agrupaciones en la variable edad: (0,40],(40,60],(60,100].

```
 bank.train <- within(bank.train, {
  age.cat <- NA # need to initialize variable
  age.cat[age <= 40] <- "Low"
  age.cat[age > 40 & age <= 60] <- "Middle"
  age.cat[age > 60] <- "High"
   } )

bank.train$age.cat <- factor(bank.train$age.cat, levels = c("Low", "Middle", "High"))
summary(bank.train$age.cat)
```

```
   Low Middle   High
  3116   2151    314
```

3.8 Valores comunes y atípicos

Los gráficos de barras relacionados con variables **cualitativas** nos han ayudado a identificar los valores más frecuentes o las categorías más repetidas en esas variables. Estos gráficos reflejan la frecuencia de cada categoría, es decir, el número de veces que aparece en el conjunto de datos. A veces ese número se representa en porcentaje respecto al número total de observaciones, proporcionando una visión relativa de la prevalencia en cada categoría. La moda es la categoría que aparece con mayor frecuencia en el conjunto de datos. Es especialmente útil para identificar la categoría más común y es aplicable a variables categóricas.

En el caso de las variables **cuantitativas**, el histograma de frecuencias se convierte en una herramienta gráfica sumamente útil para alcanzar este mismo objetivo.

3.8.1 Estadísticos resumen

En asignaturas o cursos anteriores de estadística te habrán explicado medidas que resumen el comportamiento de una variable aleatoria cuantitativa (¿verdad?). Recordemos algunas de ellas:

Media

La media aritmética es el promedio de todos los valores de la variable. Se calcula sumando todos los valores y dividiendo por el número de observaciones. La media proporciona una indicación de la tendencia central de los datos.

Mediana

La mediana es el valor central en un conjunto de datos ordenados en forma ascendente o descendente. Divide el conjunto de datos en dos mitades iguales. La mediana es menos sensible a valores extremos que la media y es especialmente útil cuando los datos no siguen una distribución (aproximadamente) normal.

Moda

La moda es el valor que ocurre con mayor frecuencia en un conjunto de datos. Puede haber una o más modas en un conjunto de datos, y esta medida es especialmente útil para variables discretas.

Rango

El rango es la diferencia entre el valor máximo y el valor mínimo en un conjunto de datos. Proporciona una indicación de la dispersión o variabilidad de los datos.

Desviación Estándar

La desviación estándar mide la dispersión de los datos con respecto a la media, y tiene sus mismas unidades de medida. Valores más altos indican mayor variabilidad. Es especialmente útil cuando se asume una distribución normal.

Cuartiles y Percentiles

Los cuartiles dividen un conjunto de datos en cuatro partes iguales, mientras que los percentiles dividen los datos en cien partes iguales. Los cuartiles y percentiles son útiles para identificar valores atípicos y comprender la distribución de los datos.

Coeficiente de Variación

El coeficiente de variación es una medida de la variabilidad relativa de los datos y se calcula como la desviación estándar dividida por la media. Se expresa como un porcentaje y es útil para comparar la variabilidad entre diferentes conjuntos de datos.

En R, podemos obtener algunos estadísticos resumen mediante la opción `summary`

```
summary(bank.train$age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  18.00   32.00   39.00   41.24   49.00   95.00 
```

Curiosamente, R no tiene una función estándar incorporada para calcular la moda. Así que creamos una función de usuario para calcular la moda de un conjunto de datos en R. Esta función toma el vector como entrada y da el valor de la moda como salida.

```
# Create the function.
summary_moda <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

summary_moda(bank.train$age)
```

```
[1] 31
```

3.8.2 Valores atípicos

Los valores atípicos (**outliers** en inglés) son observaciones inusuales, puntos de datos que no parecen encajar en el patrón o el rango de la variable estudiada. A veces, los valores atípicos son errores de introducción de datos; otras veces, sugieren nuevos datos científicos importantes.

Cuando es posible, es una buena práctica llevar a cabo el análisis con y sin los valores atípicos. Si se determina que su influencia en los resultados es insignificante y no se puede identificar su origen, puede ser razonable reemplazarlos con valores faltantes y continuar con el análisis. Sin embargo, si estos valores atípicos tienen un impacto sustancial en los resultados, no se deben eliminar sin una justificación adecuada. En este caso, será necesario investigar la causa subyacente (por ejemplo, un error en la entrada de datos) y documentar su exclusión en el informe correspondiente.

3.9 Valores faltantes

Los valores faltantes (*missing*), también conocidos como valores nulos o valores ausentes, son observaciones o datos que no están disponibles o que no han sido registrados para una o más variables en un conjunto de datos. Estos valores pueden surgir por diversas razones, como errores de entrada de datos, respuestas incompletas en una encuesta, fallos en la medición o simplemente porque cierta información no está disponible en un momento dado.

> **!** Para recordar
>
> La presencia de valores faltantes en un conjunto de datos es un problema común en el análisis de datos y puede tener un impacto significativo en la **calidad** de los resultados. Es importante abordar adecuadamente los valores faltantes, ya que pueden **sesgar los análisis** y conducir a conclusiones incorrectas si no se manejan correctamente.

Algunas de las estrategias comunes para tratar los valores faltantes incluyen:

1. **Eliminación de filas o columnas:** Si la cantidad de valores faltantes es pequeña en comparación con el tamaño total del conjunto de datos, una opción es eliminar las filas o columnas que contengan valores faltantes. Sin embargo, esta estrategia puede llevar a la pérdida de información importante.
2. **Imputación de valores:** Esta estrategia implica estimar o llenar los valores faltantes con valores calculados a partir de otros datos disponibles. Esto puede hacerse utilizando técnicas como la imputación media (rellenar con la media de la variable), imputación mediana (rellenar con la mediana), imputación de vecinos más cercanos o técnicas más avanzadas como regresión u otras técnicas de modelado.
3. **Marcadores especiales:** En algunos casos, es útil asignar un valor específico (como "N/A" o "-999") para indicar que un valor está ausente. Esto puede ser útil cuando se desea mantener un registro explícito de los valores faltantes sin eliminarlos o imputarlos. Es importante que, en este caso, el valor asignado no tenga otro significado. Por ejemplo, asignamos "-999" como marcador de valor faltante y sin embargo, es un valor plausible dentro del rango de valores de la variable.
4. **Métodos basados en modelos:** Utilizar modelos estadísticos o de ML para predecir los valores faltantes en función de otras variables disponibles. Esto puede ser especialmente eficaz cuando los datos faltantes siguen un patrón que puede ser capturado por el modelo.

La elección de la estrategia adecuada para tratar los valores faltantes depende del contexto del análisis, la cantidad de datos faltantes y la naturaleza de los datos. Es fundamental abordar este problema de manera cuidadosa y transparente, documentando cualquier procedimiento de imputación o tratamiento de valores faltantes utilizado en el análisis para garantizar la integridad y la validez de los resultados.

> **Peligro**
>
> Sustituir valores faltantes por otros obtenidos con técnicas y métodos estadísticos o de ML siempre es un riesgo, pues implica "inventar" datos allá donde no los hay.

3.10 Correlación entre variables

Existen varios métodos y técnicas para estudiar la correlación entre variables, lo que ayuda a comprender las relaciones entre las diferentes características en un conjunto de datos. En ML, especial interés van a tener las relaciones entre la variable objetivo y las variables explicativas.

Puedes desplegar los paneles siguientes para averiguar alguno de los métodos más comunes.

> **Matriz de correlación**
>
> La matriz de correlación es una tabla que muestra las correlaciones entre todas las combinaciones de variables en un conjunto de datos. Los valores de correlación varían entre -1 y 1, donde -1 indica una correlación negativa perfecta, 1 indica una correlación positiva perfecta y 0 indica la ausencia de correlación. Este método es especialmente útil para identificar relaciones lineales entre variables numéricas.

> **Gráficos de dispersión**
>
> Los gráficos de dispersión muestran la relación entre dos variables numéricas mediante puntos en un plano cartesiano. Estos gráficos permiten visualizar patrones de dispersión y tendencias entre las variables. Si los puntos se agrupan en una forma lineal, indica una posible correlación lineal.

> **Mapas de calor**
>
> Los mapas de calor son representaciones visuales de la matriz de correlación en forma de un gráfico de colores. Permiten identificar rápidamente las relaciones fuertes o débiles entre variables y son útiles para resaltar patrones en grandes conjuntos de datos.

> **Coeficiente de Correlación de Pearson**
>
> Este coeficiente mide la correlación lineal entre dos variables numéricas. Varía entre -1 y $+1$, donde valores cercanos a -1 o $+1$ indican una correlación fuerte, mientras que valores cercanos a 0 indican una correlación débil o nula.

Coeficiente de Correlación de Spearman

Este coeficiente evalúa la correlación monotónica entre dos variables, lo que significa que puede detectar relaciones no lineales. Es útil cuando las variables no siguen una distribución normal.

Coeficiente de Correlación de Kendall

Similar al coeficiente de Spearman, evalúa la correlación entre variables, pero se centra en la concordancia de los rangos de datos, lo que lo hace útil para datos no paramétricos y muestras pequeñas.

Coeficiente de Correlación de Kendall

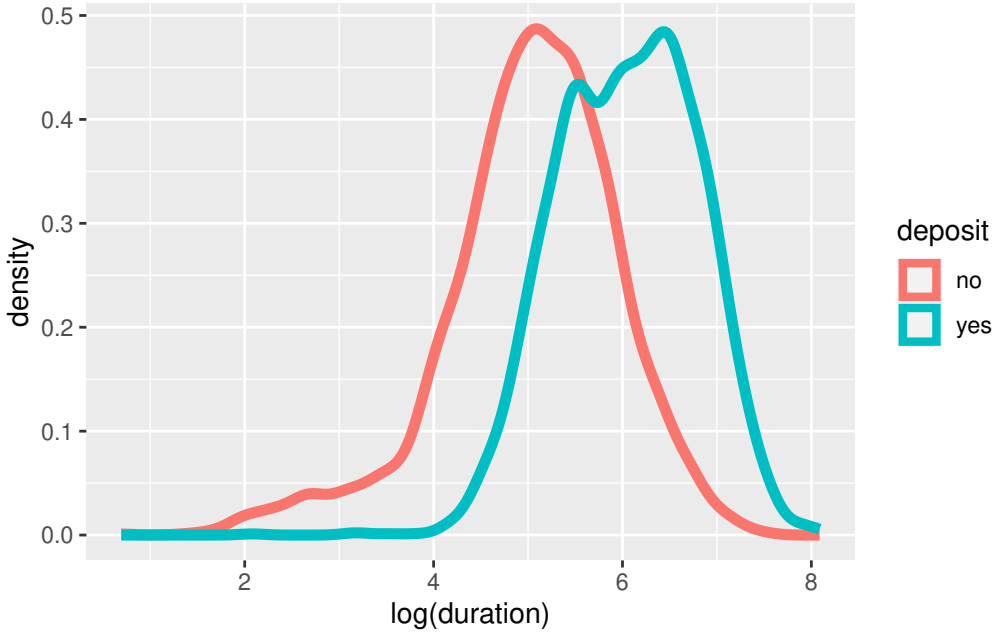
Las pruebas estadísticas, como la prueba t de Student o la ANOVA, pueden utilizarse para evaluar si existe una diferencia significativa en los promedios de una variable entre diferentes categorías de otra variable. Si la diferencia es significativa, puede indicar una correlación entre las variables.

p_valor

¿Recuerdas lo que es un contraste de hipótesis? ¿Recuerdas lo que es un intervalo de confianza? ¿y un p_valor? ¡Posiblemente te venga bien dar un repaso a esos conceptos!

Vamos a estudiar la relación existente entre la variable objetivo `deposit' y la variable`duration`de la base de datos`bank‘.

```
ggplot(bank.train, aes(x = log(duration), colour = deposit)) +
  geom_density(lwd=2, linetype=1)
```



Puede observarse una relación. Valores altos de la variable **duración** parecen estar relacionados con observaciones con **deposit** igual a 'yes'.

```
df = bank.train %>%
      select(duration,deposit)%>%
      mutate(log.duration=log(duration))

# Resumen para los casos de depósito
summary(df %>% filter(deposit=="yes") %>% .$log.duration)
```

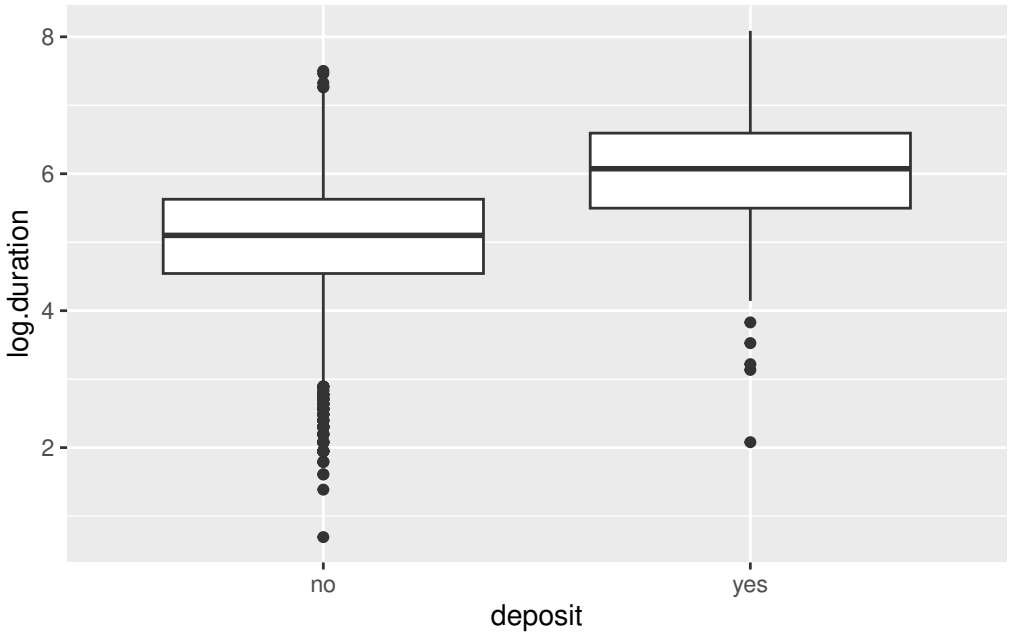
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.079   5.497   6.073   6.046   6.593   8.087
```

```
# Resumen para los casos de no depósito
summary(df %>% filter(deposit=="no") %>% .$log.duration)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.6931  4.5433  5.0999  5.0308  5.6276  7.5022
```

Gráficamente, podemos comparar los boxplots.

```r
ggplot(df, aes(deposit, log.duration)) +
        geom_boxplot()
```



Podemos constrastar dicha hipótesis. Por ejemplo, podemos realizar un test de la T para igualdad de medias.

```r
t.test(log.duration ~ deposit, data = df)
```

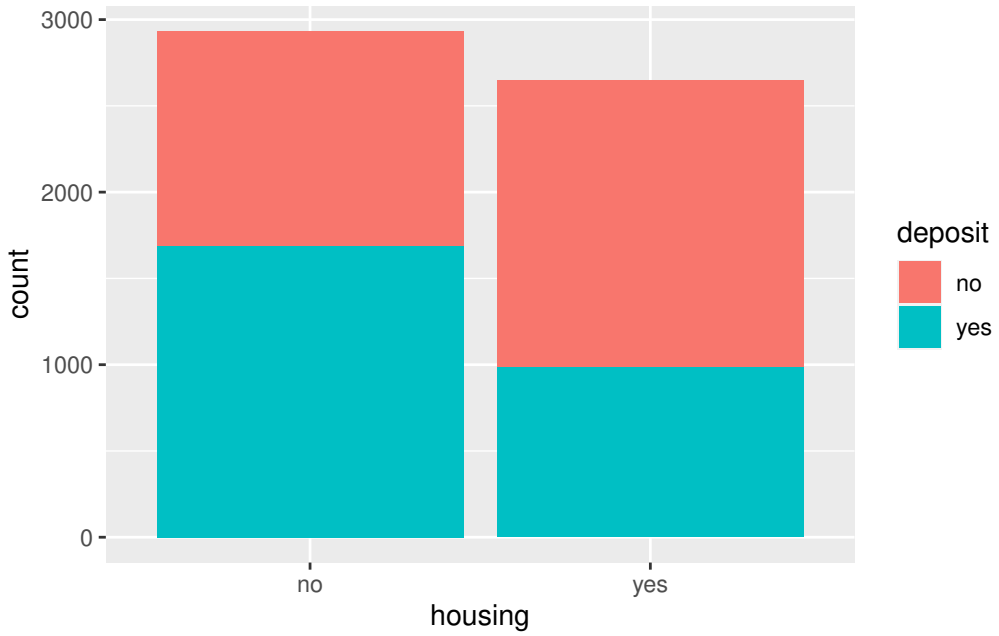
```
    Welch Two Sample t-test

data:  log.duration by deposit
t = -45.828, df = 5464.5, p-value < 2.2e-16
alternative hypothesis: true difference in means between group no and group yes is not equal
95 percent confidence interval:
 -1.0583835 -0.9715488
sample estimates:
 mean in group no mean in group yes
         5.030821          6.045787
```

Ejercicio

Dejamos como ejercicio para el alumno la interpretación del resultado del test.

Es posible estudiar la relación entre dos variables categóricas de manera gráfica.

```
ggplot(data = bank.train, aes(x = housing, fill = deposit)) +
    geom_bar()
```



Parece haber una relación, estando asociados las observaciones de personas con casa propia a un mayor porcentaje de 'no' en la variable respuesta. Podemos obtener la tabla de contingencia:

```
data1=table(bank.train$housing, bank.train$deposit)


dimnames(data1) <- list(housing = c("no", "yes"),
                        deposit = c("no", "yes"))
data1
```

```
       deposit
housing   no  yes
```

```
    no  1246 1688
    yes 1662  985
```

Y el contraste correspondiente para la hipótesis nula de no existencia de relación.

```
chisq.test(bank.train$housing, bank.train$deposit)
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  bank.train$housing and bank.train$deposit
X-squared = 229.44, df = 1, p-value < 2.2e-16
```

> Ejercicio
>
> Dejamos como ejercicio para el alumno la interpretación del resultado del test.

4 Técnicas de reducción de la dimensionalidad

En el complejo mundo de la ciencia de datos, nos encontramos a menudo con conjuntos de datos que abarcan una amplia variedad de variables. Estas variables, en su mayoría cuantitativas, pueden proporcionar una gran cantidad de información, pero también pueden convertirse en un desafío cuando se trata de comprender y analizar de manera efectiva los datos. La gestión de grandes conjuntos de variables puede ser abrumadora y puede llevar a problemas como la alta correlación entre ellas, dificultando la extracción de información valiosa y la visualización clara de los datos. Imagina un conjunto de datos con más de 50 variables. Como científico de datos, has de analizar todas y cada una de ellas, así como sus posibles relaciones.

Es en este contexto que entran en juego las "*Técnicas de Reducción de la Dimensión*". Estas técnicas son herramientas que nos permiten encontrar un equilibrio entre la riqueza de información y la simplicidad en el análisis. El objetivo principal de la reducción de la dimensión es identificar un conjunto más pequeño de variables, llamadas "*variables latentes*" o "*componentes principales*", que capturen la mayor parte de la información esencial contenida en las variables originales. Este proceso se realiza con el menor costo de información posible, lo que facilita la gestión y el análisis de datos complejos.

Las ventajas de la reducción de la dimensión son múltiples. En primer lugar, permite reducir la cantidad de información utilizada, lo que puede ser especialmente útil cuando se trabaja con grandes conjuntos de datos. Además, la eliminación de problemas de alta correlación entre variables ayuda a eliminar la redundancia en los datos y a prevenir posibles distorsiones en los resultados del análisis. Pero quizás una de las ventajas más notables es la posibilidad de visualizar los datos de manera sencilla, a menudo en solo dos dimensiones, lo que facilita la interpretación y la comunicación de resultados.

Sin embargo, es importante mencionar una desventaja potencial de estas técnicas: la falta de **explicabilidad** de las nuevas variables. Dado que estas se construyen como combinaciones de las originales, a menudo requieren la experiencia de un experto en el dominio para su correcta interpretación. Esto significa que, si bien la reducción de la dimensión simplifica el análisis, también puede introducir cierta complejidad en la comprensión de las relaciones subyacentes entre las variables.

Despliega los paneles siguientes para descrubir algunas de las técnicas más conocidas.

Análisis de Componentes Principales (PCA)

PCA es una de las técnicas más populares para la reducción de la dimensión. Transforma las variables originales en un nuevo conjunto de variables no correlacionadas llamadas componentes principales. Estos componentes capturan la mayor parte de la variabilidad en los datos y se pueden utilizar para representar los datos en un espacio de menor dimensión.

Análisis de Factor

Similar a PCA, el análisis de factor busca identificar variables latentes o factores subyacentes que expliquen las relaciones entre las variables originales. Es útil cuando se sospecha que las variables observadas están influenciadas por factores no observados.

Reducción de la Dimensión Basada en Selección

Esta técnica implica seleccionar un subconjunto de variables originales en función de algún criterio, como su importancia para el problema o su capacidad de explicar la variabilidad. Algunos métodos de selección incluyen la selección de características y la eliminación de características redundantes.

Análisis Discriminante Lineal (LDA)

LDA es una técnica que se utiliza en problemas de clasificación. Busca encontrar una combinación lineal de variables que maximice la separación entre clases en el conjunto de datos, lo que puede reducir la dimensión al tiempo que preserva la información relevante para la clasificación.

Descomposición en Valores Singulares (SVD)

SVD es una técnica matricial que se utiliza en la factorización de matrices y la reducción de la dimensión. Es fundamental en métodos como PCA y puede utilizarse para reducir la dimensión de matrices de datos.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE es una técnica de reducción de la dimensión no lineal que se utiliza comúnmente para visualización de datos. Tiene la capacidad de preservar relaciones locales entre puntos en un espacio de menor dimensión.

Autoencoders

Los autoencoders son redes neuronales que se utilizan para aprender representaciones de datos de alta dimensión en un espacio de menor dimensión. Son especialmente útiles en problemas de reducción de la dimensión no lineal.

Reducción de la Dimensión Basada en la ingeniería de características

A veces, la reducción de la dimensión se puede lograr mediante la ingeniería de características (**Feature Engineering**), donde se crean nuevas variables que resumen la información de las originales de manera más efectiva.

En este tema, exploraremos en detalle las técnicas más utilizadas para la reducción de la dimensión, proporcionando ejemplos prácticos y pautas para su aplicación efectiva. Estas técnicas son esenciales para cualquier científico de datos que desee extraer conocimiento valioso de grandes conjuntos de datos y simplificar el proceso de análisis sin perder de vista la interpretación de los resultados.

Por tanto, podemos decir que los métodos de reducción de dimensionalidad son técnicas que se utilizan para reducir la cantidad de variables en un conjunto de datos mientras se intenta retener la información relevante. Sin embargo, existen enfoques adicionales para realizar la reducción de la dimensionalidad, que se clasifican comúnmente en tres categorías: *filter*, *wrapper* y *embedded* (John, Kohavi, and Pfleger 1994).

Métodos Filter

Los métodos filter son técnicas de selección de características que se aplican antes de entrenar un modelo de ML. Estos métodos evalúan la relación entre cada variable y la variable objetivo (o alguna medida de relevancia) sin tener en cuenta un modelo específico. Los métodos *filter* utilizan estadísticas, métricas de rendimiento, pruebas de hipótesis u otras técnicas para clasificar las características en función de su relevancia. Algunos ejemplos de métodos *filter* incluyen la correlación de Pearson, la información mutua y la prueba estadística chi-cuadrado. La principal ventaja de los métodos *filter* es su eficiencia computacional, ya que no requieren entrenar modelos.

Métodos Wrapper

Los métodos *wrapper* también son técnicas de selección de características, pero al contrario que los métodos *filter*, utilizan modelos de ML as características. Se evalúan múltiples modelos mediante procedimientos que añaden y/o eliminan variables predictoras para encontrar la combinación óptima que maximice el rendimiento del modelo. Los métodos *wrapper* pueden ser más precisos que los métodos *filter*, ya que tienen en

cuenta la interacción entre las características, pero tienden a ser más computacionalmente costosos, ya que involucran el entrenamiento repetido de modelos.

i Métodos Embedded:

Los métodos *embedded* incorporan la selección de características directamente en el proceso de entrenamiento de un modelo de ML. En lugar de realizar la selección de características como un paso separado, estos métodos evalúan la relevancia de las características mientras se ajustan al modelo. Esto significa que las características se seleccionan o ponderan automáticamente durante el entrenamiento del modelo. Ejemplos de métodos *embedded* incluyen la regresión L1 (Lasso), que impone penalizaciones a los coeficientes de las características menos importantes, y los métodos de árboles de decisión, que pueden evaluar la importancia de las características durante la construcción del árbol.

La elección entre métodos *filter*, *wrapper* y *embedded* depende de la naturaleza del problema, el conjunto de datos y las necesidades específicas del análisis. Cada enfoque tiene sus propias ventajas y desventajas, y es importante considerar factores como la eficiencia computacional, la calidad del modelo y la interpretabilidad al seleccionar el enfoque adecuado para la reducción de la dimensionalidad en un proyecto de ciencia de datos o ML.

4.1 Análisis de Componentes Principales (PCA)

Tal y como se ha indicado, el Análisis de Componentes Principales (PCA, en inglés, *Principal Component Analysis*) es una técnica de reducción de la dimensionalidad cuyo principal objetivo es simplificar la estructura de datos, preservando al mismo tiempo la mayor cantidad posible de información relevante. PCA logra esto transformando un conjunto de variables correlacionadas en un conjunto nuevo de variables no correlacionadas llamadas componentes principales.

Los pasos para aplicar esta técnica sobre un conjunto de datos son:

1. **Cálculo de la matriz de covarianza:** El primer paso en PCA implica calcular la matriz de covarianza de las variables originales. La covarianza es una medida de cómo dos variables cambian juntas. Una matriz de covarianza muestra cómo todas las variables del conjunto de datos se relacionan entre sí.
2. **Obtención de los componentes principales:** A continuación, se calculan los autovectores y autovalores de la matriz de covarianza. Los autovectores son las direcciones en las cuales los datos tienen la mayor varianza, y los autovalores representan la cantidad de varianza explicada por cada autovector. Probablemente hayas estudiado técnicas para el cálculo de autovectores y autovalores en cursos de Álgebra anteriores.

3. **Selección de componentes principales:** Después de calcular los autovectores y autovalores, se ordenan en orden descendente según la cantidad de varianza que explican. Esto implica que el primer componente principal explica la mayor varianza en los datos, el segundo componente principal explica la segunda mayor varianza (una vez eliminada la variabilidad explicada por el primer componente principal), y así sucesivamente. Por lo general, se selecciona un número pequeño (¡reducción de la dimensión!) de componentes principales que capturen una cantidad significativa de la varianza total.
4. **Transformación de datos:** Finalmente, los datos originales se transforman en el espacio de los componentes principales. Esto significa que las variables originales se combinan linealmente para formar nuevas variables (los componentes principales) que son ortogonales entre sí. Es decir, se crean nuevas variables, como combinación lineal de las originales, con la particularidad de que esas nuevas variables son incorreladas y de varianza 1. Por lo tanto, estos componentes principales no tienen multicolinealidad, lo que es útil en análisis posteriores.

El PCA se utiliza en diversas aplicaciones, como reducción de dimensionalidad, compresión de imágenes, análisis de datos, reconocimiento de patrones, etc. Permite simplificar datos complejos mientras se retiene la mayor cantidad posible de información importante. Al seleccionar un número apropiado de componentes principales, es posible reducir la dimensionalidad de los datos sin perder significado, lo que puede mejorar la eficiencia del análisis y la visualización.

4.1.1 PCA en R

Trabajemos con un ejemplo de datos en R. Este conjunto de datos contiene estadísticas, en arrestos por cada 100.000 residentes por asalto (Assault), asesinato (Murder) y secuestro (Rape) en los 50 Estados de USA en 1973. También se proporciona el porcentaje de la población en áreas urbanas (UrbanPop).

```
head(USArrests)
```

```
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

```
summary(USArrests)
```

```
     Murder          Assault         UrbanPop          Rape
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

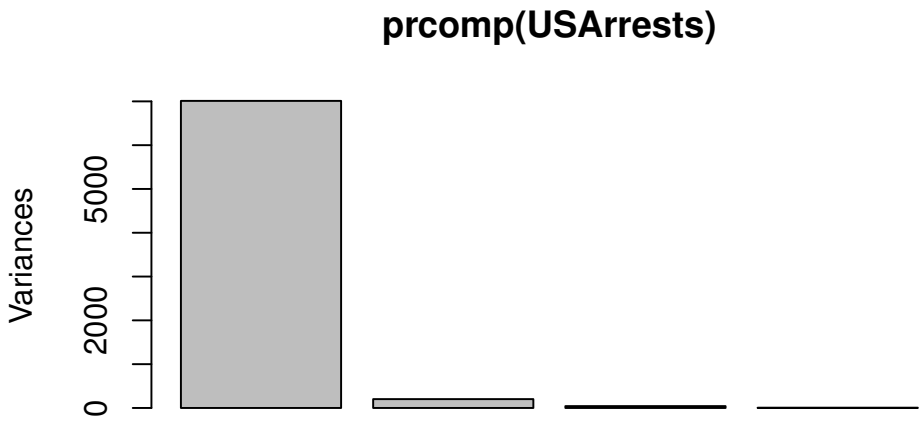
Para realizar PCA sobre los datos USArrests:

```
prcomp(USArrests)
```

```
Standard deviations (1, .., p=4):
[1] 83.732400 14.212402  6.489426  2.482790

Rotation (n x k) = (4 x 4):
                PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502
```

Las *standard deviations* son los autovalores de la matriz de correlaciones, y representan la variabilidad en cada componente. A mayor valor, más relevante es la variable correspondiente a efectos de visualización. Si queremos visualizar la importancia relativa de cada componente, haremos lo siguiente:

```
plot(prcomp(USArrests))
```



De modo numérico:

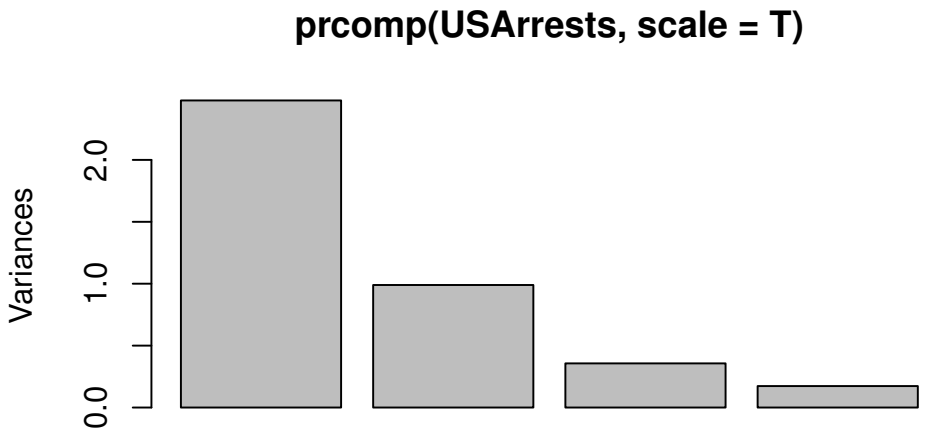
```
summary(prcomp(USArrests))
```

```
Importance of components:
                          PC1      PC2    PC3     PC4
Standard deviation     83.7324 14.21240 6.4894 2.48279
Proportion of Variance  0.9655  0.02782 0.0058 0.00085
Cumulative Proportion   0.9655  0.99335 0.9991 1.00000
```

Como vemos, la variabilidad de los datos se explica mayoritariamente por la primera componente principal PC1 que, tal y como se puede ver en la matriz *Rotation*, da un peso de 0.9952 a la variable `Assault`, y pesos cercanos a cero al resto de variables. En la tabla anterior se observa que la proporción de varianza explicada por la primera componente es del 96.6%, es decir, es prácticamente la única relevante. Queda muy poca variabilidad para ser explicada por el resto de variable. ¿Qué está sucediendo? Si nos fijamos en los datos de USArrests, la magnitud de los valores de `Assault` es mucho mayor que la magnitud de las otras variables. Por ejemplo, en el caso de Alabama, 236 para la variable `Assault` frente a 13.2 de `Murder` ó 21.2 de `Rape`. Claramente es la variable que va a tener más influencia en el resultado final, tal y como se ve en el gráfico precedente. La segunda componente más influyente es PC2, que depende de `UrbanPop`, la siguiente variable en magnitud, y así sucesivamente.

¿Tiene sentido que una variable sea más influyente que otra por el mero hecho de estar medida en diferentes unidades de medida? Realmente no.¿Cómo solucionar este problema? La respuesta es la **estandarización**. Repitamos el análisis, pero estandarizando los datos:

```
plot(prcomp(USArrests,scale=T))
```



```
summary(prcomp(USArrests,scale=T))
```

```
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

Como puede verse, con las dos primeras componentes recogemos prácticamente el 87% de la variabilidad. Esto quiere decir que un gráfico de los datos de `USAarrests`, representados por las dos primeras componentes principales será suficientemente representativo.

Antes de ir al gráfico, analicemos la matriz de rotaciones, en busca de interpretación semántica para las componentes principales:

```
prcomp(USArrests,scale=T)
```

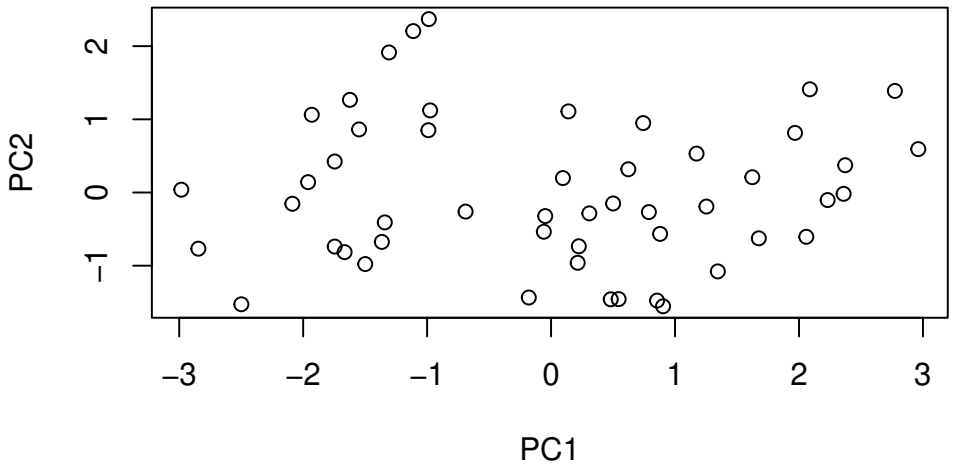
```
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

PC1 asigna pesos, todos del mismo signo a las variables. Podemos comprobar que viene a representar un promedio ponderado de las variables originales. Es decir, es una medida resumen que permite ordenar, en dicha componente, el comportamiento de los 50 estados desde el punto de vista de los delitos cometidos. Al concentrarnos en la segunda componente principal, los estados serían ordenados en cuanto a su población urbana en un sentido, ponderando en sentido contrario el número de asesinatos. En PC3, el orden viene principalmente dado por los secuestros.

En cualquier caso, debemos tener presente que no hay ninguna garantía de interpretabilidad en un análisis PCA.

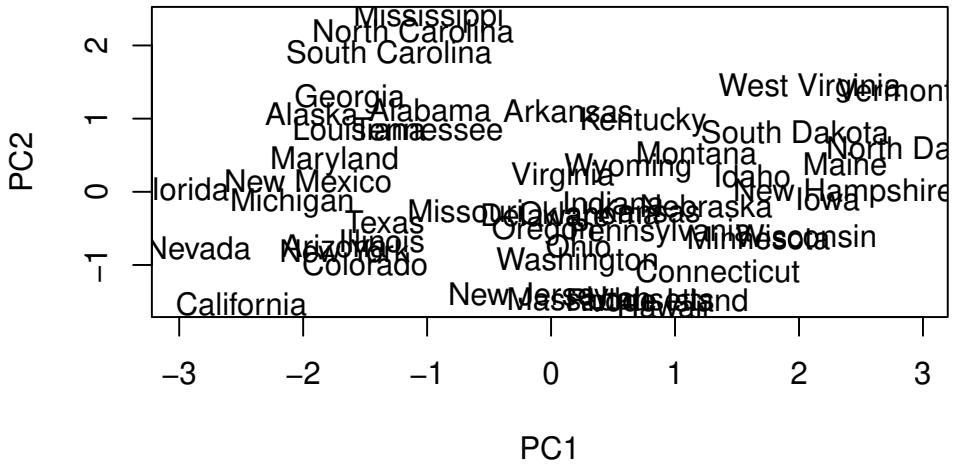
Dibujamos los datos proyectados sobre las dos primeras componentes:

```
plot(prcomp(USArrests,scale=T)$x[,1:2])
```



Sería deseable poder ver los nombres de los estados, en lugar de simples puntos. Para ello:

```
plot(prcomp(USArrests,scale=T)$x[,1:2],type="n")
text(prcomp(USArrests,scale=T)$x[,1:2],rownames(USArrests))
```

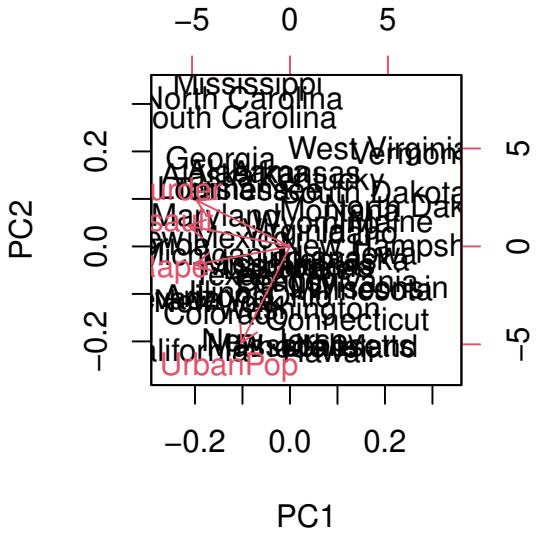


Puntos cercanos en el mapa indican comportamiento/perfil similar en cuanto a los delitos cometidos. Del gráfico podemos inferir que Florida, Nevada y California son tres puntos extremos en cuanto al comportamiento criminal. Pero, ¿en qué sentido? ¿muy baja o muy alta criminalidad? En este caso particular, la respuesta es muy alta, dado que en la matriz de rotación todos los pesos de la primera componente principal son negativos, luego a mayor valor

en las variables originales, situación más a la izquierda en PC1. En cuanto a PC2, tenemos comportamientos extremos por ejemplo en Hawai por abajo y en Mississippi y North Carolina por arriba. En el primer caso hay mucha población urbana y pocos asesinatos. En el segundo, al contrario.

¿Cómo mejorar el gráfico? Un modo posible es incorporar la información de las variables utilizando la técnica del *biplot*.

```
biplot(prcomp(USArrests,scale=T))
```



Como vemos, la primera componente viene a situarse donde se situaría el promedio de las cuatro variables. Los estados casi se ordenan en la segunda componente por `UrbanPop` en un sentido, y por `Murder` en el otro, como ya hemos deducido/comprobado.

A continuación presentamos el mismo análisis para las variables continuas de la base de datos `bank`:

```
library (dplyr)

bank = read.csv('https://raw.githubusercontent.com/rafiag/DTI2020/main/data/bank.csv')
df= bank %>%
      filter(balance>0 & previous>0 & pdays>0) %>%
      mutate( log.balance=log(balance),
              log.age=log(age),
              log.campaign=log(campaign),
              log.previous=log(previous),
```

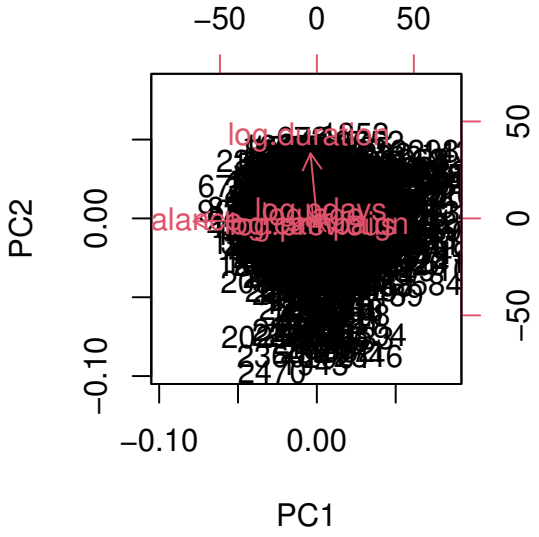
```
                log.pdays = log(pdays),
                log.duration=log(duration)) %>%
        select(log.balance,log.age,log.campaign,log.duration,log.previous,log.pdays)

  # No escalado
  prcomp(df)
```

```
Standard deviations (1, .., p=6):
[1] 1.5630485 0.8334400 0.7775226 0.7240114 0.5268730 0.2958263

Rotation (n x k) = (6 x 6):
                      PC1         PC2         PC3          PC4          PC5
log.balance  -0.997633646 -0.04632142  0.04352014 -0.006913800 -0.002158896
log.age      -0.026975335  0.01790770 -0.02192873 -0.005198189 -0.006857026
log.campaign -0.001512975 -0.09417225 -0.12317132 -0.240791927  0.958101859
log.duration -0.051244928  0.98604272 -0.13674993 -0.017279203  0.075011078
log.previous -0.025152050 -0.10278316 -0.79628880 -0.540931843 -0.248366571
log.pdays     0.027175018  0.07629283  0.57417555 -0.805629317 -0.121146742
                      PC6
log.balance   0.025198601
log.age      -0.999198047
log.campaign -0.004266074
log.duration  0.021631678
log.previous  0.020831104
log.pdays    -0.006944817
```

```
  summary(prcomp(df))
```

```
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6
Standard deviation     1.5630 0.8334 0.7775 0.7240 0.52687 0.29583
Proportion of Variance 0.5275 0.1500 0.1305 0.1132 0.05994 0.01889
Cumulative Proportion  0.5275 0.6775 0.8080 0.9212 0.98111 1.00000
```

```
  biplot(prcomp(df))
```



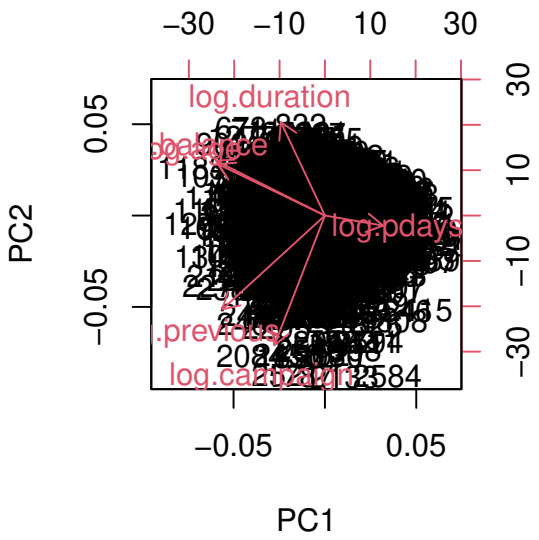

```
# Escalamos los datos:
prcomp(df, scale=T)
```

```
Standard deviations (1, .., p=6):
[1] 1.1051338 1.0832463 1.0118040 0.9678377 0.9316200 0.8814102

Rotation (n x k) = (6 x 6):
                     PC1        PC2        PC3         PC4          PC5
log.balance  -0.5333580  0.2738588 -0.1232941  0.32734897 -0.711380533
log.age      -0.5238196  0.2570122 -0.1980986  0.37723756  0.683622355
log.campaign -0.2475502 -0.6476384 -0.3372288 -0.03596251 -0.118453560
log.duration -0.2156187  0.4664755 -0.2567977 -0.79051329 -0.010808413
log.previous -0.5022597 -0.4683898  0.1226574 -0.34628989  0.111134086
log.pdays     0.2847895 -0.0502219 -0.8665116  0.06643204  0.009967902
                     PC6
log.balance  -0.1100436
log.age       0.1032814
log.campaign  0.6246884
log.duration  0.2119813
log.previous -0.6172794
log.pdays    -0.4012702
```

```
summary(prcomp(df, scale=T))
```

```
Importance of components:
                          PC1    PC2    PC3    PC4    PC5    PC6
Standard deviation     1.1051 1.0832 1.0118 0.9678 0.9316 0.8814
Proportion of Variance 0.2036 0.1956 0.1706 0.1561 0.1447 0.1295
Cumulative Proportion  0.2036 0.3991 0.5697 0.7259 0.8705 1.0000
```

```r
biplot(prcomp(df, scale=T))
```



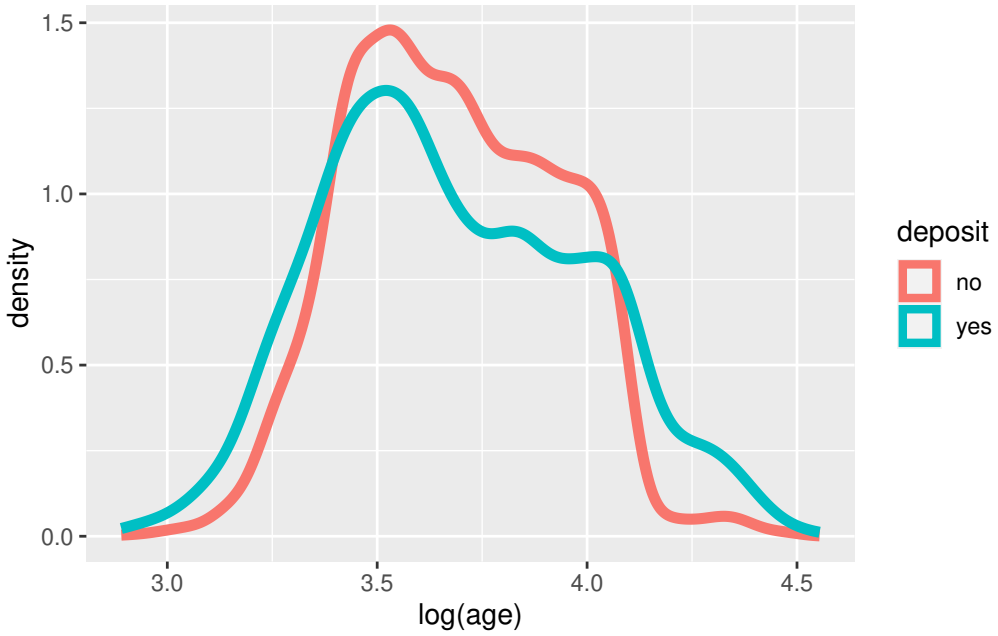
Dejamos la interpretación de los resultados como tarea para el alumno. A modo de pista, fíjate en la matriz de correlaciones en los ejemplos presentados.

> **Pregunta**
>
> ¿Qué ocurre cuando se realiza un PCA con variables independientes?

4.2 Selección de características

En el tema Chapter 3 vimos como es posible evaluar la correlación entre una variable objetivo y las variables explicativas asociadas. A continuación, a modo de ejercicio, vamos a estudiar la relación existente entre la variable objetivo `deposit` y la variable `age` de la base de datos `bank`. Tratamos de responder a la pregunta de si la variable edad del cliente influye, o no, en si el cliente suscribirá (o no) un depósito a plazo.

```
library (ggplot2)
ggplot(bank, aes(x = log(age), colour = deposit)) +
  geom_density(lwd=2, linetype=1)
```



Aprentemente, no se observa una relación significativa entre ambas variables.

```
df = bank %>%
      select(age,deposit)%>%
      mutate(log.age=log(age))

# Resumen para los casos de depósito
summary(df %>% filter(deposit=="yes") %>% .$log.age)
```

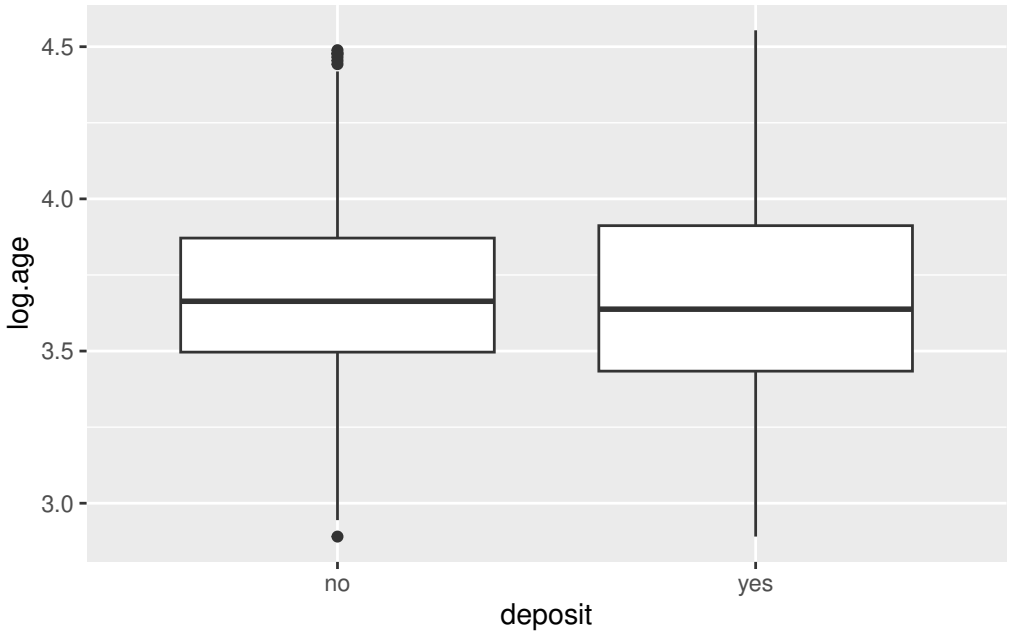
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.890   3.434   3.638   3.681   3.912   4.554
```

```
# Resumen para los casos de no depósito
summary(df %>% filter(deposit=="no") %>% .$log.age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.890   3.497   3.664   3.679   3.871   4.489
```

Efectivamente, los valores de resumen de edad en ambas categorías de la variable respuesta son muy similares. Gráficamente, podemos comparar los boxplots.

```
ggplot(df, aes(deposit, log.age)) +
        geom_boxplot()
```



Podemos constrastar la hipótesis nula de igualdad de medias mediante un test de la T:

```
t.test(log.age ~ deposit, data = df)
```

```
    Welch Two Sample t-test

data:  log.age by deposit
t = -0.26977, df = 10053, p-value = 0.7873
alternative hypothesis: true difference in means between group no and group yes is not equal
95 percent confidence interval:
 -0.011917888  0.009034321
```

```
sample estimates:
 mean in group no mean in group yes
         3.679125          3.680566
```

El $p-valor$ es mayor que 0.10 por lo que no hay evidencia estadística en las observaciones en contra de la hipótesis nula. Podemos concluir, por tanto, que no existe una relación significativa (o al menos aún no la hemos localizado) entre las variables `deposit` y `age`. Podríamos, por tanto, eliminar la varible explicativa de la base de datos. De este modo se reduce la dimensionalidad del problema. Sin embargo, es una práctica peligrosa. La variable `age` podría resultar significativamente relevante cuando se controlen los efectos de otras variables dentro del futuro modelo de ML.

> **!** Para recordar
>
> Eliminar variables explicativas en etapas tempranas del análisis para reducir la dimensionalidad del problema, conlleva el riesgo de pérdida de información que podría ser útil en etapas posteriores.

5 Aprendizaje no supervisado

Los modelos de aprendizaje no supervisado aparecen cuando no se dispone de una etiqueta sobre los datos en contraposición al aprendizaje supervisado donde sí existe dicha etiqueta (ver Chapter 1). Por lo tanto, en aprendizaje no supervisado no hay clases que aprender. En este caso el objetivo de estos modelos no será construir un modelo de clasificación capaz de separar las observaciones según unas clases predeterminadas.

> ❗ Para recordar
>
> El objetivo de los algoritmos de agrupamiento es **particionar** el conjunto de datos en grupos de observaciones, donde cada observación se parezca lo más posible a las otras observaciones de su mismo grupo y lo menos posible a las otras observaciones de los otros grupos.

Estos grupos reciben el nombre de conglomerados o clústeres.

La idea fundamental de los algoritmos de agrupamiento es que los puntos dentro de un mismo clúster sean muy similares (respecto a alguna medida de similitud) y que los puntos en diferentes clústeres sean diferentes.

Las técnicas de agrupamiento se emplean para segmentación del mercado, para visualización, para la detección de anomalías, para la imputación de valores faltantes, para la compresión de datos, etc. Los algoritmos de agrupamiento permiten obtener una visión de la complejidad de la tarea de clasificación.

> 🔥 Atención
>
> En ocasiones se recomienda comenzar el sistema de ML con un algoritmo de agrupamiento, incluso cuando tengamos etiquetas y el problema pudiera plantearse como un problema de aprendizaje supervisado.

Podéis encontrar una amplia revisión de los algoritmos de agrupamiento en (Xu and Tian 2015), siendo los más conocidos:

- los basados en **centroides**, como por ejemplo el algoritmo de las *k-medias*
- los basados en **conectividad**, como por ejemplo el *agrupamiento jerárquico*

- los basados en **densidad**, siendo el más conocido de todos el algoritmo *DBSCAN*.

En ocasiones los algoritmos de agrupamiento se clasifican en agrupamiento jerárquico y agrupamiento no jerárquico.

> **Agrupamiento jerárquico**
>
> El agrupamiento jerárquico tiene la particularidad de que encuentra grupos **anidados** de clústeres. Es decir, cuando una observación forma parte de un clúster, no lo abandona, pudiéndose unir el clúster a otros clústeres en etapas posteriores.

> **Agrupamiento no jerárquico**
>
> Por contra, el agrupamiento no jerárquico genera una clasificación mediante la partición del conjunto de datos, obteniendo un conjunto de clústeres **no superpuestos** que no tienen relaciones jerárquicas entre sí.

El agrupamiento puede ser una herramienta muy útil para el análisis de datos en un entorno no supervisado. Sin embargo, hay una serie de problemas que surgen al realizar el clustering. En este tema del curso vamos a repasar algunas de las propiedades y algunos de los problemas del clustering.

5.1 Parámetros de un modelo de ML

Un parámetro es un valor que el algoritmo del modelo de ML ajusta durante el proceso de entrenamiento para hacer que el modelo se adapte mejor a los datos de entrenamiento y, en última instancia, haga predicciones más precisas en datos no vistos (datos de prueba o datos en producción). Los parámetros son esenciales para definir la estructura y el comportamiento del modelo.

En ocasiones se diferencia entre dos tipos de parámetros en un modelo de ML:

> **Parámetros del modelo**
>
> Estos son los componentes internos del modelo que definen su estructura y su capacidad para representar relaciones en los datos. Por ejemplo, en una red neuronal, los pesos y sesgos en las capas de neuronas son parámetros del modelo. En una regresión lineal, los coeficientes son parámetros del modelo.

Hiperparámetros del modelo

A diferencia de los parámetros del modelo, los **hiperparámetros** son valores que se establecen antes del proceso de entrenamiento y controlan aspectos más generales del modelo. Ejemplos de hiperparámetros incluyen la tasa de aprendizaje, la cantidad de capas ocultas en una red neuronal, el valor k en el modelo de k vecinos, la profundidad de un árbol de decisión, etc. Los hiperparámetros afectan cómo se ajustan los parámetros del modelo durante el entrenamiento.

El proceso de ajuste de parámetros y hiperparámetros se realiza mediante la iteración y la experimentación para encontrar la combinación adecuada que permita al modelo aprender de manera efectiva y generalizar bien a datos nuevos. Esto se conoce como ajuste de hiperparámetros o búsqueda de hiperparámetros y es una parte crítica del desarrollo de modelos exitosos de ML.

5.2 k-medias

El algoritmo de las k medias es el algoritmo de ML no supervisado más utilizado para agrupar un conjunto de observaciones en un conjunto de k grupos o clústeres, donde k representa el número de grupos pre-especificados por el científico de datos. Diremos que k es un valor del modelo de ML y que su valor ha de ser fijado (o aprendido) a lo largo del proceso de aprendizaje.

La idea básica consiste en definir clústeres de manera que se reduzca al máximo la variabilidad total dentro del clúster (llamada *within-cluster variation*):

$$\sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

Existen varios algoritmos para entrenar un modelo de k-medias. El algoritmo original puede encontrarse en (Hartigan and Wong 1979), y define la variabilidad total dentro del clúster como la suma de distancias Euclídeas al cuadrado entre las observaciones y el correspondiente centroide.

1. El algoritmo comienza con k medias seleccionadas aleatoriamente del conjunto original de observaciones.

Atención

No siempre se tiene información sobre qué valor es el óptimo para el parámetro k en el modelo de las k-medias. De hecho, en ocasiones el interés de los métodos de agrupamiento

es precisamente averiguar dicho valor.

2. El algoritmo continúa asignando los registros de la base de datos al clúster con media más cercana. Es decir, para cada observación se busca su centroide más cercano dentro del conjunto de centrioides disponibles.
3. Una vez que todas las observaciones han sido agrupadas de acuerdo a su centroide más cercano, se recalculan los centroides de los k clústeres.

Se iteran estos dos últimos pasos hasta la convergencia de los centroides. Esto es, hasta que el valor de los centroides apenas se modifica (según un criterio de parada preestablecido, esto es, otro hiperparámetro).

Despliega los paneles siguientes para averiguar las principales ventajas y desventajas de este modelo de ML.

Ventajas

Las principales ventajas del algoritmo de las k-medias son su **sencillez** y su **escalabilidad** (aplicable con facilidad a grandes conjuntos de datos).

Desventajas

Las principales desventajas son la necesidad de elegir k manualmente y la alta **dependencia** de los valores iniciales, las medias con las que comienza el algoritmo. Para evitar esta última desventaja se suele replicar el algoritmo varias veces con distintas inicializaciones. Además, los centroides pueden verse fuertemente influidos por **valores atípicos**.

Para solventar esta última desventaja, en ocasiones, se usan de **medoides** en lugar de centroides. Estos medoides, al contrario de los centroides, son obligatoriamente observaciones de la muestra. Es decir, no elegimos medias aleatorias, sino observaciones reales recogidas en la base de datos.

El más común de los algoritmos de k-medoides es el **PAM: "Partitioning Around Medoids"**. Una ventaja adicional de los algoritmos basados en medoides es la interpretabilidad de los resultados. Mientras que un centroide puede no tener significado dentro de las observaciones muestrales, un medoide lo tiene por definición.

Estos algoritmos son **no-jerárquicos**, pues una observación puede cambiar de clúster durante la ejecución del mismo. Dos observaciones cualesquiera pueden pertenecer al mismo o a diferente grupo en diferentes iteraciones del algoritmo.

5.2.1 k-medias en R

Para ver cómo funciona el algoritmo de las k medias en R, vamos a emplear los datos de arrestos en USA, vistos en etapas anteriores del curso:

```
library(tidyverse)  #  manipilación de datos
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(cluster)    # algoritmos de clustering
library(factoextra) # algoritmos de clustering & visualización
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
head(USArrests)
```

```
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

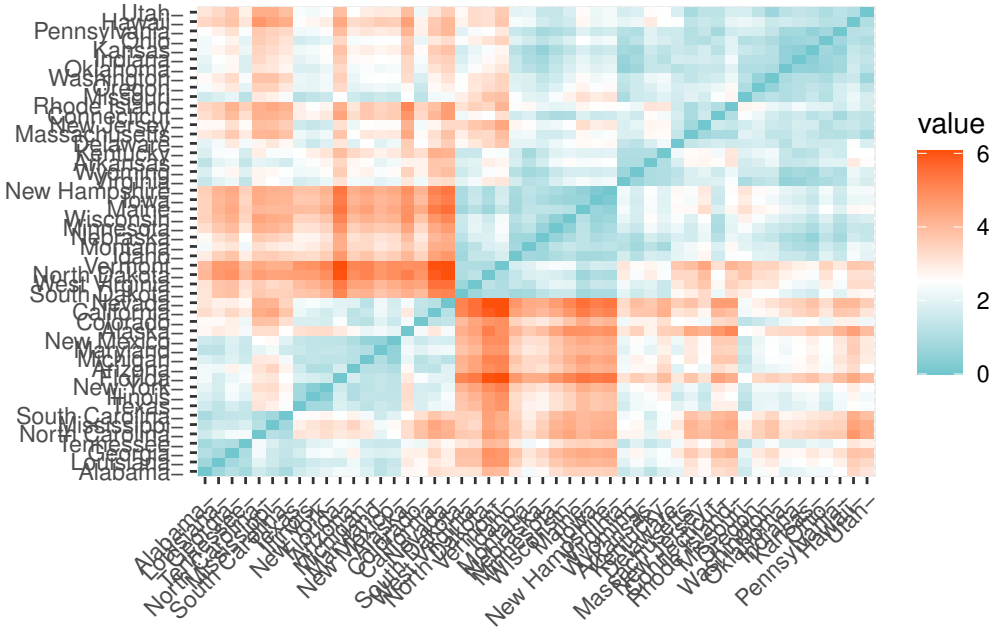
```
df <- USArrests

# eliminamos valores faltantes
df <- na.omit(df)
```

```r
# escalado de todas las variables
df <- scale(df)

head(df)
```

```
               Murder   Assault   UrbanPop         Rape
Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
Arizona    0.07163341 1.4788032  0.9989801  1.042878388
Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
California 0.27826823 1.2628144  1.7589234  2.067820292
Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

En primer lugar vamos a calcular la distancia Euclídea entre las observaciones de la base de datos. En R es sencillo calcular y visualizar la matriz de distancias utilizando las funciones `get_dist` y `fviz_dist` del paquete `factoextra` de R. La matriz de distancias se muestra a continuación. Esto empieza a ilustrar qué estados tienen grandes disimilitudes (rojo) frente a los que parecen ser bastante similares (verde azulado).

```r
distance <- get_dist(df)
fviz_dist(distance, gradient = list(low = "#00AFBB", mid = "white", high = "#FC4E07"))
```



Podemos aplicar el algoritmo de las k-medias con $k = 2$ sin más que llamar a la función `kmeans` tal y como sigue:

```
k2 <- kmeans(df, centers = 2, nstart = 25)
str(k2)
```

```
List of 9
 $ cluster     : Named int [1:50] 2 2 2 1 2 2 1 1 2 2 ...
  ..- attr(*, "names")= chr [1:50] "Alabama" "Alaska" "Arizona" "Arkansas" ...
 $ centers     : num [1:2, 1:4] -0.67 1.005 -0.676 1.014 -0.132 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:4] "Murder" "Assault" "UrbanPop" "Rape"
 $ totss       : num 196
 $ withinss    : num [1:2] 56.1 46.7
 $ tot.withinss: num 103
 $ betweenss   : num 93.1
 $ size        : int [1:2] 30 20
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

> Tarea
>
> ¿Qué significa `nstart=25` en la anterior llamada a `kmeans`?

Si imprimimos los resultados veremos que la técnica de agrupaciones dió lugar a 2 conglomerados de 30 y 20 observaciones cada uno. Vemos los centros de los conglomerados (medias) para los dos grupos en las cuatro variables (`Murder`, `Assault`, `UrbanPop`, `Rape`). También obtenemos la asignación de conglomerados para cada observación (es decir, Alabama se asignó al conglomerado 2, Arkansas se asignó al conglomerado 1, etc.).

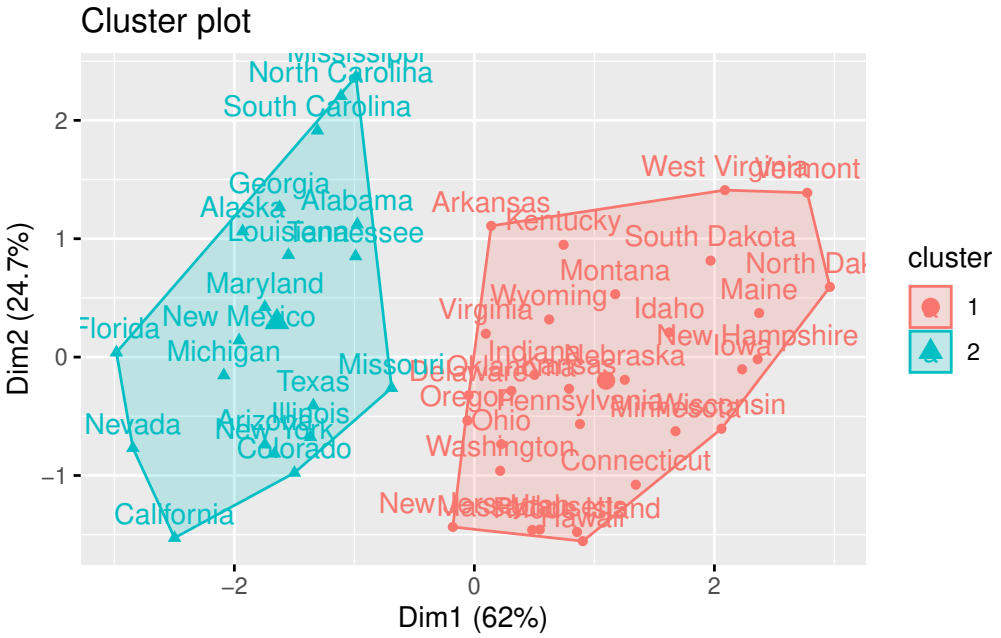
```
k2
```

```
K-means clustering with 2 clusters of sizes 30, 20

Cluster means:
     Murder    Assault   UrbanPop       Rape
1 -0.669956 -0.6758849 -0.1317235 -0.5646433
2  1.004934  1.0138274  0.1975853  0.8469650
```

```
Clustering vector:
       Alabama         Alaska        Arizona       Arkansas     California
             2              2              2              1              2
      Colorado    Connecticut       Delaware        Florida        Georgia
             2              1              1              2              2
        Hawaii          Idaho       Illinois        Indiana           Iowa
             1              1              2              1              1
        Kansas       Kentucky      Louisiana          Maine       Maryland
             1              1              2              1              2
 Massachusetts       Michigan      Minnesota    Mississippi       Missouri
             1              2              1              2              2
       Montana       Nebraska         Nevada  New Hampshire     New Jersey
             1              1              2              1              1
    New Mexico       New York North Carolina   North Dakota           Ohio
             2              2              2              1              1
      Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
             1              1              1              1              2
  South Dakota      Tennessee          Texas           Utah        Vermont
             1              2              2              1              1
      Virginia     Washington  West Virginia      Wisconsin        Wyoming
             1              1              1              1              1

Within cluster sum of squares by cluster:
[1] 56.11445 46.74796
 (between_SS / total_SS =  47.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

También podemos ver nuestros resultados utilizando `fviz_cluster`. Esto proporciona una buena visualización de los clusters. Si hay más de dos dimensiones (variables) `fviz_cluster` realizará un análisis de componentes principales (ver Chapter 4) y trazará los puntos de datos de acuerdo con los dos primeros componentes principales que explican la mayor parte de la varianza.

```
fviz_cluster(k2, data = df)
```



Tarea

¿Qué información estamos perdiendo al representar únicamente dos dimensiones del conjunto de datos? (Repasar el capítulo Chapter 4)

5.2.2 Número óptimo de clústeres

Como hemos comentado anteriormente, en numerosas ocasiones el número óptimo de clústeres de un problema no supervisado lo establece el dominio de aplicación, o más concretamente nuestra necesidad de comunicar los resultados y que estos tengan sentido y sean lo más explicables posibles.

Nombrar los grupos

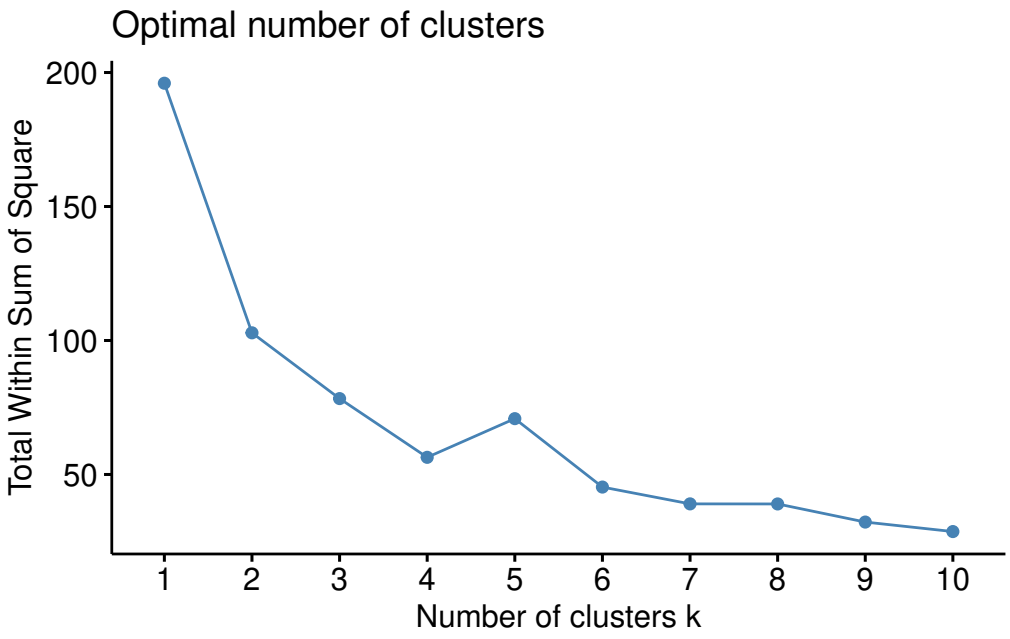
Nombrar los clusters en un análisis no supervisado es fundamental para dar significado a los resultados, comunicar eficazmente las conclusiones y facilitar la toma de decisiones informadas.

A continuación mostramos los tres métodos más populares para determinar el número óptimo de clústeres: método del *codo* , *silhoutte* y método estadístico de la *brecha* (Gap).

Método del codo

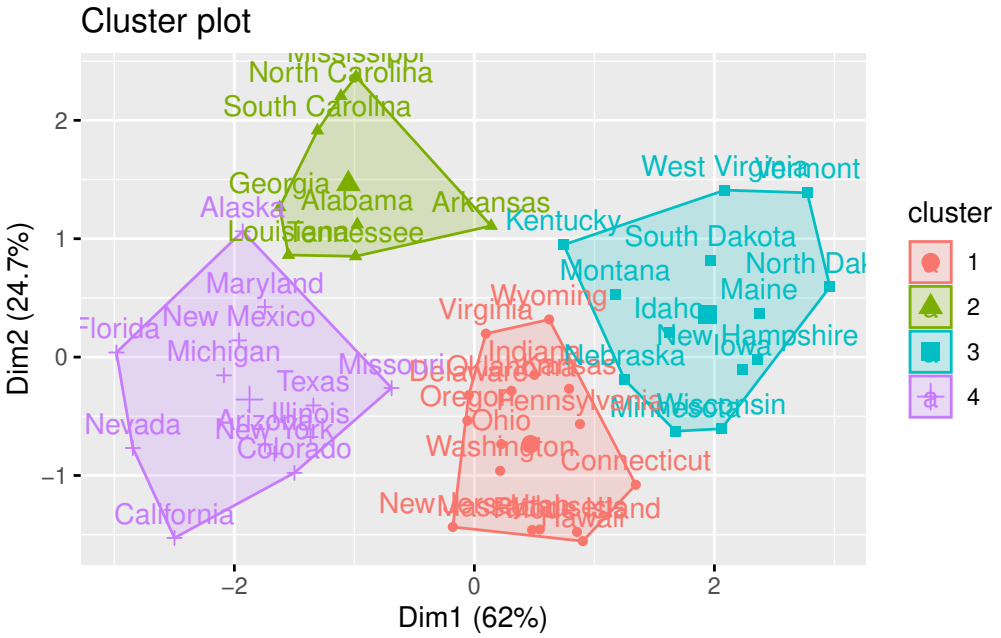
Recordemos que la idea básica de los métodos de división en clústeres, como k-medias, es definir los clústeres de manera que se reduzca al mínimo la variación total dentro de los clústeres. Podemos calcular este valor de variación total para diferentes elecciones de k y graficar dichos valores. La ubicación de un cambio de pendiente abrupto (un codo) en la figura se considera generalmente como un indicador del número apropiado de grupos. Veamos un ejemplo en R.

```
# Reproducible
set.seed(123)

fviz_nbclust(df, kmeans, method = "wss")
```



En este caso parece que 4 es una buena elección para el número óptimo de clusters. De modo qué:

```
k4 <- kmeans(df, centers = 4, nstart = 25)

fviz_cluster(k4, data = df)
```



Método de la silueta

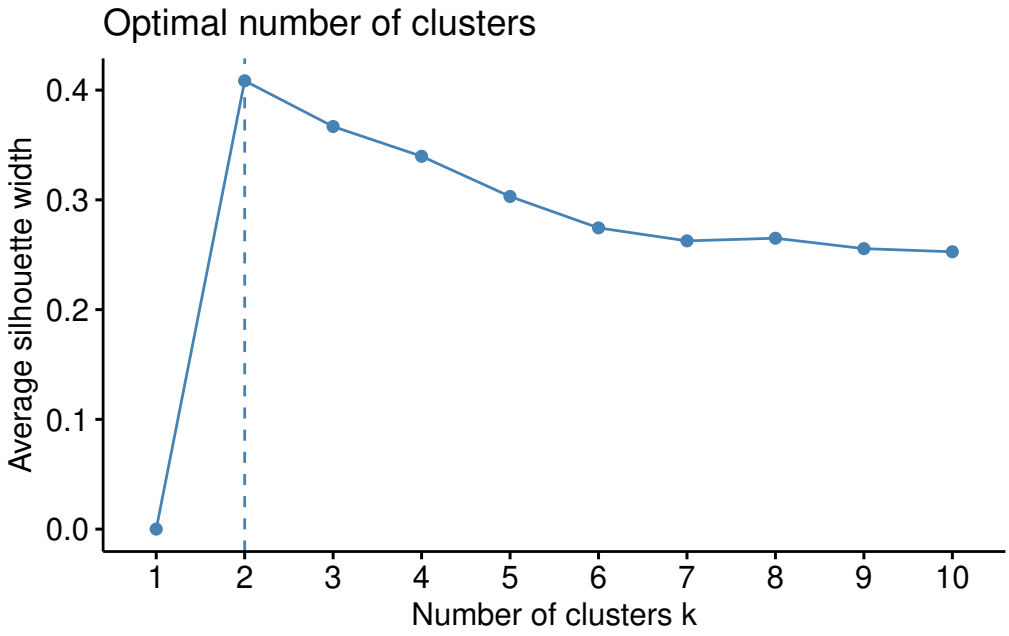
Podemos emplear otras técnicas (no supervisadas) para valorar la coherencia, o calidad, de los resultados de un algoritmo de agrupamiento. **Silhouette** es una de estas técnicas. El enfoque de la silueta media mide la calidad de una agrupación. Es decir, determina hasta qué punto cada observación se encuentra, correctamente ubicada, dentro de su agrupación. Una anchura de silueta media elevada indica una buena agrupación. El método de la silueta media calcula la silueta media de las observaciones para distintos valores de k. El número óptimo de conglomerados k es el que maximiza la silueta media en un rango de posibles valores de k. Para cada observación x_i de la base de datos calculamos la siguiente expresión:

$$s(x_i) = \frac{b(x_i) - a(x_i)}{max(a(x_i), b(x_i))},$$

donde $a(x_i)$ es la media de las distancias de la observación x_i a los puntos en su propio clúster, $b(x_i)$ es la media de las distancias de x_i a los puntos en su cluster más cercano (excluyendo el suyo propio). La interpretación es muy sencilla: las observaciones que "encajan" bien en el cluster al que pertenecen tienen valores altos, mientras que las observaciones que "no encajan" bien en el clúster al que han sido asignadas tienen valores pequeños o incluso negativos.

Es posible realizar análisis de agrupamiento con diferente número de clústeres y comparar en cada uno de esos análisis los valores de Silhoutte obtenidos.

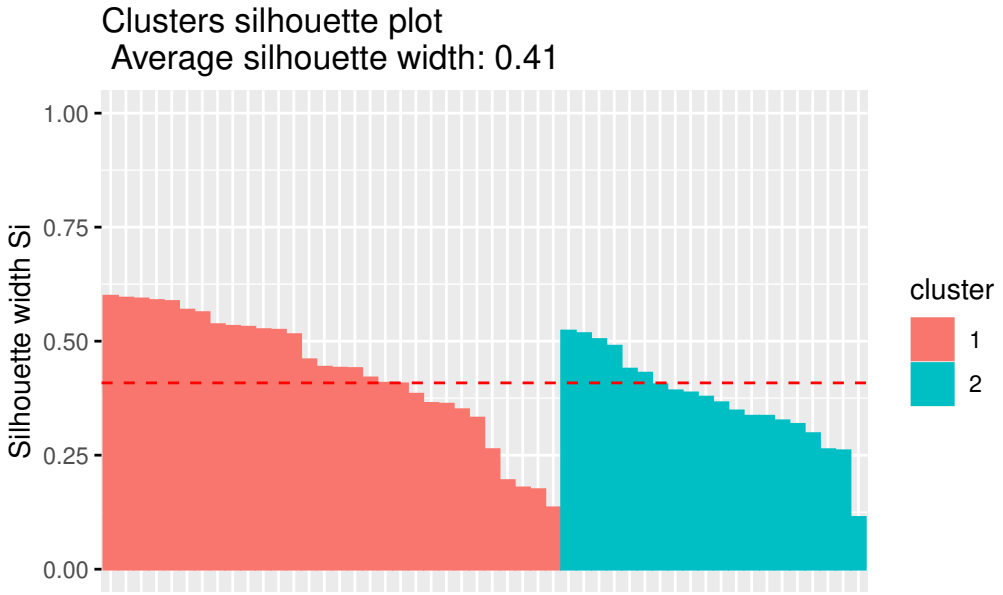
```
fviz_nbclust(df, kmeans, method = "silhouette")
```



Con esta técnica, en nuestros datos de ejemplo, parece ser que la mejor elección es k igual a 2.

```
library(cluster)
sil <- silhouette(k2$cluster, dist(df))
fviz_silhouette(sil)
```

```
  cluster size ave.sil.width
1       1   30          0.43
2       2   20          0.37
```



La interpretación del coeficiente de silueta es la siguiente: Un valor positivo significa que la observación está bien agrupada. Cuanto más se acerque el coeficiente a 1, mejor agrupada está la observación. Por contra, un valor negativo significa que la observación está mal agrupada. Finalmente, un valor igual a 0 significa que la observación se encuentra entre dos conglomerados.

El gráfico de siluetas anterior y el coeficiente de silueta medio ayudan a determinar si la agrupación es buena o no. Si una gran mayoría de los coeficientes de silueta son positivos, significa que las observaciones están situadas en el grupo correcto.

Gap

El Método Gap es útil porque ofrece una forma objetiva de determinar el número de clusters sin depender de suposiciones subjetivas. Al comparar los resultados del clustering real con datos de referencia aleatorios, ayuda a evitar la sobreelección o subelección de clusters y permite tomar decisiones más informadas sobre la estructura de los datos.

El proceso es el siguiente:

1. Se aplica el algoritmo de clustering (por ejemplo, K-Means) a los datos con diferentes valores de k, que representan el número de clusters que se desea evaluar. Se genera un conjunto de resultados de clustering para cada valor de k.
2. Se generan conjuntos de *datos de referencia aleatorios* (datos simulados) con la misma estructura y variabilidad que los datos reales, pero sin patrones de clustering. Estos datos aleatorios se utilizan como referencia para evaluar la calidad de los clusters obtenidos en el paso anterior.

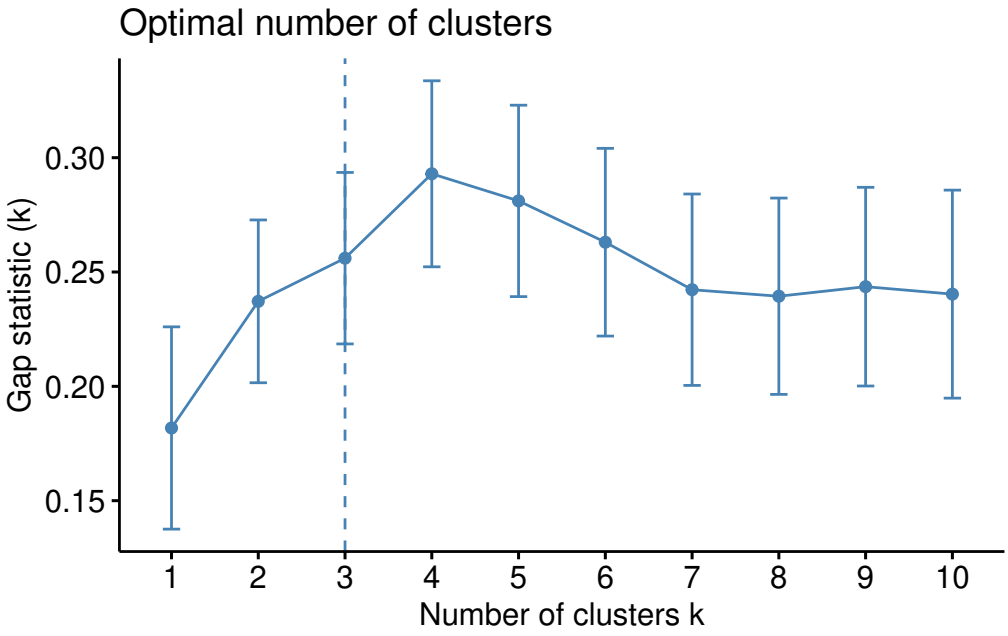
3. Se calcula el estadístico **Gap** para cada valor de k. Este estadístico compara la dispersión de los datos reales con la dispersión de los datos aleatorios generados. Cuanto más grande sea la brecha entre estas dos dispersiones, más sólido es el clustering para ese valor de k. El estadístico se calcula como la diferencia entre el logaritmo de la dispersión intra-cluster de los datos reales y el logaritmo de la dispersión intra-cluster de los datos de referencia.
4. Selección del Número Óptimo de Clusters: El valor de k que maximiza el estadístico anterior se considera el número óptimo de clusters. En otras palabras, se elige el valor de k donde la brecha entre los datos reales y los datos de referencia es más grande.

Existe una función en R que realiza todo este proceso. En los datos de ejemplo:

```r
set.seed(123)
gap_stat <- clusGap(df, FUN = kmeans, nstart = 25,
                    K.max = 10, B = 50)

print(gap_stat, method = "firstmax")
```

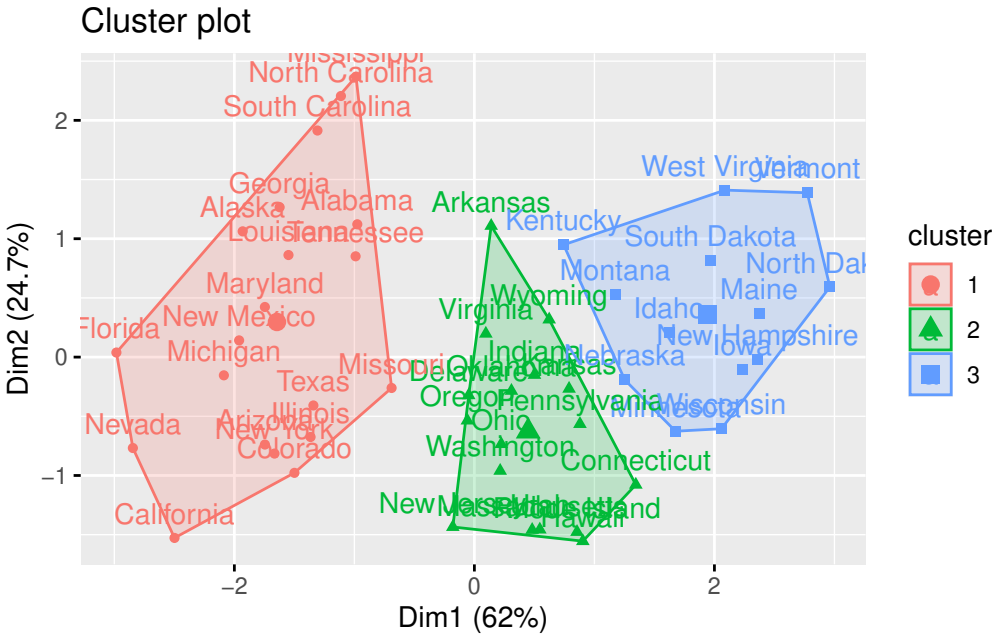
```
Clustering Gap statistic ["clusGap"] from call:
clusGap(x = df, FUNcluster = kmeans, K.max = 10, B = 50, nstart = 25)
B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
 --> Number of clusters (method 'firstmax'): 4
          logW   E.logW       gap     SE.sim
 [1,] 3.458369 3.640154 0.1817845 0.04422857
 [2,] 3.135112 3.372283 0.2371717 0.03559601
 [3,] 2.977727 3.233771 0.2560446 0.03749193
 [4,] 2.826221 3.119172 0.2929511 0.04067348
 [5,] 2.738868 3.019965 0.2810969 0.04185469
 [6,] 2.666967 2.930002 0.2630347 0.04105040
 [7,] 2.609895 2.852152 0.2422572 0.04184725
 [8,] 2.539156 2.778562 0.2394054 0.04292750
 [9,] 2.468162 2.711752 0.2435901 0.04344197
[10,] 2.407265 2.647595 0.2403307 0.04548446
```

```r
fviz_gap_stat(gap_stat)
```



Aparéntemente 3 es el número óptimo de clusters.

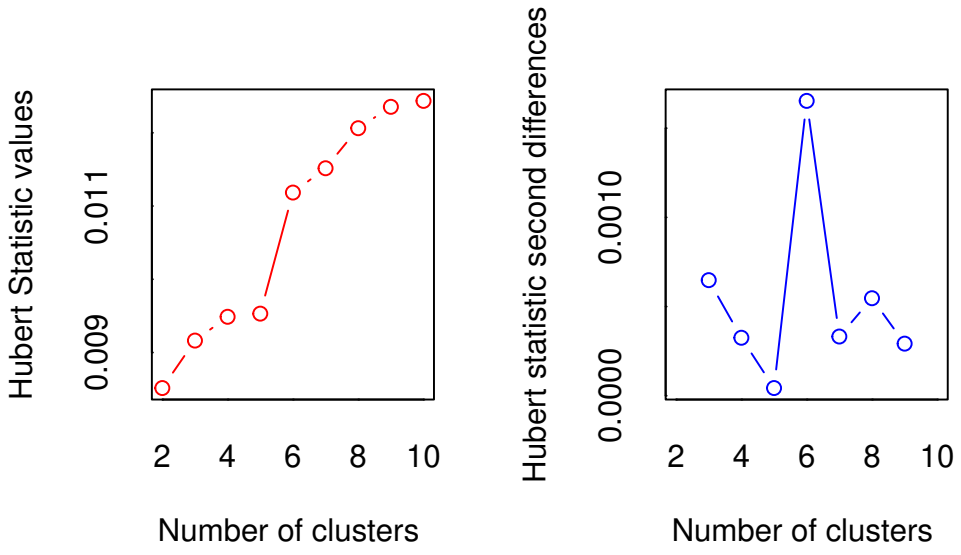
```
k3 <- kmeans(df, centers = 3, nstart = 25)

fviz_cluster(k3, data = df)
```



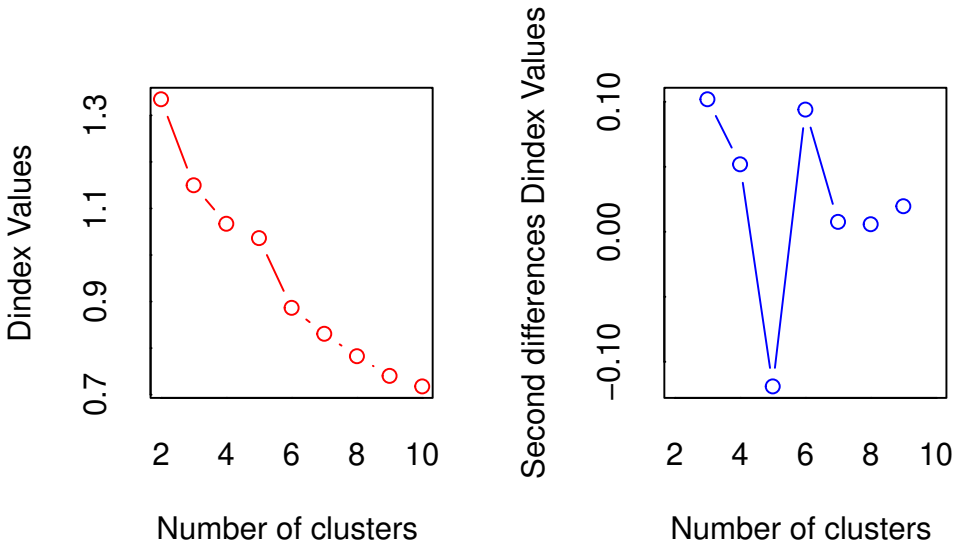
Tarea

El paquete `NbClust` proporciona **30 (treinta!!!)** índices para determinar el número relevante de clústeres y propone a los usuarios el mejor esquema de agrupación a partir de los diferentes resultados obtenidos variando todas las combinaciones de número de clústeres, medidas de distancia y métodos de agrupación. ¡Pruébalo!

```
library("NbClust")
nb <- NbClust(df, distance = "euclidean", min.nc = 2,
        max.nc = 10, method = "kmeans")
```



*** : The Hubert index is a graphical method of determining the number of clusters.
In the plot of Hubert index, we seek a significant knee that corresponds to a
significant increase of the value of the measure i.e the significant peak in
index second differences plot.



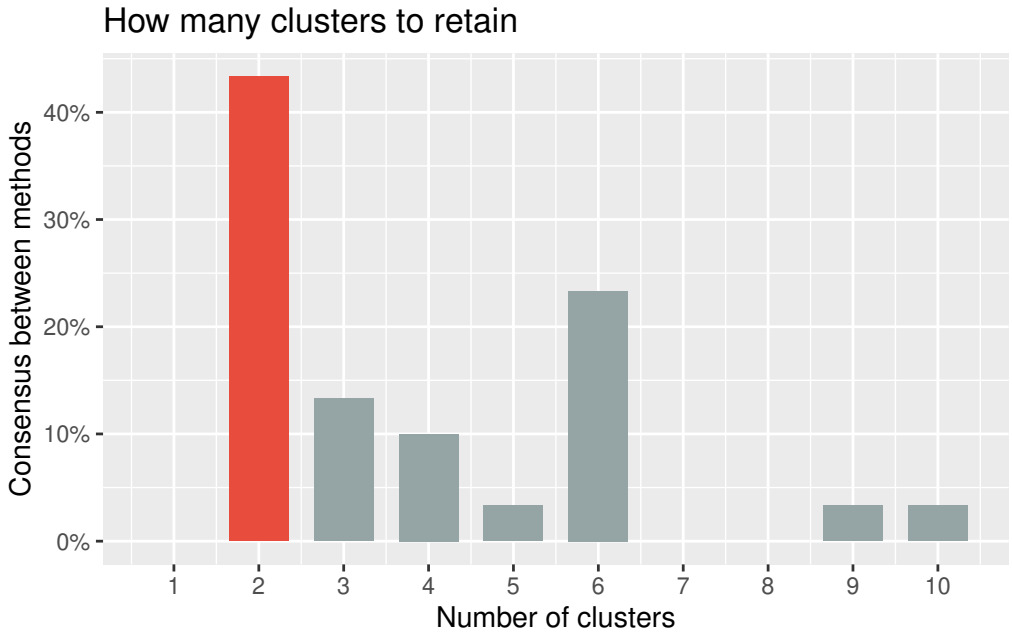
*** : The D index is a graphical method of determining the number of clusters.
In the plot of D index, we seek a significant knee (the significant peak in

```
                second differences plot) that corresponds to a significant increase of the va
                the measure.

*****************************************************************
* Among all indices:
* 11 proposed 2 as the best number of clusters
* 2 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 7 proposed 6 as the best number of clusters
* 1 proposed 9 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


*****************************************************************
```

```r
library(parameters)

n_clust <- n_clusters(as.data.frame(df),
  package = c("easystats", "NbClust", "mclust"),
  standardize = FALSE)
n_clust
```

```
# Method Agreement Procedure:

The choice of 2 clusters is supported by 13 (43.33%) methods out of 30 (Elbow, Silhouette, Ga
```

```r
plot(n_clust)
```



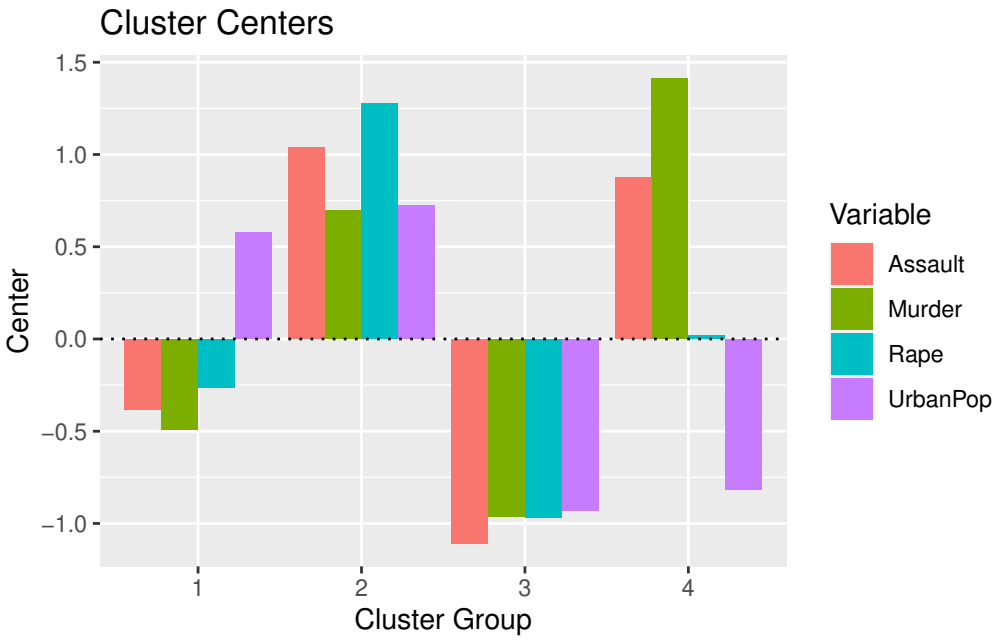
Creemos que te ha quedado claro que no existe una regla única para la elección del número óptimo de clústeres. Al contrario, hay muchos métodos para estimar el mejor número de clústeres y, obviamente, no todos ellos dan el mismo resultado. Se recomienda considerar los resultados de diferentes métodos y explorar varios números de clústeres buscando siempre una coherente interpretación de los resultados.

En el ejemplo, eligiendo 4 (quizás no sea la elección más evidente, pero ¿será interpretable?) como el número de clusters, podemos obtener los resultados finales tratando de nombrar los clusters:

```
USArrests %>%
  mutate(Cluster = k4$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

```
# A tibble: 4 x 5
  Cluster Murder Assault UrbanPop  Rape
    <int>  <dbl>   <dbl>    <dbl> <dbl>
1       1   5.66   139.      73.9  18.8
2       2  13.9    244.      53.8  21.4
3       3   3.6     78.5     52.1  12.2
4       4  10.8    257.      76    33.2
```

```r
library(parameters)

res_kmeans <- cluster_analysis(df,
  n = 4,
  method = "kmeans"
)

plot(summary(res_kmeans))
```



Tarea

Intenta "nombrar" los 4 clusters del ejemplo. Para ello deberías fijarte también en las componentes principales.

5.3 Cluster Jerárquico

Las técnicas de agrupamiento jerárquico generan una clasificación iterativa de clústeres anidados mediante la unión o la separación de clústeres creados en etapas anteriores. Existen dos alternativas posibles:

- **Aglomerativos**: en la versión aglomerativa, cada observación comienza siendo un clúster, y en cada iteración se unen en un único clúster los dos clústeres más similares, hasta alcanzar una situación final en la que todos las observaciones pertenecen a un único clúster. Esta versión se conoce como **AGNES** ("Agglomerative Nesting").
- **Divisivos**: en la versión divisiva, todas las observaciones comienzan en un único clúster y las divisiones se realizan de forma recursiva, a medida que se desciende en la jerarquía, terminando cada observación formando un único clúster individual. Esta versión se conoce como **DIANA** ("Divise Analysis").

Se necesita un criterio de conexión o "*linkage*" que especifique cómo se determina el parecido (o la disimilitud) entre dos clústeres. Este criterio no es único. Algunos de los criterios más comunes son:

i Método de Ward

Minimiza la suma de las diferencias cuadradas dentro de los clústeres. Minimiza la varianza total dentro del conglomerado.

i Agrupamiento de enlace completo

Minimiza la disimilitud máxima entre las observaciones de dos clústeres. Calcula todas las disimilitudes por pares entre los elementos del conglomerado A y los elementos del conglomerado B, y considera el mayor valor (es decir, el valor máximo) de estas disimilitudes como la distancia entre los dos conglomerados. Tiende a producir clusters más compactos.

i Agrupamiento de enlace promedio

Minimiza el promedio de las disimilitudes entre las observaciones de dos clústeres. Calcula todas las disimilitudes por pares entre los elementos del conglomerado A y los elementos del conglomerado B, y considera la media de estas disimilitudes como la distancia entre los dos conglomerados.

i Agrupamiento de enlace mínimo o simple

Minimiza las disimilitudes entre las observaciones más cercanas de dos clústeres. Es decir, calcula todas las disimilitudes por pares entre los elementos del conglomerado A y los elementos del conglomerado B, y considera la menor de estas disimilitudes como criterio de vinculación. Tiende a producir clusters largos y "dispersos".

Agrupamiento de enlace de centroides

Calcula la disimilitud entre el centroide del conglomerado A y el centroide del conglomerado B. Agrupación de enlace de centroides.

Para recordar

El criterio de agrupación o conexión es un parámetro fundamental en el resultado final del clustering jerárquico.

5.3.1 Cluster jerárquico en R

Existen diferentes funciones disponibles en R para calcular el clustering jerárquico. Las funciones más utilizadas son:

- `hclust` [en el paquete `stats`] y `agnes` [en el paquete `cluster`] para el clustering jerárquico aglomerativo
- `diana` [en el paquete `cluster`] para clustering jerárquico divisivo

```
# Clustering jerárquico usando enlace completo
hc2 <- agnes(df, method = "complete" )

hc2$ac
```

```
[1] 0.8531583
```

El **coeficiente aglomerativo** mide la cantidad de estructura de agrupamiento encontrada (los valores más cercanos a 1 sugieren una fuerte estructura de agrupamiento).

Esto nos permite encontrar ciertos métodos de clustering jerárquico que pueden identificar estructuras de agrupación más fuertes. Aquí vemos que el método de Ward identifica la estructura de agrupación más fuerte de los cuatro métodos evaluados.

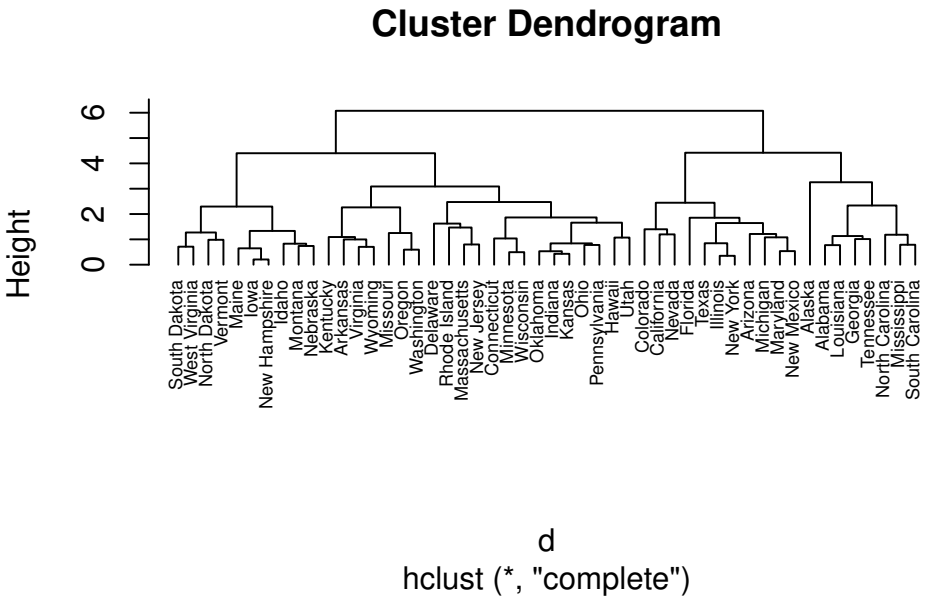
```
# Métodos evaluados
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

# Función para calcular el coeficiente de agrupamiento
ac <- function(x) {
  agnes(df, method = x)$ac
```

```
}

map_dbl(m, ac)
```

```
  average    single  complete      ward
0.7379371 0.6276128 0.8531583 0.9346210
```

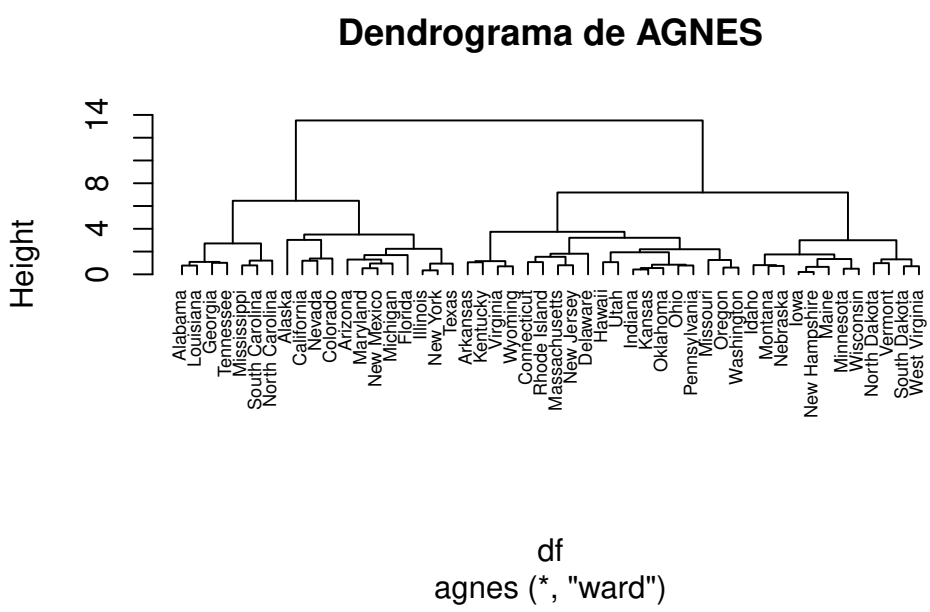
En ocasiones se emplea una representación gráfica en forma de árbol llamada **dendrograma** que ilustra las agrupaciones derivadas de la aplicación de una técnica de agrupamiento jerárquico. En el eje de ordenadas se presenta la distancia a la que se unen los diferentes clústeres. Las observaciones aparecen en el eje de abscisas.

```
# Matriz de disimilaridades
d <- dist(df, method = "euclidean")

# Clustering jerárquico usando enlace completo
hc1 <- hclust(d, method = "complete" )

# Dendrograma
plot(hc1, cex = 0.6, hang = -1)
```



Fíjate cómo obtenemos diferentes resultados según el método propuesto.

```r
hc2 <- agnes(df, method = "ward" )

# Drendrograma
pltree(hc2, cex = 0.6, hang = -1, main = "Dendrograma de AGNES")
```



La pregunta a la que nos enfrentamos es a qué distancia cortar el dendrograma, es decir, dónde dibujar una línea horizontal que determine el número óptimo de clústeres. Por ejemplo, en nuestro caso, cortar en 10 generaría dos clústeres. Sin embargo, cortar en 5 generaría cuatro clústeres, dos a la izquierda, dos a la derecha.

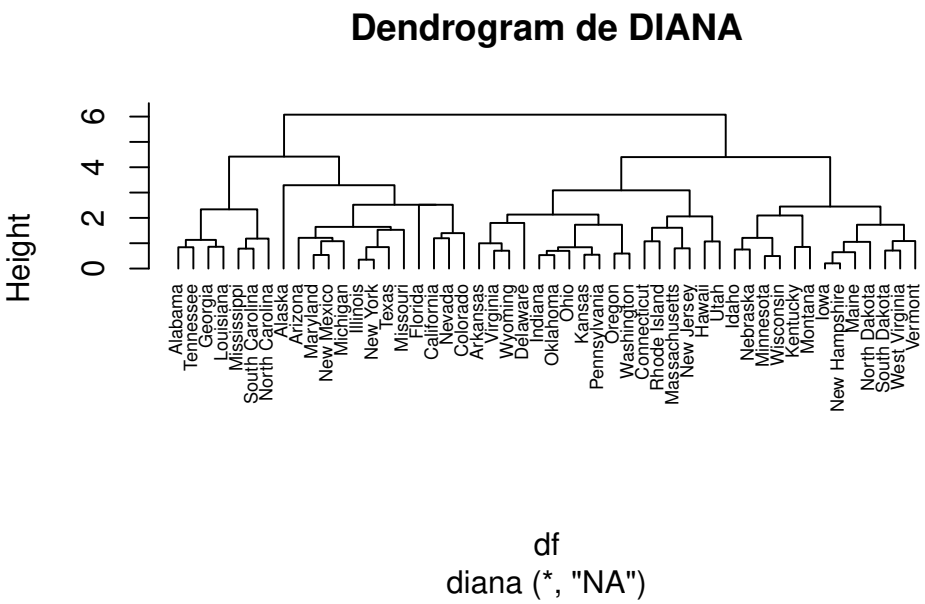
A continuación aplicamos el método divisivo.

```r
# Clustering jerárquico divisivo
hc4 <- diana(df)

# Coeficiente de división; cantidad de estructura de agrupación encontrada
hc4$dc
```

```
[1] 0.8514345
```

```r
## [1] 0.8514345
```

```r
# Drendrograma
pltree(hc4, cex = 0.6, hang = -1, main = "Dendrogram de DIANA")
```



En el dendrograma anterior, cada hoja corresponde a una observación. A medida que ascendemos en el árbol, las observaciones que son similares entre sí se combinan en ramas, que a su vez se fusionan a mayor altura.

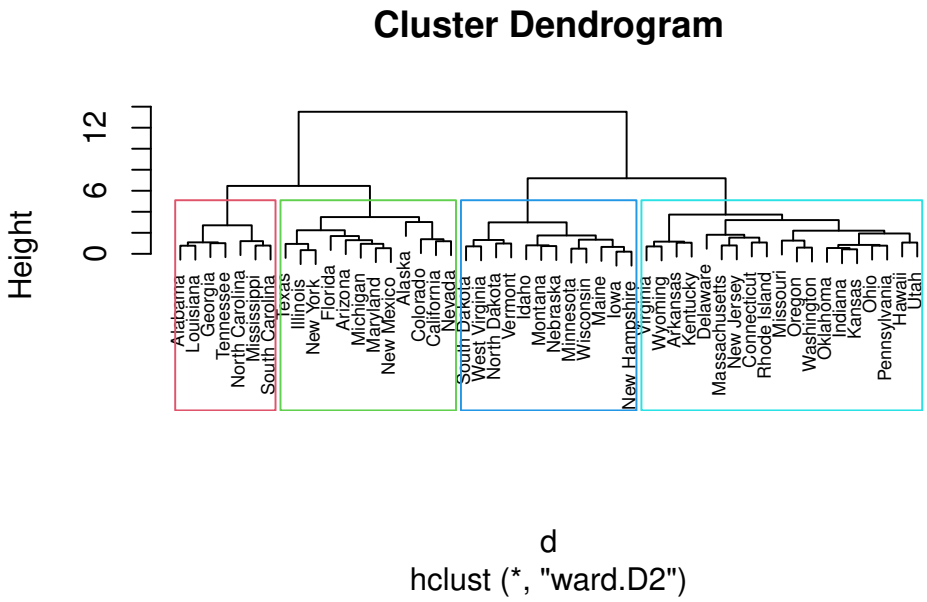
La altura de la fusión, que figura en el eje vertical, indica la (di)similitud entre dos observaciones. Cuanto mayor es la altura de la fusión, menos similares son las observaciones.

> **Atención**
>
> Cuando empleamos un **dendrograma**, las conclusiones sobre la proximidad de dos observaciones sólo pueden extraerse a partir de la altura a la que se fusionan las ramas que contienen primero esas dos observaciones. No podemos utilizar la proximidad de dos observaciones a lo largo del eje horizontal como criterio de su similitud.

Tal como hemos indicado, la altura del corte del dendrograma controla el número de clusters obtenidos. Desempeña el mismo papel que la k en la agrupación k-means. Para identificar subgrupos (es decir, clusters), podemos cortar el dendrograma con la función `cutree` de R:

```r
# Método de Ward
hc5 <- hclust(d, method = "ward.D2" )
```

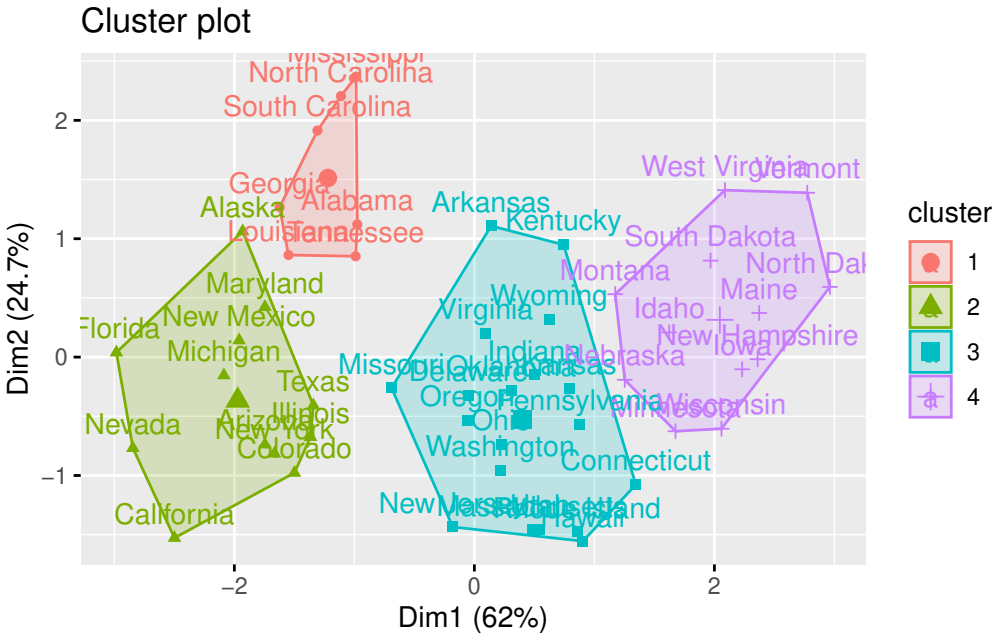
```r
# Cortamos en 4 clusters
sub_grp <- cutree(hc5, k = 4)

# Visualizamos el corte en el dendrograma
plot(hc5, cex = 0.6)
rect.hclust(hc5, k = 4, border = 2:5)
```




```r
# Número de observaciones en cada cluster
table(sub_grp)
```

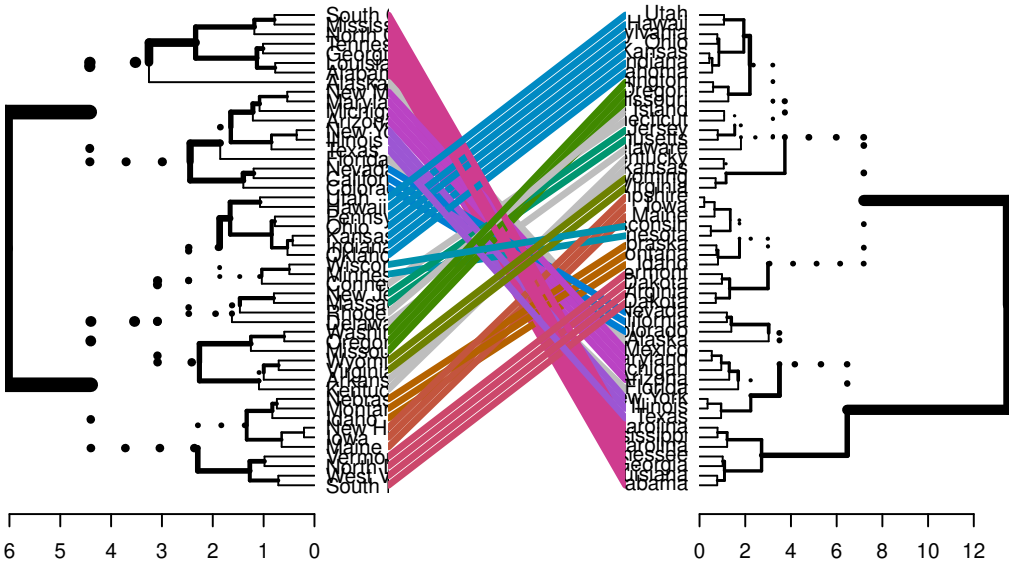
```
sub_grp
 1  2  3  4
 7 12 19 12
```

```r
# Visualización
fviz_cluster(list(data=df,cluster=sub_grp))
```



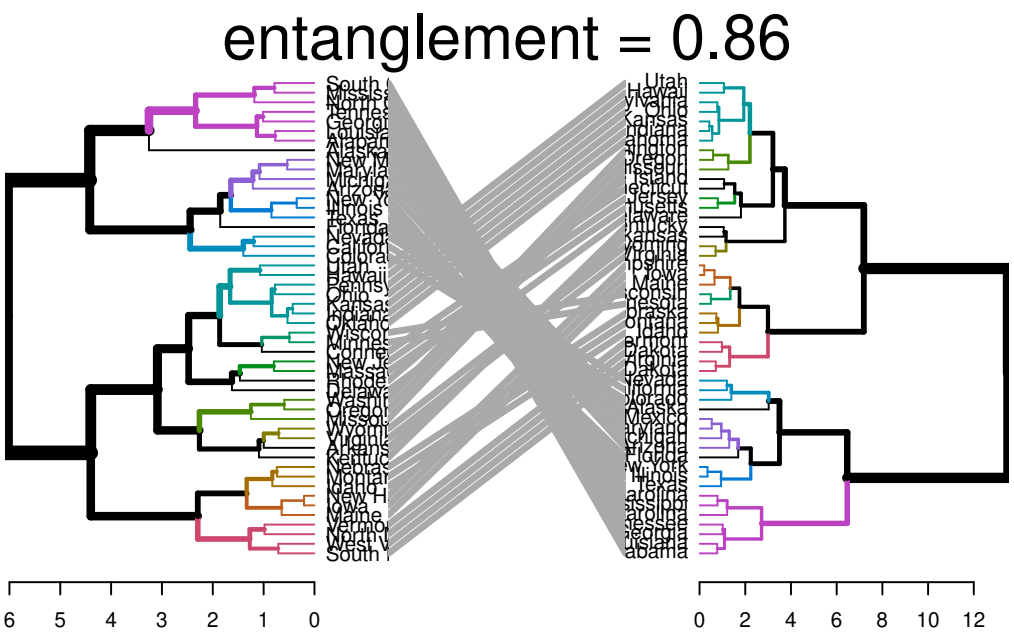
Podemos ir un paso más allá y comparar dos dendrogramas. En este ejemplo comparamos los resultados obtenidos con el método de "Ward" frente al "completo".

```
library(dendextend)

# Matriz de distancias
res.dist <- dist(df, method = "euclidean")

# Calcuamos los dos clustering jerárquicos
hc1 <- hclust(res.dist, method = "complete")
hc2 <- hclust(res.dist, method = "ward.D2")

# Dendrogramas
dend1 <- as.dendrogram (hc1)
dend2 <- as.dendrogram (hc2)

# los enfrentamos
tanglegram(dend1, dend2)
```



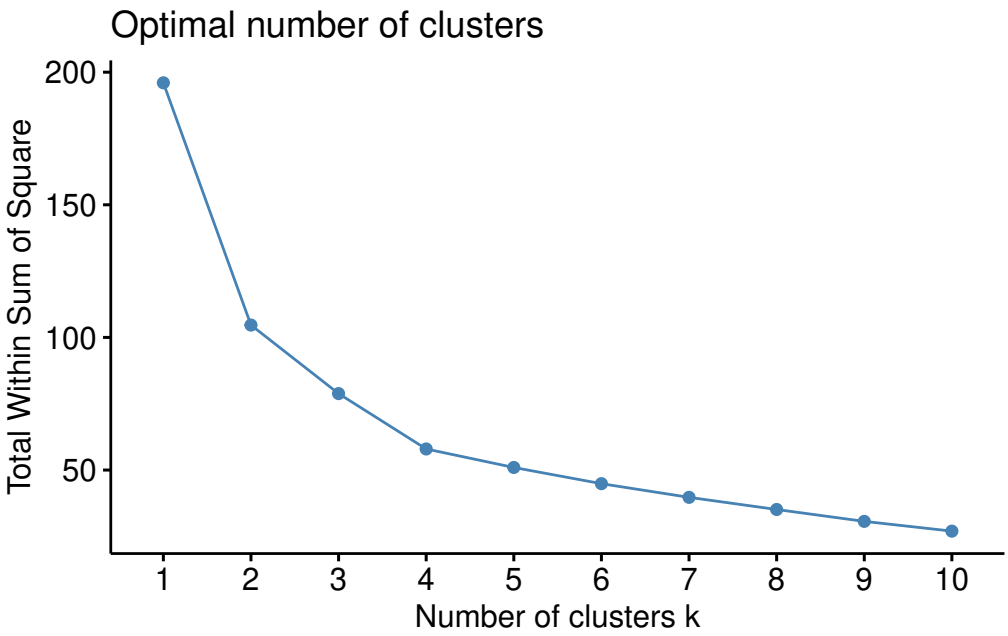
El resultado muestra nodos "únicos", con una combinación de etiquetas/elementos no presentes en el otro árbol, resaltados con líneas discontinuas. La calidad de la alineación de los dos árboles puede medirse utilizando la función de entrelazamiento. El entrelazamiento es una medida entre 1 (entrelazamiento total) y 0 (sin entrelazamiento). Un coeficiente de entrelazamiento menor corresponde a una buena alineación.

```
dend_list <- dendlist(dend1, dend2)

tanglegram(dend1, dend2,
  highlight_distinct_edges = FALSE, # Turn-off dashed lines
  common_subtrees_color_lines = FALSE, # Turn-off line colors
  common_subtrees_color_branches = TRUE, # Color common branches
  main = paste("entanglement =", round(entanglement(dend_list), 2))
  )
```



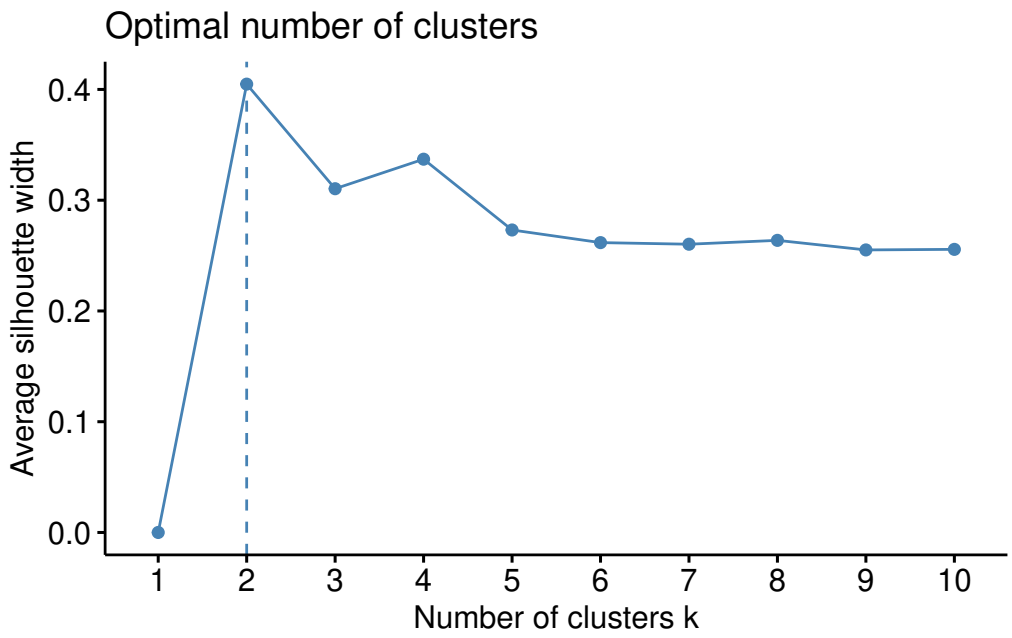
Tal y como hacíamos en el clustering no jerárquico, podemos aplicar métodos para determinar el número óptimo de clusters. Por ejemplo, el método del codo:

```
fviz_nbclust(df, FUN = hcut, method = "wss")
```

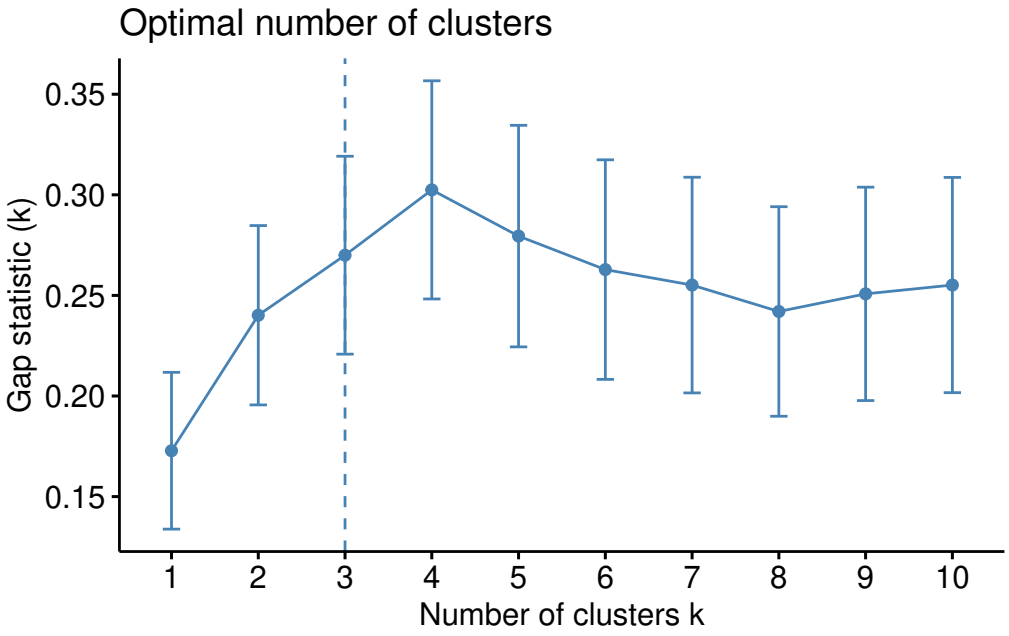


En el ejemplo propuesto, elegir 4 como número óptimo parece una buena elección. Sin embargo, y como pasaba en los métodos no jerárquicos, métodos alternativos pueden llevarnos a soluciones alternativas.

```
fviz_nbclust(df, FUN = hcut, method = "silhouette")
```




```
gap_stat <- clusGap(df, FUN = hcut, nstart = 25, K.max = 10, B = 50)
fviz_gap_stat(gap_stat)
```



Ventajas del clustering jerárquico

- **Jerarquía de Clusters**: El clustering jerárquico crea una jerarquía de clusters que permite analizar los datos a diferentes niveles de granularidad. Es posible, por tanto, explorar tanto clusters globales como subgrupos más específicos. Al tener una jerarquía de clusters, puedes tomar decisiones a diferentes niveles de detalle. Esto es valioso para la segmentación de mercado, la taxonomía de especies, la organización de documentos, entre otros.
- **Interpretación Visual**: El dendrograma facilita la interpretación visual de cómo se agrupan los datos y cómo se relacionan entre sí.
- **No Requiere Especificación Previa del Número de Clusters**: A diferencia de algunos algoritmos de clustering que requieren que especifiques el número de clusters de antemano como k-medias, el clustering jerárquico no necesita esta información.
- **Identificación de Subgrupos**: El clustering jerárquico es eficaz para la identificación de subgrupos dentro de clusters más grandes. Esto es útil en áreas como la segmentación de clientes, donde se pueden tener clusters generales y luego identificar subgrupos más específicos.
- **Detección de Outliers**: El clustering jerárquico puede ayudar a identificar outliers (valores atípicos) que no se ajustan a ningún cluster específico y que pueden ser importantes en el análisis de datos.

- **No Sensible a la Inicialización**: A diferencia de algunos algoritmos de clustering, como el k-means, el clustering jerárquico no es sensible a la inicialización de centroides, lo que puede ayudar a evitar soluciones subóptimas.
- **Análisis Exploratorio de Datos**: El clustering jerárquico es útil en la exploración inicial de datos, ya que proporciona una visión general de cómo se agrupan naturalmente los datos sin la necesidad de conocimiento previo.

Desventajas del clustering jerárquico

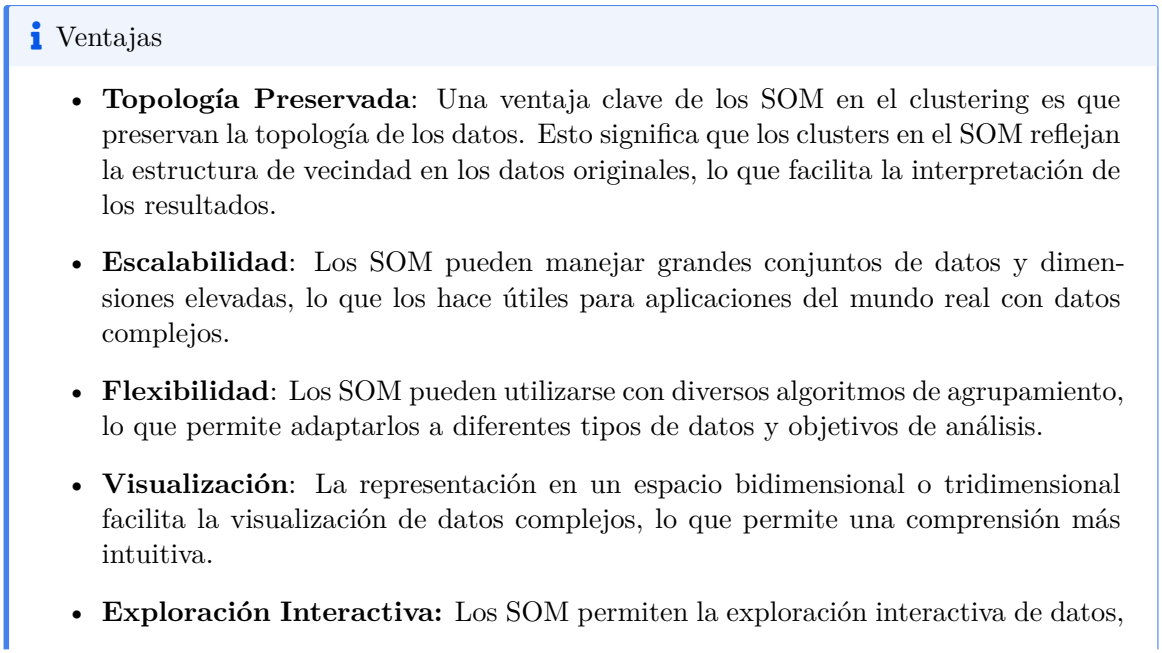
- **Requiere definir un criterio de corte**: Para convertir la jerarquía en clusters finales, es necesario definir un criterio de corte en el dendrograma. Esta elección puede ser subjetiva y afectar los resultados. Así mismo hay que definir el tipo de enlace a emplear (y la medida de disimilitud) Cada una de estas decisiones puede influir mucho en los resultados obtenidos. En la práctica, probamos varias opciones diferentes y buscamos la que ofrece la solución más útil o interpretable.
- Con estos métodos, **no hay una única respuesta correcta**: debe considerarse cualquier solución que exponga algunos aspectos interesantes de los datos.
- **No es óptimo para todos los tipos de datos**: El clustering jerárquico funciona mejor cuando los clusters tienen una estructura jerárquica natural. En algunos casos, donde no existe una jerarquía clara, otros métodos de clustering pueden ser más apropiados.
- **No es adecuado para datos de alta dimensión**: El rendimiento del clustering jerárquico puede disminuir en conjuntos de datos de alta dimensión debido a la maldición de la dimensionalidad. Este fenómeno significa que al aumentar el número de dimensiones de un problema se pueden agravar muchos de los problemas que aparecen en dimensiones bajas (curse of dimensionality, R. Bellman and Kalaba (1961)).
- **No siempre produce resultados reproducibles**: La estructura jerárquica resultante puede variar según la métrica de distancia y el enfoque de enlace utilizado, lo que puede dar lugar a resultados no siempre reproducibles.
- **Sin capacidad de predicción**: Las técnicas de agrupamiento jerárquicas no son útiles a la hora de predecir el clúster al que pertenecen nuevas observaciones.

5.4 Mapas auto-organizados

Los Mapas Auto-organizados de Kohonen (SOM por sus siglas en inglés, "*Self-Organizing Maps*") no solo son una poderosa herramienta para la **reducción de la dimensión**, sino que también son ampliamente utilizados como algoritmo de clustering. Aunque la reducción de la dimensión es una de sus aplicaciones más destacadas, los SOM también tienen la capacidad de agrupar datos de manera efectiva.

El proceso de clustering con SOM implica organizar datos en grupos o clusters de manera que los elementos dentro de un mismo grupo sean similares entre sí en función de ciertas características. Aquí se explica cómo los SOM se utilizan para esta tarea:

1. **Inicialización**: Se crea una red de nodos o neuronas en una estructura bidimensional, conocida como "*mapa SOM*". Cada neurona representa una ubicación en el espacio SOM.
2. **Asignación de Pesos**: Cada neurona en el SOM tiene asociado un vector de pesos que es del mismo tamaño que los datos originales que se están analizando.
3. **Entrenamiento**: Se presentan los datos al SOM, y cada dato se asigna a la neurona cuyos pesos son más similares a los atributos del dato. Las neuronas ganadoras (aquellas a las que se asigna un dato) y sus vecinas en el mapa SOM se ajustan para que se parezcan más al dato presentado. Este proceso de aprendizaje se repite varias veces.
4. **Agrupación en Clusters**: Después del entrenamiento, las neuronas en el mapa SOM que están cerca una de la otra representan clusters de datos. Los datos que se asignaron a estas neuronas durante el entrenamiento se consideran miembros de un mismo cluster.



ya que los usuarios pueden navegar por el mapa para inspeccionar las regiones y sus contenidos.

- **Reducción de Ruido:** Los SOM a menudo ayudan a reducir el ruido y la redundancia en los datos, lo que mejora la calidad del análisis.

Desventajas

- **Sensibilidad a la Inicialización**: Los SOM son sensibles a la inicialización de los pesos de las neuronas. Los resultados pueden variar significativamente según cómo se configuren los pesos iniciales, lo que significa que pueden converger a soluciones subóptimas si no se seleccionan adecuadamente los valores iniciales.
- **Determinación del Tamaño del Mapa**: Elegir el tamaño adecuado para el mapa SOM puede ser un desafío. Si el mapa es demasiado pequeño, puede no capturar la estructura de los datos correctamente, mientras que si es demasiado grande, puede sobreajustarse a los datos y perder la capacidad de generalización.
- **Interpretación de los Resultados**: La interpretación de los resultados de un SOM puede ser complicada, especialmente en mapas de alta dimensión. Mapear los clusters y las relaciones entre las neuronas en el mapa a menudo requiere conocimiento experto del dominio.
- **Requiere Ajuste de Hiperparámetros**: El rendimiento de un SOM puede depender de la elección adecuada de hiperparámetros, como la tasa de aprendizaje, la vecindad de las neuronas y el número de iteraciones. En algunos casos, encontrar los valores óptimos puede ser un proceso de prueba y error.
- **Puede Converger a Mínimos Locales**: Como con muchos algoritmos de optimización, los SOM pueden converger a mínimos locales en lugar del mínimo global, lo que puede llevar a soluciones subóptimas.
- **Requiere Grandes Conjuntos de Datos**: Los SOM pueden no funcionar bien en conjuntos de datos pequeños o altamente desequilibrados, ya que su eficacia se basa en la capacidad de aprender patrones significativos a partir de una cantidad suficiente de datos.

Vamos a estudiar su implementación en R con la libraría `kohonen`.

```
library(kohonen)
```

```
Attaching package: 'kohonen'
```

```
The following object is masked from 'package:purrr':

    map
```

```
library(dplyr)
library(caTools)

bank = read.csv('https://raw.githubusercontent.com/rafiag/DTI2020/main/data/bank.csv')
set.seed(2938)
sample<-sample.split(bank$deposit,SplitRatio=0.5)
bank.train <- subset(bank,sample==TRUE)
df= bank.train %>%
      filter(balance>0 & previous>0 & pdays>0) %>%
      mutate( log.balance=log(balance),
              log.age=log(age),
              log.campaign=log(campaign),
              log.previous=log(previous),
              log.pdays = log(pdays),
              log.duration=log(duration)) %>%
      select(log.balance,log.age,log.campaign,log.duration,log.previous,log.pdays,deposit)

df.st <- scale(df[,-7])
df.som <- kohonen::som(as.matrix( df.st),somgrid(4,4,"hexagonal"))

names(df.som)
```

```
 [1] "data"             "unit.classif"     "distances"        "grid"
 [5] "codes"            "changes"          "alpha"            "radius"
 [9] "na.rows"          "user.weights"     "distance.weights" "whatmap"
[13] "maxNA.fraction"   "dist.fcts"
```

```
summary(df.som)
```

```
SOM of size 4x4 with a hexagonal topology and a bubble neighbourhood function.
The number of data layers is 1.
Distance measure(s) used: sumofsquares.
Training data included: 1284 objects.
Mean distance to the closest unit in the map: 2.413.
```

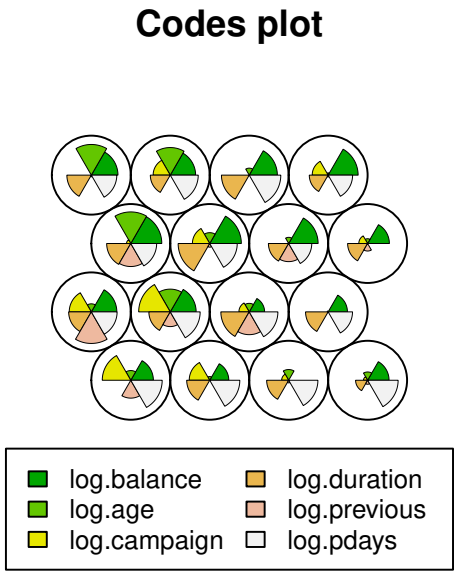
```r
# clustering
head(df.som$unit.classif)
```

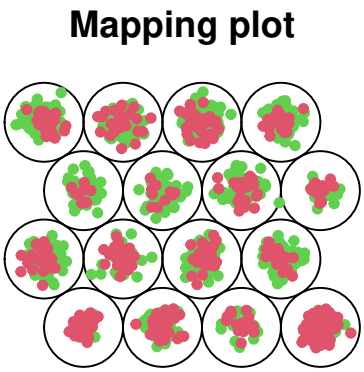
```
[1] 13 10 10 10 15 15
```

```r
# nombrar los clusters
df.som$codes
```

```
[[1]]
    log.balance     log.age log.campaign log.duration log.previous   log.pdays
V1   -0.1665504 -0.17250915   1.65905689  -2.89127875   0.75321963  0.76228798
V2   -0.5729971 -0.61916298   0.95649105   0.11360709  -0.61633860  0.53542329
V3   -2.7626873 -0.09630617  -0.22481605  -0.01775780  -0.43449514  0.39037929
V4   -0.2987460 -0.27867111  -0.48069577  -1.32194232  -0.32410665  0.60980323
V5    0.1692260 -0.29513795   1.09413528   0.06622639   1.81798163 -0.19898689
V6    0.2130609  0.79952586   1.92607192  -0.09957394   0.50713651  0.50409529
V7   -0.9925584 -0.25999586   0.06297161   0.80969889   1.13448930  0.29828886
V8   -0.5467391 -0.89454638  -0.75231500   0.05691503  -0.58716283 -0.44382395
V9    0.8004058  1.45723263  -0.46626151   0.25742877   1.15260783 -0.18585464
V10   0.8681725 -0.08718076   0.67816551   1.23176536  -0.35230246  0.09409626
V11   0.6203543 -0.43508207  -0.81201086  -0.24560976   0.77187053 -0.66260374
V12  -0.2636134 -0.53803480  -0.06677845  -0.28219715  -0.05178134 -4.61539128
V13   0.2996683  1.38922194  -0.83068969   0.29947979  -0.50925213 -0.25212519
V14   0.1516991  1.16041553   0.66077136  -0.28837446  -0.60162534  0.17447204
V15   0.4713511 -0.37448390  -0.83870156   0.73106396  -0.63544758  0.79393509
V16   0.6820996 -0.72646668   0.55768237  -0.42934114  -0.49679657 -0.42626768
```

```r
par(mfrow = c(1,2))
plot(df.som, type="codes")
plot(df.som, type="mapping", col = as.numeric(as.factor(df$deposit))+1, pch=20)
```






```
# versión supervisada
# kohmap <- xyf(as.matrix(df.st), as.factor(df$deposit),grid = somgrid(4, 4, "hexagonal"),

#plot(kohmap, type="codes", main = c("Codes X", "Codes Y"))

#plot(kohmap, type="mapping",  col = as.numeric(as.factor(df$deposit))+1, pch=20)
```

6 Medidas de rendimiento

Las medidas de rendimiento de un modelo de ML serán fundamentales para poder considerar que dicho modelo cumple con los requisitos establecidos al inicio del proyecto. Existe un principio fundamental en el análisis de datos que podríamos simplificar así:

$$DATOS = MODELO + ERROR$$

- Los **datos** representan la realidad (procesos de negocios, clientes, productos, actividades, fenómenos físicos, etc.) que se quiere comprender, predecir o mejorar.
- El **modelo** es una representación **simplificada** de la realidad que proponemos para describirla e interpretarla más fácilmente.
- El **error** refleja la diferencia entre nuestra representación simplificada de la realidad (el modelo) y los datos que relamente describen esa realidad de forma precisa.

Una vez elegido el modelo, o modelos de ML que vamos a emplear para analizar nuestros datos, y una vez que hemos entrenado dichos modelos sobre un conjunto de datos (empleando los algoritmos adecuados) surge la tarea de evaluar el rendimiento del modelo. Dentro del ciclo de vida de un proyecto de ciencia de datos (ver Figure 1.2b) nos encontramos en la etapa de "*Evaluar y criticar el modelo*". Podemos decir que estamos en la etapa de establecer si el *error* es aceptable o, por contra, es demasiado alto y no podemos aceptar el modelo como útil. En ese último caso, debemos buscar un modelo más preciso o que cometa un error asumible.

> **!** Para recordar
>
> En modelos de ML, se utilizan diferentes medidas de rendimiento para evaluar el comportamiento del modelo, y para llevar a cabo la comparación y evaluación de los modelos empleados.

6.1 Complejidad de un modelo

Cuando incrementamos la **información** (en forma de variables o combinación de las mismas) con la que se entrena el modelo, el error casi siempre suele reducirse, lo cual es bueno (¡a

primera vista!). Sin embargo, cuantas más variables de entrada tengamos en el modelo más complicado se vuelve. Esto no es tan bueno por dos motivos fundmentales:

- **Principio de parsimonia**: Preferimos modelos sencillos y fáciles de explicar, a modelos complicados. Este principio también es conocido como "*navaja de Occam*".
- **Pérdida de generalidad**: Si añadimos demasiados parámetros de entrada a un modelo es posible representar exactamente la información de los datos de entrenamiento, pero es muy probable que el modelo sea pésimo para nuevos datos. Es lo que se conoce como **sobreajuste** ("*overfitting*" en inglés).

George E.P. Box

"Todos los modelos están equivocados, pero algunos son útiles"

La figura siguiente muestra, con un ejemplo, los conceptos de *bajoajuste*, y *sobreajuste*:

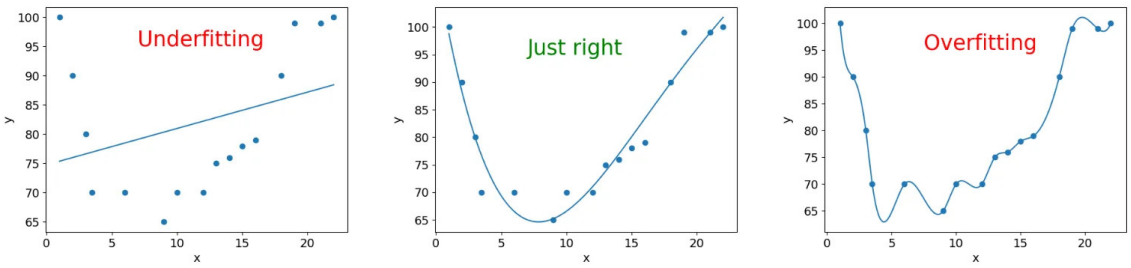


Figure 6.1: https://medium.com/(**kiprono_65591/regularization-a-technique-used-to-prevent-over-fitting-886d5b361700?**)

El fundamento de cualquier proyecto de ML es acabar con un modelo que funcione bien en datos no vistos. El **underfitting**, o bajoajuste, se produce cuando un modelo es demasiado simple para capturar la complejidad de los datos subyacentes. En otras palabras, el modelo no se ajusta lo suficiente a los datos de entrenamiento y no puede hacer predicciones precisas tanto en los datos de entrenamiento como en los datos de prueba. Esto suele ocurrir cuando se utiliza un modelo demasiado básico o se aplican suposiciones demasiado restrictivas sobre la relación entre las variables.

Por otro lado, el **overfitting**, o sobreajuste, se produce cuando un modelo es demasiado complejo y se ajusta demasiado "bien" a los datos de entrenamiento. Esto significa que el modelo no solo captura los patrones reales en los datos, sino también el ruido aleatorio o las peculiaridades únicas de los datos de entrenamiento (no generalizables). Como resultado, el modelo tiene un rendimiento excelente en los datos de entrenamiento pero un rendimiento deficiente en los datos de prueba no vistos.

6.2 Balance Sesgo-Varianza

El equilibrio entre **sesgo** ("*bias*" en inglés) y **varianza** es un concepto fundamental en ML que afecta directamente al rendimiento de los modelos. Estos dos conceptos son opuestos y encontrar el equilibrio (balance) adecuado entre ellos es esencial para construir modelos eficaces y con alta capacidad de generalización.

- **Sesgo**: El sesgo se refiere a la simplificación excesiva de un modelo, asumiendo que los datos de entrenamiento siguen una cierta estructura o patrón predefinido.
  - Un modelo con alto sesgo tiende a subajustar los datos y no captura la complejidad subyacente en los mismos.
  - Esto puede resultar en un rendimiento deficiente tanto en los datos de entrenamiento como en los datos de prueba, ya que el modelo no puede representar adecuadamente la relación entre las variables.
- **Varianza**: La varianza se relaciona con la sensibilidad excesiva de un modelo a las fluctuaciones en los datos de entrenamiento.
  - Un modelo con alta varianza se ajusta demasiado a los datos de entrenamiento, capturando incluso el ruido aleatorio en los datos.
  - Aunque puede tener un rendimiento excelente en los datos de entrenamiento, tiende a generalizar mal en nuevos datos, lo que resulta en un rendimiento deficiente en los datos de prueba.

El objetivo en el ML es encontrar un equilibrio entre estos dos extremos:

- Un modelo con **sesgo alto y varianza baja** es más simple y tiende a subajustar los datos. Puede ser adecuado cuando se dispone de pocos datos o cuando se prioriza la interpretabilidad del modelo.
- Un modelo con **sesgo bajo y varianza alta** se ajusta muy bien a los datos de entrenamiento pero generaliza mal. Puede ser útil cuando se dispone de una gran cantidad de datos y se busca la máxima precisión.
- El equilibrio adecuado se encuentra al **ajustar la complejidad del modelo y la cantidad de datos disponibles**. Esto se logra mediante técnicas como la selección de características, la regularización y la validación cruzada.

Normalmente, a medida que aumenta la complejidad del modelo, se observa una reducción del error debido a un menor sesgo del modelo. Sin embargo, esto sólo ocurre hasta cierto punto. A medida que aumente la complejidad del modelo, acabará sobreajustándolo y, por tanto, su varianza empezará a ser elevada. Por lo tanto, una disminución del sesgo del modelo aumenta la varianza y viceversa, lo que nos lleva al concepto de balance, es decir, tenemos que llegar a

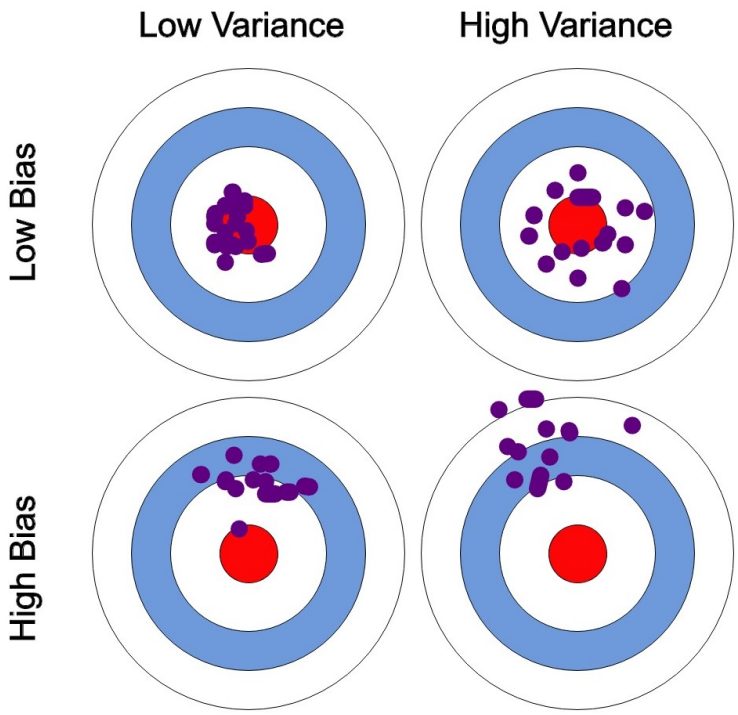


Figure 6.2: https://nvsyashwanth.github.io/machinelearningmaster/bias-variance/

un compromiso en ambos extremos. Un modelo ideal es el que tiene una varianza y un sesgo pequeños.

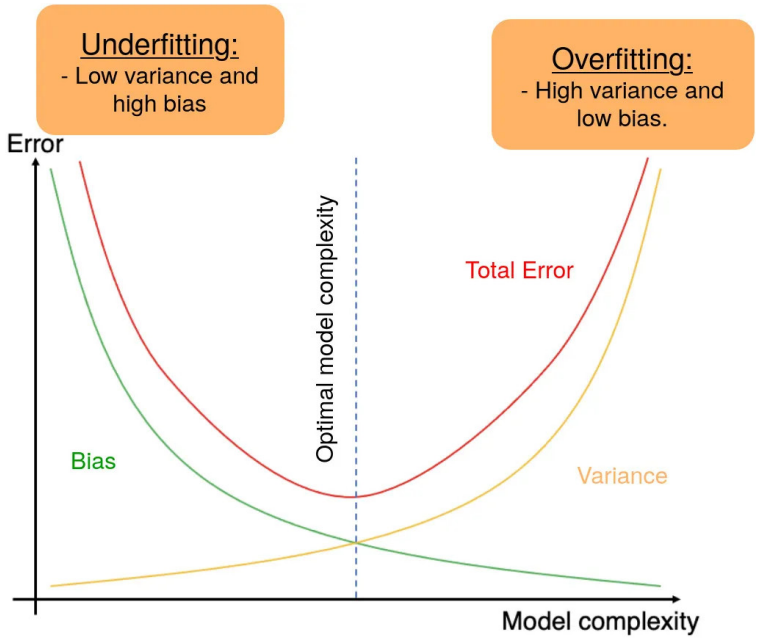


Figure 6.3: https://medium.com/(**kiprono_65591/regularization-a-technique-used-to-prevent-over-fitting-886d5b361700?**)

6.2.0.1 Evitar el sobreajuste

A continuación te explicaremos algunas técnicas para conseguir evitar el problema del sobreajuste. Despliega los paneles siguientes para obtener más información.

i Aumentar el tamaño de la muestra

Obtener más datos de entrenamiento puede ayudar a reducir el riesgo de sobreajuste. Cuanto más datos tengas disponibles, más probable es que el modelo aprenda patrones reales en lugar de ruido.

i Validación cruzada

Utiliza técnicas de validación cruzada, como la validación cruzada k-fold, para evaluar el rendimiento del modelo en diferentes subconjuntos de datos de entrenamiento y prueba. Esto puede ayudar a identificar si el modelo está sobreajustando en una sola partición

de los datos. Ver Chapter 2 para una completa descripción.

Regularización

Aplica técnicas de regularización como la regularización L1 (Lasso) y L2 (Ridge) para penalizar los coeficientes de las características menos importantes. Esto ayuda a simplificar el modelo, eliminando parámetros del modelo, y evitar que se ajuste demasiado a los datos. Es probable que estudies estas técnicas en asignaturas de cursos más avanzados.

Selección de características

Elimina características irrelevantes o redundantes que no contribuyan significativamente a la capacidad predictiva del modelo. Una menor dimensionalidad puede reducir el riesgo de sobreajuste.

Modelos más simples

Considera modelos más simples con menos parámetros, como regresiones lineales en lugar de modelos polinómicos complejos. Los modelos simples tienden a tener menos probabilidad de sobreajuste.

Dropout

En modelos de redes neuronales, el *dropout* es una técnica que consiste en apagar aleatoriamente una fracción de las neuronas durante el entrenamiento. Esto evita que las neuronas se vuelvan demasiado dependientes entre sí y reduce el riesgo de sobreajuste.

Partición de los datos

Tal y como te explicamos en el Chapter 2, divide tus datos en tres conjuntos: entrenamiento, validación y prueba. Utiliza el conjunto de validación para ajustar los hiperparámetros del modelo y el conjunto de prueba solo para la evaluación final.

Seguimiento del rendimiento

Controla regularmente el rendimiento del modelo en el conjunto de prueba durante el entrenamiento. Si el rendimiento en el conjunto de prueba comienza a empeorar mientras mejora en el conjunto de entrenamiento, es una señal de posible sobreajuste.

Ensemble Learning

Combina múltiples modelos juntos, como *Random Forests* o *Gradient Boosting*, que pueden reducir el riesgo de sobreajuste al promediar las predicciones de varios modelos. Trataremos estas técnicas en el Chapter 7.

6.3 Métricas de evaluación

Desde un punto de vista práctico, dado un modelo de ML entrenado mediante un algoritmo sobre un conjunto de datos, nuestra primera pregunta va a ser *¿cómo de eficaz es el modelo para representar los datos*?

En otras palabras, la primera consideración a tener en cuenta sobre el rendimiento de un modelo de ML es si la estimación del error es precisa. Hemos de considerar si los estimadores de los parámetros del modelo son precisos, y si todas las variables empleadas para su construcción son necesarias y suficientes. Deberemos comparar nuestro modelo, en sus diferentes versiones (resultantes de diferentes valores de los parámetros), con otros modelos alternativos. Así mismo, planteamos la evaluación del modelo sobre conjuntos de datos diferentes a aquellos sobre los que ha sido entrenado, para medir su capacidad de generalización. Tal y como venimos indicando, hemos de tener en cuenta que el modelo de ML se ajusta sobre un conjunto de datos de entrenamiento. Al ser probado sobre nuevos conjuntos de datos (prueba y validación), los resultados serán necesariamente diferentes. Tal y como hemos visto anteriormente, si el modelo se ajusta con excesiva precisión sobre los datos de entrenamiento, corremos el riesgo de sobreajuste. Por contra, si el modelo es demasiado simple, el error posiblemente sea elevado y el modelo, por tanto, de baja utilidad. Buscaremos un equilibrio, modelos parsimoniosos, con baja complejidad, pero alto rendimiento, y gran capacidad de generalización.

Para evaluar la bondad de ajuste de un modelo, vamos a diferenciar entre modelos de clasificación y modelos de regresión. En cualquier caso, para una observación concreta, el error siempre vendrá determinado por la diferencia (medida mediante alguna métrica) entre el valor observado de la variable respuesta en la observación y el valor estimado (o predicho) por el modelo de aprendizaje.

$$error_i \propto targetobservado_i - targetpredicho_i$$

6.3.1 Métricas para Regresión

Un modelo de regresión es una técnica estadística que se utiliza para analizar la relación entre una variable dependiente (respuesta) y una o más variables independientes (predictores). El objetivo principal de un modelo de regresión es comprender cómo las variables independientes

afectan o explican la variabilidad en la variable dependiente y, en última instancia, utilizar esta comprensión para hacer predicciones o inferencias. La diferencia entre las predicciones para los datos observados en la muestra de entrenamiento y el propio valor de la variable dependiente en dichos datos será una medida del error.

En el contexto de modelos de regresión, existen varias métricas comunes para evaluar el error del modelo y determinar su calidad de ajuste a los datos. Algunas de las métricas más utilizadas incluyen las siguientes:

- **Error Cuadrático Medio (Mean Squared Error, MSE)**: El MSE es una métrica que calcula el promedio de las diferencias al cuadrado entre las predicciones del modelo y los valores reales. Es especialmente sensible a los errores grandes debido al término de cuadrado. Cuanto menor sea el MSE, mejor será el ajuste del modelo:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$$

,

donde y_i es el valor de la variable objetivo en la observación x_i con $i = 1, ..., N$, y $f(x_i)$ es el valor predicho por el modelo de ML que se ha entrenado.

- **Error Absoluto Medio (Mean Absolute Error, MAE)**: El MAE es similar al MSE, pero en lugar de cuadrar los errores, toma el valor absoluto de cada error y calcula el promedio. Es menos sensible a los errores extremos en comparación con el MSE.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - f(x_i)|$$

.

- **Raíz del Error Cuadrático Medio (Root Mean Squared Error, RMSE)**: El RMSE es, simplemente, la raíz cuadrada del MSE. Ofrece una medida del error en la misma unidad que la variable objetivo, lo que facilita su interpretación.

- **Coeficiente de Determinación (R-squared, R^2)**: El R^2 es una métrica que proporciona una medida de la proporción de la variabilidad en la variable dependiente que es explicada por el modelo. Un R^2 más alto indica un mejor ajuste del modelo a los datos, con un valor máximo de 1.

$$R^2 = \frac{\sum_{i=1}^{N}(f(x_i) - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

,

donde $\bar{y}$ es el valor medio de la variable objetivo.

- **Error Porcentual Absoluto Medio (Mean Absolute Percentage Error, MAPE)**: El MAPE calcula el porcentaje promedio de error absoluto en relación con los valores reales. Es útil para comprender el error relativo en lugar del error absoluto.

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}|\frac{y_i - f(x_i)}{y_i}|$$

.

La elección de la métrica adecuada depende del tipo de problema que se esté abordando y de los objetivos específicos de modelado. Por ejemplo, si estás interesado en comprender la precisión de las predicciones en términos absolutos, elige métricas como MSE o MAE. Si prefieres evaluar el rendimiento relativo, considera métricas como R^2 o MAPE. Es importante seleccionar la métrica que mejor se alinee con tus necesidades y contexto de aplicación.

6.3.2 Métricas para Clasificación

Tal y como veremos en el Chapter 7, los modelos de clasificación son una parte importante de las técnicas de ML. El objetivo de dichos modelos es hacer una división perfecta en diferentes clases, previamente definidas en base a las etiquetas de la variable respuesta. En este capítulo nos centramos en la más popular de todos los tipos de clasificación: la clasificación binaria. De hecho, cuando nos enfrentamos a un problema de clasificación multiclase una de las soluciones más habituales es convertirlo en varios problemas de clasificación binaria usando una de las clases como clase de referencia.

El propósito de la clasificación binaria es dividir las observaciones dadas en dos clases mutuamente excluyentes *{-1,+1}*. La elección de la métrica de evaluación del rendimiento de un modelo es relevante, ya que se utilizará para seleccionar o descartar los modelos de clasificación. En general, este conjunto de métricas se obtiene de la matriz de confusión que enfrenta las clases observadas y las predicciones realizadas por el modelo de ML.

Definimos la matriz de confusión asociada a los valores predichos por un clasificador como:

		Valor observado	
		-1	+1
Valor Predicho	-1	TN	FN
	+1	FP	TP

Los elementos de la matriz de confusión son:

- **TP**: "True positive" o verdaderos positivos, corresponde con las observaciones 1 clasificadas efectivamente como 1.

- **TN**: "True negative" o verdaderos negativos, corresponde con las observaciones -1 clasificadas efectivamente como -1.
- **FP**: "False positive" o falsos positivos, corresponde con las observaciones -1, que son erróneamente clasificadas como 1.
- **FN**; "False negative" o falsos negativos, corresponde con las observaciones 1, que son erróneamente clasificadas como -1.

Es muy importante recalcar que la importancia relativa de los dos errores (FP y FN) depende del problema que estemos considerando. Por ejemplo, en un estudio de control de accesos, un FP significa dejar entrar a una persona que debería de haber sido detenida, mientras que un FN significa no dejar entrar a una persona a la que se debería de haber permitido el paso. Si estamos en un entorno de seguridad restringida, es claro que el error FP es mucho más crítico que el error FN. Sin embargo, la respuesta sobre qué error es más relevante no es única y depende fuertemente del contexto.

> **!** Para pensar
>
> Por ejemplo, ¿qué es más costoso para un sistema de detección de intrusiones, levantar falsas alarmas al detectar como anomalías situaciones que no lo son, o clasificar como normal una anomalía potencialmente peligrosa?

A continuación, definimos las métricas de rendimiento más importantes en base a la matriz de confusión, siendo N el número total de observaciones consideradas, es decir $N = TP + TN + FP + FN$:

Exactitud: probablemente la exactitud es una de las medidas de rendimiento más comunes. Representa la proporción de observaciones correctamente predichas, es decir:

$$Exactitud = \frac{TP + TN}{N}$$

Error: recíproco de la exactitud:

$$Error = \frac{FP + FN}{N}$$

Sensibilidad: también conocida como Recuperación o Tasa de Verdaderos Positivos puede verse como la probabilidad de que un 1 observado sea clasificado efectivamente como 1:

$$Recuperacin = \frac{TP}{TP + FN}$$

Especificidad: también conocida Tasa de Verdaderos Negativos puede verse como la probabilidad de que un -1 observado sea clasificado efectivamente como -1:

$$Especificidad = \frac{TN}{TN + FP}$$

Precisión: también conocida Valor Predictivo Positivo puede verse como la probabilidad de que acierto cuando se predice un valor 1:

$$Precisin = \frac{TP}{TP + FP}$$

Valor Predictivo Positivo (NPV, del inglés "Negative Predictive Value"): tasa de acierto cuando se predice un valor -1:

$$NPV = \frac{TN}{TN + FN}$$

F1-score: media armónica de Precisión y Recuperación:

$$F_1 - score = 2\frac{Precisin * Recuperacin}{Precisin + Recuperacin}$$

Es una medida que tiene en cuenta tanto el acierto cuando se predice un valor 1, como el número de observaciones con etiqueta igual a 1 que son correctamente predichas. Sin embargo, no se tienen en cuenta el número de aciertos en la clase -1. Es decir, es una medida que no involucra a los verdaderos negativos (TN). Esta limitación puede ser una ventaja en algunos casos. Sin embargo, en la práctica, los costes de clasificación errónea no suelen ser iguales. En ese caso podemos emplear la siguiente medida de la lista.

F-score generalizado: media armónica ponderada de Precisión y Recuperación:

$$F_\beta = (1 + \beta^2)\frac{Precisin * Recuperacin}{\beta^2 Precisin + Recuperacin}$$

Cuando $\beta = 1$, se tiene la medida anterior: $F_1 - score$. F_2 da el doble de peso a la Recuperación que a la Precisión. Por contra, $F_{0.5}$ da el doble de peso a la Precisión que a la Recuperación.

6.4 Rendimiento en las particiones

Tal y como se indicó en el Chapter 2, existen diversas formas de particionar los datos para llevar a cabo las labores de entrenamiento, prueba y validación. En este apartado nos vamos a centrar en realizar la evaluación de un modelo en las diferentes particiones. Para ello vamos a emplear el método de "*k-folds*". La muestra de entrenamiento será dividida en k grupos de entrenamiento. Cada uno de esos grupos queda fuera del entrenamiento una vez y se emplean las observaciones en el grupo para estimar el error. De este modo, se dispone de k errores, medidos en la muestra de entrenamiento. Podemos calcular una media de los errores, junto con su desviación típica.

6.5 Ajuste de parámetros de modelos de ML

Es común que el científico de datos necesite ajustar los parámetros de más alto nivel de los modelos de ML. Por ejemplo, en una SVM es necesario elegir el kernel y sus parámetros (aprenderás esta técnica de clasificación en asignaturas del próximo curso). En el método de los k vecinos más cercanos se necesita fijar el número k. En una Regresión Logística, el científico de datos ha de fijar los umbrales de significatividad estadística a partir de los cuales una variable debería (o no) formar parte del modelo. En un Árbol de Decisión, se necesita fijar varios parámetros, como, por ejemplo, la profundidad del árbol, o el número mínimo de observaciones para que un nodo sea considerado como un nodo terminal (lo trataremos en el próximo capítulo de estos apuntes).

Sea como fuere, para ajustar los parámetros de un modelo de ML debemos seleccionar la medida de rendimiento sobre la que comparar dos modelos diferentes. A modo de ejemplo, supongamos un modelo de clasificación basado en los k vecinos más cercanos, esto es con un único parámetro k. En este caso, el error global es la métrica elegida. Podríamos replicar el experimento de clasificación cambiando el número de vecinos. De este modo, los diferentes modelos de ML corresponden a modelos de k vecinos más cercanos con diferentes elecciones del parámetro k. La tabla siguiente muestra los resultados para valores de k entre 1 y 11:

Número de vecinos k	Media (Desv. Std.)
1	1,54 (0,33)
2	1,75 (0,41)
3	1,68 (0,28)
4	1,68 (0,35)
5	1,70 (0,34)
6	1,89 (0,38)
7	1,91 (0,36)
8	2,12 (0,45)
9	2,25 (0,46)
10	2,34 (0,34)
11	2,33 (0,38)

En este caso elegimos $k = 1$ como parámetro para el modelo de los k-vecinos más cercanos al ser el valor asociado con el menor error. Es decir, el modelo quedaría como sigue. El modelo, que veremos en el siguiente tema, funciona como sigue: para cada nueva observación calculamos su vecino más cercano en la muestra de entrenamiento y asignamos a la nueva observación la clase de dicho vecino.

Podemos incluso llevar esta estrategia a su extremo mediante la técnica de "*leave-one-out*", que consiste en hacer tantos folds como observaciones. En ese caso obtenemos errores similares. También podemos reducir el número de folds buscando mayor número de observaciones en cada

uno de ellos, reduciendo de este modo la varianza en la media de los errores. Por ejemplo, con 5 folds los errores, para diferentes elecciones de k vecinos, son:

Número de vecinos k	Media (Desv. Std.)
1	1,59 (0,23)
2	1,87 (0,32)
3	1,78 (0,16)
4	1,84 (0,12)
5	1,86 (0,25)
6	2,05 (0,25)
7	2,06 (0,36)
8	2,34 (0,20)
9	2,33 (0,19)
10	2,36 (0,17)
11	2,45 (0,20)

Vemos que, aunque la media de los errores es menor en $k = 1$, una elección como $k = 3$, o $k = 4$ podría tener más sentido al obtener resultados similares con menor variabilidad (menos desviación estándar).

A continuación, deberíamos de probar el modelo en la muestra de prueba (test). Por ejemplo, con $k = 1$ vecino más cercanose obtiene un error similar al esperado, $1,51\%$. Nótese que en este caso no hay desviación típica, pues es el error en una única partición. Cuando probamos el modelo elegido ($k = 3$) el resultado sí es algo mejor: $1,47\%$. Normalmente el modelo del primer vecino más cercano no presenta un buen equilibrio entre el sesgo y la varianza, y elecciones de $k > 1$ suelen ser preferibles (Cover and Hart 1967).

Entonces, cabe plantearse la duda ¿*cuál es la solución óptima para este problema*? La respuesta es que depende del objetivo y, en este caso, cualquier elección entre 1 y 3 vecinos parece razonable. Fíjate en la importancia de la medida de distancia elegida. El primer vecino más cercano, es el más cercano según esa media. Una medida diferente conduciría, probablemente, a otro vecino y en consecuencia a otros resultados de rendimiento.

Supongamos que elegimos un modelo de k vecinos más cercanos con k igual 3. En estos momentos, estaríamos en disposición de llevar el modelo a "*producción*". Es decir, si este fuera nuestro modelo elegido habríamos comprobado su buen funcionamiento en datos diferentes a aquellos sobre los que ha sido entrenado. El error esperado para nuevos datos está en torno al $1,5\%$. Recordemos que aún hemos reservado una partición (validación) para simular este error. La partición de validación, no obstante, sólo debería de ser empleada con un modelo, sólo uno.

! Para recordar

- **Entrenamos** los modelos de ML en la partición de **entrenamiento**.
- **Probamos** que el funcionamiento de los modelos elegidos en el entrenamiento es el correcto en la partición de **prueba**.
- **Simulamos** el comportamiento en un entorno real del **único** modelo elegido en la partición de **validación** o producción.

6.6 Comparación de modelos

La evaluación del modelo de clasificación realizada en los apartados anteriores puede ser extrapolada al resto de modelos de ML que veremos a lo largo del curso. De este modo, tendremos un montón de modelos con un montón de medidas de evaluación (preferiblemente la misma en todos ellos). Nos encontramos con el problema de comparar diferentes modelos de aprendizaje.

Muchos de estos modelos devuelven como salida, para cada observación, la **probabilidad de pertenencia** a las diferentes clases de la variable respuesta. Por ejemplo, una SVM devuelve la probabilidad de pertenencia según una ecuación que depende de la distancia del nuevo punto al hiperplano separador óptimo. Dada esa probabilidad de clase, podemos asignar las nuevas observaciones a una u otra clase. En el caso de los k vecinos más cercanos, la probabilidad de pertenencia a uno u otro grupo viene dada por la distribución de frecuencias de las clases de la variable respuesta en los k vecinos más cercanos. Así, si los k vecinos pertenecen a la misma clase, la probabilidad de pertenencia predicha será 1. Si dos tercios de los vecinos pertenecen a la misma clase, la probabilidad de pertenencia a dicha clase será de $0,667$, etc.

Lo habitual en problemas de clasificación donde se tienen clases equilibradas (frecuencia parecida de observaciones en las diferentes clases) es tomar como valor de referencia para comparar la probabilidad dada por el modelo el valor $0,5$. Es decir, si la probabilidad de clase para una nueva observación es mayor que $0,5$, entonces el modelo predice clase $+1$, y clase -1 en caso contrario. Pero ese valor, $0,5$, no es sino otro parámetro de nuestro modelo.

Fíjate que para valores bajos del umbral habrá muchas observaciones clasificadas como $+1$, probablemente más de las debidas. De este modo, se obtienen valores altos de *Sensibilidad* y *NPV*. Es decir, se recuperan la mayoría de las observaciones en la clase $+1$, pero la *Precisión* no es tan alta como cuando usamos valores altos de umbral. Por contra, cuando se usan valores altos, habrá pocas observaciones clasificadas como -1, probablemente menos de las debidas. De este modo, se obtienen valores más elevados de *Precisión* y *Especificidad*, pero más bajos valores de *Sensibilidad*.

6.6.1 Curva ROC

Existe un método gráfico para ilustrar la capacidad predictiva de un modelo de Aprendizaje Máquina binario. Se trata de la **curva ROC** (del inglés "*Receiver Operating Characteristic curve*", o curva característica de operación).

> **Curva ROC**
>
> Para dibujar la curva ROC se enfrentan la Sensibilidad o Recuperación (TPR), frente a 1 - Especificidad, o tasa de falsos positivos (FPR).

Un ejemplo de una curva ROC aparece en la figura Figure 6.4. Los valores de *False Positive Rate* y *True Positive Rate* de cada uno de los puntos de la curva se obtienen de las correspondientes matrices de confusión resultantes de modificar el umbral de decisión de clasificadores binarios. Es decir, se construye un modelo de clasificación que devuelve una probabilidad de clase para cada nuevo punto evaluado. Se establece un umbral. Se obtiene, para la partición correspondiente, la matriz de confusión. Se calculan las métricas de rendimiento y se grafica el punto. Este proceso se repite para una colección de umbrales, uniendo los puntos resultantes de la gráfica. De este modo se consigue la curva ROC.

Es claro que la mejor solución sería aquella que tiene FPR igual a 0 y TPR igual a 1. En un caso tan ideal, no hay errores de clasificación, la Precisión es perfecta y la Sensibilidad también. Se corresponde con una matriz de confusión diagonal. En el ejemplo presentado en la figura anterior vemos como el modelo de clasificación propuesto no alcanza un rendimiento tan alto en ninguno de los umbrales de decisión.

La curva ROC nos sirve, no sólo para elegir el valor óptimo (el más cercano al ùnto dideal $(0, 1)$), sino para elegir en qué modo de funcionamiento ajustar nuestro modelo. Podemos emplear un modelo con muy alta recuperación (TPR), sacrificando la FPR (baja especificidad), o bien decantarnos por un modelo con baja recuperación y alta especificidad. Lo ideal sería contar con un modelo con valores altos de las dos métricas.

Así mismo, es típico proporcionar el área bajo la curva (**AUC**, de "Area Under Curve") como medida resumen de la curva ROC. Un valor de AUC cercano a 1 indica un mejor modelo. Valores de AUC próximos a $0,5$ indican una predicción cercana al azar.

En ocasiones se grafican las curvas ROC de varios modelos simultáneamente con el objeto de realizar comparaciones entre modelos. En principio deberías de elegir aquel modelo con mayor AUC. Sin embargo, no es extraño emplear varios modelos de clasificación en función de la configuración del sistema a predecir. Hay ocasiones en las que las curvas se cruzan. Es decir, una está por encima de la otra en una zona del espacio, y por debajo en otra zona diferente. Si deseamos un modelo con muy alta Especificidad a costa de sacrificar la Sensibilidad, esto es, un modelo que recupere correctamente las observaciones -1 a costa de fallar en algunas de las $+1$, entonces nos quedaremos con un modelo A. Por contra, si

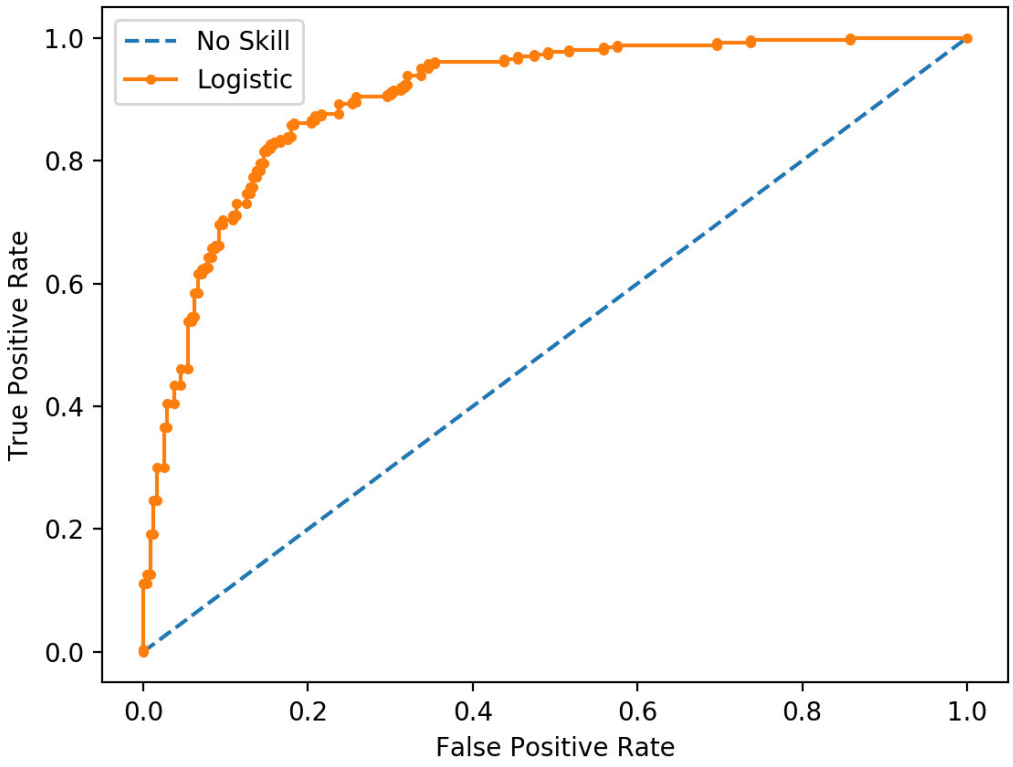


Figure 6.4: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

deseamos un modelo con muy alta Especificidad a costa de sacrificar la sensibilidad, esto es, un modelo que recupere correctamente las observaciones $+1$ a costa de fallar en algunas de las -1, entonces nos quedaremos con un modelo B. El hecho de emplear dos modelos (o más) de ML según el contexto, puede sonar un poco confuso para el científico de datos más novel, pero es algo que, desde un punto de vista meramente matemático, tiene todo el sentido del mundo. A fin de cuentas, se trata de establecer, bajo unas circunstancias determinadas (el contexto), la probabilidad de ocurrencia de un evento. Emplear métodos diferentes según esas circunstancias es algo bastante lógico.

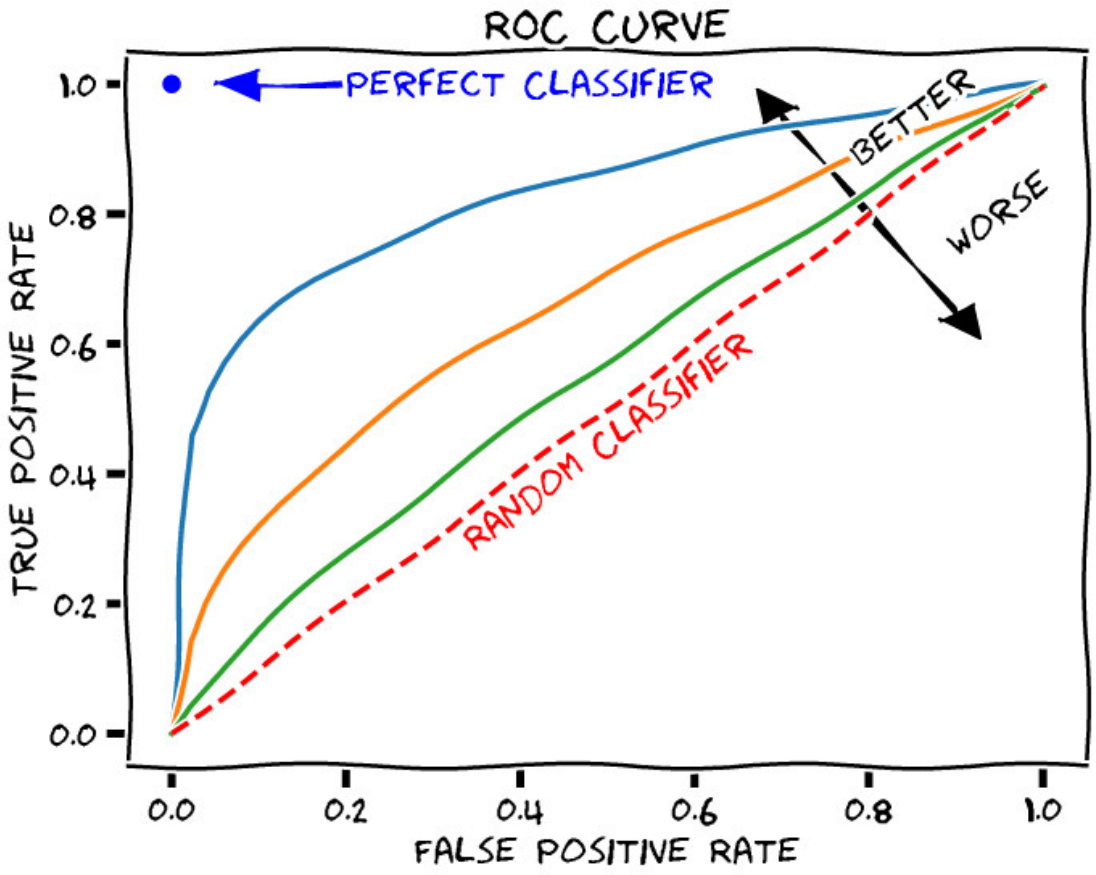


Figure 6.5: https://sefiks.com/2020/12/10/a-gentle-introduction-to-roc-curve-and-auc/

7 Aprendizaje supervisado

El aprendizaje automático supervisado es una disciplina fundamental en la ciencia de datos y la IA. En este enfoque, trabajamos con un conjunto de datos en el que conocemos la variable objetivo que deseamos predecir o clasificar. Esta variable objetivo contiene la información que queremos explicar o entender, y su comportamiento se basa en el resto de las variables o características del conjunto de datos. El objetivo principal del ML supervisado es desarrollar modelos que puedan capturar patrones y relaciones entre las variables con el fin de realizar predicciones precisas y clasificar observaciones.

Uno de los aspectos más destacados del ML supervisado es su capacidad para resolver problemas de clasificación, donde se asignan observaciones a diferentes categorías o clases. Por ejemplo, podemos aplicar modelos de clasificación para predecir si un correo electrónico es spam o legítimo, si un paciente tiene una enfermedad específica o no, o si una transacción bancaria es fraudulenta o no.

Además de la clasificación, el ML supervisado también se utiliza para abordar problemas de regresión, donde la variable objetivo es una cantidad continua. Esto nos permite realizar predicciones numéricas, como predecir el precio de una casa en función de sus características, pronosticar la demanda de productos en función de variables de mercado, o estimar la duración de un proyecto en función de factores diversos.

En este tema, exploraremos diversos algoritmos de clasificación supervisada que permiten extraer patrones a partir de los datos y realizar predicciones precisas en una variedad de aplicaciones. Aprenderemos cómo entrenar modelos, evaluar su rendimiento y aplicarlos a situaciones del mundo real. El ML supervisado es una herramienta poderosa que puede proporcionar conocimientos valiosos y respuestas a preguntas importantes en una amplia gama de dominios.

7.1 Modelos Lineales

Los modelos lineales representan uno de los pilares más robustos y extensamente empleados en el ámbito del aprendizaje supervisado en ML. Estos modelos constituyen la base de numerosas aplicaciones de predicción y clasificación, desempeñando un papel fundamental en la comprensión de los principios subyacentes de los algoritmos de ML.

La premisa fundamental de un modelo lineal es la suposición de que existe una relación lineal entre las variables de entrada (características) y la variable objetivo que se desea predecir. En otras palabras, se busca identificar una combinación lineal de las características ponderadas por coeficientes que se ajuste óptimamente a los datos observados. Los modelos lineales destacan por su alta interpretabilidad, lo que facilita un análisis claro sobre cómo cada característica influye en la variable objetivo.

En este capítulo exploraremos los modelos lineales, iniciando con los modelos lineales generalizados, entre los cuales el modelo de regresión lineal representa un caso específico. Además, examinaremos modelos lineales de clasificación, como la regresión logística, que se emplean para abordar problemas de clasificación.

7.1.1 Modelos lineales generalizados

Los Modelos Lineales Generalizados (**GLM**, de "*Generalized Linear Models*") son una extensión poderosa de los modelos lineales tradicionales que amplían su aplicabilidad a una variedad de problemas más complejos en el campo del ML. Si necesitáis una referencia completa a este tipo de modelos: (McCullagh 2019).

Estos modelos son aplicables tanto a variables respuesta cuantitativas como cualitativas. De hecho, a diferencia de los modelos lineales simples, los GLM permiten manejar una amplia gama de situaciones, incluyendo variables de respuesta que no siguen una distribución normal, relaciones no lineales entre las características y la variable objetivo, y variables categóricas. Los GLM se basan en tres componentes clave:

- la función de enlace,
- la distribución de probabilidad,
- la estructura lineal.

Llamamos y a la variable respuesta. Además, se define un predictor lineal con la combinación lineal ponderada de las p variables explicativas $x_1, x_2, ..., x_p$:

$$\eta(x) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p.$$

La **función de enlace** conecta la media de la variable respuesta con la combinación lineal de las características (el predictor lineal), lo que permite modelar relaciones no lineales y capturar la variabilidad de los datos. La elección de la función de enlace depende del tipo de problema que se esté abordando y puede incluir funciones como la logística para problemas de clasificación o la identidad para problemas de regresión.

La **estructura lineal** implica que las características se ponderan por coeficientes, similar a los modelos lineales tradicionales, pero con la capacidad de ajustar relaciones no lineales y manejar múltiples predictores, incluyendo variables categóricas.

Repaso

Quizás es un buen momento para repasar los modelos lineales (regresión) que hayas visto en otras asignaturas del grado.

Definimos una función de enlace que relaciona el valor esperado de la variable respuesta dadas las variables explicativas $E(y|x) = \mu(x)$, con el predictor lineal:

$$g(E(y|x)) = \eta(x)$$

La **distribución de probabilidad** (componente aleatoria) describe cómo se distribuyen los datos alrededor de la media y puede adaptarse a diferentes tipos de datos, como datos binarios, Poisson, binomiales o exponenciales, entre otros. Esto brinda flexibilidad para modelar una variedad de escenarios de datos. Por ejemplo, en el caso de la conocida *regresión lineal* se asume que la variable respuesta sigue una distribución *Normal*. Mediante la componente aleatoria determinamos cómo añadir el ruido o error aleatorio a la predicción que se obtiene de la función de enlace. La tabla siguiente, adaptada de (Agresti 2015), resume los GLM:

Modelo	Componente aleatoria	Función de enlace	Tipo de variables explicativas
Regresión Lineal	Normal	Identidad	Cuantitativas
ANOVA	Normal	Identidad	Cuantitativas
ANCOVA	Normal	Identidad	Cuantitativas y Cualitativas
Regresión Logística	Binomial	Logit	Cuantitativas y Cualitativas
LogLinear	Poisson	Log	Cualitativas
Regresión de Poisson	Posisson	Log	Cuantitativas y Cualitativas

Para recordar

Los GLM no asumen una relación lineal entre la variable respuesta y las variables explicativas (o independientes), pero sí suponen una relación lineal entre la variable respuesta transformada en términos de la función de enlace y las variables explicativas.

7.1.2 Regresión Logística

El modelo de Regresión Logística es uno de los modelos lineales generalizados más empleado en la práctica por la facilidad en la interpretación de sus resultados. Es un modelo adecuado para modelar la relación entre una variable respuesta binaria (0 o 1) y un conjunto de variables explicativas cuantitativas y/o cualitativas.

> **Multiclase**
>
> Existe una versión de la Regresión Logística adaptada al caso de una variable respuesta con más de dos categorías, que recibe el nombre de Regresión Logística Politómica o Multinomial.

Asumiremos que la variables explicativas son independientes entre sí y que la probabilidad de 1 (a veces llamada probabilidad de éxito) solo depende de los valores de dichas variables. Si las variables explicativas no fueran independientes, podríamos añadir interacciones al modelo.

Tratamos de modelar la probabilidad condicionada, $\mu(x)$, de que la variable respuesta sea un 1 dados los valores de las variables explicativas. En este caso, la componente de la variables respuesta se corresponde con una distribución binomial, $Binomial(n, \pi)$, donde $\pi = \mu(x)$ es la probabilidad de éxito. Así,

$$\begin{gathered} g(E(y|x) = g(\mu(x)) = \eta(x) = \\ = logit(\mu(x)) = log\left(\frac{\mu(x)}{(1-\mu(x))}\right) = log\left(\frac{E(y|x)}{1-E(y|x)}\right) \\ = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p \end{gathered}$$

que modela la log odds de probabilidad de 1 como función de las variables explicativas. Esto es, se modela el logaritmo de la probabilidad de 1 dadas las variables explicativas, frente a la probabilidad de 0 dadas las variables explicativas.

De este modo:

$$\mu(x) = \frac{1}{1 + exp(-\eta(x))} = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p))}$$

Se emplean métodos de Máxima Verosimilitud para la estimación de los parámetros $\beta_i, i = 0, ..., p$. Es posible realizar contrastes de hipótesis sobre dichos parámetros, tratando de eliminar del modelo las variables no significativas. Es decir, se contrasta si en los datos hay suficiente información contraria a la siguiente hipótesis nula, como para rechazarla:

$$H_o : \beta_i = 0$$

La interpretación de los coeficientes se realiza mediante una razón de ventajas ("*odds ratio*" en inglés), tomando la exponencial de los estimadores de los parámetros del modelo.

Con los parámetros estimados, y sus desviaciones típicas, podremos hacer selección de variables, quedándonos con las más significativas (aquellas con un $p-valor$ por debajo de un umbral).

Vamos a trabajar con el ejemplo de los bancos que se estudió en el Chapter 3. Como primera aproximación a la regresión logística, aplicamos un modelo con una única variable explicativa. En este caso elegimos la variable `housing`:

```
#Lectura de datos

library(tidyverse)
bank = read.csv('https://raw.githubusercontent.com/rafiag/DTI2020/main/data/bank.csv')
dim(bank)
```

```
[1] 11162    17
```

```
bank=as.tibble(bank)

# Parciticionamos los datos
set.seed(2138)
n=dim(bank)[1]
indices=seq(1:n)
indices.train=sample(indices,size=n*.5,replace=FALSE)
indices.test=sample(indices[-indices.train],size=n*.25,replace=FALSE)
indices.valid=indices[-c(indices.train,indices.test)]

bank.train=bank[indices.train,]
bank.test=bank[indices.test,]
bank.valid=bank[indices.valid,]

# Estudiamos la influencia de housing en deposit
tabla1=xtabs(~housing+deposit,data=bank.train)
tabla1
```

```
       deposit
housing   no  yes
    no  1246 1688
    yes 1662  985
```

```r
chisq.test(tabla1)
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  tabla1
X-squared = 229.44, df = 1, p-value < 2.2e-16
```

Podemos observar que existe una relación relevante entre ambas variables. El $p - valor$ es, claramente, inferior al valor de referencia 0.05. Empleamos la regresión logística para cuantificar la relación:

```r
factor.deposit <- factor(bank.train$deposit)
logit1 <- glm(factor.deposit ~ housing, data = bank.train, family = "binomial")
summary(logit1)
```

```
Call:
glm(formula = factor.deposit ~ housing, family = "binomial",
    data = bank.train)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.30361    0.03735   8.129 4.34e-16 ***
housingyes  -0.82674    0.05488 -15.064  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7727  on 5580  degrees of freedom
Residual deviance: 7495  on 5579  degrees of freedom
AIC: 7499

Number of Fisher Scoring iterations: 4
```

¿Qué vemos en la salida? En primer lugar, los 3 asteriscos (" *** ") en la fila correspondiente a la variable `housing` indican su alta significatividad estadística. En otras palabras, es una variable muy relacionada con la respuesta (como ya habíamos averiguado). El valor del

parámetro β_1 es de -0.82674 con una error estándar de 0.05488. ¿Qué significa? Es complicado interpretar valores negativos en los coeficientes. Para una mayor comprensión vamos a darle la vuelta. Es decir, el valor -0.82674 está asociado a"*housing=yes*", de modo que el valor 0.82674 está asociado a "*housing=no*". La interpretación se realiza en base al exponente de dicho valor tal y como sigue:

$$exp(.82674) = 2.29$$

significa que el "riesgo" de contratar un depósito es 2.29 veces mayor en aquellos que no tienen casa, en relación con el mismo riesgo para aquellos que sí tienen casa en propiedad. Si hubíeramos mantenido la interpretación original (con el valor del coeficiente negativo), el resultado sería: "el riesgo de contratar un depósito es $exp(-0.82674) = 0.44$ veces menor en aquellos propietarios de su propia casa, respecto a otros clientes que no tienen casa. Como puede verse es mucho más sencillo interpretar coeficientes positivos (odds ratio mayor que 1) que coeficientes negativos.

En base al valor del parámetro y a su error, podemos obtener intervalos de confianza:

```
exp(cbind(OR=coef(logit1),confint.default(logit1)))
```

```
                   OR     2.5 %   97.5 %
(Intercept) 1.3547352 1.2591064 1.457627
housingyes  0.4374726 0.3928586 0.487153
```

```
confint.default(logit1)
```

```
                2.5 %     97.5 %
(Intercept)  0.2304023  0.3768097
housingyes  -0.9343056 -0.7191770
```

Y por tanto, el riesgo de contratar un depósito es $2.29 \in [2.05, 2.54]$ veces mayor en aquellos que no tienen casa, en relación con el mismo riesgo para aquellos que sí tienen casa en propiedad.

Vamos ahora a trabajar en el caso multivariante. Como primer paso estudiemos, una a una las variables que se van a incluir en el modelo. A modo de ejemplo, consideramos las siguientes variables:

- `housing`
- `marital`
- `education`
- `impago`

- saldo

```
# para que la categoría de referencia sea "married"
bank.train$marital=relevel(as.factor(bank.train$marital),ref=2)

logit2 <- glm(factor.deposit ~ marital, data = bank.train, family = "binomial")
summary(logit2)
```

```
Call:
glm(formula = factor.deposit ~ marital, family = "binomial",
    data = bank.train)

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.24428    0.03575  -6.832 8.35e-12 ***
maritaldivorced  0.19139    0.08662   2.209   0.0271 *
maritalsingle    0.43555    0.05974   7.290 3.09e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7727.0  on 5580  degrees of freedom
Residual deviance: 7673.4  on 5578  degrees of freedom
AIC: 7679.4

Number of Fisher Scoring iterations: 3
```

La variable `marital` es estadísticamente significativa. Los clientes solteros tienen un riesgo del orden de 1.5 veces mayor de `deposit=yes` respecto a los clientes casados. Los clientes divorciados tienen un riesgo del orden de 1.2 veces mayor.

```
logit2 <- glm(factor.deposit ~ education, data = bank.train, family = "binomial")
summary(logit2)
```

```
Call:
glm(formula = factor.deposit ~ education, family = "binomial",
    data = bank.train)
```

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.42876    0.07457  -5.750 8.92e-09 ***
educationsecondary  0.25901    0.08385   3.089 0.002008 **
educationtertiary   0.59332    0.08820   6.727 1.73e-11 ***
educationunknown    0.48005    0.14220   3.376 0.000736 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7727  on 5580  degrees of freedom
Residual deviance: 7671  on 5577  degrees of freedom
AIC: 7679

Number of Fisher Scoring iterations: 4
```

La variable `education` es estadísticamente significativa. A medida que la educación aumenta, los clientes tienen un riesgo mayor de `deposit=yes` (1 categoría de referencia, 1.3 educación secundaria, 1.8 educación terciaria). Los clientes con nivel de educación desconocido también tienen un riesgo positivo. Aquí la interpretación es más complicada al desconocer el nivel real de educación de estos clientes.

```r
logit2 <- glm(factor.deposit ~ default, data = bank.train, family = "binomial")
summary(logit2)
```

```
Call:
glm(formula = factor.deposit ~ default, family = "binomial",
    data = bank.train)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.07800    0.02701  -2.887  0.00388 **
defaultyes  -0.39200    0.21713  -1.805  0.07102 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 7727.0  on 5580  degrees of freedom
Residual deviance: 7723.7  on 5579  degrees of freedom
AIC: 7727.7

Number of Fisher Scoring iterations: 3
```

La variable `default` no es estadísticamente significativa. Su $p - valor$ es mayor que 0.05. Sin embargo, está cerca de dicho valor de referencia. Eliminar esta variable del modelo en una etapa tan temprana puede ser un error.

```
logit2 <- glm(factor.deposit ~ balance, data = bank.train, family = "binomial")
summary(logit2)
```

```
Call:
glm(formula = factor.deposit ~ balance, family = "binomial",
    data = bank.train)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.667e-01  3.036e-02  -5.489 4.04e-08 ***
balance      5.621e-05  1.004e-05   5.600 2.15e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7727.0  on 5580  degrees of freedom
Residual deviance: 7689.1  on 5579  degrees of freedom
AIC: 7693.1

Number of Fisher Scoring iterations: 3
```

La variable `saldo` es claramente significativa. Un incremento de una unidad en el `saldo` está asociado a un incremento de 1.00006 en el riesgo de depóstivo. En variables continuas es muy habitual que un incremento tan bajo (1 dolar esté asociado a incrementos tan pequeños en el riesgo). Para mejorar la interpretación se suele multiplicar esta cantidad como sigue:

```
exp(logit2$coefficients[2]*1000)
```

```
 balance
1.057816
```

De este modo, la interpretación sería: los clientes con 1000 dólares más en el salto incrementan su riesgo de depósito en 1.06 respecto a los que tienen 1000 dólares menos.

> **Métodos Wrapper de selección de características**
>
> En el Chapter 4 vimos algunas técnicas de reducción de la dimensión basadas en la selección de características de acuerdo con la calidad de las mismas. Esta calidad se evalua empleando algoritmos de ML.

A continuación, ajustamos un modelo de regresión logística con todas las características significativas.

```
logit2 <- glm(factor.deposit ~ housing+marital+education+default+ balance, data = bank.tra
summary(logit2)
```

```
Call:
glm(formula = factor.deposit ~ housing + marital + education +
    default + balance, family = "binomial", data = bank.train)

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -1.982e-01  8.389e-02  -2.362  0.01815 *
housingyes         -7.770e-01  5.585e-02 -13.912  < 2e-16 ***
maritaldivorced     1.776e-01  8.901e-02   1.995  0.04604 *
maritalsingle       3.728e-01  6.211e-02   6.002 1.95e-09 ***
educationsecondary  2.475e-01  8.666e-02   2.856  0.00429 **
educationtertiary   4.323e-01  9.166e-02   4.717 2.40e-06 ***
educationunknown    3.178e-01  1.464e-01   2.170  0.03000 *
defaultyes         -2.920e-01  2.222e-01  -1.314  0.18872
balance             4.495e-05  1.007e-05   4.462 8.12e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7727.0  on 5580  degrees of freedom
Residual deviance: 7397.1  on 5572  degrees of freedom
```

```
AIC: 7415.1

Number of Fisher Scoring iterations: 4
```

Podemos ver como la variable `default` pierde significatividad estadística. Esto es normal. Al tener en consideración el efecto de otras variables, aquellas que estaban al límite de la significatividad pueden ser eliminadas. Entrenamos el modelo sin esa variable.

```
logit2 <- glm(factor.deposit ~ housing+marital+education+ balance, data = bank.train, fami
summary(logit2)
```

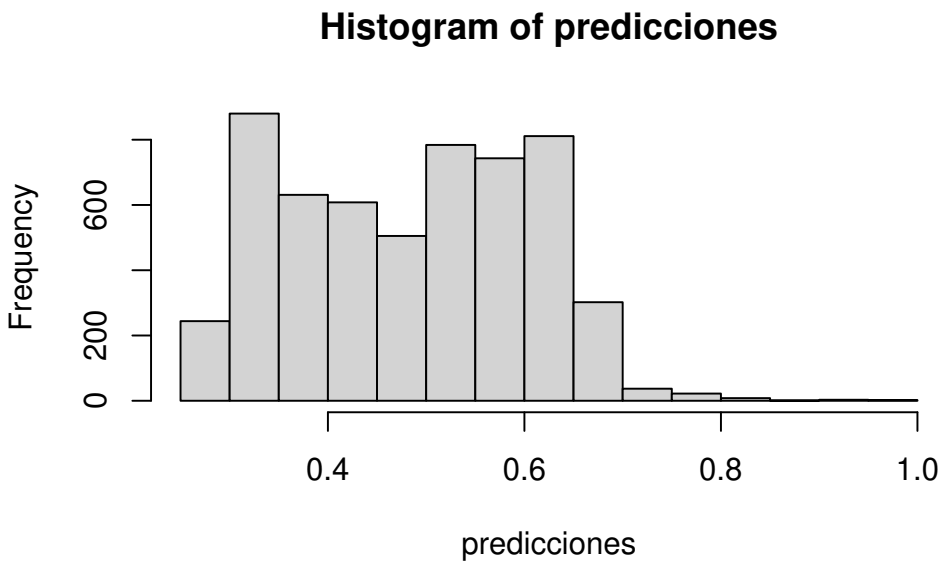
```
Call:
glm(formula = factor.deposit ~ housing + marital + education +
    balance, family = "binomial", data = bank.train)

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        -2.042e-01  8.377e-02  -2.438  0.01479 *
housingyes         -7.772e-01  5.584e-02 -13.917  < 2e-16 ***
maritaldivorced     1.751e-01  8.898e-02   1.968  0.04908 *
maritalsingle       3.735e-01  6.209e-02   6.014 1.80e-09 ***
educationsecondary  2.479e-01  8.667e-02   2.861  0.00423 **
educationtertiary   4.328e-01  9.167e-02   4.721 2.35e-06 ***
educationunknown    3.151e-01  1.464e-01   2.152  0.03137 *
balance             4.582e-05  1.009e-05   4.543 5.54e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7727.0  on 5580  degrees of freedom
Residual deviance: 7398.8  on 5573  degrees of freedom
AIC: 7414.8

Number of Fisher Scoring iterations: 4
```

Todas las variables del modelo son estadísticamente significativas. Si las variables fueran independientes, los coeficientes de la regresión múltiple coincidirían al 100% con los coeficientes de las regresiones simples. Por ejemplo, el coeficiente asociado a la variable `housing` era 2.29 y ahora es 2.18, muy similar. Podemos intuir la existencia de una pequeña correlación entre las variables del modelo.

7.1.3 Probabilidad de clase

Podemos calcular la probabilidad que ofrece el modelo para cada una de las observaciones en la muestra de entrenamiento.

```
predicciones <- predict(logit2, type="response")
hist(predicciones)
```



Y podemos emplear esas predicciones, con un umbral de decisión (0.5 en el ejemplo), para clasificar.

```
library(caret)
clase.pred=ifelse(predicciones>0.5,"yes","no")
confusionMatrix(data=as.factor(clase.pred),reference=as.factor(bank.train$deposit),positiv
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  1790 1078
       yes 1118 1595

               Accuracy : 0.6065
                 95% CI : (0.5936, 0.6194)
```

```
   No Information Rate : 0.5211
   P-Value [Acc > NIR] : <2e-16

                 Kappa : 0.2121

 Mcnemar's Test P-Value : 0.4053

            Sensitivity : 0.5967
            Specificity : 0.6155
         Pos Pred Value : 0.5879
         Neg Pred Value : 0.6241
             Prevalence : 0.4789
         Detection Rate : 0.2858
   Detection Prevalence : 0.4861
      Balanced Accuracy : 0.6061

       'Positive' Class : yes
```

En la salida tenemos algunas de las medidas de rendimiento estudiadas en el Chapter 6. Sin embargo, no aparece el $F_1 - score$, media harmónica de la Precisión y la Recuperación. En cualquier caso, es fácil de programar:

```
clase.pred=ifelse(predicciones>0.5,"yes","no")

confmat1=confusionMatrix(data=as.factor(clase.pred),reference=as.factor(bank.train$deposit
Precision=confmat1$byClass[3]
Recall=confmat1$byClass[1]
F1score=2*Precision*Recall/(Precision+Recall)
names(F1score)="F1Score"
F1score
```

```
  F1Score
0.6198061
```

Este es el funcionamiento del modelo en entrenamiento, pero ¿cuál es su funcionamiento en la partición de prueba? Recuerda, si el valor es mucho menor entonces estamos sobreajustando el modelo de entrenamiento.

```
predicciones <- predict(logit2, newdata=bank.test, type="response")
clase.pred=ifelse(predicciones>0.5,"yes","no")
```

```
confmat2=confusionMatrix(data=as.factor(clase.pred),reference=as.factor(bank.test$deposit)
Precision=confmat2$byClass[3]
Recall=confmat2$byClass[1]
F1score=2*Precision*Recall/(Precision+Recall)
names(F1score)="F1Score"
F1score
```

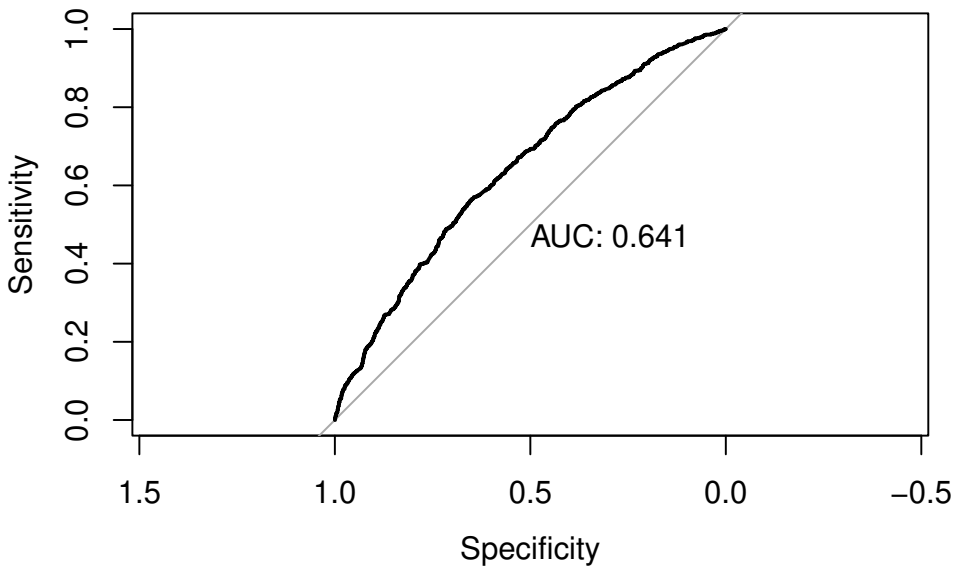
```
  F1Score
0.6137859
```

Vemos que el valor es muy similar al obtenido en entrenamiento. No estamos sobreajustando el modelo.

> **Tarea**
>
> Dejamos como labor del estudiante la tarea de mejorar la capacidad predictiva del modelo. Puedes emplear algunas de las variables que no han sido consideradas y que puedes consultar en el Chapter 3.

Podemos calcular la curva *ROC* presentada en el Chapter 6.

```
library(pROC)
roc_score=roc(bank.test$deposit, predicciones,plot=TRUE,print.auc=TRUE)
```



Y ahora, podemos plantear si este modelo es adecuado. Es decir ¿estamos satisfechos con el rendimiento del modelo? ¿Es suficientemente bueno?

7.1.4 Análisis Discriminante Lineal

El *análisis discriminante lineal (Linear Discriminant Analysis, LDA)* (Hastie et al. 2009), (Murphy 2012) es una técnica estadística utilizada para encontrar una combinación lineal de variables que discrimine dos o más grupos.

Se trata de un algoritmo de *clasificación*, es decir, la variable objetivo es categórica. Sus dos supuestos principales son que las variables explicativas (que deben ser continuas) siguen una distribución Normal y que comparten varianza.

El **objetivo** de LDA es construir la combinación lineal de las variables explicativas que mejor discrimina las clases. En base a dicho hiperplano separador se clasificarán las nuevas observaciones.

Para explicar mejor cómo funciona este algoritmo, comencemos recordando el teorema de Bayes en una caso de clasificación, es decir, cuando tenemos una variable objetivo Y y las variables explicativas $\mathbf{X}$:

$$P(C = 1|\mathbf{X} = \mathbf{x}) = \frac{f_1(\mathbf{x})P(1)}{\sum_{c=1}^{C_n} f_c(\mathbf{x})P(c)}$$

siendo $P(C = 1|\mathbf{X} = \mathbf{x})$ la probabilidad a posteriori, es decir, la probabilidad de la clase 1 dada la observación $\mathbf{x}$. $P(c)$ es la probabilidad a priori de la clase c. C_n es el número total de clases. Nótese que $\sum_{c=1}^{C_n} P(c) = 1$. Finalmente $f_c(\mathbf{x})$ es la función de densidad condicional de las $\mathbf{X}$ en la clase c. En el caso del LDA se asume que la densidad de cada clase es una Normal multivariante

$$f_c(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^t \Sigma_k^{-1}(\mathbf{x}-\mu_k)}$$

Además, también se asume que las clases comparten la misma matriz de varianzas covarianzas $\Sigma_c = \Sigma,\ \forall c$.

En base a estas asunciones, comparemos las probabilidades de ambas clases:

$$log\left(\frac{P(C = 1|\mathbf{X} = \mathbf{x})}{P(C = 0|\mathbf{X} = \mathbf{x})}\right) = log\left(\frac{f_1(\mathbf{x})}{f_0(\mathbf{x})}\right) + log\left(\frac{P(1)}{P(0)}\right) =$$

$$= log\left(\frac{P(1)}{P(0)}\right) - \frac{1}{2}(\mu_1 + \mu_0)^t \Sigma^{-1}(\mu_1 - \mu_0) + \mathbf{x}^t \Sigma^{-1}(\mu_1 - \mu_0)$$

Obtenemos una ecuación que es lineal en $\mathbf{x}$. Es decir, la frontera de decisión entre las clases 0 y 1 es una ecuación lineal, un hiperplano en dimensión p.

En la práctica, se desconocen los parámetros de la distribución Normal. Se estiman con los datos de entrenamiento:

- $\widehat{P(c)} = n_c/n$, siendo n_c el tamaño de la clase c y n el total
- $\hat{\mu}_c$ media muestral de los elementos de la clase c
- $\widehat{\Sigma}$ varianza muestral de los elementos de la clase c

¿Cómo se lleva entonces a cabo la clasificación? Como la frontera es un hiperplano separador, se comparará a qué población está más cercano cada observación. Esto es, para clasificar una observación como perteneciente a la clase 1, tendrá que ocurrir que

$$log\left(\frac{P(C=1|\mathbf{X}=\mathbf{x})}{P(C=0|\mathbf{X}=\mathbf{x})}\right) > 0$$

y esto es equivalente a

$$\mathbf{x}^t\widehat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0) > \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_0)^t\widehat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0) - log(n_1/n_0)$$

Si, en esta ecuación, llamamos $\mathbf{w} = \widehat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0)$, podemos reescribir la expresión anterior anterior como

$$\mathbf{x}^t\mathbf{w} > \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_0)^t\mathbf{w} - log(n_1/n_0)$$

La frontera de decisión entre ambas clases es

$$\mathbf{x}^t\mathbf{w} = \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_0)^t\mathbf{w} - log(n_1/n_0)$$

que es una combinación lineal de las variables explicativas.

¿Cómo se interpreta este procedimiento? Se está proyectando la observación a clasificar y las medias de las clases en una recta y se asigna la observación a la clase cuya media sea la más cercana.

i Ventajas

- **Simple y rápido**
- **Eficiente cuando se cumple las hipótesis**
- **Clasificación de observaciones en grupos determinados**. LDA se utiliza comúnmente para clasificar observaciones en grupos predeterminados, lo que facilita la interpretación y la toma de decisiones.

- **Combinación de información para la frontera de decisión**. Al considerar la información combinada de varias variables, LDA puede capturar patrones que podrían no ser evidentes al analizar cada variable por separado

Desventajas

- **Asume normalidad y homocedasticidad**. LDA puede ser sensible a la falta de normalidad en las variables y a la diferente variabilidad en las mismas.
- **Sensible a outliers**. LDA puede ser sensible a la presencia de valores atípicos en los datos, lo que puede afectar negativamente la calidad del modelo.
- **Es un clasificador lineal**.Como su nombre indica, LDA es lineal y puede no funcionar bien en situaciones donde la relación entre las variables predictoras y la variable dependiente es no lineal.
- **Requiere cierto tamaño de muestra**. Para obtener resultados confiables, LDA requiere un tamaño de muestra adecuado en cada grupo, y puede no funcionar bien con tamaños de muestra desequilibrados

7.2 k-Vecinos

El método de los k vecinos más cercanos ($k-NN$), abreviatura de "k nearest neighbors" en inglés, se cuenta entre los enfoques más simples y ampliamente utilizados en el campo del ML. Este método se basa en la noción de similitud (o distancia) entre observaciones.

Para recordar

La principal asunción del modelo de los k vecinos más cercanos es la existencia de un espacio de características en el cual observaciones similares se encuentran en proximidad.

A pesar de su simplicidad aparente, esta suposición plantea desafíos importantes. En primer lugar podemos plantearnos la elección adecuada del espacio de características, es decir, en la correcta selección de la métrica que definirá la similitud. No existe una fórmula mágica para determinar, a partir de un conjunto de datos dado, cuál es la métrica óptima a utilizar.

El segundo desafío se relaciona con la noción de "*cercanía*". ¿Cómo definimos la cercanía entre dos observaciones? ¿Cuál es el volumen máximo del espacio dentro del cual consideramos que dos observaciones están cerca, y más allá del cual las consideramos distantes?

A pesar de estas cuestiones aparentemente simples, el método de los k vecinos más cercanos se ha demostrado efectivo en una amplia variedad de aplicaciones, tanto en clasificación como

en regresión. En este apartado, exploraremos cómo funciona este algoritmo, cómo ajustar sus parámetros y cómo aplicarlo de manera efectiva en situaciones del mundo real. A pesar de su simplicidad conceptual, los k vecinos más cercanos siguen siendo una herramienta valiosa en el repertorio de ML.

El algoritmo de los k vecinos más cercanos es un sencillo método de clasificación y regresión. En primer lugar, se eligen los parámetros del modelo:

- la métrica,
- el número de vecinos, k.

En los ejemplos siguientes, se podrá observar que es habitual realizar pruebas con diferentes configuraciones de parámetros para identificar cuáles son los más apropiados. El principio fundamental del algoritmo implica obtener los k vecinos más cercanos para cada observación en la muestra de datos. En un problema de clasificación, el algoritmo devolverá la clase que predomina entre los k vecinos, es decir, la moda de la variable objetivo. En cambio, en un problema de regresión, el algoritmo proporcionará la media de la variable respuesta de los k vecinos más cercanos.

Para determinar el valor óptimo de k en función de los datos, ejecutamos el algoritmo $k-NN$ múltiples veces, utilizando distintos valores de k, y seleccionamos aquel valor que minimice la tasa de errores, al mismo tiempo que preserva la capacidad del algoritmo para realizar predicciones precisas en datos no vistos previamente. Es importante destacar que valores muy bajos de k pueden generar inestabilidad en el algoritmo, mientras que valores extremadamente altos de k pueden aumentar el error en las predicciones. En este proceso, se busca el equilibrio adecuado que optimice el rendimiento del algoritmo en cada situación.

Ventajas

7.3 Ventajas

- **Sencillez y Facilidad de Implementación**: Es uno de los algoritmos más sencillos y fáciles de entender en ML. No requiere suposiciones complejas o entrenamientos prolongados.
- **Versatilidad**: Se puede aplicar tanto a problemas de clasificación como de regresión.
- **No Paramétrico**: A diferencia de algunos algoritmos que asumen una distribución específica de los datos, $k-NN$ no hace suposiciones sobre la forma de los datos, lo que lo hace adecuado para una amplia gama de situaciones.
- **Adaptabilidad a Datos Cambiantes**: Puede adaptarse a cambios en los datos sin necesidad de volver a entrenar el modelo por completo, lo que lo hace útil en escenarios de datos en constante evolución. Ver Chapter 10.

- **Interpretabilidad**: Las predicciones de k-NN se basan en la observación de datos cercanos, lo que facilita la interpretación de las decisiones del modelo.
- **Robustez frente al ruido**: Puede manejar datos con ruido, que no se adaptan a la distribución de la mayoría de los datos, sin un gran impacto en su rendimiento.
- **Facilidad para Multiclases**: Es aplicable a problemas de clasificación con múltiples clases sin necesidad de modificaciones significativas.

Desventajas

7.4 Desventajas

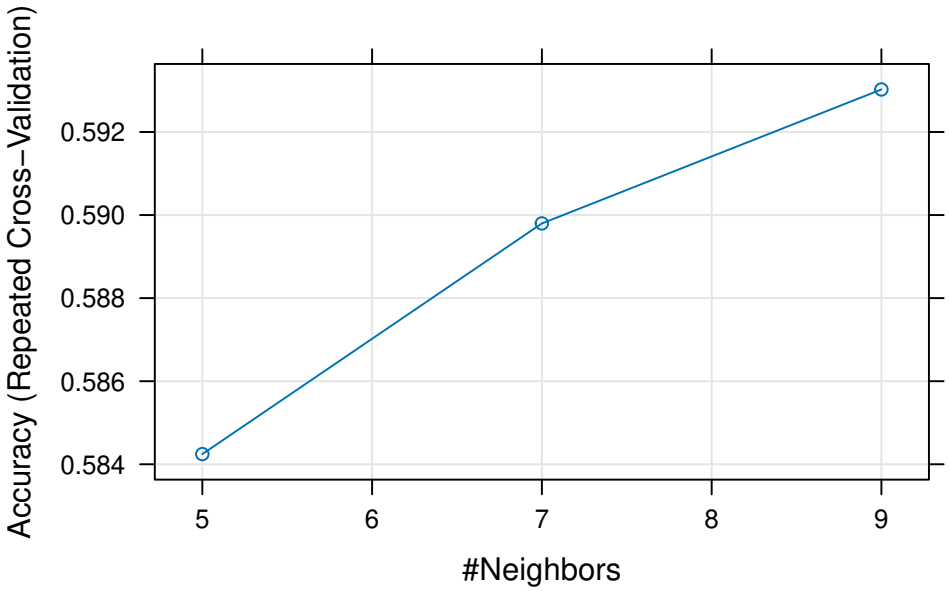
- **Sensibilidad a la elección de k**: La elección del parámetro que determina el número de vecinos puede ser crítica. Un valor demasiado pequeño puede hacer que el modelo sea sensible al ruido en los datos y cause sobreajuste, mientras que un valor demasiado grande puede hacer que el modelo sea demasiado suave y cause subajuste.
- **Sensibilidad a la Elección de la Métrica de Distancia**: La elección de la métrica de distancia es crucial y puede tener un impacto significativo en el rendimiento del algoritmo. No existe una métrica única que funcione para todos los problemas, y la elección de la métrica adecuada puede ser un desafío. Es muy probable que el científico de datos deba programar la métrica más adecuada para cada caso particular.
- **Coste computacional**: En conjuntos de datos grandes, calcular las distancias entre una nueva observación y todos los puntos de datos en el conjunto de entrenamiento puede ser computacionalmente costoso y lento. Esto limita su eficiencia en aplicaciones con grandes volúmenes de datos.
- **Incapacidad para Capturar Relaciones Complejas**: $k-NN$ es un algoritmo simple que no puede capturar relaciones complejas entre las características, como lo harían modelos más avanzados. No es adecuado para problemas con patrones no lineales.
- **Problemas en Datos Desbalanceados**: En problemas de clasificación con clases desequilibradas, donde una clase tiene muchas más muestras que otra, $k-NN$ tiende a sesgarse hacia la clase mayoritaria y puede no detectar bien la clase minoritaria.

Aplicamos el algoritmo a nuestro problema de los datos bancarios.

```r
# Uzamos 10-fold cross validation
trainControl <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "Accuracy"

set.seed(128)
df <- bank.train %>%
      select(housing,marital,education,balance,deposit)
df[,4] = scale(df[,4])
fit.knn <- train(deposit~., data=df, method="knn",
                 metric=metric ,trControl=trainControl)
knn.k1 <- fit.knn$bestTune
print(fit.knn)
```

```
k-Nearest Neighbors

5581 samples
   4 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 5023, 5023, 5023, 5022, 5023, 5023, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.5842447  0.1655072
  7  0.5897985  0.1765086
  9  0.5930238  0.1835312

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

```r
plot(fit.knn)
```



Según el gráfico anterior, podríamos elegir el mejor valor de k, si bien puede observarse pocas diferencias entre las diferentes opciones.

A continuación, obtenemos la predicción para el conjunto de datos de prueba e imprimimos la matriz de confusión.

```
set.seed(128)
media=mean(bank.train$balance)
stddev=sqrt(var(bank.train$balance))
df.test <- bank.test %>%mutate(balance=(balance-media)/stddev)

prediction <- predict(fit.knn,newdata=df.test)
cf <- confusionMatrix(prediction, as.factor(df.test$deposit),positive="yes")
print(cf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  900 578
       yes 568 744

               Accuracy : 0.5892
                 95% CI : (0.5707, 0.6076)
```

```
   No Information Rate : 0.5262
   P-Value [Acc > NIR] : 1.216e-11

                 Kappa : 0.1759

 Mcnemar's Test P-Value : 0.7903

            Sensitivity : 0.5628
            Specificity : 0.6131
         Pos Pred Value : 0.5671
         Neg Pred Value : 0.6089
             Prevalence : 0.4738
         Detection Rate : 0.2667
   Detection Prevalence : 0.4703
      Balanced Accuracy : 0.5879

       'Positive' Class : yes
```

Tarea

En el código anterior, ¿Porqué se calcula la media y la varianza de la variable `balance`?

7.5 Grid search

La "*grid search*" (búsqueda en cuadrícula) es una técnica comúnmente utilizada en ML para encontrar los **mejores hiperparámetros** para un modelo. Tal y como vimos en el Chapter 5, los hiperparámetros son configuraciones que no se aprenden del conjunto de datos, sino que se establecen antes de entrenar el modelo. Estos hiperparámetros pueden incluir aspectos como la tasa de aprendizaje, la profundidad máxima de un árbol de decisión, el número de vecinos en el algoritmo de $k-NN$ y otros ajustes que afectan cómo se entrena y se ajusta el modelo.

La *grid search* consiste en definir una "*cuadrícula*" de posibles valores para los hiperparámetros que se desean optimizar. Por ejemplo, si se está trabajando con un modelo de máquinas de vectores de soporte (SVM), se podría tener una cuadrícula para los hiperparámetros de la función kernel y el parámetro de regularización. Luego, se entrena y evalúa el modelo utilizando todas las combinaciones posibles de valores en la cuadrícula.

El objetivo es encontrar la combinación de hiperparámetros que produce el mejor rendimiento del modelo según una métrica específica, como precisión, exactitud, $F1$-score, error cuadrático medio, etc.

> **!** Para recordar
>
> La búsqueda en cuadrícula es un enfoque sistemático y automatizado que ahorra tiempo en comparación con probar manualmente diferentes combinaciones de hiperparámetros.

Si se tiene una cuadrícula de búsqueda para dos hiperparámetros, cada uno con tres valores posibles, se probarían un total de nueve combinaciones diferentes (3x3). El proceso de *grid search* evaluaría el rendimiento del modelo en nueve configuraciones distintas y seleccionaría aquella que obtiene los mejores resultados según la métrica definida.

Es importante mencionar exiten otras técnicas para la selección de hiperparámetros. Otros métodos, como la búsqueda aleatoria ("*random search*"), la optimización bayesiana y técnicas más avanzadas, también se utilizan en la selección de hiperparámetros en función de la complejidad y los recursos (computacionales, personales, temporales, etc) disponibles para la tarea.

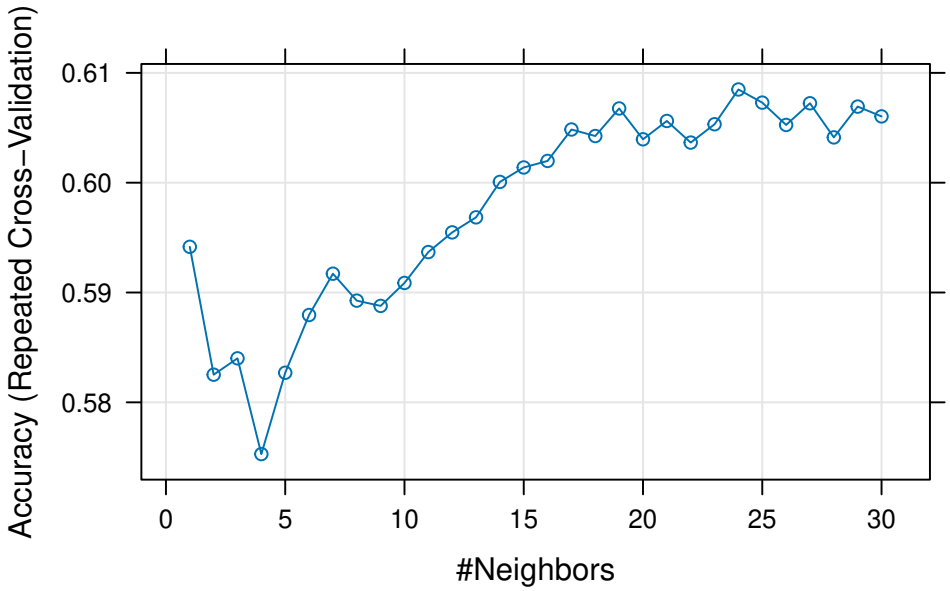
En el ejemplo que nos ocupa, buscamos el mejor valor del parámetro k entre 1 y 30:

```
set.seed(1271)
grid <- expand.grid(.k=seq(1,30,by=1))
fit.knn <- train(deposit~., data=df, method="knn",
                 metric=metric, tuneGrid=grid, trControl=trainControl)
knn.k2 <- fit.knn$bestTune
print(fit.knn)
```

```
k-Nearest Neighbors

5581 samples
   4 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 5023, 5024, 5024, 5023, 5023, 5023, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.5941624  0.1869556
   2  0.5825199  0.1634009
   3  0.5840047  0.1656685
   4  0.5752866  0.1478903
   5  0.5826935  0.1625400
```

```
 6  0.5879512  0.1727362
 7  0.5917138  0.1802904
 8  0.5892608  0.1756808
 9  0.5887821  0.1747647
10  0.5908722  0.1789482
11  0.5936836  0.1845274
12  0.5954742  0.1880368
13  0.5968459  0.1908545
14  0.6000722  0.1972805
15  0.6013864  0.1999850
16  0.6019835  0.2010479
17  0.6048438  0.2067664
18  0.6042492  0.2055601
19  0.6067594  0.2104826
20  0.6039522  0.2049647
21  0.6056218  0.2082912
22  0.6036533  0.2040438
23  0.6053225  0.2074437
24  0.6084891  0.2139799
25  0.6072912  0.2116389
26  0.6052587  0.2074789
27  0.6072328  0.2115650
28  0.6041256  0.2050824
29  0.6069315  0.2107200
30  0.6060367  0.2089063

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 24.
```

```
plot(fit.knn)
```



Encontramos óptimo $k = 24$, el número de instancias más cercanas que hay que recopilar para hacer una predicción óptima. Ahora, usamos el modelo ajustado para predecir la clase para nuestro conjunto de prueba, e imprimir la matriz de confusión:

```
set.seed(128)
prediction.knn <- predict(fit.knn,newdata=df.test,type="prob")[,2]
clase.pred.knn=ifelse(prediction.knn>0.5,"yes","no")
cf <- confusionMatrix(as.factor(clase.pred.knn), as.factor(df.test$deposit),positive="yes"
print(cf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  1028  603
       yes  440  719

               Accuracy : 0.6262
                 95% CI : (0.6079, 0.6442)
    No Information Rate : 0.5262
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2457
```

```
Mcnemar's Test P-Value : 5.271e-07

           Sensitivity : 0.5439
           Specificity : 0.7003
        Pos Pred Value : 0.6204
        Neg Pred Value : 0.6303
            Prevalence : 0.4738
        Detection Rate : 0.2577
  Detection Prevalence : 0.4154
     Balanced Accuracy : 0.6221

      'Positive' Class : yes
```

Métrica de disimilaridad

Hasta ahora hemos empleado la distancia Euclídea, pero ¿es lo más adecuado en este caso? Averigua cómo modificar este hiperparámetro del modelo y estudia qué consecuencias tiene.

7.6 Árboles de Decisión

Los Árboles de Decisión (DT, de *Decision Tree* en inglés) son algoritmos de ML basados en la **información**. Este tipo de algoritmos determinan qué variables explicativas proporcionan la mayor parte de la información (**ganancia de información**) para medir la variable objetivo y hacen predicciones probando secuencialmente las características en orden a su información (Kelleher and Tierney 2018).

Un DT se crea mediante un proceso de particionamiento recursivo en el conjunto de variables explicativas, es decir, probando el valor de una característica y creando una rama del árbol para cada uno de sus posibles valores. Para una mejor comprensión de la estructura de un árbol, necesitamos definir los siguientes conceptos:

- **Nodo raíz**: es el nodo origen, inicialmente todas las observaciones forman parte de este nodo.
- **Nodos internos**: son los nodos que se crean al definir reglas sobre una variable explicativa.
- **Nodos hojas**: son los nodos terminales del árbol.

Habitualmente, favorecemos árboles que empleen el menor número de preguntas posible, lo que significa que buscamos árboles poco profundos. La construcción de árboles con poca profundidad implica que las características más informativas, es decir, aquellas que mejor distinguen entre observaciones con diferentes valores en la variable objetivo, deben ubicarse en la parte superior del árbol. Para llevar a cabo esta tarea, es fundamental contar con una métrica formal que evalúe cuán efectivamente una característica discrimina entre los niveles de la variable objetivo.

> Entropía
>
> La **entropía** es una medida teórica de la "*incertidumbre*" contenida en un conjunto de datos, debido a la presencia de más de una posibilidad de clasificación.

La entropía es una medida de la impureza (heterogeneidad), de los datos. Desde un punto de vista formal, la entropía de un conjunto de N valores distintos que son igualmente probables es el menor número de preguntas de tipo *sí/no* necesarias para determinar un valor desconocido extraído de las N posibilidades:

$$Entropa = -\sum_{i=1}^{N} p_i log_2 p_i$$

Podemos ver que la entropía, por definición, toma valores mayores o iguales a cero. Su valor máximo se alcanza cuando las observaciones se distribuyen equitativamente entre las N posibles clases. La entropía toma su valor mínimo si, y sólo si, todas las observaciones están en la misma clase. Supongamos dos clases, esto es $N = 2$. En ese caso, consideramos la probabilidad de clase 1 y la probabilidad de clase 2 (p_1 y p_2 respectivamente). Si todas las observaciones están en la misma clase (por ejemplo, la clase 1), entonces: $p1 = 1$, $p2 = 0$. Y así:

$$Entropa = -(p_1 log_2 p_1 + p_2 log_2 p_2) = -(1 log_2 1 + 0 log_2 0) = 0$$

.

El algoritmo de DT basado en ganancia de la información sería como sigue:

1. Calcular la **entropía** del conjunto de datos original. ¿Cuánta información se requiere para organizar el conjunto de datos en conjuntos puros, donde todas las observaciones dentro del conjunto pertenecen a la misma clase? Esta medida nos da una idea de lo "complejo" que es nuestro problema.
2. Para cada variable explicativa, se crean los conjuntos resultantes **dividiendo las observaciones** en el conjunto de datos utilizando un umbral para dicha variable. A continuación, se suman los valores de entropía de cada uno de estos conjuntos. Calcular la información que sigue siendo necesaria para organizar las observaciones en conjuntos puros después de dividirlos utilizando la variable explicativa.

3. Calcular la **Ganancia de Información** restando la entropía restante (paso 2) del valor de entropía original (paso 1)

Es decir, la idea es emplear los valores de las características medidas sobre las observaciones para dividir el conjunto inicial de datos en subconjuntos más pequeños repetidamente, hasta que la entropía de cada uno de los subconjuntos sea cero (o pequeña, menor que un umbral). Para cada partición, la entropía media de los subconjuntos resultantes debería ser menor que la del conjunto anterior.

En términos generales, la **ganancia de información** se define como la diferencia entre la entropía anterior y la nueva entropía. Por lo tanto, para cada nodo en el árbol, se elige para la división aquella variable que maximiza la ganancia de información.

> **Índice de Gini**
>
> Existen otras métricas posibles para medir la calidad de la división. Por ejemplo, el **índice de Gini**:
>
> $$Gini = -\sum_{i=1}^{N} p_i(1 - p_i)$$

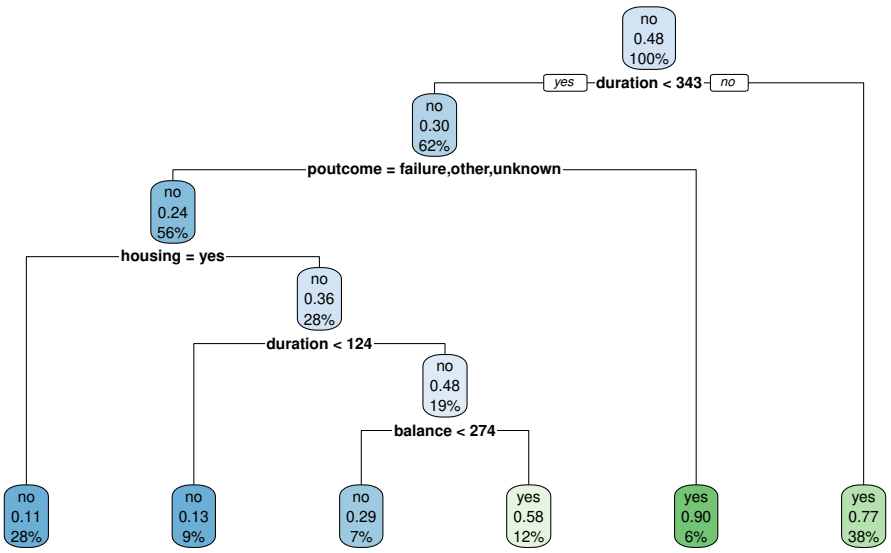
En el caso de árboles de regresión, donde la variable objetivo es cuantitativa, se define la reducción de la varianza como la disminución total de la varianza de la variable objetivo como resultado de la división del conjunto total en dos subconjuntos.

Existen varios criterios para detener la ejecución de este algoritmo de división. En un caso extremo podemos no detenerlo hasta que el error de clasificación sea 0. En dicho caso posiblemente cometamos errores de sobreajuste. Es decir, seleccionar divisiones sobre las variables hasta que todas las observaciones de la muestra de entrenamiento están perfectamente clasificadas funcionará bien, pero sólo sobre las observaciones de entrenamiento. Cuando ese mismo criterio se aplique a nuevas observaciones el resultado será, probablemente, mucho peor. En ocasiones se detiene el proceso de particionado cuando la profundidad del árbol supera un umbral (esto es, cuando se han empleado un número determinado de preguntas sobre las variables explicativas), o bien cuando el número de observaciones en el nodo a dividir es menor que un valor prefijado.

> **Para recordar**
>
> La salida de un DT es un conjunto de reglas que se puede representar como un árbol donde cada nodo representa una decisión binaria.

A continuación aplicamos el método de DT al conjunto de datos `bank`.

```
library(rpart)
library(rpart.plot)

set.seed(128)
df <- bank.train %>%
      select(age,job,housing,marital,education,duration,poutcome,balance,deposit)
fit.dt <- rpart(deposit~., data = df, method = 'class')
rpart.plot(fit.dt, extra = 106)
```



Como podemos observar, la interpretación del modelo de DT es muy sencilla. Todas las observaciones están en el primer nodo (el nodo raíz). Sin realizar "preguntas" sobre las variables explicativas, el nodo es de clase "*no*" con una probabilidad de "*yes*" del 0.48.

La primera variable de interés es la `duration`:

- Si el valor es menor que 343 (este valor es aprendido por el algoritmo y puede ser difícil de interpretar) entonces la observación va hacia la rama izquierda, cayendo en un nodo dónde la probabilidad de "yes" ha disminuido a 0.30. En ese nodo aparecen el 62% de todas las observaciones del conjunto de entrenamiento.
- Por contra, si el valor de `duration`es mayor que 343 entonces la observación va hacia la rama derecha, cayendo en un nodo dónde la probabilidad de "yes" ha aumentado a 0.77. En ese nodo aparecen el 38% de todas las observaciones del conjunto de entrenamiento. Además, fíjate que se trata de un nodo terminal.

Tarea

No se están teniendo en cuenta *todas* las variables disponibles en el conjunto de datos. El objetivo de la mayoría de los ejercicios de estos apuntes es mostrar a los alumnos cómo funcionan los métodos de ML. Si tu objetivo es obtener la más alta precisión, o el menor error, entonces te animamos a que intentes construir mejores modelos con, probablemente, más y mejores variables.

Podemos interpretar el resto de los nodos terminales del ejemplo como sigue:

Nodo	Descripción	Tamaño relativo (%)	Prob("yes")
1	duration < 343, poutcome=failure,other or unkonwn, housing=yes	26.0	0.11
2	duration < 124, poutcome=failure,other or unkonwn, housing=yes	9.0	0.13
3	124<=duration < 343, poutcome=failure,other or unkonwn, housing=yes, balance<274	7.0	0.29
4	124<=duration < 343, poutcome=failure,other or unkonwn, housing=yes, balance>=274	12.0	0.58
5	duration < 343, poutcome=success	6.0	0.90
6	duration>343	38.0	0.77

Usamos el modelo de DT para hacer predicciones:

```
# sobre la partición de entrenamiento
prediction <- predict(fit.dt, df, type = 'class')
cf <- confusionMatrix(prediction, as.factor(df$deposit),positive="yes")
```

```r
print(cf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  2090  349
       yes  818 2324

               Accuracy : 0.7909
                 95% CI : (0.78, 0.8015)
    No Information Rate : 0.5211
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.584

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8694
            Specificity : 0.7187
         Pos Pred Value : 0.7397
         Neg Pred Value : 0.8569
             Prevalence : 0.4789
         Detection Rate : 0.4164
   Detection Prevalence : 0.5630
      Balanced Accuracy : 0.7941

       'Positive' Class : yes
```

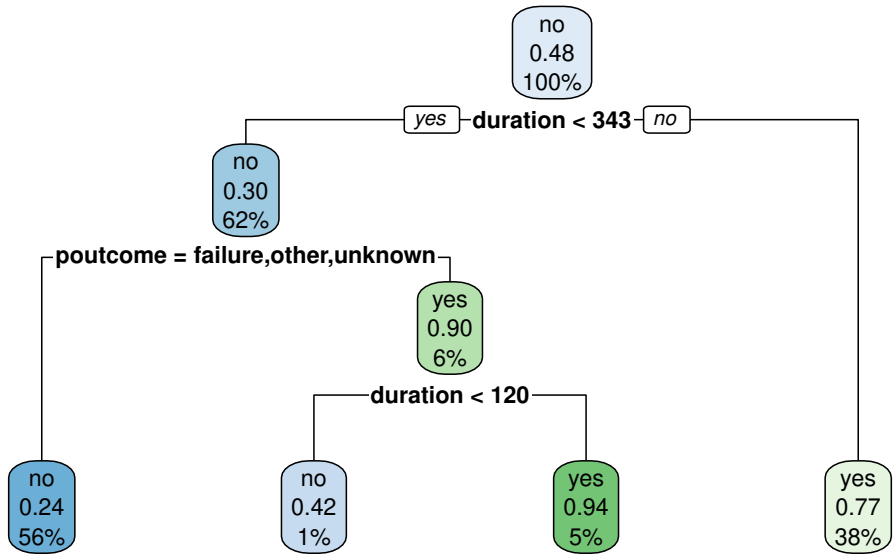
```r
# sobre la partición de prueba
prediction.dt <- predict(fit.dt, df.test, type = 'prob')[,2]
clase.pred=ifelse(prediction.dt>0.5,"yes","no")
cf <- confusionMatrix(as.factor(clase.pred), as.factor(df.test$deposit),positive="yes")
print(cf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
```

```
      no  1178  381
      yes  290  941

               Accuracy : 0.7595
                 95% CI : (0.7432, 0.7753)
    No Information Rate : 0.5262
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.516

 Mcnemar's Test P-Value : 0.000512

            Sensitivity : 0.7118
            Specificity : 0.8025
         Pos Pred Value : 0.7644
         Neg Pred Value : 0.7556
             Prevalence : 0.4738
         Detection Rate : 0.3373
   Detection Prevalence : 0.4412
      Balanced Accuracy : 0.7571

       'Positive' Class : yes
```

Al igual que en otros métodos de ML, podemos ajustar los hiperparámetros del modelo. En este caso ajustamos la máxima profundidad del árbol, el mínimo número de observaciones en un nodo para poder ser particionado en dos y el mínimo número de observaciones que un nodo terminal debe tener.

```
control <- rpart.control(minsplit = 4,
    minbucket = round(5 / 3),
    maxdepth = 3,
    cp = 0)
tune.fit <- rpart(deposit~., data = df, method = 'class', control = control)
rpart.plot(tune.fit, extra = 106)
```



El nuevo modelo presenta un rendimiento ligeramente menor que el original pero respondiendo a condiciones que antes no imponíamos.

```
# sobre la partición de prueba
prediction <- predict(tune.fit, df.test, type = 'class')
cf <- confusionMatrix(prediction, as.factor(df.test$deposit),positive="yes")
print(cf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  1186  393
       yes  282  929

               Accuracy : 0.7581
                 95% CI : (0.7417, 0.7739)
    No Information Rate : 0.5262
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5128

 Mcnemar's Test P-Value : 2.297e-05
```

```
            Sensitivity : 0.7027
            Specificity : 0.8079
         Pos Pred Value : 0.7671
         Neg Pred Value : 0.7511
             Prevalence : 0.4738
         Detection Rate : 0.3330
   Detection Prevalence : 0.4341
      Balanced Accuracy : 0.7553

       'Positive' Class : yes
```

Ventajas

- **Explicabilidad**: Los DT son altamente interpretables, lo que significa que sus resultados se pueden entender y comunicar de manera sencilla (ver Chapter 10). Esto los hace ideales para tomar decisiones basadas en modelos en entornos donde se necesita entender el razonamiento detrás de las predicciones.
- **Compatibilidad con Datos Mixtos**: Pueden manejar tanto variables explicativas cuantitativas como cualitativas (categóricas). Esta versatilidad les permite trabajar con una amplia variedad de tipos de datos.
- **Clasificación Multiclase**: Los DT se pueden extender de manera natural para la clasificación multiclase, lo que significa que pueden asignar observaciones a más de dos categorías.
- **Escalado de Variables no Requerido**: A diferencia de algunos otros algoritmos, los DT no requieren que las variables se escalen o estandaricen antes de su uso. Esto facilita el proceso y ahorra tiempo en la preparación de datos.
- **Manejo de Datos Faltantes**: Los árboles pueden manejar datos faltantes sin necesidad de preprocesamiento adicional. Tratan los valores faltantes como una categoría adicional y no descartan observaciones con datos faltantes.
- **Captura de No Linealidades**: Los DT son capaces de captar relaciones no lineales entre variables, lo que es valioso cuando las relaciones entre las características y la variable objetivo son complejas y no se ajustan bien a modelos lineales.
- **Robustez a Valores Atípicos**: Los árboles son menos sensibles a valores atípicos en los datos en comparación con algunos otros algoritmos de aprendizaje automático. Los valores atípicos no tienen un impacto tan drástico en la construcción de árboles como en modelos lineales, por ejemplo.

- **Eficiencia**: Son relativamente eficientes en términos computacionales, especialmente para conjuntos de datos de tamaño moderado. La construcción de árboles puede realizarse de manera rápida.

Desventajas

- **Menos Precisión en Modelos Simples:** Los DT a menudo son menos precisos que métodos más avanzados, como los bosques aleatorios o las redes neuronales, especialmente cuando se trata de problemas complejos. Su simplicidad a veces limita su capacidad para modelar relaciones sofisticadas en los datos.
- **Sensibilidad a Cambios en los Datos:** Pequeños cambios en los datos de entrenamiento pueden resultar en DT significativamente diferentes. Esto significa que los modelos basados en árboles son menos estables y robustos en comparación con algunos otros algoritmos. Un cambio en una observación o característica puede dar como resultado un árbol completamente diferente, lo que dificulta la confianza en las predicciones.
- **Propensión al Sobreajuste:** Los DT tienden a sobreajustarse a los datos de entrenamiento cuando se construyen demasiado profundos. Esto significa que el modelo se adapta demasiado a los datos de entrenamiento y puede no generalizar bien a nuevos datos. La elección de la profundidad óptima del árbol es crítica para evitar el sobreajuste.
- **Dificultad para Modelar Relaciones Lineales:** A pesar de su capacidad para capturar relaciones no lineales, los DT no son ideales para modelar relaciones lineales en los datos. En tales casos, otros modelos, como la regresión lineal, pueden ser más apropiados.
- **Limitación en la Predicción de Valores Continuos:** Los DT son adecuados para la clasificación, pero no son la mejor opción para la regresión o la predicción de valores continuos. Pueden proporcionar predicciones discretas, lo que puede ser una limitación en aplicaciones donde se requieren estimaciones precisas de valores numéricos.
- **Sensibilidad a Valores Atípicos:** Aunque son menos sensibles a los valores atípicos que algunos otros algoritmos, los DT todavía pueden verse afectados por valores extremos en los datos, lo que puede influir en la construcción del árbol y, por lo tanto, en las predicciones.
- **No Son Óptimos para Datos de Alta Dimensión:** Para conjuntos de datos de alta dimensión, la construcción de árboles puede volverse compleja y computacionalmente costosa. En tales casos, es posible que otros algoritmos, como las máquinas de vectores soporte (SVM) o el aprendizaje profundo, sean más apropiados.

- **Difícil Interpretación en Árboles Profundos:** Si los árboles se construyen muy profundos, pueden volverse difíciles de interpretar y visualizar, lo que reduce su utilidad en aplicaciones que requieren explicaciones claras de las decisiones del modelo.

Si bien los DTs son una herramienta valiosa, es importante considerar estas limitaciones al seleccionar la técnica de ML adecuada para un problema específico. Las desventajas mencionadas anteriormente son algunas de las razones por las que se han desarrollado métodos más avanzados, basados en ensamblado como los Bosques Aleatorios, para abordar algunas de estas limitaciones.

7.7 Métodos de ensamblado

Los métodos de ensamblado son una potente estrategia en el campo del ML que se basa en la idea de que la unión de múltiples modelos puede mejorar significativamente el rendimiento de predicción en comparación con un solo modelo.

La unión hace la fuerza

En lugar de depender de un solo algoritmo para tomar decisiones, los métodos de ensamblado combinan las predicciones de varios modelos para obtener resultados más precisos y robustos.

Esta técnica se asemeja a la *sabiduría colectiva*: al reunir a un grupo diverso de personas, cada una con su conjunto único de conocimientos y perspectivas, se puede tomar una decisión más sólida y precisa. Del mismo modo, los métodos de ensamblado combinan múltiples modelos, cada uno de los cuales puede sobresalir en diferentes aspectos del problema, para lograr una predicción más precisa y confiable.

Los métodos de ensamblado se han convertido en un componente esencial en la caja de herramientas de los científicos de datos y profesionales del ML. Estas técnicas utilizan una variedad de algoritmos base, como DT, regresión logística, máquinas de vectores soporte y más, y los combinan de manera inteligente para mejorar la capacidad de generalización del modelo. En este enfoque, se pueden distinguir dos categorías principales de métodos de ensamblado:

- el ensamblado de **bagging**
- el ensamblado de **boosting**

7.7.1 Bagging

El ensamblado de **Bagging**, (de "*Bootstrap Aggregating*" en inglés), es una técnica de ML diseñada para mejorar la precisión y la estabilidad de los modelos predictivos. Se basa en la idea de construir múltiples modelos similares y combinar sus predicciones para obtener un resultado final más robusto y generalizable.

Damos, a continuación, una explicación detallada de cómo funciona el ensamblado de Bagging:

1. **Bootstrap**: El proceso comienza dividiendo el conjunto de datos original en múltiples subconjuntos llamados conjuntos de entrenamiento, utilizando un método llamado muestreo con reemplazo. Esto significa que, en cada conjunto de entrenamiento, algunas muestras se seleccionan más de una vez, mientras que otras pueden quedar fuera. Este proceso genera múltiples conjuntos de entrenamiento, cada uno ligeramente diferente del original.
2. **Modelo Base**: Luego, se entrena un modelo base, como un DT, regresión logística o cualquier otro algoritmo, en cada uno de estos conjuntos de entrenamiento. Cada modelo base se entrena en datos diferentes debido al muestreo con reemplazo, lo que da como resultado una serie de modelos que pueden variar en pequeñas diferencias.
3. **Predicciones**: Una vez que se han entrenado todos los modelos base, se utilizan para hacer predicciones individuales sobre un conjunto de datos de prueba.
4. **Combinación**: Finalmente, las predicciones de todos los modelos base se combinan para obtener una predicción agregada. En problemas de clasificación, esto suele implicar votación, donde se elige la clase con más votos, y en problemas de regresión, se calcula un promedio de las predicciones.

i Ventajas

- **Reducción de la varianza**: Al crear múltiples conjuntos de entrenamiento y modelos base, el ensamblado de Bagging reduce la varianza de las predicciones, lo que hace que el modelo sea más robusto y menos propenso al sobreajuste.
- **Mayor precisión**: La combinación de múltiples modelos base suele resultar en una precisión general superior en comparación con un solo modelo.
- **Estabilidad**: Al construir modelos ligeramente diferentes a partir de diferentes subconjuntos de datos, el ensamblado de Bagging aumenta la estabilidad y la resistencia del modelo ante datos ruidosos o atípicos.
- **Mayor generalización**: Debido a su capacidad para reducir el sobreajuste, el ensamblado de Bagging es efectivo para problemas de alta dimensionalidad y datos con ruido.

Desventajas

- **Mayor complejidad computacional:** Debido a la necesidad de entrenar múltiples modelos base, el Bagging puede requerir más recursos computacionales y tiempo de entrenamiento en comparación con un solo modelo. Esto es especialmente relevante cuando se trabaja con conjuntos de datos grandes o algoritmos de modelo base complejos.
- **Menos interpretabilidad:** La combinación de múltiples modelos base dificulta la interpretación del modelo global. A diferencia de modelos individuales, como DT simples, los modelos Bagging no proporcionan una descripción sencilla de cómo se llega a una decisión o predicción.
- **No garantiza la mejora:** Aunque el Bagging suele mejorar la precisión y reducir la varianza en comparación con un solo modelo base, no garantiza un mejor rendimiento en todos los casos. En algunos conjuntos de datos o problemas, el Bagging puede no proporcionar mejoras significativas y, en raras ocasiones, podría incluso empeorar el rendimiento.
- **Menos efectivo con modelos base inestables:** Si se utilizan modelos base que son inherentemente inestables o propensos al sobreajuste, el Bagging puede no ser tan efectivo en mejorar su rendimiento. En tales casos, otros métodos de ensamblado, como Boosting, podrían ser más apropiados.
- **Limitaciones en problemas de desequilibrio de clases:** En problemas de clasificación con desequilibrio de clases, el Bagging puede no abordar adecuadamente el desafío de predecir clases minoritarias. En tales situaciones, se requieren técnicas específicas, como el ajuste de pesos de clase, para abordar el desequilibrio.

El algoritmo más conocido que utiliza Bagging es el "**Random Forest**", que combina múltiples DTs para lograr un modelo predictivo altamente preciso.

7.7.1.1 Random Forest

Un Bosque Aleatorio es una potente técnica de ensamblado que utiliza un conjunto de DTs para mejorar la precisión de las predicciones. En lugar de confiar en un solo árbol, se construye un bosque compuesto por numerosos árboles individuales. La singularidad de un Bosque Aleatorio radica en cómo se crean y combinan estos árboles.

Cada árbol dentro del Bosque Aleatorio no se crea a partir de todo el conjunto de datos, sino que se entrena con un subconjunto aleatorio de variables y un conjunto de observaciones seleccionadas al azar. Este proceso de muestreo aleatorio introduce diversidad en la construcción de cada árbol, lo que ayuda a mitigar la tendencia de DT a sobreajustar los datos.

Cada árbol individual en el Bosque Aleatorio genera una predicción para la variable objetivo, pero no se espera que cada árbol sea altamente efectivo por sí solo. La fortaleza del método radica en la combinación de numerosos árboles, cada uno de los cuales opera en una región diferente del espacio de características. El Bosque Aleatorio se basa en una regla de decisión que cuenta los votos de cada árbol para determinar la predicción final. En teoría, un gran número de modelos relativamente no correlacionados que funcionan como un comité superarán a cualquier modelo individual.

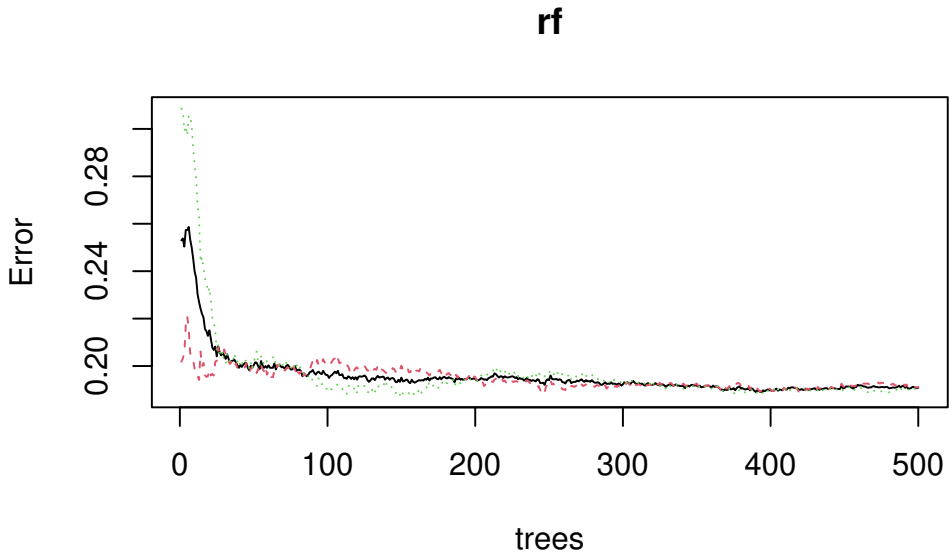
Una ventaja fundamental del Bosque Aleatorio es que aprovecha la sensibilidad de DTs a los datos en los que se entrenan. Cada árbol individual se entrena con una muestra aleatoria del conjunto de datos, permitiendo la selección con reemplazo. Esto da como resultado árboles diferentes en el bosque, lo que mejora la generalización del modelo.

Aplicamos este nuevo método de ML a nuestro conjunto de datos de `bank`.

```
library(randomForest)
library(caret)

rf <- randomForest(as.factor(deposit)~., data=df, importance=TRUE,proximity=TRUE)
print(rf)
```

```
Call:
 randomForest(formula = as.factor(deposit) ~ ., data = df, importance = TRUE,      proximity
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 19.08%
Confusion matrix:
      no  yes class.error
no  2352  556   0.1911967
yes  509 2164   0.1904227
```

```
plot(rf)
```




```
# sobre la partición de prueba
df.test <- bank.test %>%
      mutate(fmarital=as.factor(marital))%>%
      select(age,job,housing,marital=fmarital,education,duration,poutcome,balance,deposit)
df.test$deposit=as.factor(df.test$deposit)
prediction.rf <- predict(rf, df.test,type="prob")[,2]
clase.pred.rf=ifelse(prediction.rf>0.5,"yes","no")
cf <- confusionMatrix(as.factor(clase.pred.rf), as.factor(df.test$deposit),positive="yes")
print(cf)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  1157  242
       yes  311 1080

               Accuracy : 0.8018
                 95% CI : (0.7865, 0.8164)
    No Information Rate : 0.5262
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6035
```

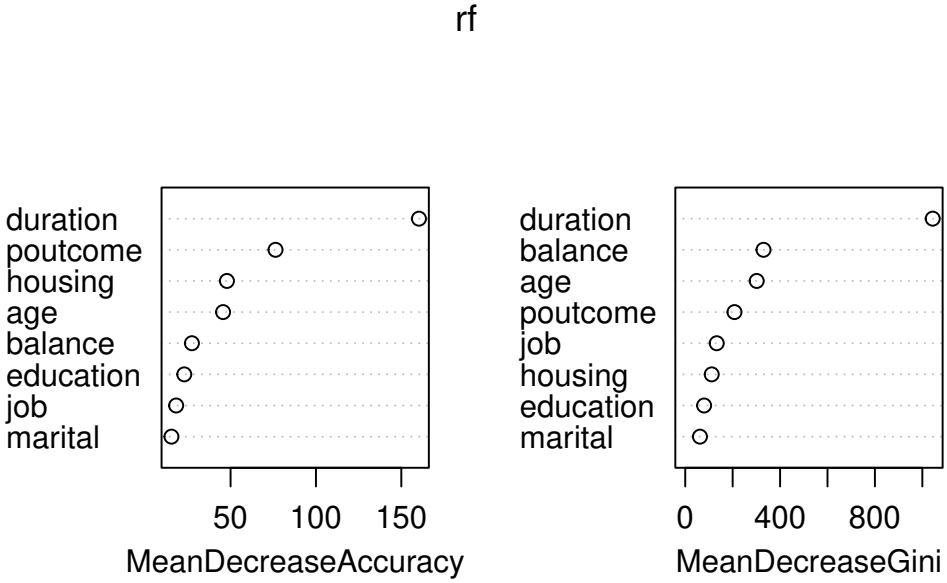
```
 Mcnemar's Test P-Value : 0.003832

            Sensitivity : 0.8169
            Specificity : 0.7881
         Pos Pred Value : 0.7764
         Neg Pred Value : 0.8270
             Prevalence : 0.4738
         Detection Rate : 0.3871
   Detection Prevalence : 0.4986
      Balanced Accuracy : 0.8025

       'Positive' Class : yes
```

Podemos averiguar la importancia de las variables en el modelo.

```
varImp(rf)
```

```
                no        yes
age         32.275333  32.275333
job         11.934786  11.934786
housing     37.597656  37.597656
marital      9.492926   9.492926
education   15.716137  15.716137
duration   143.795574 143.795574
poutcome    63.717485  63.717485
balance     19.018701  19.018701
```

```
varImpPlot(rf)
```



Este gráfico muestra la importancia de las variables, ordenadas de mayor a menor. En este caso particular la variable más relevante dentro del modelo es la duración de la llamada.

Además podemos obtener una visualización del funcionamiento del Bosque con la función `MDSplot`:

```
#MDSplot(rf,as.factor(df$deposit),k=3)
```

La función grafica las k primeras componentes principales (ver Chapter 4) de la matriz de proximidad, que es una matriz de medidas de proximidad entre la observaciones de entrenamiento (basada en la frecuencia con que los pares de puntos de datos se encuentran en los mismos nodos terminales).

7.7.2 Boosting

El ensamblado de **Boosting** es una técnica de ML que se utiliza para mejorar la precisión de los modelos de clasificación o regresión. A diferencia del Bagging, que se basa en la construcción de múltiples modelos independientes y su promediación, el Boosting se centra en mejorar iterativamente un solo modelo débil (o sencillo).

El proceso de Boosting funciona de la siguiente manera:

1. Comienza con un **modelo base** (generalmente un modelo simple o débil), que se entrena en el conjunto de datos original.

2. Luego, se evalúa el rendimiento del modelo base. Las instancias clasificadas incorrectamente o **las predicciones con errores se ponderan más fuertemente** para la siguiente iteración.
3. En la siguiente iteración, se entrena un **segundo modelo débil**, pero esta vez se da más énfasis a las instancias que se clasificaron incorrectamente en la iteración anterior.
4. Los resultados de los modelos base se combinan ponderando sus predicciones en función de su **rendimiento relativo**. Los modelos que tienen un mejor rendimiento reciben más peso en la predicción final.
5. Este proceso de entrenamiento, evaluación y asignación de pesos se repite durante **varias iteraciones** (también llamadas etapas de Boosting). Cada nuevo modelo se enfoca en corregir las deficiencias del modelo anterior.
6. Al final de las iteraciones, se obtiene un **modelo ensamblado fuerte** que es capaz de mejorar significativamente la precisión de las predicciones en comparación con un solo modelo base.

Algunos algoritmos de Boosting populares incluyen AdaBoost, Gradient Boosting, y XGBoost.

Ventajas

- **Mejora del rendimiento:** El Boosting es conocido por aumentar significativamente la precisión de las predicciones en comparación con modelos individuales o modelos base.
- **Reducción del sesgo:** A través del proceso iterativo, el Boosting puede reducir el sesgo del modelo, lo que significa que se vuelve más capaz de ajustarse a los datos y hacer predicciones precisas.
- **Manejo de datos desequilibrados:** Es eficaz en la clasificación de conjuntos de datos con clases desequilibradas, ya que se enfoca en las instancias mal clasificadas.
- **Adaptabilidad:** Funciona bien con una variedad de algoritmos base, lo que brinda flexibilidad al elegir el modelo débil más adecuado.
- **Versatilidad:** El Boosting se utiliza en problemas de clasificación y regresión, lo que lo hace adecuado para una amplia gama de aplicaciones.
- **Capacidad para detectar patrones complejos:** Al mejorar iterativamente el modelo, el Boosting puede capturar patrones complejos en los datos y generar modelos más precisos.
- **Robustez:** Puede manejar ruido en los datos y es menos propenso al sobreajuste en comparación con algunos otros métodos de aprendizaje automático.

Desventajas

- **Sensibilidad al ruido:** Los modelos de Boosting pueden ser sensibles al ruido y a valores atípicos en los datos. Si el conjunto de datos contiene observaciones erróneas o extremas, el Boosting puede sobreajustarse a estos valores inusuales.
- **Tiempo de entrenamiento:** El proceso de Boosting implica entrenar múltiples modelos base de manera secuencial, lo que puede llevar más tiempo en comparación con otros métodos de ensamblado más simples.
- **Menos interpretabilidad:** A medida que se agregan más modelos al ensamblado, la interpretabilidad del modelo general puede disminuir. Esto hace que sea más difícil comprender y explicar las predicciones del modelo.
- **Necesidad de ajuste de hiperparámetros:** Los modelos de Boosting tienen varios hiperparámetros que deben ajustarse adecuadamente para obtener el mejor rendimiento. Encontrar la combinación óptima de hiperparámetros puede requerir tiempo y recursos computacionales.
- **Potencial sobreajuste:** Si no se ajustan cuidadosamente los hiperparámetros, los modelos de Boosting pueden estar sujetos al sobreajuste, especialmente si el número de iteraciones (número de modelos base) es demasiado alto.
- **Requisitos de recursos computacionales:** Dependiendo de la implementación y la cantidad de modelos base, los modelos de Boosting pueden requerir recursos computacionales significativos, lo que podría ser un problema en sistemas con recursos limitados.

7.7.2.1 XGBoost

XGBoost, que significa "*Extreme Gradient Boosting*", es un modelo de ML de tipo ensamblado que ha ganado gran popularidad en competiciones de ciencia de datos y aplicaciones prácticas debido a su alto rendimiento y eficacia en una variedad de problemas de clasificación y regresión. XGBoost es una poderosa técnica de ML que ha demostrado su eficacia en una amplia gama de aplicaciones. Su capacidad para manejar datos ruidosos, seleccionar características importantes y su velocidad de entrenamiento lo hacen valioso tanto para profesionales de datos como para científicos de datos en competiciones y proyectos del mundo real.

XGBoost es una extensión del método de *Gradient Boosting*, que se centra en mejorar las debilidades de dicho algoritmo. ¿Cómo funciona el método del Gradient Boosting?. Lo vemos en la siguiente figura:

Pasamos por ciclos que construyen repetidamente nuevos modelos y los combinan en un modelo de ensamblado. Empezamos el ciclo tomando un modelo existente y calculando los errores de

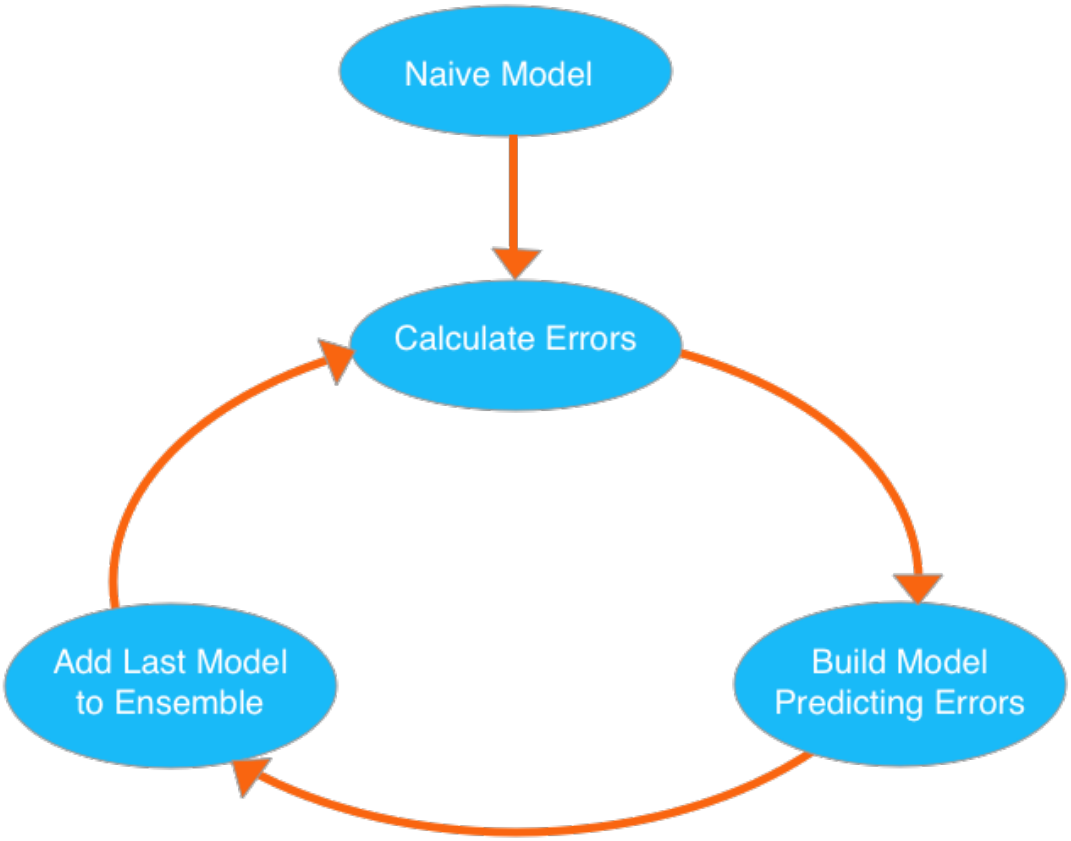


Figure 7.1: https://www.kaggle.com/code/rtatman/machine-learning-with-xgboost-in-r

cada observación del conjunto de datos. A continuación, construimos un nuevo modelo para predecir estos errores. Añadimos las predicciones de este modelo de predicción de errores al ensamblado de modelos.

Para hacer una predicción, sumamos las predicciones de todos los modelos anteriores. Podemos utilizar estas predicciones para calcular nuevos errores, construir el siguiente modelo y añadirlo al ensamblado.

```
library(purrr)
library(dplyr)
library(caret)
library(xgboost)

#convertimos los datos a numéricos
df.train <- map_df(df, function(columna) {
  columna %>%
    as.factor() %>%
    as.numeric %>%
    { . - 1 }
})

datos <- list()
datos$train <- df.train

df.test <- map_df(df.test, function(columna) {
  columna %>%
    as.factor() %>%
    as.numeric %>%
    { . - 1 }
})

datos$test <- df.test


# Convertimos los datos al formato Dmatrix
datos$train_mat  <-
  datos$train %>%
  select(-deposit) %>%
  as.matrix() %>%
  xgb.DMatrix(data = ., label = datos$train$deposit)
datos$test_mat  <-
  datos$test %>%
```

```
    select(-deposit) %>%
    as.matrix() %>%
    xgb.DMatrix(data = ., label = datos$test$deposit)

  # Entrenamiento del modelo
  datos$modelo_01 <- xgboost(data = datos$train_mat,
                             objective = "binary:logistic", #clasificación binaria
                             nround = 10, # número máximo de iteraciones boosting
                             max_depth=2, # número de nodos de bifurcación de los árboles de
                             eta =0.3, # La tasa de aprendizaje del modelo
                             nthread =2) #  El número de hilos computacionales que serán usa
```

```
[1] train-logloss:0.620493
[2] train-logloss:0.576189
[3] train-logloss:0.547507
[4] train-logloss:0.527672
[5] train-logloss:0.507268
[6] train-logloss:0.496584
[7] train-logloss:0.485207
[8] train-logloss:0.473366
[9] train-logloss:0.467606
[10]    train-logloss:0.462035
```

```
  datos$modelo_01
```

```
##### xgb.Booster
raw: 12.8 Kb
call:
  xgb.train(params = params, data = dtrain, nrounds = nrounds,
    watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
    early_stopping_rounds = early_stopping_rounds, maximize = maximize,
    save_period = save_period, save_name = save_name, xgb_model = xgb_model,
    callbacks = callbacks, objective = "binary:logistic", max_depth = 2,
    eta = 0.3, nthread = 2)
params (as set within xgb.train):
  objective = "binary:logistic", max_depth = "2", eta = "0.3", nthread = "2", validate_parame
xgb.attributes:
  niter
callbacks:
```

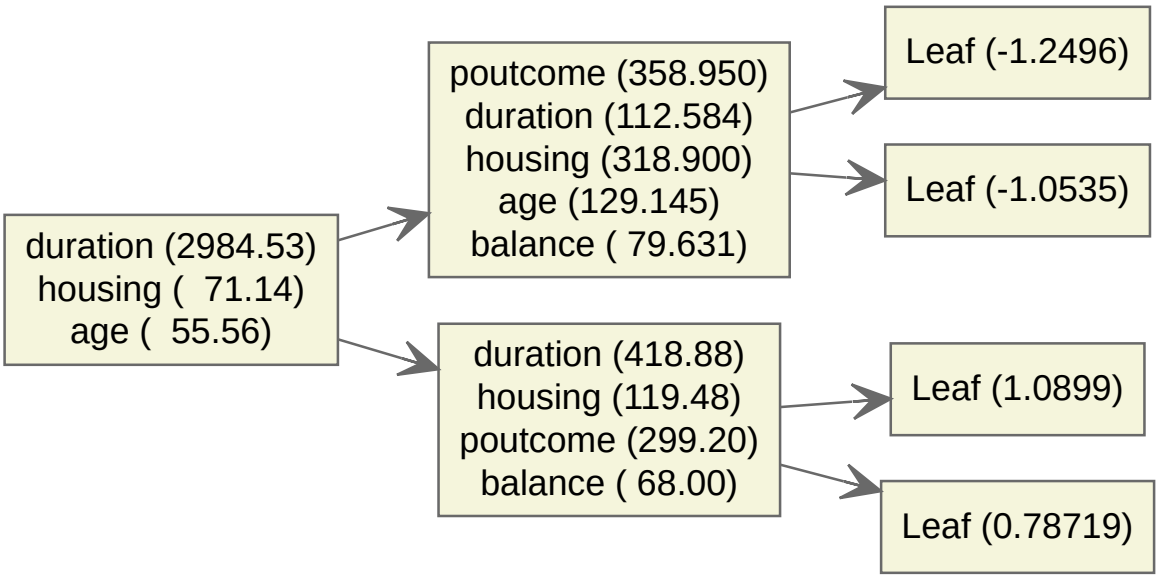
```
  cb.print.evaluation(period = print_every_n)
  cb.evaluation.log()
# of features: 8
niter: 10
nfeatures : 8
evaluation_log:
    iter train_logloss
       1     0.6204931
       2     0.5761893
---
       9     0.4676060
      10     0.4620354
```

Ahora podemos examinar e interpretar nuestro modelo XGBoost. Una forma en que podemos examinar nuestro modelo es observando una representación de la combinación de todos los DTs en nuestro modelo. Dado que todos los árboles tienen la misma profundidad podemos apilarlos unos encima de otros y elegir las características que aparecen con más frecuencia en cada nodo.

```
# Visualización del conjunto de árboles como una única unidad
xgb.plot.multi.trees(model = datos$modelo_01)
```

```
Column 2 ['No'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list
```

```
PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installe
```



En esta figura, la parte superior del árbol está a la izquierda y la parte inferior a la derecha. En el caso de las características, el número que aparece al lado es la "*calidad*", que ayuda a indicar la importancia de la característica en todos los árboles. Una mayor calidad significa que la característica es más importante. Por tanto, podemos decir que `duration` es con diferencia la característica más importante en todos nuestros árboles, tanto porque está más arriba en el árbol como porque su puntuación de calidad es muy alta.

Para los nodos "*hoja*", el número es el valor medio del modelo devuelto a través de todos los árboles cuando una determinada observación terminó en esa hoja. Como estamos utilizando un modelo logístico, nos devuelve diciendo que el logaritmo de probabilidades en lugar de la probabilidad. Sin embargo, podemos convertir fácilmente las probabilidades logarítmicas en probabilidad.

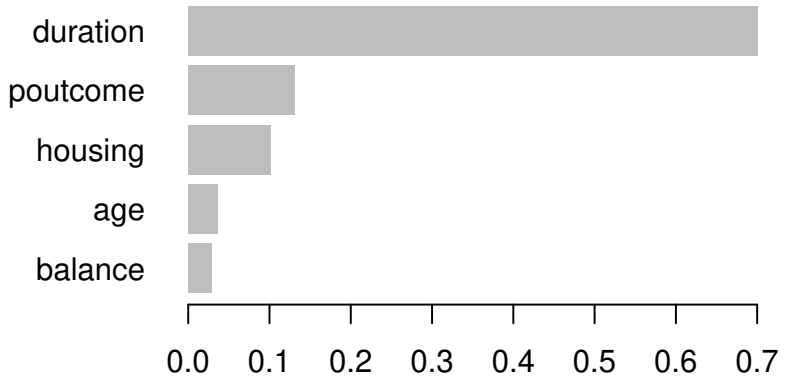
```
# convertimos log odds a probabilidades
odds_to_probs <- function(odds){
    return(exp(odds)/ (1 + exp(odds)))
}


# ejemplo
odds_to_probs(-1.2496)
```

```
[1] 0.2227694
```

Así, en los árboles en los que una observación acabó en esa hoja, la probabilidad media de que una observación fuera un "*yes*" en la variable `deposit` es del 22%.

¿Y si queremos ver rápidamente qué características son las más importantes? Podemos hacerlo creando, y luego trazando, la matriz de importancia, como sigue.

```
# importancia sobre cada característica
importance_matrix <- xgb.importance(model = datos$modelo_01)

xgb.plot.importance(importance_matrix)
```



Pasamos a obtener las predicciones. El resultado es un vector de valores numéricos, cada uno representando la probabilidad de que un caso en particular pertenezca al valor 1 de nuestra variable objetivo. Es decir, la probabilidad de que esa observación sea un "*yes*".

```
# Predicciones en la muestra de prueba
datos$predict_01 <- predict(datos$modelo_01, datos$test_mat)
cbind(datos$predict_01 > 0.5, datos$test$deposit) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix(positive="1")
```

```
Confusion Matrix and Statistics

   X2
X1     0    1
  0 1188  319
  1  280 1003

               Accuracy : 0.7853
```

```
                95% CI : (0.7696, 0.8004)
    No Information Rate : 0.5262
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5688

 Mcnemar's Test P-Value : 0.1205

            Sensitivity : 0.7587
            Specificity : 0.8093
         Pos Pred Value : 0.7818
         Neg Pred Value : 0.7883
             Prevalence : 0.4738
         Detection Rate : 0.3595
   Detection Prevalence : 0.4599
      Balanced Accuracy : 0.7840

       'Positive' Class : 1
```

! Time consuming

La tarea que más tiempo consume al usar el modelo XGBoost es encontrar los mejores hiperparámetros para alcanzar la mayor precisión posible de un modelo.

Tarea

Queda como tarea del alumno modificar los hiperparámetros del modelo, buscando una mejora en su rendimiento

7.8 Naive Bayes

El clasificador ingenuo de Bayes (en inglés "*Naïve Bayes*") es un clasificador sencillo que se basa en el conocido **teorema de Bayes**. A pesar de su simplicidad, los clasificadores "ingenuos" de Bayes son especialmente útiles para problemas con muchas variables de entrada, variables de entrada cuantitativas con un número muy grande de valores posibles y clasificación de texto. Es uno de los algoritmos básicos que suele aparecer en problemas de *Procesamiento de Lenguaje Natural*. Podrás estudiar aspectos fundamentales de Lenguaje Natural a lo largo de tus estudios de grado.

El teorema de Bayes relaciona la probabilidad condicional de dos eventos A y B. La idea es modificar nuestras creencias iniciales sobre lo que ha sucedido (a priori) proporcionalmente con la forma en que nuestras observaciones se relacionan con sus posibles causas (probabilidad inversa):

$$P(A \cap B) = P(A, B) = P(A) * P(B|A) = P(B) * P(A|B) \Rightarrow P(B|A) = \frac{P(B)P(A|B)}{P(A)}$$

Su aplicación a un problema de clasificación es como sigue. Supongamos $X_1, X_2, ..., X_n$ variables explicativas independientes dado el valor de la variable objetivo Y_k. Es decir, suponemos:

$$P(X_1|X_2, ..., X_n, Y_k) = P(X_1|Y_k); P(X_2|X_3, ..., X_n, Y_k) = P(X_2|Y_k)$$

, etc.

De este modo, aplicando el teorema de Bayes de manera recursiva, se tiene:

$$P(X_1, X_2, ..., X_n, Y_k) = P(Y_k)P(X_n|Y_k)P(X_{n-1}|X_nY_k)...P(X_1|X_2, ..., X_n, Y_k) = P(Y_k)P(X_n|Y_k)P(X_{n-1}|Y_k)...P(X_1|Y_k)$$

Por tanto:

$$P(Y_k|X_1, ..., X_n) = \frac{P(Y_k)\prod_{j=1}^{n} P(X_j|Y_k)}{P(X_1, X_2, ..., X_n)}$$

Esta es la expresión de la probabilidad de un valor de la variable respuesta dado el conjunto de valores de las variables explicativas. El denominador es constante, de modo que bastará con considerar sólo el numerador para comparar las probabilidades de las diferentes clases condicionadas a los valores de las variables explicativas. Se evalúa el numerador para todos los valores de la variable objetivo y se obtienen las diferentes probabilidades de clase.

La probabilidad de clase $P(Y_k)$ se denomina probabilidad a priori, y:

$$\prod_{j=1}^{n} P(X_j|Y_k)$$

es la verosimilitud. La verosimilitud mide cómo de verosímiles son las observaciones de las variables explicativas dado que se ha observado una determinada etiqueta en la variable respuesta. Si la variable es cualitativa, habitualmente se asume una distribución Normal. En caso de variables cuantitativas, se suele asumir la distribución Multinomial.

Finalmente se predice para cada conjunto de variables explicativas la clase con mayor probabilidad de clase condicionada $P(Y_k|X_1, \ldots, X_n)$.

```
library(naivebayes)
library(dplyr)
library(caret)

model <- naive_bayes(as.factor(deposit) ~ ., data = df.train, usekernel = T)
probabilities <- predict(model, df.test[,-9],type="prob")[,2]
classes <- as.numeric(probabilities>0.5)
confusionMatrix(table(classes, datos$test$deposit),positive="1")
```

```
Confusion Matrix and Statistics


classes    0    1
      0 1166  264
      1  302 1058

               Accuracy : 0.7971
                 95% CI : (0.7817, 0.8119)
    No Information Rate : 0.5262
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5937

 Mcnemar's Test P-Value : 0.1199

            Sensitivity : 0.8003
            Specificity : 0.7943
         Pos Pred Value : 0.7779
         Neg Pred Value : 0.8154
             Prevalence : 0.4738
         Detection Rate : 0.3792
   Detection Prevalence : 0.4875
      Balanced Accuracy : 0.7973

       'Positive' Class : 1
```

7.9 Modelos de mezcla de Gaussianas

Los modelos de mezcla de gaussianas (**GMM**, por sus siglas en inglés, "*Gaussian Mixture Models*") son una técnica de ML utilizada para modelar datos con estructuras complejas y distribuciones múltiples. Estos modelos se basan en la idea de que un conjunto de datos puede estar compuesto por varias distribuciones gaussianas (también conocidas como normales). En otras palabras, los GMM permiten descomponer un conjunto de datos en múltiples componentes gaussianas, cada una de las cuales describe una parte de la distribución de los datos.

Los datos en el mundo real rara vez siguen una distribución simple y unimodal (¡ninguna teoría es tan compleja como un conjunto de datos!). Los GMM abordan esta complejidad al permitir que múltiples componentes gaussianas se superpongan y se combinen para representar mejor la verdadera distribución de los datos. Esto los convierte en una herramienta efectiva para modelar datos que pueden ser una mezcla de diferentes poblaciones o grupos subyacentes.

Cada componente gaussiana representa una subdistribución de datos, y se caracteriza por su media (promedio) y desviación estándar (dispersión). En un modelo de mezcla de gaussianas, se asume que cada componente es una distribución normal.

Cada observación en el conjunto se asigna a una de las componentes gaussianas en función de la probabilidad de que pertenezca a esa componente. Esto permite una descripción detallada de cómo se distribuyen los datos entre los diferentes grupos subyacentes.

Los GMM se utilizan en una variedad de aplicaciones, como la segmentación de imágenes, la detección de anomalías, la compresión de datos, la clasificación de patrones y la generación de datos sintéticos.

La estimación de los parámetros de un GMM, es decir, las medias y desviaciones estándar de las componentes gaussianas, se realiza mediante algoritmos de optimización que buscan maximizar la probabilidad conjunta de los datos observados.

Los GMM son especialmente útiles en la detección de anomalías, ya que permiten identificar regiones en el espacio de características donde los datos son inusuales o atípicos en comparación con el modelo GMM.

Aunque los GMM son efectivos en la modelación de datos complejos, pueden ser sensibles al número de componentes elegido y a la inicialización. La selección de un número adecuado de componentes y una buena inicialización son aspectos importantes en su aplicación.

8 Comparación de modelos

Hemos aplicado una variedad de modelos (y en otras asignaturas verás nuevos modelos de ML). Podemos comparar el rendimiento de modelos de diversas maneras. Una de ellas, tal y como vimos en el Chapter 6 es la curva ROC:

```
library(pROC)
roc_score
```

```
Call:
roc.default(response = bank.test$deposit, predictor = predicciones,     plot = TRUE, print.au

Data: predicciones in 1468 controls (bank.test$deposit no) < 1322 cases (bank.test$deposit ye
Area under the curve: 0.6411
```

```
par(pty = "s") # square
roc(df.test$deposit, probabilities,plot=TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "#377eb8", lwd = 2,
    print.auc = TRUE,legend=TRUE,  brier.in.legend =TRUE)
```

```
Call:
roc.default(response = df.test$deposit, predictor = probabilities,     percent = TRUE, plot =

Data: probabilities in 1468 controls (df.test$deposit 0) < 1322 cases (df.test$deposit 1).
Area under the curve: 87.33%
```

```
roc(df.test$deposit, predicciones, percent=TRUE, col="#4daf4a",lwd= 2,
         print.auc =TRUE, add=TRUE,print.auc.y = 40,plot=TRUE,legend=TRUE)
```

```
Call:
roc.default(response = df.test$deposit, predictor = predicciones,     percent = TRUE, plot =

Data: predicciones in 1468 controls (df.test$deposit 0) < 1322 cases (df.test$deposit 1).
Area under the curve: 64.11%
```

```
roc(df.test$deposit, prediction.dt, percent=TRUE, col="goldenrod",lwd= 2,
          print.auc =TRUE, add=TRUE,print.auc.y = 30,plot=TRUE,legend=TRUE)
```

```
Call:
roc.default(response = df.test$deposit, predictor = prediction.dt,     percent = TRUE, plot =

Data: prediction.dt in 1468 controls (df.test$deposit 0) < 1322 cases (df.test$deposit 1).
Area under the curve: 81.39%
```

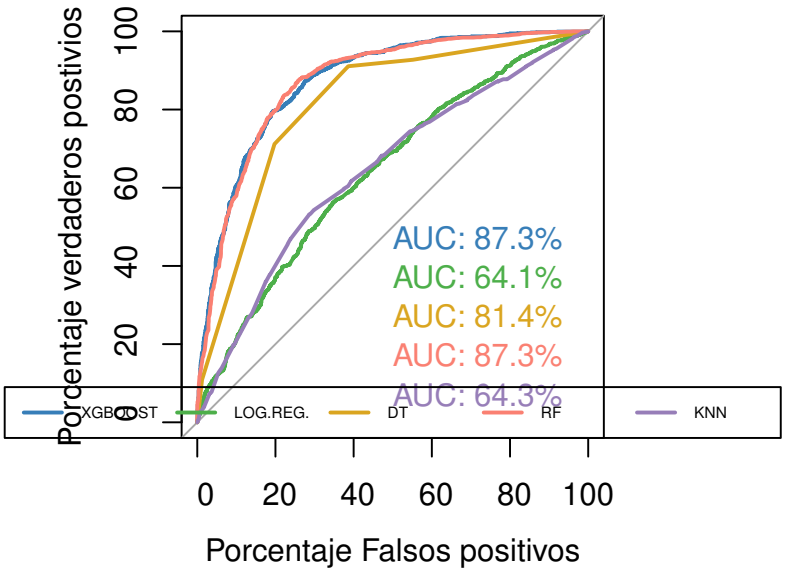
```
roc(df.test$deposit, prediction.rf, percent=TRUE, col="salmon",lwd= 2,
          print.auc =TRUE, add=TRUE,print.auc.y = 20,plot=TRUE,legend=TRUE)
```

```
Call:
roc.default(response = df.test$deposit, predictor = prediction.rf,     percent = TRUE, plot =

Data: prediction.rf in 1468 controls (df.test$deposit 0) < 1322 cases (df.test$deposit 1).
Area under the curve: 87.29%
```

```
roc(df.test$deposit, prediction.knn, percent=TRUE, col="#977eb8",lwd= 2,
          print.auc =TRUE, add=TRUE,print.auc.y = 10,plot=TRUE,legend=TRUE)
```

```
Call:
roc.default(response = df.test$deposit, predictor = prediction.knn,     percent = TRUE, plot

Data: prediction.knn in 1468 controls (df.test$deposit 0) < 1322 cases (df.test$deposit 1).
Area under the curve: 64.33%
```

```r
#prediction <- predict(tune.fit, df.test, type = 'prob')

legend("bottom",
       legend=c("XGBOOST", "LOG.REG.", "DT", "RF","KNN"),
       col=c("#377eb8", "#4daf4a", "goldenrod","salmon","#977eb8"),
       lwd=2, cex =.5, xpd = TRUE, horiz = TRUE)
```



Tarea

A la vista de los resultados. ¿Qué modelo es el más adecuado? Piensa sobre ello.

Modelo óptimo vs modelo útil

En este documento hemos aplicado una serie de modelos a un conjunto de datos. Nuestro objetivo es mostrar a los estudiantes varios métodos de ML. Nuestro objetivo nunca ha sido encontrar el modelo *óptimo*. Es decir, no buscamos el mejor modelo, sino un modelo *útil*. Un modelo que proporcione información y de ella podamos extraer conocimiento.

9 Reglas de asociación

Las reglas de asociación son una técnica fundamental en ML que se utiliza para descubrir patrones significativos y relaciones ocultas en conjuntos de datos, especialmente aquellos que involucran transacciones o **cestas de la compra**. Estas reglas desempeñan un papel crucial en la extracción de conocimiento de datos, y su aplicación abarca una amplia variedad de dominios.

Descubriendo Patrones Relevantes: Las reglas de asociación son un enfoque de ML diseñado para identificar patrones de co-ocurrencia entre elementos o características en conjuntos de datos. Estas reglas permiten revelar conexiones interesantes y relaciones significativas que a menudo pasan desapercibidas a simple vista. Los escenarios ideales para su aplicación incluyen, pero no se limitan a, datos de transacciones de compras de clientes, registros de usuarios en línea y registros de interacciones de productos.

Ejemplo

Supongamos que disponemos de un conjunto de datos de compras de clientes en una tienda en línea. Los datos pueden verse de la siguiente manera:

Transacción	Productos comprados
1	Pan, Leche, Huevos
2	Leche, Queso, Yogur
3	Pan, Leche, Mantequilla
4	Pan, Huevos
5	Leche, Huevos, Queso

En este conjunto de datos, cada fila representa una transacción y los productos comprados en esa transacción. Las reglas de asociación se utilizan para descubrir patrones de co-ocurrencia entre productos. Por ejemplo, podríamos encontrar las siguientes reglas de asociación:
Si un cliente compra Pan y Leche, entonces también compra Huevos. Si un cliente compra Leche y Queso, entonces también compra Yogur.

La Estructura de "Si... Entonces...": Como acabas de ver en el ejemplo, estas reglas de asociación se expresan típicamente en una estructura condicional "si… entonces…", donde se establece una relación entre los elementos o características. Esta estructura es fácilmente

interpretable y ofrece una base sólida para la toma de decisiones y la generación de recomendaciones.

Mediciones de Fuerza de Asociación: Para cuantificar la importancia de estas reglas, se utilizan dos métricas clave: el soporte (support) y la confianza (confidence). El soporte mide la frecuencia con la que aparece una asociación en el conjunto de datos, lo que indica su relevancia en términos de ocurrencia. La confianza, por otro lado, mide la probabilidad de que se cumpla una regla dada. Estas métricas ayudan a determinar la solidez y la utilidad de las reglas de asociación descubiertas. **Aplicaciones en Diversos Campos:** Las reglas de asociación tienen una amplia gama de aplicaciones en el mundo real. Desde el ámbito del comercio electrónico, donde se utilizan para generar recomendaciones de productos personalizadas, hasta la investigación de mercado, donde ayudan a identificar tendencias y patrones de consumo. También se aplican en la optimización de la colocación de productos en tiendas y la detección de anomalías en datos, como fraudes o comportamientos inusuales.

Algoritmos Especializados: Para extraer reglas de asociación, se utilizan algoritmos especializados como Apriori y FP-Growth. Estos algoritmos son capaces de manejar grandes conjuntos de datos y descubrir patrones complejos de manera eficiente.

9.1 Reglas de asociación en R

Vamos a emplear un ejemplo en R. El conjunto de datos `Groceries` contiene datos de ventas de una tienda de comestibles con 9835 transacciones y 169 artículos (grupos de productos).

```
library(arulesViz)
```

```
Loading required package: arules
```

```
Loading required package: Matrix
```

```

Attaching package: 'arules'
```

```
The following objects are masked from 'package:base':

    abbreviate, write
```

```
# Limitamos el número de decimales en la salida
 options(digits = 2)
```

```r
# reproducible
 set.seed(1234)

 data("Groceries")
 summary(Groceries)
```

```
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.026

most frequent items:
      whole milk other vegetables       rolls/buns             soda
            2513             1903             1809             1715
          yogurt          (Other)
            1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
  17   18   19   20   21   22   23   24   26   27   28   29   32
  29   14   14    9   11    4    6    1    1    1    1    3    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       2       3       4       6      32

includes extended item information - examples:
       labels  level2           level1
1 frankfurter sausage meat and sausage
2     sausage sausage meat and sausage
3  liver loaf sausage meat and sausage
```

El resumen muestra algunas estadísticas básicas del conjunto de datos. Por ejemplo, que el conjunto de datos es bastante disperso, con una densidad ligeramente superior al $2,6\%$, que la *`leche entera`* es el artículo más popular y que la `transacción media` contiene menos de 1.000 unidades.

A continuación, extraemos reglas de asociación mediante el algoritmo *Apriori* implementado en la librería `arules`.

```r
rules <- apriori(Groceries, parameter=list(support=0.001, confidence=0.5))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.5    0.1    1 none FALSE            TRUE       5   0.001      1
 maxlen target  ext
     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [5668 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
rules
```

```
set of 5668 rules
```

El resultado es un conjunto de 5668 reglas de asociación. Las tres reglas más importantes con respecto a la *medida lift*, una medida popular de la fuerza de las reglas, son:

```
inspect(head(rules, n = 3, by ="lift"))
```

```
    lhs                               rhs              support confidence
[1] {Instant food products, soda} => {hamburger meat} 0.0012  0.63
[2] {soda, popcorn}               => {salty snack}    0.0012  0.63
[3] {flour, baking powder}        => {sugar}          0.0010  0.56
    coverage lift count
[1] 0.0019   19   12
[2] 0.0019   17   12
[3] 0.0018   16   10
```

Aquí podemos ver el soporte, esto es, la relevancia en términos de la ocurrencia de cada regla, así como la confianza, o probablidad de que se cumpla.

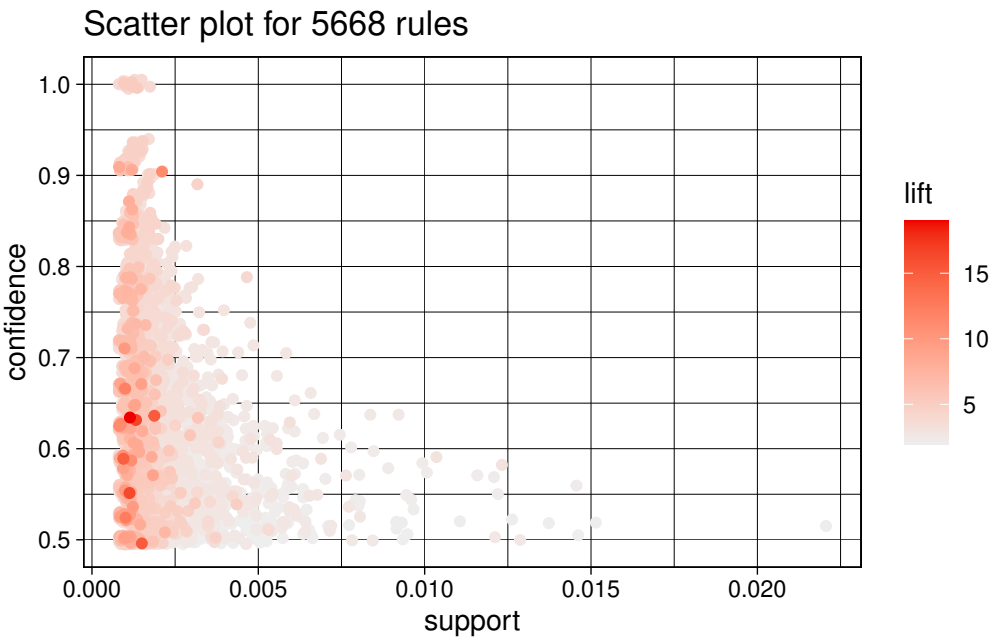
9.2 Visualizando las reglas

En conjuntos de datos reales, es imposible repasar manualmente el elevado número de reglas de asociación que se crean. ¡En el ejemplo anterior se habían creado 5668 reglas!

Una visualización sencilla de las reglas de asociación es utilizar un gráfico de dispersión con dos medidas de interés en los ejes.

```
plot(rules)
```

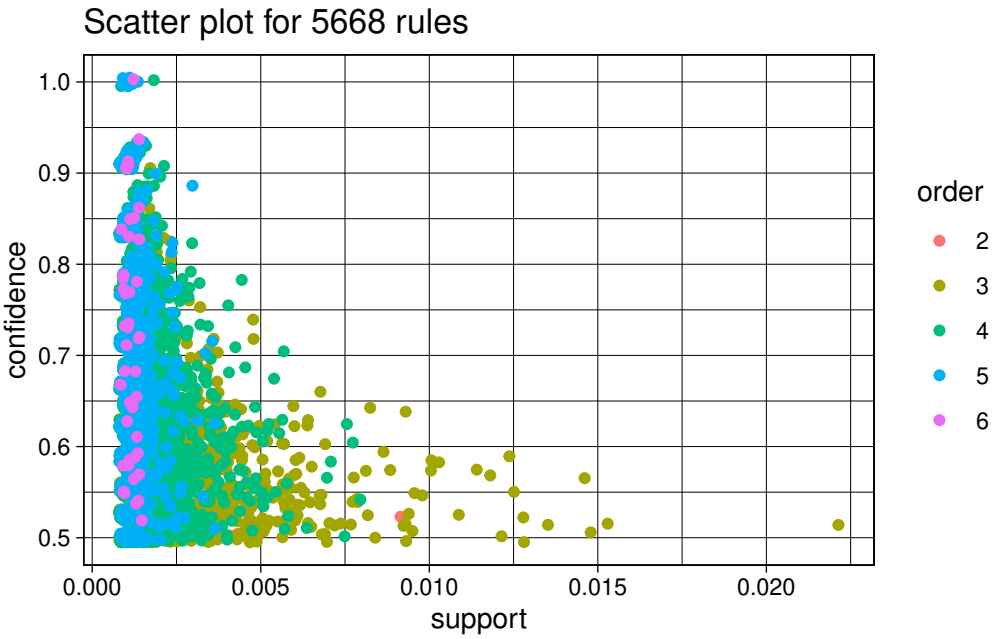
```
To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



Podemos ver que las normas con gran elevación (*lift*) suelen tener un soporte relativamente bajo. Las reglas más interesantes aparecen en la frontera *support/confidence*. Otro gráfico intersante es el siguiente. En este caso, , el soporte y la confianza se utilizan para los ejes X e Y, y el color de los puntos se utiliza para indicar el "*orden*", es decir, el número de elementos que contiene la regla.

```
plot(rules, method = "two-key plot")
```

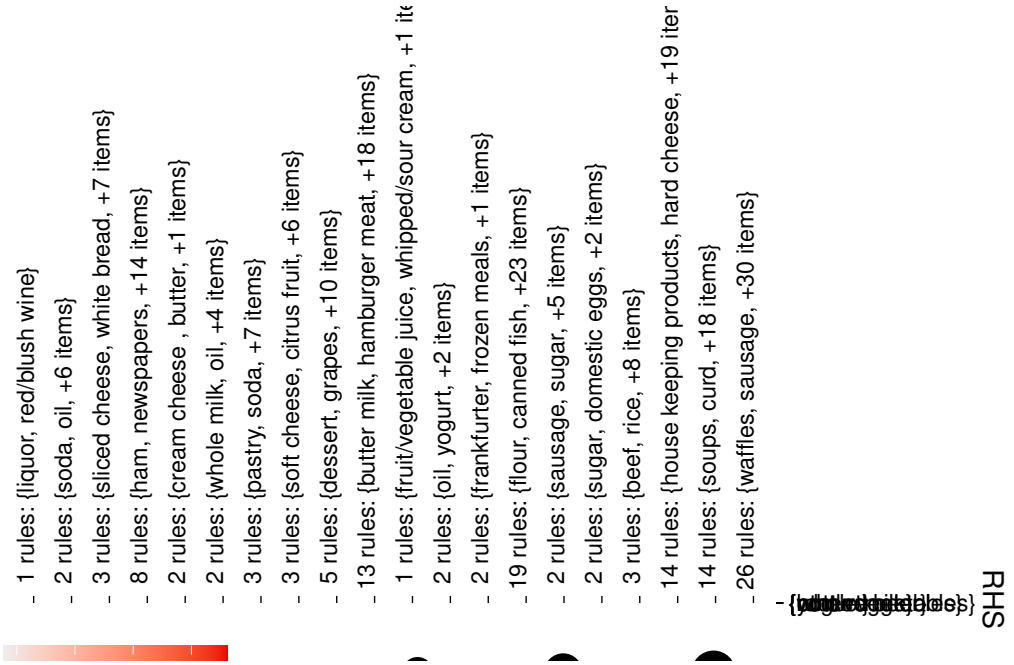
```
To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



Del gráfico se desprende claramente que el orden y el soporte tienen una relación inversa muy fuerte, lo que es un hecho conocido para las reglas de asociación.

Podemos visualizar grupos de reglas más específicamente:

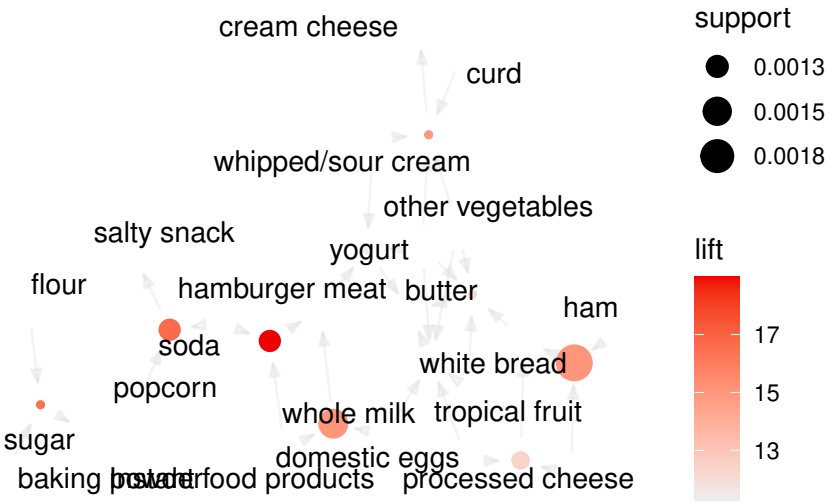
```
subrules <- rules[quality(rules)$confidence > 0.9]
plot(subrules,method="grouped")
```



El grupo de reglas más interesantes según (la medida por defecto) se muestran en la esquina superior izquierda del gráfico. Hay 3 reglas que contienen "*productos alimenticios instantáneos*" y hasta otros 2 elementos en el antecedente y el consecuente es "*carne de hamburguesa*".

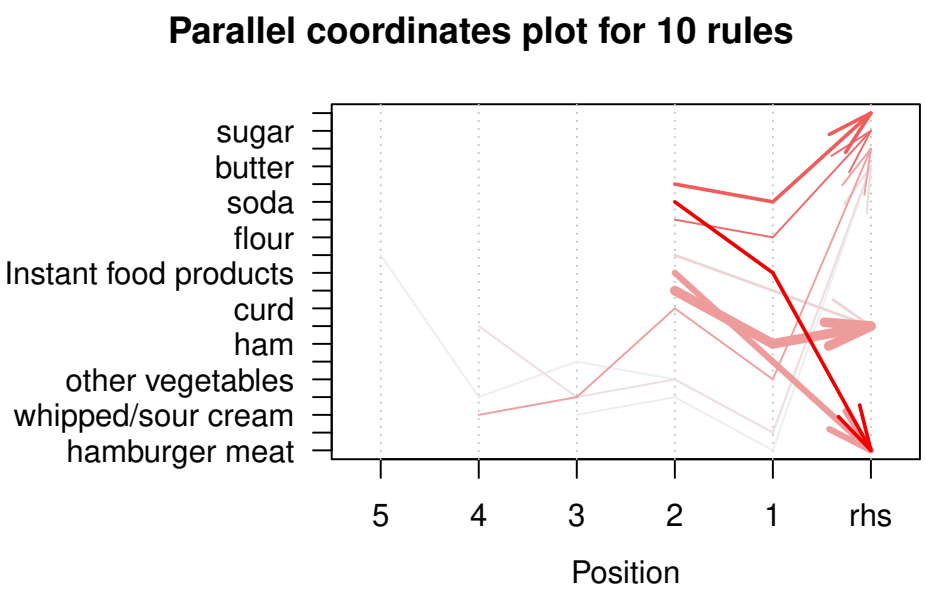
Finalmente, podemos visualizar las reglas de asociación mediante vértices y aristas en los que los vértices anotados con etiquetas de elementos representan elementos y los conjuntos de elementos o reglas se representan como un segundo conjunto de vértices. Los elementos se conectan con los conjuntos de elementos/reglas mediante flechas. En el caso de las reglas, las flechas que apuntan de los artículos a los vértices de las reglas indican los artículos del LHS y una flecha de una regla a un artículo indica el RHS. Las medidas de interés suelen añadirse al gráfico mediante el color o el tamaño de los vértices que representan los conjuntos de elementos/reglas. La visualización basada en gráficos ofrece una representación muy clara de las reglas, pero tienden a saturarse con facilidad, por lo que sólo son viables para conjuntos de reglas muy pequeños.

```
subrules2 <- head(rules, n = 10, by = "lift")
plot(subrules2, method = "graph")
```



La representación anterior se centra en cómo las reglas se componen de elementos individuales y muestra qué reglas comparten elementos.

```
plot(subrules2, method = "paracoord", control = list(reorder = TRUE))
```



10 Nuevas tendencias

10.1 Ética en el Aprendizaje Automático

La ética en ML es un tema crucial en la actualidad, ya que el uso de algoritmos y modelos de aprendizaje automático está cada vez más presente, a veces sin darnos cuenta, en nuestra vida cotidiana. La toma de decisiones automatizada basada en datos plantea **desafíos éticos y sociales** que deben ser abordados de manera responsable.

> Ética
>
> 1. Disciplina filosófica que estudia el bien y el mal y sus relaciones con la moral y el comportamiento humano.
> 2. Conjunto de costumbres y normas que dirigen o valoran el comportamiento humano en una comunidad.

La ética en ML se centra en garantizar que los sistemas de AI tomen **decisiones justas, imparciales y éticas**. Esto implica considerar la equidad en el tratamiento de diferentes grupos de personas, la transparencia en cómo se toman las decisiones y la privacidad de los datos utilizados en el entrenamiento de modelos.

Uno de los desafíos clave en la ética del ML es la **discriminación algorítmica**. Los modelos de ML pueden aprender **sesgos** presentes en los datos de entrenamiento, lo que resulta en decisiones discriminatorias en áreas como la selección de candidatos, la concesión de créditos o la aplicación de la ley. Una de las vías de trabajo e investigación en ML es la de tratar de mitigar estos sesgos y garantizar una toma de decisiones más justa.

La **transparencia** es otra preocupación importante. Comprender por qué un modelo de ML toma ciertas decisiones es esencial, especialmente en aplicaciones críticas como la atención médica. Los investigadores están desarrollando técnicas de explicabilidad, que veremos más adelante, que permiten a los usuarios comprender el razonamiento detrás de las decisiones de los algoritmos.

La **privacidad** de los datos también es un aspecto fundamental de la ética en ML. La recopilación y el uso de datos personales deben realizarse de manera ética y respetando las normativas de protección de datos. Garantizar la **seguridad** de los datos es esencial para prevenir brechas de privacidad.

10.2 Aprendizaje Automático Explicable

Los modelos computacionales han alcanzado niveles muy elevados de precisión y capacidad en la toma de decisiones. Sin embargo, a medida que estos modelos se vuelven más complejos, su falta de transparencia se ha convertido en un desafío crítico y probablemente, en uno de sus puntos débiles más críticos para su implantación en la sociedad. La necesidad de comprender cómo y por qué un modelo de ML toma ciertas decisiones se ha vuelto fundamental, especialmente en aplicaciones críticas, como el diagnóstico médico y la detección de fraudes financieros.

El aprendizaje máquina explicable (*XML*, "Explainable Machine Learning" en inglés) se refiere a la capacidad de los modelos de ML para proporcionar explicaciones claras y comprensibles de sus decisiones. Esto implica no solo producir resultados precisos, sino también presentar una justificación o una razón para cada predicción o clasificación realizada. La **transparencia** es esencial porque permite a los usuarios, ya sean médicos, reguladores o consumidores, confiar en las decisiones del modelo y tomar decisiones informadas en consecuencia.

Las técnicas de XML incluyen: La interpretabilidad y explicabilidad en el aprendizaje automático abarcan una variedad de modelos y métodos que permiten comprender cómo funcionan los modelos de machine learning y por qué toman ciertas decisiones. A continuación, proporcionamos una taxonomía de estos modelos y métodos:

i Métodos basados en reglas

- **Reglas de decisión**: Estos modelos generan reglas lógicas que explican el razonamiento detrás de las predicciones del modelo.
- **Árboles de decisión**: Los árboles muestran la secuencia de decisiones tomadas por el modelo en cada nodo, lo que facilita la interpretación.

i Métodos basados en características

- **Importancia de características**: Calcula la importancia de cada característica en el modelo, lo que permite identificar las variables más influyentes en las predicciones.
- **Análisis de efectos parciales**: Evalúa el impacto de una sola característica en las predicciones, manteniendo las demás constantes.

i Métodos basados en ejemplos

- **Prototipos**: Encuentra ejemplos representativos o prototipos de datos que explican cómo el modelo toma decisiones.
- **Casos de prueba**: Genera instancias que muestran cómo el modelo reacciona a

diferentes escenarios.

Métodos de aproximación

- **Modelos locales interpretables**: Crea modelos más simples (lineales, regresiones, etc.) en regiones locales del espacio de características para comprender decisiones en áreas específicas.
- **Regresiones lineales localmente ponderadas (LWLR)**: Asigna pesos a las instancias cercanas para ajustar una regresión lineal local.

Métodos basados en atención

- **Atención y atención saliente**: Modelos basados en atención destacan características o regiones de interés que influyen en las predicciones.
- **Redes neuronales con atención**: Las redes neuronales con mecanismos de atención permiten entender qué partes de la entrada son relevantes para la salida.

Métodos basados en métricas

- **Métricas de proximidad**: Evalúan la similitud entre entradas y cómo se relacionan con las predicciones.
- **SHAP (SHapley Additive exPlanations)**: Utiliza conceptos de teoría de juegos para asignar valores de importancia a las características.

Métodos basados en resumen

- **Reglas de decisión generadas por el modelo**: El modelo crea reglas que resumen su comportamiento.
- **Análisis de componentes**: Reduce la dimensión de los datos para visualizar y resumir características significativas.

Métodos de muestreo y generación de datos

- **Perturbación de datos**: Se modifican las características de entrada para entender cómo afectan a las predicciones.
- **Muestreo de datos contrapuestos**: Se generan ejemplos que muestran cómo las predicciones cambiarían si los datos fueran diferentes.

Métodos de comparación

- **Modelos interpretables frente a modelos de caja negra**: Compara modelos interpretables con modelos complejos en términos de rendimiento y capacidad de explicación.

Herramientas de visualización

- **Gráficos interactivos**: Visualizaciones que muestran cómo las características afectan a las predicciones.
- **Heatmaps y perfiles de importancia**: Muestran la importancia de las características en un formato visual.

Usemos, a modo de ejemplo, alguna de estas técnicas en un ejemplo práctico. Buscamos una interpretación del modelo mediante la técnica **SHAP**. El objetivo de SHAP es proporcionar explicaciones claras y coherentes para las predicciones del modelo de ML, un modelo de *Random Forest* en el ejemplo. La técnica se basa en el principio de que cada característica o atributo de entrada de un modelo contribuye de alguna manera a la predicción final. SHAP cuantifica esta contribución para cada característica, permitiendo una interpretación más profunda de cómo el modelo llega a sus conclusiones.

```
library(DALEX)
library(dplyr)
library(tidyverse)
library(randomForest)

bank = read.csv('https://raw.githubusercontent.com/rafiag/DTI2020/main/data/bank.csv')
dim(bank)
```

```
[1] 11162    17
```

```
bank=as.tibble(bank)

# Parciticionamos los datos
set.seed(2138)
n=dim(bank)[1]
indices=seq(1:n)
indices.train=sample(indices,size=n*.5,replace=FALSE)
indices.test=sample(indices[-indices.train],size=n*.25,replace=FALSE)
indices.valid=indices[-c(indices.train,indices.test)]
```

```
bank.train=bank[indices.train,]
bank.test=bank[indices.test,]

set.seed(128)
df <- bank.train %>%
     mutate(fmarital=as.factor(marital))%>%
      select(age,job,housing,fmarital,education,duration,poutcome,balance,deposit)

df.test <- bank.test %>%
      mutate(fmarital=as.factor(marital))%>%
      select(age,job,housing,fmarital,education,duration,poutcome,balance,deposit)

rf <- randomForest(as.factor(deposit)~., data=df, importance=TRUE,proximity=TRUE)

new.test <- df.test[1,-9]
new.test
```

```
# A tibble: 1 x 8
    age job        housing fmarital education duration poutcome balance
  <int> <chr>      <chr>   <fct>    <chr>        <int> <chr>      <int>
1    47 technician yes     single   secondary      973 unknown     -239
```

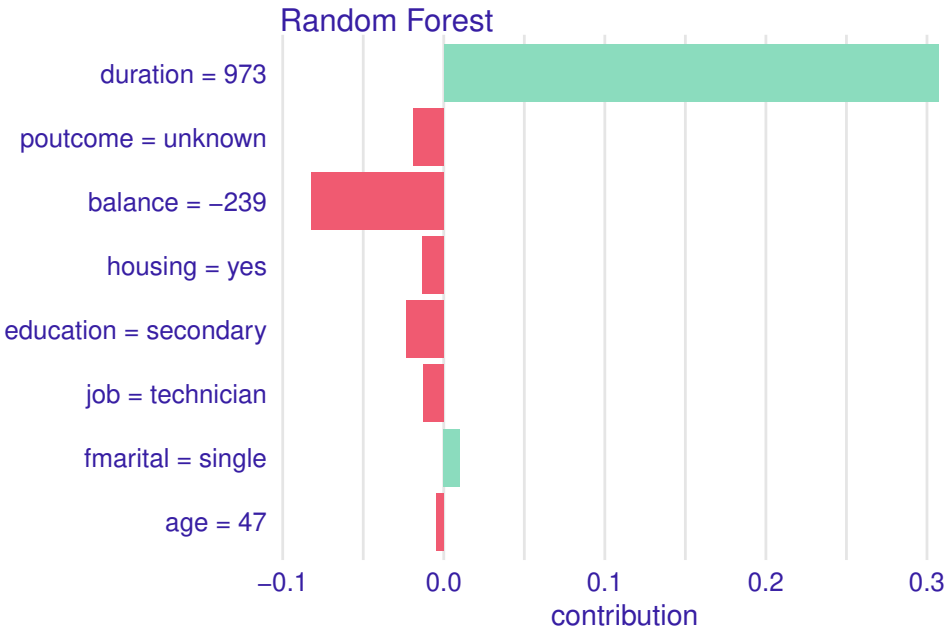
```
explain.rf <- DALEX::explain(model=rf,data=df[,-9],y=df[,9]=="yes",label="Random Forest")
```

```
Preparation of a new explainer is initiated
  -> model label       :  Random Forest
  -> data              :  5581  rows  8  cols
  -> data              :  tibble converted into a data.frame
  -> target variable   :  5581  values
  -> predict function  :  yhat.randomForest  will be used (  default  )
  -> predicted values  :  No value for predict function target column. (  default  )
  -> model_info        :  package randomForest , ver. 4.7.1.1 , task classification (  defaul
  -> model_info        :  Model info detected classification task but 'y' is a matrix . Conve
  -> predicted values  :  numerical, min =  0 , mean =  0.4759208 , max =  1
  -> residual function :  difference between y and yhat (  default  )
  -> residuals         :  numerical, min =  -0.65 , mean =  0.003025623 , max =  0.764
  A new explainer has been created!
```

```
predict(explain.rf, new.test)
```

```
[1] 0.638
```

```
shap.new <- predict_parts(explainer = explain.rf,
                          new_observation = new.test,
                                     type = "shap",
                                        B = 25)
plot(shap.new,show_boxplots=FALSE)
```



La predicción para la nueva observación es `yes` (*probabilidad* > 0.5). En el gráfico podemos observar los llamados valores *Shapley*, y encontramos los motivos para dicha predicción. Vemos como las dos variables que están asociadas con esa predicción positiva son la larga duración de la llamada y su estado civil (`single`). El resto de valores de las variables para ese individuo estarían indicando que la respuesta (predicha) es `no`.

Si repetimos el experimento para el segundo de los datos en `test` podemos observar que la corta duración de la llamada, el no saber qué resultado tuvo la campaña de marketing anterior y el no tener casa en propiedad, son las variables más asociadas a la predicción `no` para el valor de la variable respuesta.

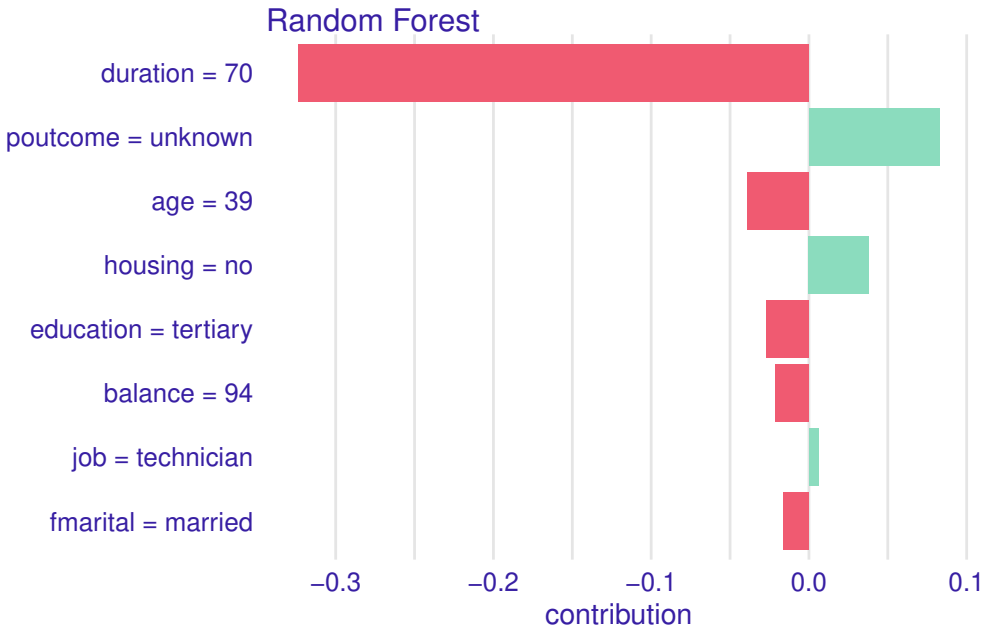
```
new.test <- df.test[2,-9]
new.test
```

```
# A tibble: 1 x 8
    age job        housing fmarital education duration poutcome balance
  <int> <chr>      <chr>   <fct>    <chr>        <int> <chr>      <int>
1    39 technician no      married  tertiary        70 unknown       94
```

```
predict(explain.rf, new.test)
```

```
[1] 0.176
```

```
shap.new <- predict_parts(explainer = explain.rf,
                          new_observation = new.test,
                                     type = "shap",
                                        B = 25)
plot(shap.new,show_boxplots=FALSE)
```



Puedes encontrar una taxonomía completa aquí: (Schwalbe and Finzel 2023) y aquí (Molnar 2020). Es importante recordar que la elección de un método o modelo de interpretabilidad

depende del contexto y los objetivos del análisis. La interpretabilidad y explicabilidad son esenciales para comprender y confiar en los modelos de ML, especialmente en aplicaciones críticas donde las decisiones tienen un alto impacto.

Además, la **regulación** y las directrices **éticas** están comenzando a exigir la explicabilidad en ciertas aplicaciones críticas para garantizar la rendición de cuentas y la equidad.

El XML no solo mejora la confianza en los modelos de ML, sino que también permite una colaboración más efectiva entre humanos y máquinas. Al comprender las razones detrás de las decisiones del modelo, los expertos pueden realizar ajustes cuando sea necesario y mejorar aún más la precisión y la utilidad de la IA en una variedad de campos.

10.2.1 Contrafácticos

En el contexto de la AI y la explicabilidad de modelos de ML, los contrafácticos (o contrafactuales) son una técnica utilizada para evaluar y comprender cómo los modelos toman decisiones al considerar escenarios hipotéticos o situaciones alternativas. Los contrafácticos son preguntas o declaraciones que plantean "¿Qué habría ocurrido si…?" con el propósito de analizar cómo un modelo habría respondido si las condiciones o las entradas hubieran sido diferentes. Esta técnica se utiliza para obtener información sobre cómo un modelo realiza sus predicciones y para explicar su razonamiento. Además, los contrafácticos contribuyen a garantizar la transparencia y la equidad en las aplicaciones de AI.

Algunas características clave de los contrafácticos son:

- **Evaluación de modelos**: Los contrafácticos permiten evaluar la coherencia y el comportamiento de un modelo de ML. Al plantear preguntas hipotéticas, se puede determinar si el modelo da respuestas razonables y lógicas en diferentes situaciones.
- **Comprensión de decisiones**: Los contrafácticos ayudan a comprender por qué un modelo toma decisiones específicas. Al explorar cómo habría respondido el modelo en diferentes circunstancias, se puede inferir qué factores influyen en sus predicciones.
- **Validación de decisiones**: Los contrafácticos se utilizan para validar o justificar las decisiones de un modelo. Al proporcionar explicaciones sobre cómo habría actuado en situaciones hipotéticas, se puede respaldar la idoneidad de las decisiones tomadas.
- **Sensibilidad a entradas**: Los contrafácticos también ayudan a determinar la sensibilidad de un modelo a diferentes entradas. Pueden revelar si el modelo realiza predicciones consistentes cuando las variables de entrada cambian.
- **Detección de sesgos y fallos**: Al plantear contrafácticos, es posible identificar sesgos o problemas en el modelo. Si las respuestas del modelo en situaciones hipotéticas parecen sesgadas o incoherentes, esto puede señalar áreas problemáticas que requieren atención.

Por ejemplo, en un sistema de crédito que utiliza un modelo de ML para tomar decisiones de aprobación, se podría plantear la pregunta contrafáctica: "*¿Habría aprobado el crédito si el ingreso del solicitante fuera un 10% más alto?*" Esto permite evaluar si el modelo es sensible a los cambios en los ingresos y si realiza decisiones justas y coherentes.

Si quieres aprender un poquito más sobre contrafácticos, prueba esta lectura (Fernández et al. 2020).

10.3 Deriva conceptual

En el mundo del ML, la deriva conceptual es un fenómeno crítico que se refiere a la **evolución de los datos** y la cambiante naturaleza de las relaciones entre las variables a lo largo del tiempo. A medida que se acumulan nuevos datos, los patrones subyacentes pueden cambiar, lo que puede afectar drásticamente la eficacia de los modelos de previamente entrenados.

Este fenómeno es particularmente relevante en aplicaciones en las que los datos son dinámicos y cambian con el tiempo, como el **monitoreo de sistemas** en tiempo real, el análisis de tendencias de mercado o la detección de anomalías en redes informáticas. En tales casos, los modelos de ML que se entrenan inicialmente en un conjunto de datos pueden volverse obsoletos o ineficaces a medida que las condiciones cambian y las relaciones entre las variables evolucionan.

La deriva conceptual puede manifestarse de varias formas:

- cambios en la **distribución de los datos**
- cambios en la **importancia relativa de las características**
- cambios en las **relaciones entre variables**
- aparición de **nuevos patrones** que antes no estaban presentes.

Reconocer y abordar la deriva conceptual es esencial para mantener la precisión y la relevancia de los modelos a lo largo del tiempo.

Para abordar la deriva conceptual, es necesario implementar técnicas de **adaptación** de modelos que permitan a los algoritmos de ML ajustarse a los cambios en los datos. Esto puede implicar:

- **reentrenamiento periódico** de los modelos
- **monitorización constante de la calidad** del modelo
- **identificación de los momentos** en que se produce una deriva conceptual significativa.

La capacidad de adaptarse a esta deriva es esencial para garantizar que los modelos sigan siendo efectivos en entornos dinámicos y cambiantes.

La deriva conceptual es un desafío continuo en ML y destaca la importancia de mantener los modelos actualizados y capaces de adaptarse a entornos de datos en constante evolución.

Tener un equipo de científicos de datos dedicados para la monitorización de modelos en producción es esencial para garantizar que los modelos sigan siendo eficaces, justos, cumplidores y útiles a lo largo del tiempo. Además, este equipo puede proporcionar una capa adicional de seguridad y confiabilidad en el uso de la AI y el ML en aplicaciones del mundo real.

Rendimiento continuo

Los modelos de ML no son estáticos; tal y como se ha indicado, su rendimiento puede degradarse con el tiempo debido a cambios en los datos de entrada o en el entorno. Un equipo de científicos de datos puede monitorear de cerca el rendimiento del modelo y tomar medidas para mantener su eficacia.

Detección de deriva de datos

Hemos visto que la deriva de datos es un fenómeno en el que las características de los datos de entrada cambian con el tiempo. Esto puede llevar a predicciones inexactas y, en algunos casos, a resultados desastrosos. Los científicos de datos pueden identificar la deriva de datos y ajustar o reentrenar el modelo en consecuencia.

Gestión de sesgo y equidad

La equidad y la falta de sesgo son consideraciones críticas al implementar modelos en producción, especialmente en áreas como la toma de decisiones y la selección de personal. Un equipo de científicos de datos puede evaluar y abordar problemas de sesgo y equidad de manera proactiva.

Actualización de modelos

Con el tiempo, es probable que se desee mejorar los modelos existentes o desarrollar nuevos. Los científicos de datos pueden dirigir el proceso de actualización y garantizar que los modelos reflejen las necesidades cambiantes del negocio.

Supervisión de errores y problemas técnicos

Los errores pueden ocurrir, y es fundamental identificarlos y corregirlos rápidamente. Los científicos de datos pueden implementar sistemas de monitorización de errores y procesos de solución de problemas.

Cumplimiento normativo

En sectores regulados, como la atención médica o las finanzas, se requiere un cumplimiento estricto con regulaciones específicas. Un equipo de científicos de datos puede garantizar que los modelos cumplan con las normativas vigentes.

Comunicación con partes interesadas

Los resultados de los modelos deben comunicarse de manera efectiva a partes interesadas que pueden no estar familiarizadas con la ciencia de datos. Los científicos de datos pueden actuar como intermediarios y explicar los resultados de manera comprensible.

Optimización de recursos

Los recursos computacionales y el tiempo de los científicos de datos son valiosos. Un equipo de científicos de datos puede optimizar la asignación de recursos, enfocándose en los aspectos más críticos de la monitorización y ajuste de modelos.

Puedes aprender sobre la deriva conceptual aquí: (Lu et al. 2018).

11 Conclusiones

En este capítulo final, te proponemos reflexionar sobre las principales lecciones aprendidas a lo largo del curso "Aprendizaje Automático I" y destacaremos la importancia de las técnicas abordadas en la formación de futuros profesionales en Ciencia e Ingeniería de Datos. Además, alentaremos a los estudiantes a continuar explorando y ampliando sus conocimientos en cursos posteriores.

11.1 Recapitulación de Técnicas Clave

En las secciones anteriores, hemos explorado un amplio espectro de técnicas de ML. Desde la preparación de datos hasta las técnicas de reducción de dimensionalidad, y desde el aprendizaje no supervisado hasta el supervisado, hemos adquirido una comprensión sólida de los fundamentos de este emocionante campo.

11.2 Aplicaciones Prácticas

A lo largo del curso, hemos analizado cómo estas técnicas se aplican en situaciones del mundo real. Desde la identificación de patrones en grandes conjuntos de datos hasta la creación de modelos predictivos, estas habilidades son esenciales en la resolución de problemas complejos en la ciencia de datos.

11.3 El Poder de la Exploración Continua

Aunque hemos abordado una variedad de técnicas, el ML es un campo en constante evolución. Nuevas tendencias y algoritmos emergen regularmente, y los desafíos que enfrentaréis en el futuro requerirán enfoques innovadores.

11.4 La Invitación a la Exploración

Este curso es solo el comienzo de un (esperemos) apasionante viaje en el mundo del ML. En cursos posteriores vais a profundizar en áreas específicas, como Deep Learning, Procesamiento de Lenguaje Natural o Aprendizaje por Refuerzo. Estas extensiones no solo ampliarán vuestro conjunto de habilidades, sino que también os permitirán abordar problemas más complejos y fascinantes.

11.5 Un Futuro en Aprendizaje Automático

El ML es una disciplina dinámica y esencial en la era de los datos. Os animamos a continuar explorando, experimentando y aplicando lo que habéis aprendido. El viaje apenas comienza, y vuestra participación activa en la evolución de este campo jugará un papel crucial en la resolución de desafíos futuros y en la creación de soluciones innovadoras.

¡Buena suerte!

Bibliografía

Agresti, Alan. 2015. *Foundations of Linear and Generalized Linear Models*. John Wiley & Sons.

Bellman, Richard Ernest. 1978. "Artificial Intelligence: Can Computers Think?" *(No Title)*.

Bellman, Richard, and Robert Kalaba. 1961. "Reduction of Dimensionality, Dynamic Programming, and Control Processes."

Bradshaw, Shannon, Eoin Brazil, and Kristina Chodorow. 2019. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media.

Cover, Thomas, and Peter Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13 (1): 21–27.

Dixit, Bharvi, Rafał Kuć, Marek Rogoziński, and Saurabh Chhajed. 2017. "Elasticsearch: A Complete Guide."

Fernández, Rubén R, Isaac Martín De Diego, Víctor Aceña, Alberto Fernández-Isabel, and Javier M Moguerza. 2020. "Random Forest Explainability Using Counterfactual Sets." *Information Fusion* 63: 196–207.

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge university press.

Fox, John, and Sanford Weisberg. 2018. *An r Companion to Applied Regression*. Sage publications.

Hao, Jiangang, and Tin Kam Ho. 2019. "Machine Learning Made Easy: A Review of Scikit-Learn Package in Python Programming Language." *Journal of Educational and Behavioral Statistics* 44 (3): 348–61.

Harrington, Jan L. 2016. *Relational Database Design and Implementation*. Morgan Kaufmann.

Hartigan, John A, and Manchek A Wong. 1979. "Algorithm AS 136: A k-Means Clustering Algorithm." *Journal of the Royal Statistical Society. Series c (Applied Statistics)* 28 (1): 100–108.

Hastie, Trevor, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer.

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

John, George H, Ron Kohavi, and Karl Pfleger. 1994. "Irrelevant Features and the Subset Selection Problem." In *Machine Learning Proceedings 1994*, 121–29. Elsevier.

Kelleher, John D, Brian Mac Namee, and Aoife D'arcy. 2020. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*.

MIT press.
Kelleher, John D, and Brendan Tierney. 2018. *Data Science*. MIT Press.
Kurzweil, Ray, Robert Richter, Ray Kurzweil, and Martin L Schneider. 1990. *The Age of Intelligent Machines*. Vol. 580. MIT press Cambridge.
Lu, Jie, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. "Learning Under Concept Drift: A Review." *IEEE Transactions on Knowledge and Data Engineering* 31 (12): 2346–63.
McCullagh, Peter. 2019. *Generalized Linear Models*. Routledge.
Meier, Andreas, and Michael Kaufmann. 2019. *SQL & NoSQL Databases*. Springer.
Molnar, Christoph. 2020. *Interpretable Machine Learning*. Lulu. com.
Murphy, Kevin P. 2012. *Machine Learning: A Probabilistic Perspective*. MIT press.
Nelson, Jeremy. 2016. *Mastering Redis*. Packt Publishing Ltd.
Osmani, Addy. 2012. *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*. " O'Reilly Media, Inc.".
Oualline, Steve. 2003. *Practical c++ Programming*. " O'Reilly Media, Inc.".
Poole, David I, Randy G Goebel, and Alan K Mackworth. 1998. *Computational Intelligence*. Vol. 1. Oxford University Press Oxford.
Ra, YG, HA Kuno, and Elke A Rundensteiner. 1994. *A Flexible Object-Oriented Database Model and Implementation for Capacity-Augmenting Views*. Citeseer.
Russell, Stuart J. 2010. *Artificial Intelligence a Modern Approach*. Pearson Education, Inc.
Schwalbe, Gesina, and Bettina Finzel. 2023. "A Comprehensive Taxonomy for Explainable Artificial Intelligence: A Systematic Survey of Surveys on Methods and Concepts." *Data Mining and Knowledge Discovery*, 1–59.
Tukey, John W et al. 1977. *Exploratory Data Analysis*. Vol. 2. Reading, MA.
Webber, Jim. 2012. "A Programmatic Introduction to Neo4j." In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, 217–18.
Weiss, Mark Allen. 2012. *Data Structures and Algorithm Analysis in Java*. Pearson Education, Inc.
Winston, Patrick Henry. 1992. *Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc.
Wirth, Rüdiger, and Jochen Hipp. 2000. "CRISP-DM: Towards a Standard Process Model for Data Mining." In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 1:29–39. Manchester.
Xu, Dongkuan, and Yingjie Tian. 2015. "A Comprehensive Survey of Clustering Algorithms." *Annals of Data Science* 2: 165–93.
Agresti, Alan. 2015. *Foundations of Linear and Generalized Linear Models*. John Wiley & Sons.
Bellman, Richard Ernest. 1978. "Artificial Intelligence: Can Computers Think?" *(No Title)*.
Bellman, Richard, and Robert Kalaba. 1961. "Reduction of Dimensionality, Dynamic Programming, and Control Processes."
Bradshaw, Shannon, Eoin Brazil, and Kristina Chodorow. 2019. *MongoDB: The Definitive*

Guide: Powerful and Scalable Data Storage. O'Reilly Media.

Cover, Thomas, and Peter Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13 (1): 21–27.

Dixit, Bharvi, Rafał Kuć, Marek Rogoziński, and Saurabh Chhajed. 2017. "Elasticsearch: A Complete Guide."

Fernández, Rubén R, Isaac Martín De Diego, Víctor Aceña, Alberto Fernández-Isabel, and Javier M Moguerza. 2020. "Random Forest Explainability Using Counterfactual Sets." *Information Fusion* 63: 196–207.

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge university press.

Fox, John, and Sanford Weisberg. 2018. *An r Companion to Applied Regression*. Sage publications.

Hao, Jiangang, and Tin Kam Ho. 2019. "Machine Learning Made Easy: A Review of Scikit-Learn Package in Python Programming Language." *Journal of Educational and Behavioral Statistics* 44 (3): 348–61.

Harrington, Jan L. 2016. *Relational Database Design and Implementation*. Morgan Kaufmann.

Hartigan, John A, and Manchek A Wong. 1979. "Algorithm AS 136: A k-Means Clustering Algorithm." *Journal of the Royal Statistical Society. Series c (Applied Statistics)* 28 (1): 100–108.

Hastie, Trevor, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer.

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

John, George H, Ron Kohavi, and Karl Pfleger. 1994. "Irrelevant Features and the Subset Selection Problem." In *Machine Learning Proceedings 1994*, 121–29. Elsevier.

Kelleher, John D, Brian Mac Namee, and Aoife D'arcy. 2020. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT press.

Kelleher, John D, and Brendan Tierney. 2018. *Data Science*. MIT Press.

Kurzweil, Ray, Robert Richter, Ray Kurzweil, and Martin L Schneider. 1990. *The Age of Intelligent Machines*. Vol. 580. MIT press Cambridge.

Lu, Jie, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. "Learning Under Concept Drift: A Review." *IEEE Transactions on Knowledge and Data Engineering* 31 (12): 2346–63.

McCullagh, Peter. 2019. *Generalized Linear Models*. Routledge.

Meier, Andreas, and Michael Kaufmann. 2019. *SQL & NoSQL Databases*. Springer.

Molnar, Christoph. 2020. *Interpretable Machine Learning*. Lulu. com.

Murphy, Kevin P. 2012. *Machine Learning: A Probabilistic Perspective*. MIT press.

Nelson, Jeremy. 2016. *Mastering Redis*. Packt Publishing Ltd.

Osmani, Addy. 2012. *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide*. " O'Reilly Media, Inc.".

Oualline, Steve. 2003. *Practical c++ Programming*. " O'Reilly Media, Inc.".

Poole, David I, Randy G Goebel, and Alan K Mackworth. 1998. *Computational Intelligence*. Vol. 1. Oxford University Press Oxford.

Ra, YG, HA Kuno, and Elke A Rundensteiner. 1994. *A Flexible Object-Oriented Database Model and Implementation for Capacity-Augmenting Views*. Citeseer.

Russell, Stuart J. 2010. *Artificial Intelligence a Modern Approach*. Pearson Education, Inc.

Schwalbe, Gesina, and Bettina Finzel. 2023. "A Comprehensive Taxonomy for Explainable Artificial Intelligence: A Systematic Survey of Surveys on Methods and Concepts." *Data Mining and Knowledge Discovery*, 1–59.

Tukey, John W et al. 1977. *Exploratory Data Analysis*. Vol. 2. Reading, MA.

Webber, Jim. 2012. "A Programmatic Introduction to Neo4j." In *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, 217–18.

Weiss, Mark Allen. 2012. *Data Structures and Algorithm Analysis in Java*. Pearson Education, Inc.

Winston, Patrick Henry. 1992. *Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc.

Wirth, Rüdiger, and Jochen Hipp. 2000. "CRISP-DM: Towards a Standard Process Model for Data Mining." In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 1:29–39. Manchester.

Xu, Dongkuan, and Yingjie Tian. 2015. "A Comprehensive Survey of Clustering Algorithms." *Annals of Data Science* 2: 165–93.