Facultad de Ciencias Económicas y Empresariales

Análisis de datos financieros con técnicas de Machine Learning

Autor: Álvaro Campos Martín

Director: María Coronado Vaca

MADRID | Junio 2019

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

RESUMEN

El Machine Learning es un fenómeno que surgió a mediados del siglo XX, pero cuyo auge ha tenido lugar en la última década. El radical incremento en la generación de datos y la capacidad computacional, aunado a la reducción del coste de almacenamiento, han permitido la rápida consolidación de esta tecnología. Mediante la confección de dos modelos de Aprendizaje Supervisado, y dos de Aprendizaje No Supervisado, se pondrá de manifiesto la relevancia de esta tecnología en el actual paradigma social.

Palabras clave: Inteligencia Artificial, Machine Learning, Aprendizaje Supervisado, Aprendizaje No Supervisado, finanzas, clasificación, árbol de decisión, random forest, clustering, clustering jerárquico, k-means clustering.

ABSTRACT

The phenomenon of Machine Learning was born in the mid-20th century, but it´s booming did not take place until this last decade. The rapid consolidation of this technology has been underpinned by a radical increase in data generation and computational capacity, as well as a decrease in the cost of data storage. The relevance of this technology in the actual social paradigm will be explained through two Supervised Learning models and two Unsupervised Learning models.

Key words: Artificial Intelligence, Machine Learning, Supervised Learning, Unsupervised Learning, finance, classification, decision tree, random forest, clustering, hierarchical clustering, k-means clustering.

ABREVIATURAS

AGI	Artificial General Intelligence.
AI	Artificial Intelligence.
ANI	Artificial Narrow Intelligence.
ASI	Artificial Super Intelligence.
AUC	Area Under the Curve.
CAPM	Capital Asset Pricing Model.
ID3	Iterative Dichotomiser 3.
IoT	Internet of Things.
LCM	Línea del Mercado de Capitales.
MAE	Mean Absolute Error.
MIT	Massachusetts Institute of Technology.
ML	Machine Learning.
MSE	Mean Square Error.
ROC	Receiver Operating Characteristics.
SAS	Statistical Analysis System.
SL	Supervised Learning.
SSL	Semi-Supervised Learning.
UL	Unsupervised Learning

1. INTRODUCCIÓN

1.1. OBJETIVOS

Con este trabajo se pretende exponer el paradigma actual del Machine Learning, explicando los motivos por los cuales se ha llegado a la situación en la que nos encontramos actualmente, así como las tendencias futuras.

Se busca que un lector sin ningún tipo de conocimiento previo en la materia, sea capaz de comprender cómo se construyen los modelos que se emplean en este trabajo, siendo capaz de formar conclusiones propias en base a los resultados obtenidos.

1.2. JUSTIFICACIÓN DEL TEMA OBJETO DE ESTUDIO

En un mundo con clara vocación por lo digital, con ineludible tendencia por la automatización y optimización de procesos, las posibilidades del Machine Learning son infinitas. El contexto que nos rodea está plagado de este tipo de algoritmos, aunque no seamos conscientes de ello. Desde un coche autónomo de Tesla, hasta un juego de ajedrez que permite al usuario competir contra la máquina, es prácticamente imposible no interactuar con uno de estos modelos a día de hoy.

Dada la repercusión social de este fenómeno, se ha buscado estudiar su impacto en la industria financiera en concreto. Esta industria cada vez va siendo más consciente del potencial de las nuevas tecnologías. Desde hace tiempo, las instituciones financieras han indagado en las posibles aplicaciones de la tecnología *Blockchain*, la Inteligencia Artificial o el Machine Learning en los campos de mejora de la experiencia del consumidor, entrada en nuevos mercados o maximización de ingresos (optimizando sus esfuerzos regulatorios).

1.3. METODOLOGÍA

Para la realización de la parte teórica de este trabajo, se ha realizado una lectura exhaustiva de la literatura relativa a la materia, además de diversas consultas a expertos en materia de Inteligencia Artificial y Machine Learning.

Para la construcción de los algoritmos, se han realizado cursos de programación en lenguaje R, y se ha contado con la asesoría de dos programadores.

1.4. ESTRUCTURA

El siguiente trabajo comprende siete secciones. En la primera, se establecen los objetivos que se pretenden alcanzar y la metodología empleada para lograrlos, así como las razones que han motivado la elección de este tema.

En la segunda, se lleva a cabo una contextualización del fenómeno de la Inteligencia Artificial, desde su origen en los 1950 hasta nuestros días, analizando los motivos que han dado pie a este auge.

En la tercera, se profundiza en el Machine Learning, distinguiendo entre los distintos tipos de aprendizaje.

En la cuarta, se ahonda en el tipo de aprendizaje supervisado, a fin de asegurar que el lector comprende las nociones técnicas necesarias para entender el proceso de construcción de un algoritmo supervisado.

En la quinta, se explican los modelos supervisados y no supervisados que se construyen, con el objetivo de que el lector sea capaz de formar su opinión en cuanto la validez de los resultados obtenidos.

En la sexta, se lleva a cabo un análisis de crédito mediante dos modelos clasificatorios de árbol de decisión y *random forest*, cuyos resutados serán comparados.

En la séptima, se analizan las empresas que componen el índice IBEX 35, mediante dos modelos *clustering*, atendiendo a los criterios de rendimiento medio y volatilidad.

2. CONTEXTUALIZACIÓN DE LA INTELIGENCIA ARTIFICIAL Y DEL MACHINE LEARNING

2.1. INTRODUCCIÓN

Según el SAS Institute[MCV1][1], la Inteligencia Artificial ("AI" por sus siglas en inglés[2]) [MCV2][A3][A4]es un área de lo que conocemos como computer science[MCV5], que permite que las máquinas sean capaces de aprender de su propia

[1] Instituto desarrollador de software analítico dedicado al análisis y gestión de datos para optimizar el proceso de toma de decisiones.
[2] A lo largo del trabajo, se utilizarán ambas expresiones indistintamente.

experiencia, ajustándose a nuevos *inputs*, y llevando a cabo actividades propias de un ser humano [Thompson *et al* (2019)].

Muchas de las actividades de las que oímos hablar hoy en día, desde la detección del detección del fraude bancario hasta ordenadores capaces de jugar al ajedrez, dependen de lo que se conoce como Machine Learning[3] [MCV6](sub-sector de la AI). A través del uso de este tipo de tecnologías[MCV7], podemos entrenar a un ordenador para que sea capaz de cumplir tareas específicas, mediante el procesamiento de cantidades de datos [MCV8] ingentes y la identificación de patrones en los mismos.

A pesar de que gran parte de la sociedad piensa que la AI es un fenómeno de creación reciente, lo cierto es que lleva siendo estudiado desde hace mucho tiempo.

En la segunda mitad del siglo XIX, la ciencia ficción comienza a familiarizar al mundo con el concepto de robots con AI. Empezó con el Hombre de Hojalata de "El Mago de Oz" y continuó con el Hombre de Hierro de la novela de Ted Hughes. De esta manera, en la década de los 50s, exisitía ya una generación de matemáticos, científicos y filósofos con la idea de AI asimilada culturalmente.

El término como tal fue acuñado por primera vez en 1956 por el científico John McCarthy[MCV9]. Definió el concepto como "la ciencia e ingeniería de hacer máquinas inteligentes" [Guillén (2016)] [MCV10][A11]. Sin embargo, el viaje para entender si las máquinas pueden realmente pensar comenzó antes, con el Test de Turing (o Imitation Game), creado por el matemático británico Alan Turing. Turing (1950), [MCV12]argumentaba cómo [MCV13]construir una máquina inteligente, y cómo testar dicha inteligencia. El test es un objetivo a largo plazo en la investigación de la AI – ¿llegaremos a construir algún día un ordenador capaz de imitar lo suficiente a un ser humano como para que un juez suspicaz sea incapaz de distinguir entre hombre y máquina? La premisa de Turing es que, si la máquina es capaz de engañar al juez, será considerada inteligente. Esto dio lugar al aún existente debate en la comunidad científica, sobre si las máquinas son o serán realmente capaces de

[3] Término que será explicado a lo largo del trabajo.

pensar. Así, debemos introducir los distintos tipos de Inteligencia Artificial: especializada, generalizada y súper inteligencia.

2.2. INTELIGENCIA ARTIFICIAL ESPECIALIZADA (NARROW O WEAK AI)

Se refiere al tipo de AI que existe hoy en día. Este tipo de inteligencia está programada para llevar a cabo una sola tarea, ya sea informar acerca del tiempo o analizar datos para escribir un artículo periodístico. Los sistemas de Inteligencia Artificial Especializada (ANI) pueden desarrollar una tarea en tiempo real, sin embargo sólo pueden extraer información de una base de datos[4] específica. El resultado es que estos sistemas no pueden operar fuera de esa tarea singular para la que han sido diseñados.

A diferencia de la Inteligencia Artificial Generalizada y de la Súper Inteligencia, la ANI no presenta una inteligencia cognitiva-relacional propia del ser humano, sino que opera dentro de unos límites predefinidos, a pesar de que en ocasiones aparente una mayor complejidad (como por ejemplo un coche que no requiere un piloto, simplemente se compone de numerosos sistemas de ANI). Es por esto que se dice que esta "Inteligencia" Artificial no es verdaderamente inteligente. Sergio Álvarez-Teleña (2017), economista experto en trading algorítmico y AI, describe la ANI como "fuerza bruta". Así hace referencia al proceso de hacer uso de una robusta capacidad computacional para procesar un gran volumen de datos. Asimismo, hace alusión a esta falta de inteligencia cognitiva-relacional al poner de ejemplo la donación de $80 millones de la Fundación de Bill y Melinda Gates para eliminar la carencia de datos de género y acelerar el progreso de mujeres y niñas en países en desarrollo. Si no crean una huella digital, la máquina no va a tenerlas en cuenta, dando lugar a situaciones discriminatorias. Gates (2016), aseguró que "*No podemos eliminar la desigualdad de género sin antes eliminar la brecha de datos. No sabemos lo suficiente acerca de las barreras que obstaculizan a mujeres y niñas. Nos comprometemos a cambiar esta situación invirtiendo en mejores datos y políticas*".

[4] A lo largo del trabajo, se utilizarán las expresiones "base de datos" o "*data-set*" indistintamente.

Ello pone de manifiesto las limitaciones de la ANI, ya que efectivamente ninguno de los sistemas basados en este tipo de inteligencia es capaz de operar fuera de la tarea específica para la que fue programado, ni extrapolar conclusiones de *data-sets* de los que no dispone.

2.3. Inteligencia Artificial Generalizada (General/Strong AI)

Se refiere a las máquinas que exhiben inteligencia propia de un ser humano. En otras palabras, un sistema con Inteligencia Artificial Generalizada (AGI), es capaz de llevar a cabo cualquier actividad propia de un humano, gozando de esa inteligencia cognitiva-relacional a la que he aludido previamente. Este tipo de inteligencia es propia de personajes como C-3PO de "Star Wars", máquinas y sistemas operativos con conciencia, sentimiento e impulsados por emociones.

Actualmente, las máquinas son capaces de procesar datos de manera más rápida y eficiente que cualquier ser humano. Sin embargo, como hombres somos capaces de pensar de forma abstracta, crear estrategias, y hacer uso de nuestros recuerdos y pensamientos para tomar decisiones. Este tipo de inteligencia nos hace superiores a las máquinas, y es difícil de definir o integrar en una máquina, dado que está primordialmente impulsada por nuestra habilidad para ser criaturas sensibles.

Se espera que la AGI sea capaz de resolver problemas, razonar, tomar decisiones desde la incertidumbre, aprender y ser innovadora y creativa. Sin embargo, este tipo de AI, a diferencia de la ANI, no tiene más de 50 años, y se espera que tarde en llegar.

2.4. Súper Inteligencia Artificial (Artificial Super Intelligence)

Nick Bostrom (2006), filósofo de la Universidad de Oxford, definió la súper inteligencia ("ASI", por sus siglas en inglés) de la siguiente manera:

"*Cuando hablamos de 'súper inteligencia' nos referimos a un intelecto que es mucho más listo que los mejores cerebros de humanos en prácticamente cualquier campo, incluyendo la creatividad científica, sabiduría general y capacidades sociales… …Entidades como compañías o la comunidad científica no son súper inteligencias de acuerdo con esta definición. Aunque pueden llevar a cabo un gran número de tareas que ningún otro ser humano puede, no son intelectos y hay muchos campos en los que se*

desenvuelven peor que un cerebro humano – por ejemplo, no puedes tener una conversación en tiempo real con la 'comunidad científica' –"

La gran paradoja es que, si bien es cierto que tardaremos en llegar a la AGI, el salto de AGI a ASI será prácticamente "instantáneo". Esto se debe a que, una vez existan AIs al nivel de seres humanos (AGI), que a su vez tengan acceso a bases de datos masivas y servidores, harán uso de esa "*fuerza bruta*" que mencionaba previamente para desarrollarse de forma exponencial. Las AIs contribuirán a construir mejores AIs, que a su vez construirán mejores AIs, y así sucesivamente. Además, al contrario que un intelecto biológico, probablemente sea posible copiar las habilidades o módulos cognitivos de una AI a otra. De esta forma, si una AI ha logrado la eminencia en un determinado campo, posteriores AIs podrían descargar el programa de la pionera, logrando así el mismo rendimiento.

Es más, en el artículo anteriormente citado, Bostrom asegura que, aunque no se produjeran nuevos desarrollos de software y las AIs no adquirieran nuevas habilidades a través del aprendizaje propio, se seguirían volviendo más inteligentes si la velocidad de los procesadores sigue aumentando. Si tras un año el hardware se mejorara hasta doblar la velocidad anterior, tendríamos una AI capaz de pensar el doble de rápido que su implementación original. De esta manera, se llegaría a lo que conocemos como ASI débil, un intelecto equiparable al del ser humano, pero de velocidad notablemente superior.

Asimismo, la utilidad marginal en las mejoras en AI una vez llegásemos a la AGI incrementaría sobremanera, volviéndose un fenómeno todavía más atractivo para las fuentes de financiación. De esta forma, podemos hacer la predicción de que, una vez lleguemos a una AI equiparable a la de un humano, no tardaremos mucho en llegar a la súper inteligencia.

2.5. AUGE RECIENTE DE LA AI

Una vez expuestos los distintos tipos de AI, debemos entender cuáles han sido los factores que han permitido que la AI evolucione hasta llegar al nivel de desarrollo actual. Para ello, debemos discernir qué es lo que impidió a Turing solucionar el problema que planteó en 1950. En primer lugar, uno de los obstáculos más notables que propuso el matemático fue la capacidad de almacenamiento de los ordenadores, que vaticinó sería suficiente "en unos cincuenta años". Antes de 1949, los ordenadores no eran capaces de

almacenar comandos, sólo ejecutarlos. En segundo lugar, disponer de capacidad computacional era extremadamente caro. A principios de 1950, el precio de alquilar un ordenador ascendía a $200,000, por lo que los científicos de alto perfil debían ser capaces de convencer a sus fuentes de financiación de que la AI merecía la pena [Anyoha (2017)].

Así, este fenómeno comienza a cobrar una mayor popularidad en la época de los 1990s-2000s, donde se lograron numerosos de los éxitos más importantes para la AI. En 1997, el campeón de ajedrez Garry Kasparov fue derrotado por Deep Blue [Kasparov vs. Deep Blue (2018)], el programa desarrollado por la compañía americana IBM, en 19 movimientos (su derrota más rápida). En el mismo año, Dragon Systems lanzó al mercado NaturallySpeaking 1.0, su primer software de reconocimiento de voz, que fue integrado en el sistema Windows 95 [Anyoha (2017)]. Breazeal (1998), del Massachusetts Institute of Techology (MIT), evidenció que incluso la emoción humana podía ser alcanzable por las máquinas, a través de Kismet, un robot capaz de reconocer y mostrar emociones.

¿A qué se debieron entonces estas rompedoras innovaciones? No fueron fruto de una notable mejora en la manera de escribir código orientado a la AI, sino a un incremento en el volumen de datos disponibles, algoritmos más avanzados, así como mejoras en almacenamiento y capacidad computacional (el gran límite subrayado por Turing en los 1950s). De esta manera, pasaré a exponer la relevancia de estas 3 tendencias en facilitar lo que se conoce como "*Big Data Revolution*"[5]:

1. **Incremento exponencial en la cantidad de datos disponible**[MCV14]**:** Winans (2016) estimaron que el 90% de los datos del mundo a 31 de diciembre de 2016, habían sido generados entre 2015 y 2016. Se prevé que esta afluencia de datos continúe aumentando el acumulado de datos en el mundo digital de 4.4 zettabytes (o billones de gigabytes) en 2015 a 44 zettabytes en 2020 [The digital universe of opportunities (2014)].

   La población global goza de un acceso cada vez mayor a internet. Según Data Never Sleeps 6.0 (2018), la población conectada a internet creció un 11.8% de 2016 a 2017, pasando de 3.4 mil millones a 3.8 mil millones. Por este motivo, el fenómeno

[5] Basándome en Kolanovic y Krishnamachari (2017).

que conocemos como *Internet of Things* (IoT) jugará un papel determinante en la recolección de fuentes de datos alternativas, apoyado por:

- *Integración de sensores conectados a la red en aplicaciones domésticas:* Guide to the IoT (2015), demostró que el número de dispositivos inteligentes (o "s*mart devices*") aumentó de 2 mil millones en 2006 a 15 mil millones en 2015. Asimismo, predijo que esta cifra rozará los 200 mil millones en 2020, lo cual implicará una media de 26 dispositivos inteligentes por persona.
- *Recolección de datos a través de smartphones:* Data Never Sleeps 6.0 (2018) también reveló que realizamos un 50% de nuestras búsquedas en internet desde un *smartphone*, dato muy relevante teniendo en cuenta que en 2018 se vendieron 1.56 mil millones de unidades en todo el mundo [Number of smartphones sold (2019)].
- *Reducción del coste de tecnología satélite:* La industria espacial ha madurado en los últimos años. Al contrario que en los comienzos de NASA, ha pasado a incluir docenas de compañías privadas alrededor del globo, en una carrera por ver quién puede enviar material al espacio de manera más económica. Ello ha implicado una reducción en los costes de lanzamiento. Christensen [en Tartar y Qiu (2018)], fundadora y directora general de la consultora Bryce Space and Technology, asegura que se ha producido una reducción del precio (en términos reales) de lanzamiento del 15% desde 2010 [Tartar y Qiu (2018)].

2. **Incremento de la capacidad computacional y de almacenamiento:** El acceso a capacidad de almacenamiento y los beneficios de *parallel/distributed computing*[6] [MCV15]queda garantizado gracias a su puesta a disposición por vía remota. A este fenómeno lo conocemos como "Cloud Computing", y se estima que en 2020, más de un tercio de todos los datos vivirán o pasarán por la nube (*cloud*).
3. **Uso de Machine Learning para analizar bases de datos extensas y complejas:** Hemos visto desarrollos significativos en el campo de reconocimiento de patrones y

[6] Son procesos mediante los cuales una misma tarea se disgrega en diversas sub-tareas, para ser ejecutadas por varios procesadores simultáneamente, reduciendo el tiempo de procesamiento y aumentando la eficiencia y seguridad (si uno de los ordenadores cae, el sistema sigue funcionando).

aproximación de funciones (descubriendo relaciones entre variables). A estos métodos analíticos se les denomina "Machine Learning" (ML), y forman parte de disciplinas más amplias como Estadística o "Computer Science". Las técnicas de ML permiten el análisis de *data-sets* extensos y no estructurados, así como la construcción de estrategias de trading.

A pesar de la excitación generada en torno al Big Data o al ML, los investigadores estiman que sólo un 0.5% de todos los datos de los que disponemos a nivel global se analiza [Regalado (2013)]. Wolfe (2014), [MCV16]reveló en una entrevista para Business Insider que este porcentaje se reducía a medida que aumenta la generación de información. En 2013, sólo el 22% de toda la información digital generada era susceptible de análisis (es decir, sería útil si estuviese etiquetada). Se estima que para 2020 ese porcentaje de información útil aumente hasta llegar a 35% [The digital universe of opportunities (2014)]. No sólo habrá más datos, sino de mayor calidad y representatividad.

Además, nos encontramos en la época no sólo de mayor disponibilidad de información, sino además más barata. Por ejemplo, el gobierno de Estados Unidos pone a disposición de cualquier individuo más de 300.000 bases de datos de forma gratuita (lo que llamamos "open data") a través de su plataforma www.data.gov.

En conclusión, todos estos desarrollos incitan a los agentes del mercado a invertir en el descubrimiento de nuevos *data-sets* y herramientas de ML.

3. MACHINE LEARNING

Esta sección pretende aportar las nociones científicas suficientes para entender el proceso de elaboración de modelos en Machine Learning, que posteriormente serán aplicados sobre datos financieros. Por ende, se comienza introduciendo la definición del término que denominamos Machine Learning, para después explicar qué objetivos se persiguen con esta técnica. Más adelante, se introducen los diversos tipos de aprendizajes que existen, estableciendo brevemente sus diferencias.

3.1. INTRODUCCIÓN

Este apartado introducirá el fenómeno de Machine Learning (ML), explicando a un nivel general en qué consiste y qué objetivos se persiguen.

3.1.1. Definición

Como se ha explicado anteriormente, la AI busca dotar a las máquinas con la inteligencia cognitiva propia de un ser humano. Los primeros intentos de lograr AI consistían en introducir manualmente ("*hardcode*") un gran número de reglas e información en la memoria de un ordenador. A esto lo conocemos como AI Simbólica, la cual no dio muy buenos resultados. Así, se buscaron otros tipos de enfoque en la búsqueda de esta AI, entre los que destacamos el Machine Learning.

Pero, ¿qué es exactamente el Machine Learning? Podríamos definirlo de la siguiente manera:

"Machine Learning *es una ciencia orientada a conseguir que los ordenadores puedan aprender/actuar de la misma forma que un ser humano, adaptando y mejorando su aprendizaje de forma autónoma, a través de información en la forma de observaciones e interacciones con el mundo real*".

Esta definición es un conglomerado de las definiciones [MCV17]que han proporcionado varios expertos en el campo, tales como:

"*ML es una parte de la investigación de la AI, que busca dotar a los ordenadores de conocimiento a través de datos, observaciones e interacciones con el mundo. El*

conocimiento adquirido permite a los ordenadores adaptarse a nuevos entornos" [Bengio[7] en Faggella (2019b)].

"*ML es la ciencia que busca conseguir que los ordenadores actúen sin estar programados de forma explícita*" [Stanford en Machine Learning (2019)].

"*ML es la ciencia orientada a lograr que los ordenadores aprendan de la misma forma que lo hacen los humanos, o mejor*" [Yampolskiy[8] en Faggella (2019a)].

3.1.2. Objetivo

La finalidad del ML es descubrir patrones presentes en una base de datos, para poder realizar predicciones basadas en esas complejas relaciones, lo cual nos ayudará a resolver problemas o a modificar nuestras estrategias empresariales.

Para analizar dichos datos se utilizan algoritmos o modelos, que serán entrenados para identificar los mencionados patrones, y posteriormente optimizados para mejorar la precisión de sus predicciones. Aquí entran en juego los diversos *performance metrics* (métricas de rendimiento), utilizados para medir el rendimiento del algoritmo que se ha implementado, que varía en función del tipo de modelo que se emplee.

Así, el objetivo es crear un algoritmo que sea capaz de predecir eventos futuros de la forma más precisa, y para ello se buscará maximizar o minimizar (según el tipo de función que se esté empleando) los *performance metrics*.

3.2. TIPOS DE APRENDIZAJE

El ML tiene una infinidad de aplicaciones, empleándose tanto en buscadores de páginas web, como en detectores de fraudes o diseños de drogas. En el mundo de las finanzas, en concreto, podemos ver el ML como un intento de descubrir relaciones entre variables, donde dados unos patrones históricos (*inputs* y *outputs*), la máquina es capaz de predecir un resultado (que puede ser el precio de una acción, por ejemplo).

[7] *Computer Scientist* de la Universidad de Montréal.
[8] *Computer Scientist* de la Universidad de Louisville.

La consecuencia de esta pluralidad de aplicaciones es la existencia de diversos tipos de ML, que conocemos como métodos de aprendizaje:

3.2.1. Descripción del Supervised Learning

Se trata del tipo de ML más maduro y más usado. Una máquina basada en este tipo de aprendizaje es capaz de hacer uso de sus experiencias pasadas y aplicarla a nuevos datos. Así, empleará datos etiquetados para predecir patrones y eventos futuros. Estos datos etiquetados (*labelled data*) son una serie de *training examples*, donde cada ejemplo es una pareja consistente en un *input* y un *output* con el valor deseado (lo cual conocemos como *supervisory signal*) [Fumo (2017)]. Es por ello que se dice que este tipo de aprendizaje está apoyado en un "profesor", ya que a lo largo del proceso de aprendizaje éste le explicará a la máquina cuál es la respuesta correcta.

Las aplicaciones de este método son útiles cuando los datos históricos permiten obtener predicciones fiables sobre casos futuros. Por ejemplo, se puede emplear para determinar qué transacciones son potencialmente fraudulentas, qué clientes de una compañía de seguros tienen más probabilidad de reclamar una determinada cantidad monetaria, etc.

3.2.2. Descripción del Unsupervised Learning (UL)

Al contrario que en *Supervised Learning*, este método encuentra patrones donde no tenemos una base de datos que contenga respuestas correctas. Esto puede ser debido a diversos motivos: que estas respuestas correctas sean inobservables, que no sea posible obtenerlas, o que no exista una respuesta correcta *per se*.

Este sistema se emplea sin usar ningún tipo de dato etiquetado. No recibe *outputs* determinados ni correlaciones entre *outputs* y respuestas "correctas", sino que el algoritmo debe explorar los datos y encontrar algún tipo de patrón o estructura.

Este tipo de aprendizaje tiene sentido cuando los datos son de carácter transaccional. Un ejemplo es encontrar grupos de clientes con características similares sobre los que poder enfocar una campaña de marketing.

Este proceso es más complejo que el *Supervised Learning*, ya que el sistema inteligente parte a "ciegas", y debe hacer uso de su lógica para guiarse. Para entender el proceso, intentaré exponer un ejemplo. Imaginemos que como persona nunca hemos oído hablar de lo que es el baloncesto. Nos llevan a un partido de la NBA y nos dejan ahí para intentar entender qué es lo que estamos observando. No podemos acceder a los conocimientos que tenemos de otros deportes para tratar de encontrar similitudes y diferencias, y así acabar entendiendo mejor lo que es el baloncesto. No tenemos nada más aparte de nuestra habilidad cognitiva. Siguiendo el método de *Unsupervised Learning*, el objetivo es que acabemos siendo capaces de reconocer una serie de patrones y estructuras que nos permitan no sólo comprender qué es el baloncesto, sino poder identificar también qué jugadores son buenos y por qué.

3.2.3. Descripción del Semi-Supervised Learning (SSL)

Este tipo es una especie de híbrido entre *Supervised* y *Unsupervised Learning*. Se emplea en casos en los que los datos necesarios para resolver un determinado problema están a nuestro alcance, pero se presentan incompletos y con imprecisiones. De esta forma, el SSL puede acceder a sus datos de referencia y hacer uso de las técnicas de *Unsupervised Learning* para rellenar esos huecos.

El sistema aprende de los datos etiquetados a los que he aludido previamente, para poder emitir recomendaciones o llegar a conclusiones en relación con los datos no etiquetados, reconociendo patrones o estructuras.

Este tipo de aprendizaje es útil en la mitigación del fallo humano en el proceso, debido a que estas etiquetas presentes en el *Supervised Learning* pueden presentar potenciales errores, al haber sido asignadas por un programador.

3.2.4. Descripción del Reinforcement Learning

Es un tipo de programación dinámico que entrena a los algoritmos siguiendo un sistema de recompensa y castigo. El algoritmo aprende interactuando con su entorno, recibiendo una recompensa si actúa de forma óptima, o un castigo si lo hace de manera incorrecta. Así, aprende sin ser enseñado por ningún humano, buscando la mayor recompensa

y minimizando el castigo. Evidentemente este tipo de aprendizaje está ligado a un contexto determinado, ya que este sistema de incentivos puede variar por completo de uno a otro.

Este tipo de aprendizaje depende de tres componentes: agente (el sistema que toma las decisiones), contexto (con lo que interactúa el agente) y acciones (lo que el agente puede hacer). Así, el agente aprenderá a maximizar su recompensa en un contexto determinado. El aprendizaje viene en la forma de *feedback*, que se conoce como la *reinforcement signal*.

Esta metodología se emplea en sectores, tales como la robótica o la navegación. El algoritmo detecta una serie de pasos que llevan a la maximización de recompensa a través de la prueba y el error. Cuando este proceso se repite, el problema se conoce como el Proceso de Decisión de Markov.

Un ejemplo que tenemos en nuestro día a día es nuestro tablón de Facebook. Esta empresa hace uso del Machine Learning para personalizar los anuncios y actividad que se presenta en nuestro tablón. Si frecuentemente pulsamos “me gusta” en determinadas actividades, fotos de amigos, etc., comenzaremos a verlos con más frecuencia en nuestro tablón.

4. SUPERVISED (INDUCTIVE) LEARNING[MCV18]

En esta sección se expondrá la metodología detrás de la construcción de un modelo de *Supervised Learning* (aprendizaje automático supervisado), aunque se profundizará en aquellas nociones teóricas que serán necesarias para comprender el funcionamiento de los modelos que han sido utilizados en este trabajo (sección 4). Se ha decidido ahondar en el método SL, debido a que se ha estimado necesario contextualizar determinados conceptos, con carácter previo a la explicación del modelo construido (árbol de decisión)[9].

Se comenzará hablando de la división del conjunto de datos en distintas partes, para posteriormente explicar qué tipo de métricas se usan a la hora de determinar los hiperparámetros y el grado de complejidad del algoritmo.

4.1. INTRODUCCIÓN

Como se explicó previamente, en este tipo de aprendizaje partimos de un *data-set* con datos etiquetados, lo cual implica que, para cada caso de entrada, conocemos el valor real de salida. Con estos casos proporcionados, el algoritmo debe ser capaz de predecir el valor de salida de puntos no etiquetados, es decir, que no ha visto.

Podemos ver el *Supervised Learning* como algo muy parecido al concepto de aproximación de una función, ya que básicamente se entrena a un algoritmo y al final del proceso se escoge la función que mejor describe el *input data*. En otras palabras, la función (f) que, dado un *input* (X) lleve a cabo la mejor estimación para un *output* (Y): $Y = f(X)$. De esta forma, lo que más nos importa es que nuestro algoritmo sea capaz de establecer, de la manera más precisa posible, una relación entre las *input features* y los *target prediction outputs*. La estimación de dicha relación o $f(X)$ se conoce como *statistical learning* (aprendizaje estadístico). Así, será capaz de predecir *outputs* de variables que aún no haya visto.

[9] En el caso de los modelos de UL (*hierarchical* y *k-means clustering*), se explicarán en la sección 4.

4.2. DIVISIÓN DE LOS DATOS

El conjunto de datos suele disgregarse en tres tipos: *training data* (conjunto de aprendizaje), *validation data* (conjunto de validación), y *test data* (conjunto de prueba). El ratio de división no es fijo (y es imperativo que los datos se distribuyan en cada conjunto de manera aleatoria, para evitar caer en sesgos[MCV19]), sino que depende del número de muestras que contiene nuestra base de datos, además del modelo algorítmico que deseamos construir.

4.2.1. Conjunto de aprendizaje (training dataset)

Se trata del conjunto de datos que sirven al algoritmo de experiencia de aprendizaje, mediante los cuales se ajustarán algunos parámetros del modelo. Como mencioné previamente, el objetivo de todo modelo es ser capaz de describir la relación entre variables con la mayor precisión posible. En otras palabras, que el valor de salida del modelo se aproxime lo máximo posible al valor real. Esta medición se lleva a cabo a través de las funciones de coste o *loss functions*, y el objetivo de todo modelo es encontrar el punto que las minimice.

Podemos destacar (en modelos regresivos) las funciones de error cuadrático (*mean square error*) y error absoluto (*mean absolute error*). Es necesario comprender las diferencias entre una y otra, y por qué una se usa más que la otra, ya que así entenderemos uno de los grandes riesgos que encontramos al aplicar ML al mundo de las finanzas.

A. Mean Square Error[10] (MSE)

Es la suma de la distancia cuadrática entre el valor real y los valores de salida del modelo.

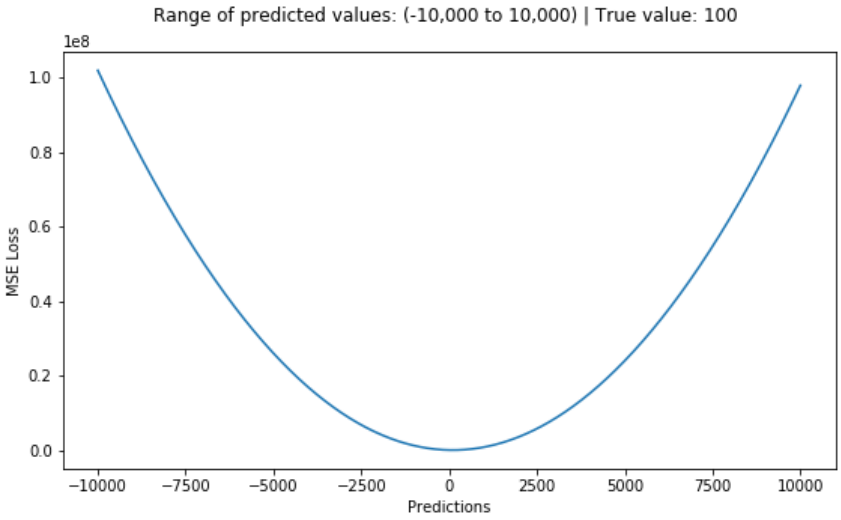
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

En la Figura 1, podemos observar la representación de una función de error cuadrático, donde el valor real es 100, y predicciones que oscilan en un rango entre los valores

[10] Función de error cuadrático medio en inglés.

-10.000 y 10.000. El error cuadrático (eje Y) logra su valor mínimo con la predicción (eje X) = 100.

Figura 1. Representación de la función de error cuadrático medio.



Fuente: Grover (2018).

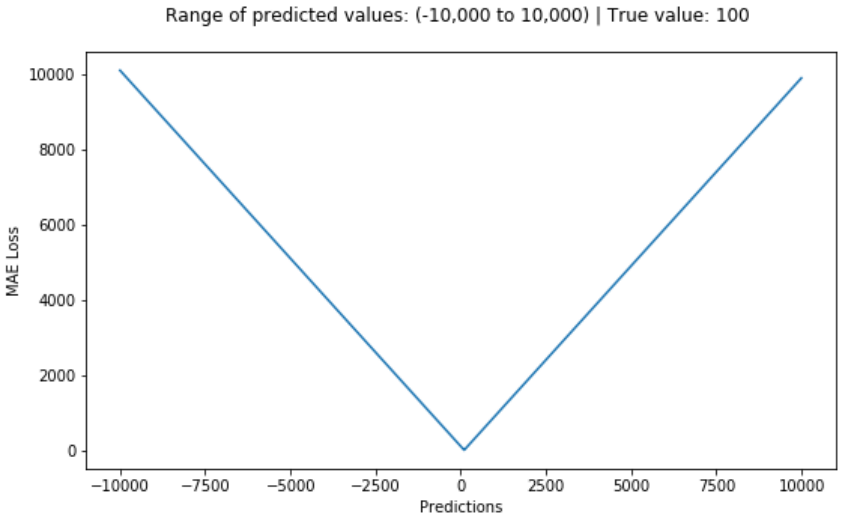
B. Mean Absolute Error[11] (MAE)

Es la suma absoluta de diferencias entre el valor real y los valores de salida del modelo. Mide la magnitud media de errores en una serie de predicciones, sin tener en cuenta su dirección.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

[11] Función de error absoluto en inglés.

Figura 2. Representación de la función de error absoluto.



Fuente: Grover (2018).

C. MSE vs. MAE

El hecho de que la función MSE recoge el error al cuadrado, implica que es muy sensible a *outliers* (ya que el valor del error aumenta notoriamente si $e > 1$), mientras que la MAE, al usar valores absolutos, es más robusta. Podemos comprobar esto observando la Figura 3.

Figura 3. Variación de resultados en las funciones MSE y MAE por la presencia de *outliers*.

MAE vs. RMSE for cases with slight variance in data

ID	Error	|Error|	Error²
1	0	0	0
2	1	1	1
3	-2	2	4
4	-0.5	0.5	0.25
5	1.5	1.5	2.25

MAE: 1 RMSE: 1.22

MAE vs. RMSE for cases with outliers in data

ID	Error	|Error|	Error²
1	0	0	0
2	1	1	1
3	1	1	1
4	-2	2	4
5	15	15	225

outlier

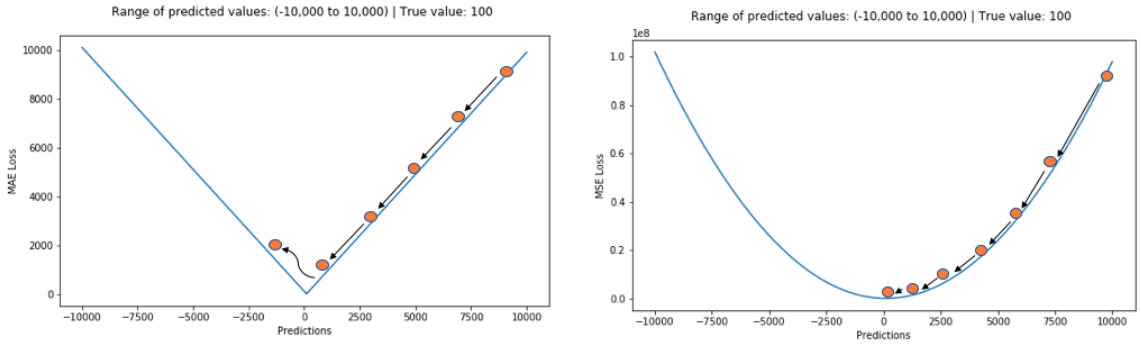
MAE: 3.8 RMSE: 6.79

Fuente: Grover (2018).

A pesar de ello, en el mundo financiero cobra una mayor relevancia la función de error cuadrático. Este es el peligro al que hacía referencia previamente. En esencia, debido a su forma convexa, lo que nos viene a decir esta función es que, por ejemplo, fallar por una cantidad de 1.000€ es más grave que fallar diez veces por una cantidad de 100€, cuando financieramente, no es una asunción correcta. Además, dicha función también descarta modelos precisos. Por ejemplo, podemos crear un algoritmo cuyos errores oscilen siempre entre -1 y 1. Sin embargo, en nuestro conjunto de datos puede haber una minoría de *outliers*, con errores de -20 y 20. Pues bien, esta función preferirá un modelo que sistemáticamente falle en un rango entre -5 y 5, debido a que pondera los *outliers* de forma desproporcionada, descartando modelos válidos a causa de la varianza de los datos.

Siendo conscientes de este riesgo, ¿por qué se sigue entrenando un modelo con la función de errores cuadráticos? La respuesta es que es más eficiente para el aprendizaje algorítmico. El motivo es que la función de error absoluto, como hemos podido comprobar en las Figuras 1 y 2, tiene el mismo gradiente en cualquier punto, mientras que la de errores cuadráticos es alta para errores significativos, y decrece cuanto más se aproxima a 0. Así, esta función se vuelve más precisa al final del entrenamiento/aprendizaje.

Figura 4. Comparativa de las funciones MSE y MAE en relación al aprendizaje algorítmico.



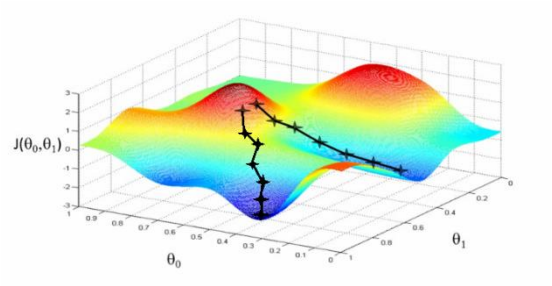
Fuente: Grover (2018).

D. Algoritmo de Gradient Descent

Siendo conscientes de que el objetivo de un modelo es minimizar la función de coste, debemos tratar de entender cómo se hace. Para lograr este objetivo, se hace uso de un algoritmo de optimización conocido como *gradient descent*, un método iterativo que

modifica los parámetros de una función convexa para encontrar su mínimo local [Donges (2018)].

Figura 5. Representación de la función de *gradient descent*.



Fuente: Vryniotis (2013).

Matemáticamente, podemos expresarla de la siguiente manera:

$$\theta_{n+1} = \theta_n - \alpha \frac{\partial}{\partial \theta_n} J(\theta_n)$$

Siendo θ un parámetro, α el *learning rate*, y $\frac{\partial}{\partial \theta_n} J(\theta_n)$ el gradiente de la función J para θ_n.

El *learning rate* es el parámetro que determina la velocidad de aproximación a la ponderación óptima. Si es demasiado grande, nos saltaremos la solución óptima, mientras que si es demasiado pequeño, serán necesarias demasiadas iteraciones.

4.2.2. Conjunto de validación (validation dataset)

Se trata de un conjunto de datos (integrado en el conjunto de aprendizaje) que proporciona una evaluación no sujeta a sesgo [MCV20](explicaré este término y su relevancia más adelante) de la precisión del modelo en relación con el *training data*, y sirve para ajustar los hiperparámetros (profundizaré en este término más adelante) [Shah (2017)]. Esta evaluación se lleva a cabo diversas veces con el objetivo de reajustar el modelo. Ello implica que, a medida que vamos incorporando al modelo el conocimiento que hemos aprendido con el *validation data-set*, la evaluación cada vez se ve más sujeta a sesgo.

Si bien es cierto que existen métodos para ajustar hiperparámetros usando sólo el conjunto de entrenamiento, la selección del modelo en sí (es decir, si se crea un árbol de decisión o una red neuronal u otro modelo) requiere usar el conjunto de validación.

4.2.3. Conjunto de prueba (*test data-set*)

Este conjunto de datos se utiliza para analizar la precisión del algoritmo con datos no empleados para entrenar el modelo. A diferencia del conjunto de validación, no tenemos ese vaivén por el cual se vuelve a entrenar el modelo una vez analizada la precisión e identificados los fallos. De esta manera, este conjunto proporciona una evaluación del modelo final, no sometida a sesgo [Shah (2017)].

4.3. GENERALIZATION ERROR Y SELECCIÓN DE HIPERPARÁMETROS

Comenzaré introduciendo el concepto de *generalization error*, y luego explicaré su desglose en diversos componentes, para posteriormente analizar la relevancia de los mismos en la selección de hiperparámetros (grado de complejidad algorítmico).

4.3.1. Generalization Error

Para medir cómo se va a comportar nuestro modelo con datos futuros, empleamos el *Generalization Error*, que mide el error sobre todas las posibles variables x e y. El problema con esta métrica es que desconocemos cuáles serán los datos (x_i) y las etiquetas (y_i) futuros. A pesar de ello, podemos tratar de aproximar u obtener un rango de error mediante el *test error* (calculando un intervalo de confianza), que mide el error en el conjunto de datos que no hemos usado ni para entrenar ni para validar nuestro algoritmo [Lavrenko (2014)].

Su expresión matemática es la siguiente:

$$R\,(f) = \mathbb{E}[f(X) - Y^2]$$

De esta forma, el cálculo de $R(f)$ lo hacemos elevando al cuadrado el error medio que esperamos devolverá nuestro modelo, menos el valor del modelo real.

Otra manera de entender este error es a través del hecho de que para construir el modelo, tomamos n muestras de Y, y realizamos una estimación (f) de la relación de las

variables X e Y. Sin embargo, nuestra estimación f es aleatoria, ya que de haber escogido otra muestra (Y), nuestro f variaría.

Podemos desglosar el *generalization error* en dos tipos: irreducible y reducible, cuyo análisis nos ayudará a determinar cómo optimizar los parámetros de complejidad de nuestro algoritmo.

Para explicar los distintos componentes que integran el *generalization error*, partimos de la siguiente regresión:

$$Y = f*(X) + \epsilon \qquad \epsilon \sim \mathrm{N}(0, \sigma^2)$$

Donde $f^*(X)$ es un modelo determinístico (que puede ser una recta, curva, etc.); y ϵ representa el ruido presente en el *data-set*, siguiendo una distribución normal de media 0 y varianza σ^2.

Podemos desglosar la fórmula anterior de la siguiente manera:

$$R(f) = \mathbb{E}[(f(X) - \mathbb{E}[f(X)])^2] + \mathbb{E}[(\mathbb{E}[f(X)] - f^*(X))^2] + \sigma^2$$

Donde $f(X)$ es la función que trataremos de aproximar al modelo determinístico $f^*(X)$, y $\mathbb{E}[]$ es la esperanza de la variable.

4.3.1.1. Varianza

Pasamos a describir el primer componente, denominado *variance* o varianza (encuadrado de color azul en la fórmula). Mide la variabilidad de la predicción del modelo para un dato o valor concretos, informándonos así del *spread* (diferencia) de los estimadores de los coeficientes. Nos informa de la sensibilidad de nuestro modelo a la muestra. Una posibilidad para reducirlo es aumentar el muestreo con el que construimos nuestro modelo.

Los modelos con varianza significativa prestan mucha atención al *training data*, pero no generalizan de manera óptima con datos que aún no han visto. El resultado son rendimientos altos en *training data*, pero niveles de error notorios en *test data*.

4.3.1.2. Sesgo

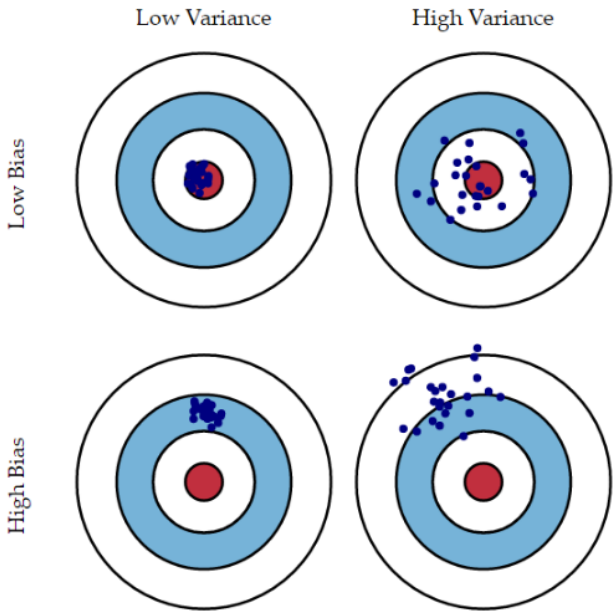
La segunda parte de nuestra ecuación (encuadrada en rojo), analiza lo que denominamos sesgo[12]. Pretende analizar cómo se aproxima nuestro modelo al determinístico, lo cual está relacionado con el grado de complejidad algorítmico. Es un fenómeno que tiene lugar cuando un algoritmo produce resultados sistemáticamente sometidos a prejuicios debido a la existencia de presunciones erróneas en el proceso de ML. Ocurre cuando se intenta aproximar un problema real, que puede ser extremadamente complejo, con un modelo mucho más simple. Así, si la verdadera relación es extremadamente compleja y se intenta usar una regresión lineal, indudablemente incurriremos en sesgo en la estimación de $f(X)$, sin importar el número de observaciones del que dispongamos.

Mide la diferencia entre la media de predicciones de nuestro modelo y el valor real de la relación que intenta aproximar. Es decir, para un punto dado, cuánto falla de media nuestro modelo en relación con la función determinística.

Por ello, decimos que un modelo con un sesgo significativamente alto presta poca atención al *training data* y está simplificado en exceso, llevando a niveles altos de error en *training* y *test data* [Singh (2018)].

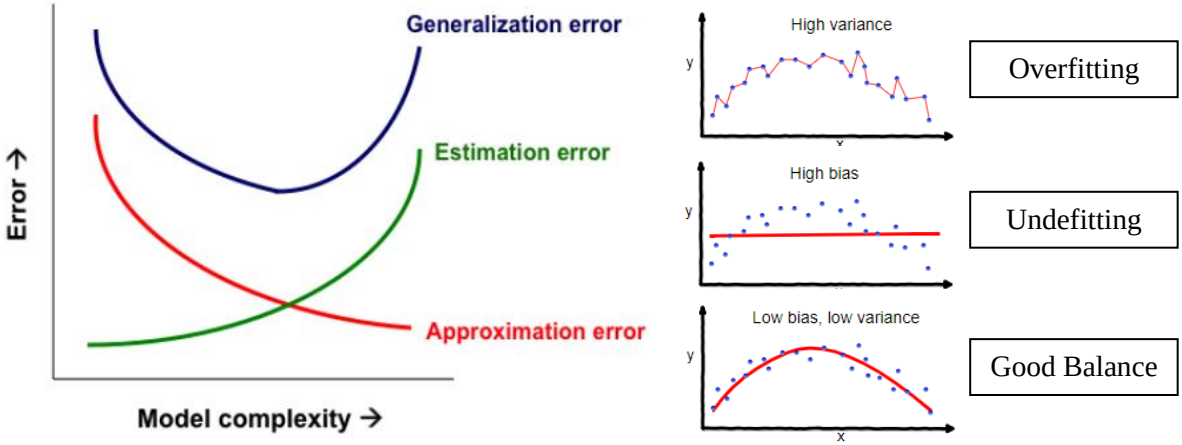
[12] En la ecuación debemos notar que el segundo componente es sesgo^2.

Figura 6. Representación de los fenómenos de sesgo y varianza.



Fuente: James et al (2013).

Figura 7. Contribución al error por sesgo o varianza en función de la complejidad del modelo y el fenómeno de *overfitting*.



Fuente: Agarwal (2018).

Como podemos apreciar en la Figura 7, en un muestreo de tamaño fijo, a medida que incrementa la complejidad del modelo, se reduce el *generalization error*, hasta llegar a un punto óptimo, tras el cual volverá a incrementar como consecuencia del *overfitting*.

4.3.1.3. Error irreducible

Si bien es cierto que nuestro objetivo siempre será hacer que nuestro modelo sea lo más preciso posible (en otras palabras, que $f(X) = f^*(X)$), esta precisión nunca será plena debido al error irreducible, nuestro tercer componente (encuadrado en verde).

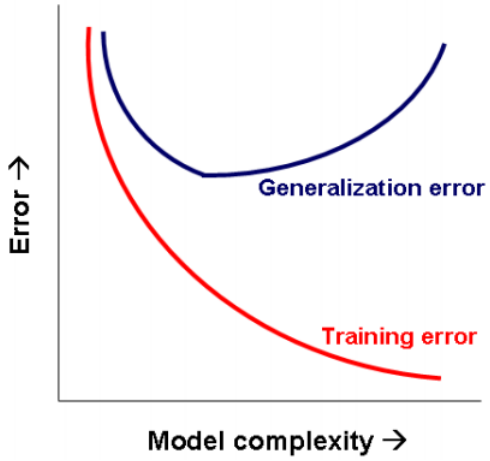
Representa la información de Y que no puede explicar X, y mide el ruido presente en nuestro *data-set*. Ésta puede contener tanto información de las variables que sería útil a la hora de predecir Y (como no medimos esta información, al no encontrarse en nuestro *training set*, la función f no la tiene en cuenta); como variación que no puede medirse (por ejemplo, el riesgo de que un paciente padezca una reacción adversa en un día concreto, puede depender de ciertas variaciones en la producción de la droga, o simplemente cómo se encuentre el paciente en ese día concreto).

Es por esto que el objetivo de todo algoritmo de ML es reducir el denominado *excess risk* (exceso de riesgo), o en otras palabras, la suma de varianza y sesgo^2.

4.3.2. Complejidad del Modelo

La complejidad de la relación $f(X)$ entre los *inputs* y las variables de respuesta, es un factor importante a tener en cuenta en el aprendizaje de un *data-set*, como hemos podido comprobar. En muchos algoritmos, tenemos cierta flexibilidad a la hora de escoger el parámetro de complejidad de un modelo: el número de *hidden nodes* (nodos ocultos) en un *neural network* (red neuronal), la profundidad de un árbol de decisión, el número de *neighbors* en el método de *nearest neighbor*, etc.

Figura 8. Relación del error del modelo y el error en el conjunto de datos de entrenamiento.



Fuente: Agarwal (2018).

La Figura 8 muestra cómo, a medida que la complejidad del modelo aumenta, el *training error* disminuye, pero el *generalization error* tiene forma convexa. Es alto en modelos de escasa complejidad, decrece hasta que la complejidad del modelo se equipara a la distribución de datos desconocidos (modelo determinístico), e incrementa de nuevo en modelos de alta complejidad.

Como hemos visto en la Figura 7, los modelos de baja complejidad se ven sometidos a un problema de *underfitting*. En otras palabras, no son lo suficientemente flexibles como para describir de forma adecuada los patrones en los datos. Los modelos de alta complejidad tienen el problema contrario, *overfitting*. Son tan flexibles que describen no sólo patrones generales, sino que se ven afectados por el ruido presente en el *training data*.

Por ende, nuestro objetivo es seleccionar el parámetro de complejidad del modelo que devuelva el *generalization error* más bajo (en otras palabras, que no caiga ni en *underfit* ni *overfit*). El reto, tal y como subraya Argawal (2018), es que la complejidad óptima depende de la distribución de datos desconocidos, y debe estimarse en base a los datos con los que contamos. A este problema se le denomina "*model selection*".

Para dar solución a este problema, surgen métodos como el *cross-validation*, que se emplea como medio para estimar el *generalization error* para diversos grados de complejidad del modelo.

4.3.3. Optimización de Hiperparámetros

El proceso de decidir si los resultados numéricos que cuantifican la relación hipotética entre las variables son aceptables como una descripción de los datos se denomina *validation* (validación). Generalmente la estimación del error para el modelo se lleva a cabo con posteridad al *training*, lo cual conocemos también como *evaluation of residuals* (evaluación de los residuos o errores). En este proceso, se lleva a cabo una estimación numérica de la diferencia entre las respuestas que predice el modelo frente a las originales, también llamado *training error*. Sin embargo, esto sólo nos ofrece información sobre cómo lo está haciendo nuestro modelo en relación a los datos empleados a lo largo del *training*.

El problema con esta técnica de evaluación es que no nos indica cómo de bien va a ser capaz de generalizar nuestro *learner* (modelo) frente a un *data-set* independiente. Para hacerse a una idea de esto recurrimos a distintos métodos, como *cross-validation* o *rolling windows*.

4.3.3.1. Cross-Validation

A. Holdout Method

Consiste en "guardar" parte del *training data*, para posteriormente usarlo para hacer predicciones sobre el modelo. Es una técnica simple de *cross-validation*, que no conlleva aparejado ningún sobrecoste computacional y es más fiable que la tradicional técnica de validación. A pesar de ello, está expuesta a problemas de *high variance*, ya que no queda claro qué *data points* irán destinados al *validation set* (la parte del *training data* que "guardamos"), y el resultado podría variar enormemente dependiendo del *set*.

B. K-Fold Cross-Validation

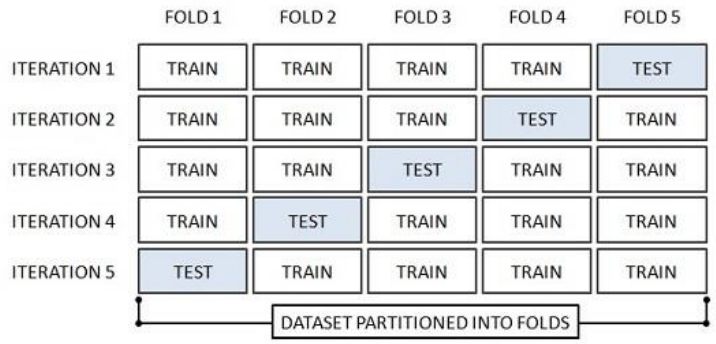
Domingos (2012) asegura que, "por regla general, un algoritmo 'tonto' que dispone de muchos datos gana a uno más 'inteligente' que disponga de una cantidad de datos más

modesta". Reduciendo el *training data* disponible, nos arriesgamos a perder importantes patrones de nuestro *data-set*, lo cual incrementa el error inducido vía sesgo.

Así, el método K-Fold proporciona amplios datos para entrenar el modelo y validarlo. Esta técnica divide los datos en *k subsets*, para posteriormente aplicar el método *holdout k* veces, de manera que cada vez uno de los *k subsets* se utiliza como *validation set*, mientras que el resto de *subsets* ($k - 1$) se unen para formar el *training set*.

Para finalizar, se calcula una media de la estimación del error entre los *k* intentos, por lo que cada dato sirve a su vez para entrenar y validar el modelo, reduciendo así el sesgo y la varianza.

Figura 9. Tabla explicativa de un modelo *K-Fold* con k = 5.



Fuente: Mueller y Massaron (sin fecha).

C. Stratified K-Fold Cross Validation

En algunos casos, puede haber notables disparidades en las variables de respuesta. Por ejemplo, en un *data-set* que contiene precios de viviendas, puede haber muchas con precios muy elevados (regresión), o muchos más ejemplos negativos que positivos (clasificación). Para este tipo de problemas, se produce un ajuste en la técnica de K-Fold, por el cual cada *fold* pasará a contener un porcentaje similar de cada *target class* (clase objetivo), o en el caso de problemas de predicción, la media de los valores de repuesta será similar en todos los *folds*.

D. Leave-P-Out Cross Validation

Esta técnica deja p datos fuera del *training data*. Si tenemos n datos en nuestro *data-set*, $n - p$ se emplearán para entrenar el modelo, y p se destinarán a validarlo. Esto se repite para todas las combinaciones en las que el *data-set* original puede dividirse siguiendo este proceso, y después se calcula una media del error para todos los intentos.

Este método es exhaustivo, pero si la p es moderadamente grande, se convierte en inviable, computacionalmente hablando.

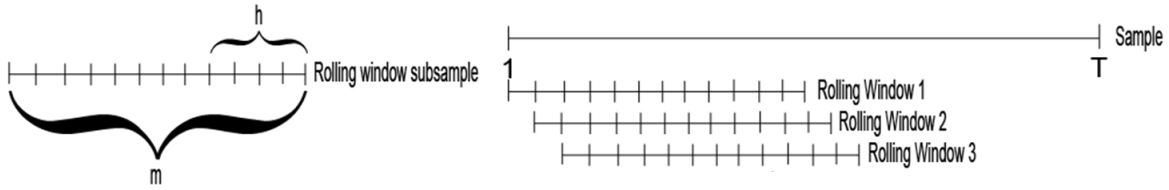
4.3.3.2. Rolling Windows for Predictive Performance

El método de *cross-validation* es muy útil cuando nos encontramos ante un problema de clasificación. Sin embargo, si nos enfrentamos a una serie temporal, cuyos datos contienen información implícita dependiendo del marco temporal en el que han sido generados, el método previamente definido no funciona.

Aunque existen diversos métodos regresivos (modelo ARIMA, *temporal features*, etc.), me centraré en el conocido como *rolling windows*.

Este modelo utiliza el *backtesting* para analizar el rendimiento predictivo de diversos modelos de series temporales [Zivot y Wang (2006)]. Para ello, comienza eligiendo un *rolling window* de tamaño m, que dependerá del tamaño muestral T, y la periodicidad de los datos (por norma general, el tamaño es menor cuanto menor sea el intervalo de recolección de datos). Posteriormente, se escoge un horizonte temporal h. Por último, se debe seleccionar el incremento entre rolling windows sucesivas k, para así partir la base de datos en N sub-muestras, siendo $N = T - m + k$.

Figura 10. Representación del método *rolling window*.



Fuente: Zivot y Wang (2006).

De esta manera, se podrá computar el error medio de cada sub-muestra (*rolling window*), determinando así el modelo cuyas predicciones son más precisas (menor error medio).

4.3.3.3. Conclusión

Los procesos que he descrito anteriormente se llevan a cabo para distintos niveles de complejidad, lo cual nos permite optimizar los hiperparámetros de nuestro modelo. Para facilitar la comprensión, pondré un ejemplo. Pongamos que construimos una red neuronal. Recordemos que, para optimizar los parámetros del modelo (en este caso las ponderaciones de cada neurona), usamos el método de *gradient descent*, previamente explicado.

Sin embargo, para optimizar los hiperparámetros debemos recurrir a la "fuerza bruta". Podemos construir un *tunegrid*, cuyo eje *X* recoja el número de capas, mientras que el eje *Y* representa el número de neuronas por capa. De esta forma, se elaborará un modelo ajustado a los hiperparámetros definidos en el *tunegrid*, para posteriormente analizar el error de cada modelo, y determinar el más preciso.

5. MODELOS QUE SERÁN APLICADOS EN EL SIGUIENTE TRABAJO

En esta sección me centraré en dos tipos de algoritmo concretos. Explicaré cómo construir cada modelo, aportando ejemplos simplificados para facilitar la comprensión. El objetivo es que un lector sin ningún tipo de conocimiento de ML sea capaz de entender cuál es el proceso detrás de un modelo (aunque el algoritmo que aplique a datos financieros sea de mayor complejidad).

5.1. INTRODUCCIÓN

Habiendo definido los métodos de *Supervised Learning* y *Unsupervised Learning*, debemos pasar a analizar algunos de los distintos tipos de algoritmos. A pesar de que contamos con una gran variedad de algoritmos (*Nearest Neighbor*, *Naive Bayes*, *Support Vector Machines*, *Linear Regression*, *Neural Networks*, etc.), me centraré en dos: los árboles de decisión (SL) y el método de clustering (UL).

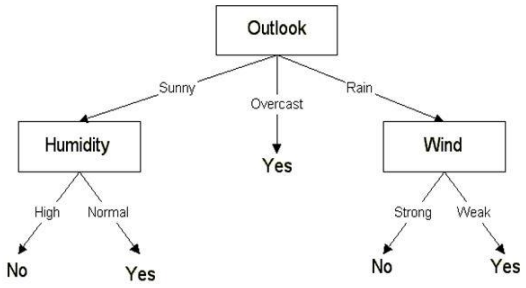
Antes de profundizar en los mismos, debo resaltar que en ML existe un teorema conocido como "*No Free Lunch*". Wolpert (1996) demuestra que en un escenario sin ruido (*noise-free scenario*) donde la *loss function* (función de coste) es el *missclassification rate* (error en la clasificación), si uno está interesado en *off-training-set error* (error en el conjunto de datos no usados para entrenar el modelo), no hay ninguna distinción a priori entre algoritmos de aprendizaje. En otras palabras, no existe un algoritmo que funcione siempre mejor que el resto para todos los problemas.

5.2. ÁRBOLES DE DECISIÓN

Es el más utilizado en *Supervised Learning*. Éstos algoritmos suelen llamarse CART o *Classification and Regression Trees*. Así, podemos distinguir entre los *Classification Trees*, que son aquellos cuyo objetivo es predecir valores discretos, como un clasificador de emails en spam o no spam; y *Regression Trees*, orientados a predecir valores continuos, como el precio de una vivienda.

De esta forma, pasamos a representar nuestro algoritmo como un árbol. El siguiente ejemplo ilustra un árbol de decisión que analiza si nuestro amigo querrá o no jugar al golf hoy:

Figura 11. Ejemplo de árbol de decisión.



Fuente: Kulkarni (2017).

Se dibuja como un árbol invertido, con la raíz en lo más alto. De esta forma, podemos encontrar los *internal nodes* (representados con cajas), que comprueban el valor de un determinado atributo *X* (*outlook*, humidity, wind), para proceder a ramificar de acuerdo con los resultados de este test. El nodo de decisión que se encuentra en la parte superior del árbol corresponde al mejor predictor, denominado *root node*. El final de una rama que ya no se divide más es denominado *leaf* (hoja), y especificará la clase $h(X)$, que en nuestro caso será una respuesta afirmativa o negativa frente a la prerrogativa de si nuestro amigo querrá o no jugar al golf hoy.

Las bases de datos cuentan con un número muy superior de observaciones y variables, que harán que nuestro árbol de decisión tenga un mayor número de ramas. De esta forma, para construir el algoritmo, recurrimos a dos procesos, denominados *induction* y *pruning*.

5.2.1. Induction

Este proceso consiste en una serie de pasos a seguir a la hora de construir un árbol de decisión:

1. Partimos de nuestra base de datos, que deberá consistir en una serie de atributos/variables (X) y un *output* aparejado (Y).
2. Determinar el "mejor atributo" en la base de datos para comenzar la división/ramificación. Profundizaré en este proceso valiéndome del ejemplo a continuación.
3. Dividir los datos en subgrupos que contengan los posibles valores para este "mejor atributo". A esta división es a lo que denominamos nodo en nuestro árbol.

4. Generar nuevos nodos de forma sistemática partiendo de los subgrupos creados en el paso 3, hasta llegar a un punto en el que se haya optimizado la precisión con el menor número de divisiones/nodos.

5.2.2. Ejemplo

Para explicar los pasos expuestos previamente, utilizaré un modelo de árbol clasificatorio simplificado. Uno de los algoritmos más empleados para la elaboración de los mismos es el ID3 (Iterative Dichotomiser 3), creado por J. R. Quinlan.

Para seguir con el ejemplo que expuse previamente, elaboraré un árbol de decisión que estudie si mi amigo querrá jugar o no al golf conmigo. Para ello, partiré de la siguiente base de datos:

Figura 12. Base de datos con distintas observaciones para la construcción del árbol de decisión.

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Fuente: Elaboración propia.

La elección de qué atributo emplear y en que división específica suele hacerse empleando un método de "*greedy search*". El algoritmo ID3 hace uso de la entropía para calcular la homogeneidad de la muestra.

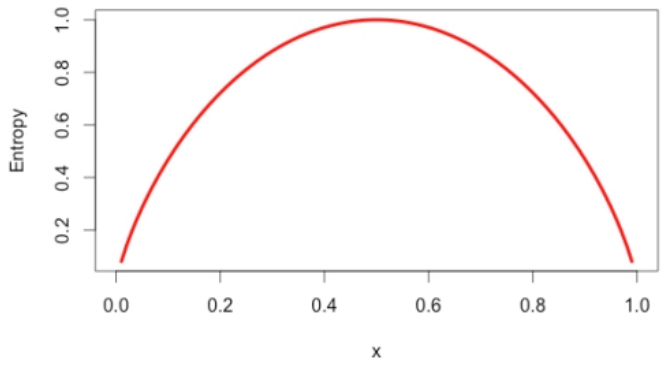
5.2.2.1. Entropía

La entropía, también llamada *Shannon Entropy*, se denota $H(S)$ para un set finito de S, y mide la incertidumbre o aleatoriedad en los datos. Se calcula siguiendo la siguiente fórmula:

$$H(S) = \sum_{x \epsilon X} p(x)\, log_2 \frac{1}{p(x)}$$

De manera intuitiva, nos dará información acerca de la previsibilidad de un evento. Por ejemplo, si lanzamos una moneda al aire, la probabilidad de que salga cara o cruz es 0.5 en ambos casos. Este es el caso con mayor entropía (1), al no haber manera posible de determinar el resultado. Sin embargo, si ambos lados de la moneda fueran cruz, la entropía de este evento puede determinarse fácilmente (0), ya que sabemos que siempre saldrá cruz, al no haber aleatoriedad.

Figura 13. Representación de la curva de entropía.[MCV21]



Fuente: Elaboración propia.

Así, nuestro primer paso será calcular la entropía de jugar al golf, utilizando la tabla de frecuencia de la siguiente manera:

Figura [MCV22]**14. Cálculo de la entropía de jugar al golf.**

Play Golf	
Yes	**No**
9	5

Fuente: Elaboración propia.

Entropy (PlayGolf) = Entropy (9, 5)

= Entropy (0.64, 0.36)

= - (0.64 $\log_2$ 0.64) - (0.36 $\log_2$ 0.36)

Obtenemos un resultado de 0.94, lo cual nos indica que la distribución entre las clases es bastante aleatoria (recordemos que 1 era el resultado más alto). El siguiente paso será calcular el atributo que nos ofrezca el mayor *information gain* (que explicaré a continuación), que conformará el *root node*.

5.2.2.2. Information Gain

El *information gain*, también denominado *Kullback-Leibler divergence*, se denota $IG(S, A)$ para un set finito de S, e indica el cambio efectivo en entropía habiendo escogido un particular atributo A. Mide el cambio relativo en entropía con respecto a las variables independientes. Se calcula siguiendo la siguiente fórmula:

$$IG(S, A) = H(S) - H(S, A)$$

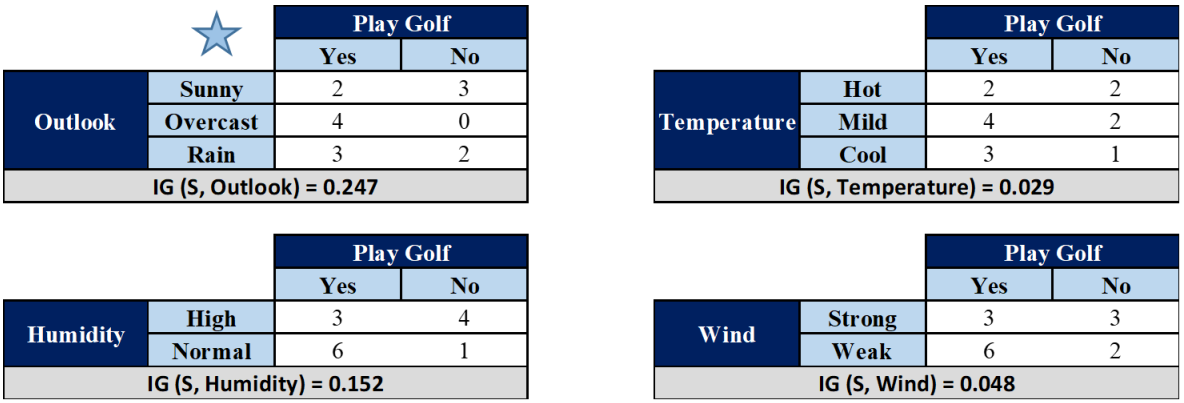
Alternativamente,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Donde $H(S)$ es la entropía de todo el *set*, mientras que el segundo término calcula la entropía después de dividir en función del atributo A, siendo $P(x)$ la probabilidad del evento x.

De esta manera, el siguiente paso en nuestro ejemplo será calcular el *Information Gain* aportado por cada atributo:

Figura 15. *Information Gain* aportado por cada variable en la primera división del árbol.



		Play Golf	
		Yes	No
Outlook	Sunny	2	3
	Overcast	4	0
	Rain	3	2
IG (S, Outlook) = 0.247			

		Play Golf	
		Yes	No
Temperature	Hot	2	2
	Mild	4	2
	Cool	3	1
IG (S, Temperature) = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
IG (S, Humidity) = 0.152			

		Play Golf	
		Yes	No
Wind	Strong	3	3
	Weak	6	2
IG (S, Wind) = 0.048			

Fuente: Elaboración propia.

De esta manera, podemos escoger el atributo *outlook* como nuestro *root* (o raíz) o *decision node* (nodo de decisión), al ofrecer el mayor *information gain*. Dividiremos nuestros datos en función de sus ramas, repitiendo este mismo proceso para cada una de ellas. Lo que podemos observar a primera vista, es que cada vez que el *outlook* del día es *overcast*, nuestro amigo siempre quiere jugar al golf. De esta manera, al ser la entropía de esta rama 0, nos encontramos ante lo que hemos denominado antes como *leaf node*, y no hará falta seguir dividiendo.

Por ello, sólo debemos repetir el proceso anterior para las ramas *sunny* y *rain*. Partiremos de nuestra tabla que contiene los casos en los que el día ha sido soleado:

Figura 16. Observaciones con variable *outlook* = *sunny*.

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Fuente: Elaboración propia.

Como antes lo primero que debemos calcular es la entropía, es decir, $H(Sunny)$:

Figura 17. Cálculo de la entropía de que el día esté soleado.

Sunny	
Yes	**No**
2	3

Fuente: Elaboración propia.

Entropy (Sunny) = Entropy (2, 3)

= Entropy (0.40, 0.60)

= - (0.40$\log_2$ 0.40) - (0.60$\log_2$ 0.60)

Así, pasamos a calcular el *information gain* para nuestros 3 atributos restantes:

Figura 18. *Information Gain* aportado por cada variable en la división de la rama *sunny*.

★		Play Golf	
		Yes	No
Humidity	High	0	3
	Normal	2	0
IG (S, Humidity) = 0.971			

		Play Golf	
		Yes	No
Wind	Strong	1	1
	Weak	1	2
IG (S, Wind) = 0.020			

		Play Golf	
		Yes	No
Temperature	Hot	0	2
	Mild	1	1
	Cool	1	0
IG (S, Temperature) = 0.571			

Fuente: Elaboración propia.

Como podemos comprobar, el atributo que aporta un mayor *information gain* es *humidity*. De hecho, reduce la entropía a 0, dividiendo los datos en dos clases de forma perfecta.

Si repetimos el mismo proceso con la rama de *rain*, partiendo de los siguientes datos:

Figura 19. Observaciones con variable *outlook* = *rain*.

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Fuente: Elaboración propia.

Podemos calcular su entropía:

Figura 20. Cálculo de la entropía de que el día sea lluvioso.

Rain	
Yes	**No**
3	2

Fuente: Elaboración propia.

Entropy (Rain) = Entropy (3, 2)

= Entropy (0.60, 0.40)

= - (0.60$\log_2$ 0.60) - (0.40$\log_2$

Así, volvemos a calcular el *information gain* para los mismos 3 atributos, esta vez condicionados a que llueva:

Figura 21. *Information Gain* aportado por cada variable en la división de la rama *rain*.

		Play Golf	
		Yes	No
Wind	Strong	0	2
	Weak	3	0
IG (S, Wind) = 0.971			

		Play Golf	
		Yes	No
Temperature	Mild	2	1
	Cool	1	1
IG (S, Temperature) = 0.571			

		Play Golf	
		Yes	No
Humidity	High	1	1
	Normal	2	1
IG (S, Humidity) = 0.020			

Fuente: Elaboración propia.

En este caso, *wind* es el atributo que aporta un mayor *information gain*, y como sucedía anteriormente, la entropía se reduce a 0, por lo que sabemos que hay una división de clases perfecta.

En un árbol de mayor complejidad, podríamos seguir dividiendo sus ramas hasta el infinito, lo cual no es una buena idea. Nuestro árbol sería inmenso, lento, y sería objeto de *overfitting*. Por ello, existen diversas formas de definir un criterio que ponga fin a la construcción del árbol, como por ejemplo definir de antemano un grado de entropía por debajo del cual no realizar más divisiones, o establecer un número mínimo de *instances* por *leaf node*, por debajo del cual no se producirían más divisiones.

5.2.2.3. Pruning

Establecer las condiciones que he mencionado previamente no es una tarea ni mucho menos sencilla. Es por ello que la mayoría de las veces se decide escoger la vía más segura, y fijar ese mínimo de *instances* en una cifra bastante baja. Ello generará un árbol excesivamente grande y complejo, donde muchas de esas divisiones terminan siendo redundantes, y no incrementan necesariamente la precisión de nuestro modelo.

Pruning (podar en inglés) es el nombre que recibe la técnica de remover esas divisiones redundantes de nuestro modelo. Así, se comprime un árbol sujeto a restricciones más rígidas, para obtener unas que sean más fácilmente generalizables, reduciendo así la complejidad del modelo. Esta complejidad la indica el número de nodos que comprende nuestro árbol de decisión.

Una manera simple pero efectiva de llevar a cabo este proceso es ir nodo a nodo, evaluando el efecto de eliminarlo sobre la *cost function* o función de coste. Así, si el efecto de eliminar dicho nodo no es muy perceptible en nuestra función, procederíamos a "podarlo".

A. Función de Coste

En función de si nuestro árbol de decisión es de clasificación o regresión, empleamos una función distinta.

En modelos **regresivos**, se utiliza la fórmula de error cuadrático:

$$E = \sum (y - \hat{y})^2$$

Donde y es el *output* de nuestro *instance*, e $\hat{y}$ su predicción en el nodo concreto.

En modelos **clasificatorios**, empleamos el índice de Gini:

$$G = \sum (p_k * (1 - p_k))$$

Donde p_k es la proporción (probabilidad) de *instances* de clase k en el nodo de interés. De esta manera, un nodo cuyas *instances* pertenezcan todas a la misma clase (*perfect class purity*) tendrá un $G = 0$, mientras que uno con una división de 50/50 tendrá $G = 0.5$.

Si estamos ante un problema de clasificación binaria (como es el caso del ejemplo expuesto previamente), nuestra fórmula puede expresarse como $G = 2 * p1 * p2$. Cada división en un árbol de estas características genera una tabla de contingencia de 2x2:

Figura 22. Tabla de contingencia en clasificación binaria.

	Class 1	Class 2	
$>$ split	n_{11}	n_{12}	n_{+1}
$\leq$ split	n_{21}	n_{22}	n_{+2}
	n_{1+}	n_{2+}	n

Fuente: Muñoz *et al* (2018).

- $n_{11(2),}$ número de instances en el nodo 1 para la clase 1(2).
- $n_{21(2)}$, número de instances en el nodo 2 para la clase 1(2).
- $n_{+1(2)}$, número de instances en el nodo 1(2).
- $n_{1(2)+}$, número de instances pertenecientes a la clase 1(2).
- n, número de instances del parent node.

De esta manera, el índice de Gini para un concreto punto de división en un problema de clasificación binaria se calcula de la siguiente manera:

$$Gini\ (\text{after split}) = \frac{n_{+1}}{n}\left[2\left(\frac{n_{11}}{n_{+1}}\right)\left(\frac{n_{12}}{n_{+1}}\right)\right] + \frac{n_{+2}}{n}\left[2\left(\frac{n_{21}}{n_{+2}}\right)\left(\frac{n_{22}}{n_{+2}}\right)\right]$$
$$= 2\left[\left(\frac{n_{11}}{n}\right)\left(\frac{n_{12}}{n_{+1}}\right) + \left(\frac{n_{21}}{n}\right)\left(\frac{n_{22}}{n_{+2}}\right)\right]$$

5.2.3. Ventajas de un CART[13]

- Fácil de entender, interpretar y visualizar.
- De manera implícita, esta clase de modelos lleva a cabo una selección de atributos.
- Son capaces de manejar datos tanto numéricos como categóricos. Además, también pueden abordar problemas con múltiples *outputs*.

[13] Basado en Gupta (2017).

- No requieren demasiado esfuerzo del usuario en cuanto a la preparación de datos.
- Las relaciones no lineares entre parámetros no afecta al rendimiento del modelo.

5.2.4. Desventajas de un CART[14]

- El modelo puede crear un árbol demasiado complejo que no sea capaz de generalizar los datos de manera óptima (*overfitting*).
- Los árboles de decisión pueden ser inestables, debido a que pequeñas variaciones en los datos pueden resultar en la generación de un modelo completamente distinto. Este fenómeno, explicado anteriormente (varianza), se reduce mediante la aplicación de métodos como los denominados *bagging* y *boosting* (que explicaré en el apartado de "Ensemble Algorithms en ML").
- Los *greedy algorithms* que empleamos para la selección de los atributos que aportan un mayor *information gain* no garantizan la elaboración del árbol globalmente óptimo. Esto se debe a que puede darse la situación en la que dividir las ramas en base a un atributo X no aporte el mayor *information gain* de primeras, sin embargo la realización de una posterior división con un atributo Y, globalmente resulta ser la óptima. Sin embargo, el *greedy algorithm* desechará la primera división (atributo X) por no ser el "mejor" atributo para esa rama concreta. En otras palabras, este algoritmo no estudia el árbol en su conjunto, sino rama a rama, lo cual puede dar lugar a esta clase de errores. Este problema puede mitigarse entrenando múltiples árboles, donde los atributos se muestreen de forma aleatoria y con reemplazamiento.
- Los *learners* de esta clase de modelos pueden crear árboles sometidos a sesgo si hay alguna clase dominante. Es por ello que es recomendable balancear el *data-set* antes de construir el modelo.

B) Ensemble Algorithms en ML

Como introduje previamente, la aleatoriedad en la selección de la muestra tiene una gran influencia en la precisión del modelo, debido a la presencia de varianza. Para reducirla, encontramos distintos tipos de *ensemble methods*, que son algoritmos que construyen una

[14] Basado en Gupta (2017).

serie de clasificadores para posteriormente clasificar un *data-set* a partir de una media ponderada de sus predicciones [Dietterich (2000)]. Entre ellos podemos destacar dos: *bootstrap aggregation* y *boosting*.

a. Bootstrap Aggregation

Fue propuesto en 1996 por Leo Breiman, distinguido estadístico en la Universidad de California, Berkeley. Antes de analizar este método, también denominado *bagging*, debemos entender la técnica que le sirve de piedra angular, llamada *bootstrap*.

Bootstrap es un potente instrumento estadístico empleado para estimar una cantidad de una muestra, como por ejemplo una media o desviación típica [Brownlee (2016)]. Supongamos que disponemos de una muestra compuesta por 100 valores (X), y queremos obtener una estimación de su media. Podemos calcularla de manera sencilla con la siguiente fórmula:

$$Mean(X) = \frac{1}{100} * Sum(X)$$

Sin embargo, sabemos que nuestra muestra es pequeña, y que el resultado va a contener error. Por ello, podemos mejorar nuestra estimación a través del procedimiento que denominamos *bootstrap*:

- Se crean n sub-muestras aleatorias de nuestro *data-set*, con reemplazo (lo cual implica que el mismo valor puede seleccionarse en diversas ocasiones).
- Se calcula la media de cada sub-muestra.
- Se calcula la media de todas las medias recogidas, obteniendo de esta manera la media estimada de nuestros datos. Por ejemplo, pongamos que hemos creado 5 sub-muestras con medias 2.4, 3.8, 4.3, 2.7 y 4.1. Nuestra media estimada será 3.46.

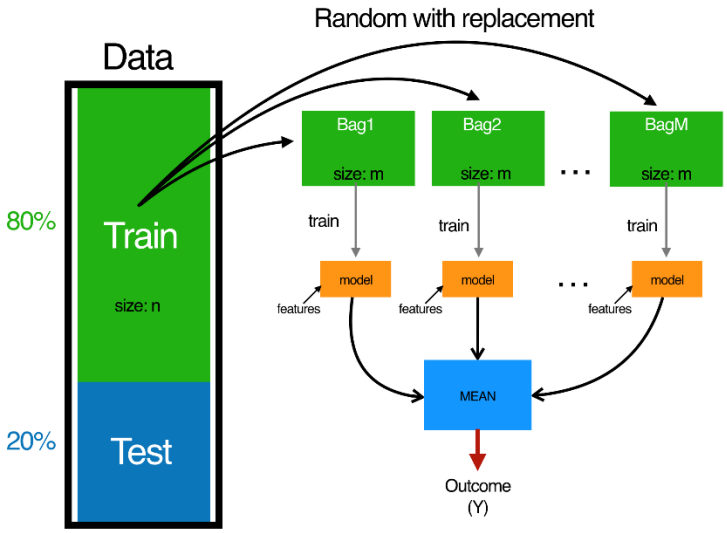
Este proceso se emplea en ML para calcular todo tipo de cantidades, desde desviación típica hasta coeficientes.

Breiman (1996), describe *bagging* como un método empleado para generar múltiples versiones de un predictor, usándolas para obtener un único predictor agregado. La agregación calcula una media de todas las versiones cuando predice un resultado numérico, y lleva a

cabo un *plurality vote* cuando predice una clase. Estas múltiples versiones se forman creando réplicas *bootstrap* del *learning set*, usando éstas como nuevos *learning sets*.

Breiman demostró que, dado el elemento de inestabilidad en un modelo predictivo, la técnica de *bootstrap aggregation* podía mejor sustancialmente la precisión de los modelos.

Figura 23. Representación de la técnica de *bagging*.



Fuente: Regression Trees (sin fecha).

b. Random Forests

Es conveniente hacer mención de la técnica *random forest*, que introduce mejoras sobre árboles creados usando el método *bagging*. Fue propuesta por Tin Kam Ho en 1995, que alegaba que los árboles de decisión podían construirse con una complejidad arbitraria para prevenir una posible pérdida de precisión en la generalización sobre datos no vistos. Kam Ho (1995) defendía que ésta limitación en la complejidad normalmente conllevaba niveles de precisión no óptimos en el *training data*, proponiendo como alternativa su modelo de *random forest*.

El problema con los árboles de decisión, como introduje previamente, es que para la elaboración del modelo utilizan un *greedy algorithm*, eligiendo en la división el atributo que aporta un mayor *information gain*. Es por ello que tienen muchas similitudes estructurales, y por ende una alta correlación en sus predicciones.

La combinación de predicciones de múltiples modelos en *ensembles* funciona mejor si las predicciones de los sub-modelos no están tan correlacionadas. Este tipo de técnica cambia el algoritmo que dicta la manera en la que se crean los sub-modelos para reducir su correlación. Así, en este caso el algoritmo, a la hora de hacer la selección del "mejor" atributo, ve restringida de forma aleatoria la muestra de atributos entre los cuales puede escoger (se reduce el *feature space*, mientras que en un modelo CART, puede escoger entre todos).

c. Boosting

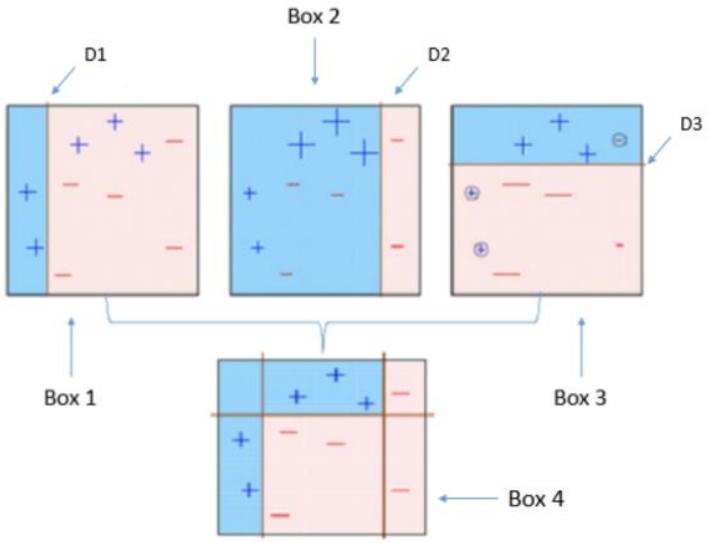
Esta técnica fue introducida por Yoav Freund y Robert E. Schapire en 1995 a través de su algoritmo AdaBoost. Schapire (2013) la describe como un método destinado a crear una regla de predicción altamente precisa a partir de diversas reglas débiles y poco precisas. Freund y Schapire (1999) explican esta técnica con el siguiente ejemplo: un hombre que apuesta a carreras de caballos desea crear un algoritmo que pueda predecir qué caballo ganará una determinada carrera, basándose en información usual (número de carreras ganadas por caballo, cuotas de apuestas, etc.). Para ello, acude a un jugador apostador profesional para que le explique su estrategia, que es incapaz de articular una serie de reglas para seleccionar un caballo. Sin embargo, cuando se le presentan los datos para carreras concretas, es capaz de dar una serie de "reglas de oro" (como "apuesta por el caballo que ha ganado más carreras recientemente" o "apuesta por el caballo con cuota más favorable"). Aunque cada una de estas reglas es poco precisa, podemos esperar que ofrezcan mejores predicciones que la selección aleatoria.

Para poder usar estas "reglas de oro" de forma óptima, el hombre se enfrenta a dos problemas: (i) cómo elegir las carreras que presentarle al experto para que las reglas extraídas aporten la mayor utilidad posible, y (ii) una vez recolectadas dichas reglas, cómo combinarlas en una única y precisa regla de predicción.

La diferencia entre *bagging* y *boosting* es que, mientras en la primera todos los modelos funcionan de manera independiente, para posteriormente agregar sus resultados sin ninguna preferencia por ningún modelo concreto; en la segunda hay "trabajo en equipo", y cada modelo dicta el atributo en el que debe centrarse el siguiente modelo (entran en juego las ponderaciones).

Para entender el funcionamiento de este modelo, usaré un ejemplo gráfico:

Figura 24. Representación gráfica de la técnica *boosting*.



Fuente: Shubham (2018).

- **Box 1:** Se asignan las mismas ponderaciones a todos los datos, y el modelo los clasifica como positivos o negativos. El modelo D1 genera una línea vertical para clasificar los datos, y como podemos comprobar clasifica de forma errónea tres positivos como negativos.
- **Box 2:** El tamaño aumentado de los tres positivos clasificados incorrectamente representa la mayor ponderación que reciben (esto se hace para que los *learners* posteriores se enfoquen en ellos durante la fase de *training*). El modelo D2 traza otra línea vertical para clasificar los datos, y en este caso clasifica de forma incorrecta tres negativos. Por ello, se asigna una mayor ponderación a estos tres datos, y se aplica otro modelo.
- **Box 3:** Esta vez, el modelo traza una línea horizontal basándose en la mayor ponderación de los datos clasificados de forma incorrecta anteriormente.
- **Box 4:** Combinando los modelos D1, D2 y D3 se logra una predicción robusta, con reglas complejas en comparación con los débiles *learners* individualmente considerados.

5.3. CLUSTERING

El *clustering* es una técnica de UL, que suele utilizarse en tareas de *data mining* exploratorio o análisis estadístico. Persigue agrupar objetos con atributos similares en un *set*, que denominamos *cluster*. De esta manera, los objetos de un *cluster* deberían presentar distintas características a aquellos agrupados en otro. Podemos distinguir entre *hard* (fuerte) y *soft* (débil) *clustering*. En el primero, un objeto será miembro de un *cluster* o no, mientras que en el segundo podría pertenecer a varios. Por ello, se concluye que el objetivo que persigue esta técnica es la de agrupar datos no etiquetados.

El *clustering* no es un algoritmo en sí mismo, sino que existen distintos métodos de llevar a cabo esta técnica, entre los que podemos destacar *hierarchical* (jerárquico), *k-means* o *DBSCAN*. La existencia de esta pluralidad de algoritmos reside en la dificultad de definir el concepto de *cluster*, debido a que la tarea del *clustering* es eminentemente subjetiva. Dichos modelos difieren en propiedades, y por eso son aplicados en función del *data-set* en cuestión, así como el objetivo que se pretende. Tal y como indica Kaushik (2016), este conjunto de algoritmos puede disgregarse en 4 tipos:

1. **Connectivity models**: Tal y como indica su nombre, estos modelos se fundamentan en la noción de que la proximidad entre *data points* en un espacio indica una mayor similitud entre ellos que con otros ubicados a una distancia superior. Estos modelos pueden tomar dos caminos: (i) Clasificar todos los *data points* en *clusters* distintos, para posteriormente agregarlos a medida que la distancia se reduce; (ii) clasificar todos los *data points* en un solo *cluster*, y dividirlo a medida que la distancia aumenta. La elección de la función de distancia es subjetiva. Estos modelos son tan fáciles de interpretar como lo son poco escalables. Entre ellos destacamos *hierarchical clustering*.
2. **Centroid models**: Son algoritmos iterativos (que buscan el óptimo local) cuya noción de similitud deriva de la proximidad de un *data point* al *centroid* (centroide). En estos modelos, el número de *clusters* requeridos al final debe especificarse de antemano, por lo que es importante conocer el *data-set* de antemano. Un ejemplo de estos algoritmos es *k-means*.

3. **Distribution models**: Estos modelos se fundamentan en cuál es la probabilidad de que los *data points* de un *cluster* pertenezcan a la misma distribución (*i.e.* Normal). Suelen tener problemas de *overfitting*, y uno de los más utilizados es el algoritmo de *expectation-maximization* (maximización de la expectativa), que utiliza distribuciones normales multivariantes[15].
4. **Density models**: Examinan el *data space* en busca de áreas con distinta densidad de *data points*, aislando estas regiones con determinada densidad, para posteriormente asignar los *data points* de dichas regiones a *clusters* concretos. Un ejemplo son los modelos *DBSCAN*.

En este trabajo se emplearán los modelos de *k-means* y *hierarchical clustering*, que serán explicados a continuación.

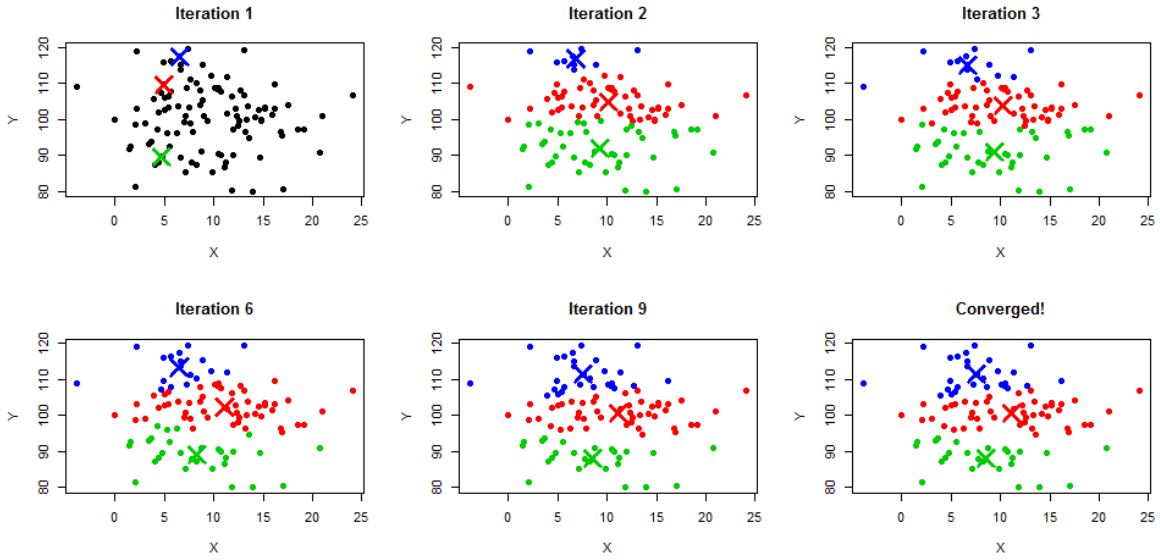
5.3.1. K-Means Clustering

Como se introdujo previamente, se trata de un método iterativo que persigue encontrar el óptimo local en cada iteración. El funcionamiento del modelo puede resumirse en una serie de pasos:

- Especificar el número deseado de *clusters*.
- Se asigna de forma aleatoria un *data point* a cada *cluster*.
- Se computan los centros de cada *cluster*.
- Se reasigna cada punto al *cluster* a cuyo centro se encuentre más próximo.
- Se vuelven a computar los centros de los *clusters*.
- Se repiten los últimos dos pasos hasta que no pueda realizarse ninguna mejora (se habrá llegado al óptimo global), marcando la terminación del algoritmo.

[15] Se trata de una generalización a dimensiones superiores de la distribución normal unidimensional.

Figura 25. Ejemplo del funcionamiento de un modelo *k-means clustering*.

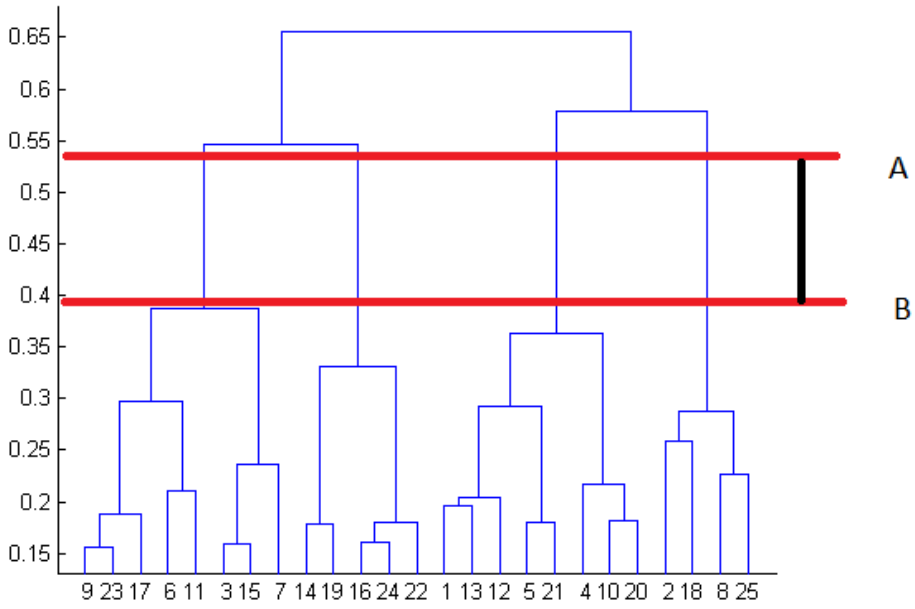


Fuente: K-Means Clustering (sin fecha).

5.3.2. Hierarchical Clustering

Como su propio nombre indica, se trata de un algoritmo que construye una jerarquía entre *clusters*. El algoritmo comienza asignando un *cluster* a cada *data point*. Después, los dos *clusters* más cercanos, se unen. Este proceso se repite hasta que sólo queda uno. El resultado de este proceso se puede mostrar utilizando un dendrograma, como el que podemos apreciar en la Figura 26.

Figura 26. Ejemplo de construcción de un dendrograma.



Fuente: Kaushik (2016).

Partimos de 25 *data points*, cada uno asignado a un *cluster* distinto. Los dos clusters más cercanos convergen hasta que sólo queda uno. La altura del dendrograma en la que dos *clusters* se unen representa la distancia entre ambos en el *data space*.

El número de *clusters* óptimo para dividir el *data-set* es el número de líneas verticales en le dendrograma, cortado por una línea horizontal que atraviese la máxima distancia vertical sin que haya intersección con otro *cluster*. En el caso de este dendrograma en concreto, el número óptimo sería 4, ya que la línea horizontal de color rojo cubre la mayor distancia vertical entre los puntos A y B.

En este ejemplo en concreto se ha implementado un método *bottom-up* (de abajo arriba), pero también es posible hacerlo a la inversa, mediante un método *top-down* (de arriba abajo).

Por último, debe mencionarse que la decisión de unir dos *clusters* se toma en base a la proximidad de los mismos, existiendo diversas métricas para decidir acerca de ésta:

- *Euclidean distance*: $||a - b||_2 = \sqrt{(\Sigma(a_i - b_i))}$

- *Squared Euclidean distance*: $||a - b||_2{}^2 = \Sigma((a_i - b_i)^2)$
- *Manhattan distance*: $||a - b||1 = \Sigma|a_i - b_i|$
- *Maximum distance*: $||a - b||_{INFINITY} = max_i|a_i - b_i|$
- *Mahalanobis distance*: $\sqrt{((a-b)^T S^{-1}(-b))}$ $\{s: covariance\ matrix\}$

5.3.3. Diferencias entre K-Means y Hierarchical Clustering

El *clustering* jerárquico tiene peor rendimiento cuando el número de datos a analizar es alto, mientras que el modelo *k-means* no tiene ese problema. Ello se debe a que la complejidad temporal del segundo es linear, mientras que la del primero es cuadrática.

En *k-means* los resultados de ejecutar el algoritmo diversas veces pueden ser dispares, al comenzar con una elección de *clusters* aleatoria, mientras que los de *hierarchical clustering* son reproducibles.

El modelo *k-means* obtiene mejores resultados cando la forma de los *clusters* es híper esférica (*i.e.* círculo en 2D, esfera en 3D, etc.).

Un modelo *k-means* requiere un conocimiento previo de la base de datos, ya que se necesita una noción de *k* (*i.e.* número de *clusters* en los que dividir el *data-set*) antes de modelar.

Como nota final, a pesar de que el *clustering* es un modelo sencillo de implementar, deben tenerse en cuenta una serie de factores para asegurar su utilidad, como tratar los *outliers* o que cada *cluster* tenga población suficiente.

6. ANÁLISIS DE RIESGO CREDITICIO A TRAVÉS DE MÉTODOS CLASIFICATORIOS

El objetivo de esta sección es poner en práctica la teoría introducida en el apartado 4.2, dedicado al modelo de árboles de decisión, a través de un experimento programado en lenguaje R. Se ha construido un modelo de árbol de decisión, y uno de *random forest*, con el objetivo de tratar de predecir si el prestatario de una hipoteca va a ser capaz de repagar la deuda, en base a una serie de variables. Se analizarán individualmente los resultados obtenidos con ambos métodos, para posteriormente compararlos.

6.1. INTRODUCCIÓN

La base de datos [MCV23]que se analiza se compone de una selección aleatoria de préstamos hipotecarios de los portfolios que integran el *U.S. Residential Mortgage-Backed Securities* (RMBS)[16]. Contiene observaciones para el rendimiento de 1.045 prestatarios de hipotecas en Estados Unidos. Consta de una serie de variables que explican un *output* binario: *Default* (impago). Éstas variables son:

1. **Age:** Tiempo transcurrido desde el origen hasta la observación.
2. **Time_to_maturity:** Tiempo restante hasta la conclusión de la hipoteca.
3. **Balance:** Balance remanente al tiempo de la observación.
4. **LTV:** Ratio *Loan-to-value* al tiempo de la observación, en %. Se obtiene mediante la siguiente fórmula:

$$LTV\ ratio = \frac{Mortgage\ amount}{Appraised\ property\ value}$$[17]

5. **Interest_rate:** El tipo de interés al tiempo de la observación, en %.
6. **SingleFamily:** Vivienda unipersonal = 1, de lo contrario = 0.
7. **Balance_orig:** Balance hipotecario al tiempo de origen.
8. **FICO_orig:** *FICO score*[18] al tiempo de origen, en %.
9. **LTV_orig:** Ratio *Loan-to-value* al tiempo de origen, en %.

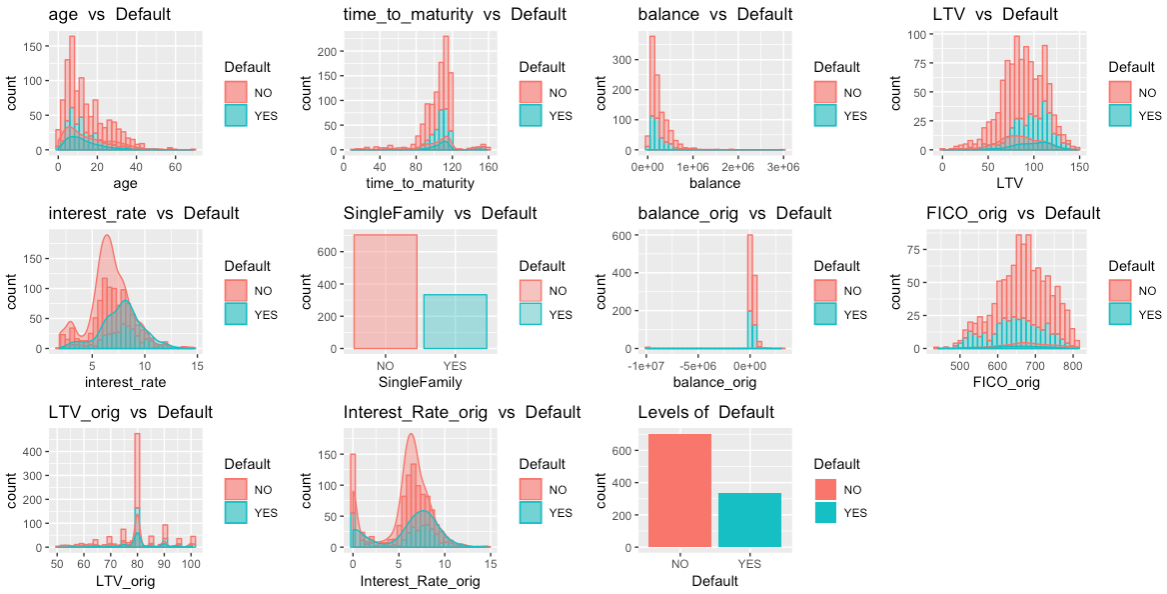
[16] Se trata de instrumentos financieros de deuda, similares a un bono, respaldados por el interés pagado sobre hipotecas. Para mitigar el riesgo de impago de estos préstamos individuales, se forman grupos de las mismas (término denominado "*pooling*" en inglés).
[17] Se trata de la valoración de una propiedad en un determinado punto del tiempo.
[18] Se trata de un sistema que analiza la calidad crediticia de un individuo a través de un sistema de puntuación basado en una serie de atributos. Fue creado en 1989 por la compañía Fair Isaac Corporation.

10. **Interest_Rate_orig:** Tipo de interés al tiempo de origen, en %.

6.2. Análisis Exploratorio

En primer lugar, se ha realizado un análisis exploratorio de las variables, creando una serie de histogramas que representan cada una de las 10 variables en función del *output*.

Figura 27. Histogramas para todas las variables separadas por clases.



Fuente: Elaboración propia.

De esta manera, lo primero que podemos observar es que la base de datos está compuesta por un mayor número de ejemplos de la clase "YES" que "NO". Ello servirá para evaluar la relevancia de los resultados obtenidos. Asimismo, también debe notarse que la función es incapaz de representar la variable binaria "SingleFamily". Mientras que la gráfica debería mostrar cuántos puntos toman valores 1 o 0 en esta variable, para ambas clases, sólo muestra la proporción de puntos en cada clase.

6.3. Algortimo de Árbol de Decisión

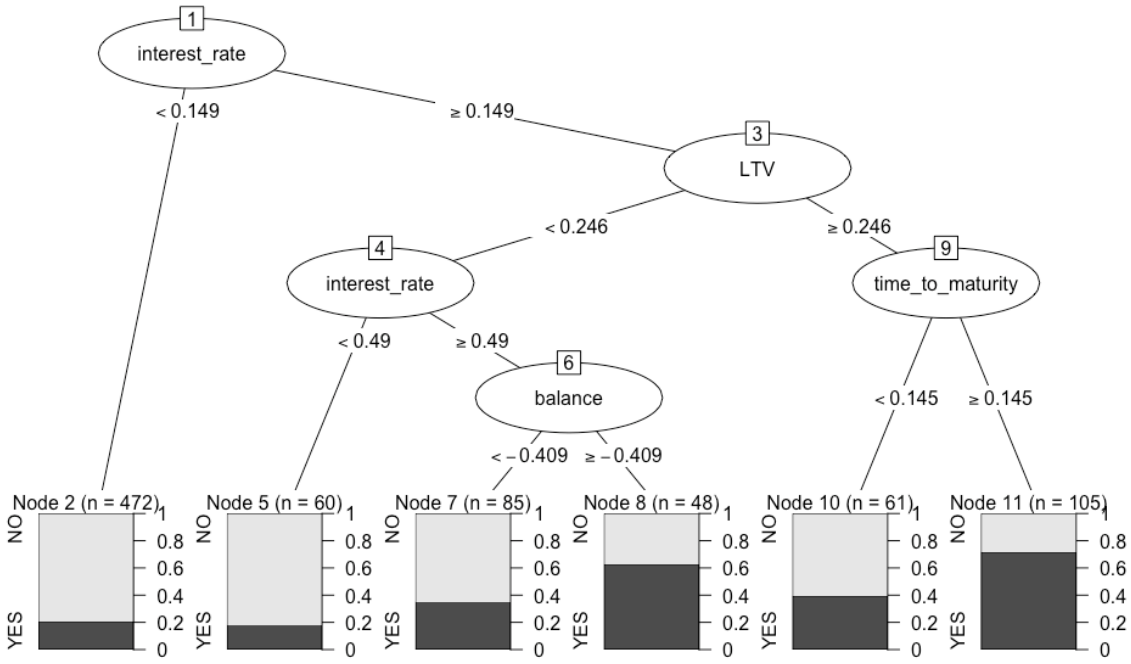
6.3.1. Construcción del Árbol de Decisión

Para la construcción del árbol de decisión, se hace uso de la métrica *Cp* (*Complexity Parameter*). Se utiliza para controlar el tamaño óptimo del árbol, penalizando la complejidad

del modelo para evitar el *overfitting*. Para encontrar el nivel óptimo de *pruning*, se prueban distintos valores de Cp. En el caso de este modelo, se probaron todas las posibilidades para un rango entre 0 y 0.5, con saltos de 0.001. El árbol final utiliza un $Cp = 0.01$, que ha sido obtenido restringiendo el número mínimo de observaciones por nodo para continuar la división en 5.

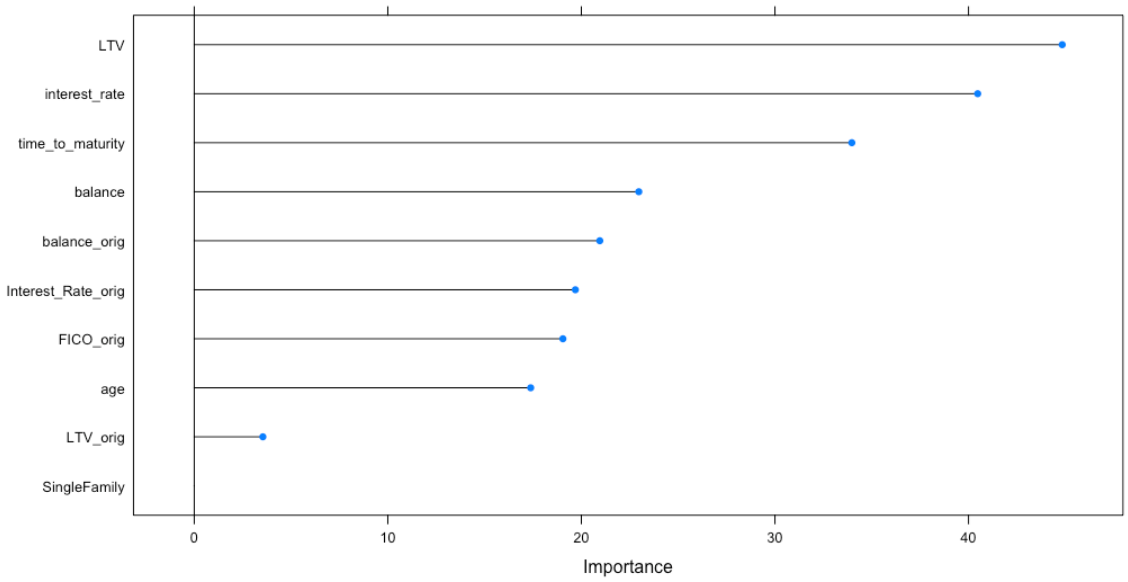
Tal y como se puede apreciar en la Figura 28, no se han utilizado todas las variables. Ello puede explicarse atendiendo a la Figura 29, que muestra la importancia de las variables en la construcción del modelo, clasificadas atendiendo a su contribución a la reducción de entropía.

Figura 28. Representación del árbol clasificatorio final.



Fuente: Elaboración propia.

Figura 29. Importancia de las variables.



Fuente: Elaboración propia.

6.3.2. Validación de los Resultados

La precisión del modelo sobre el conjunto de prueba fue de 0.7524. En la Figura 30 se recoge la matriz de confusión para el *test data*. En ella, se recogen las posibilidades de clasificación de las distintas observaciones. Como podemos comprobar, la predicción puede ser correcta (verdadero positivo (VP) y verdadero negativo (VN)) o incorrecta (falso positivo (FP) o falso negativo (FN)).

Figura 30. Matriz de confusión para el conjunto de prueba.

		Reference	
Prediction		NO	YES
	NO	129	40
	YES	11	26

Fuente: Elaboración propia.

Es por ello que debemos analizar otras dos métricas de validez: sensibilidad y especificidad [Pita y Pértegas (2003)].

- La sensibilidad mide la probabilidad de clasificar correctamente el impago de una hipoteca, en otras palabras, la capacidad del modelo para detectar el impago.

$$Sensibilidad = \frac{VP}{VP + FN}$$

- La especificidad mide la probabilidad de clasificar correctamente el pago de una hipoteca, es decir, la capacidad del modelo para detectar que un individuo va a ser capaz de satisfacer las cuotas hipotecarias.

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

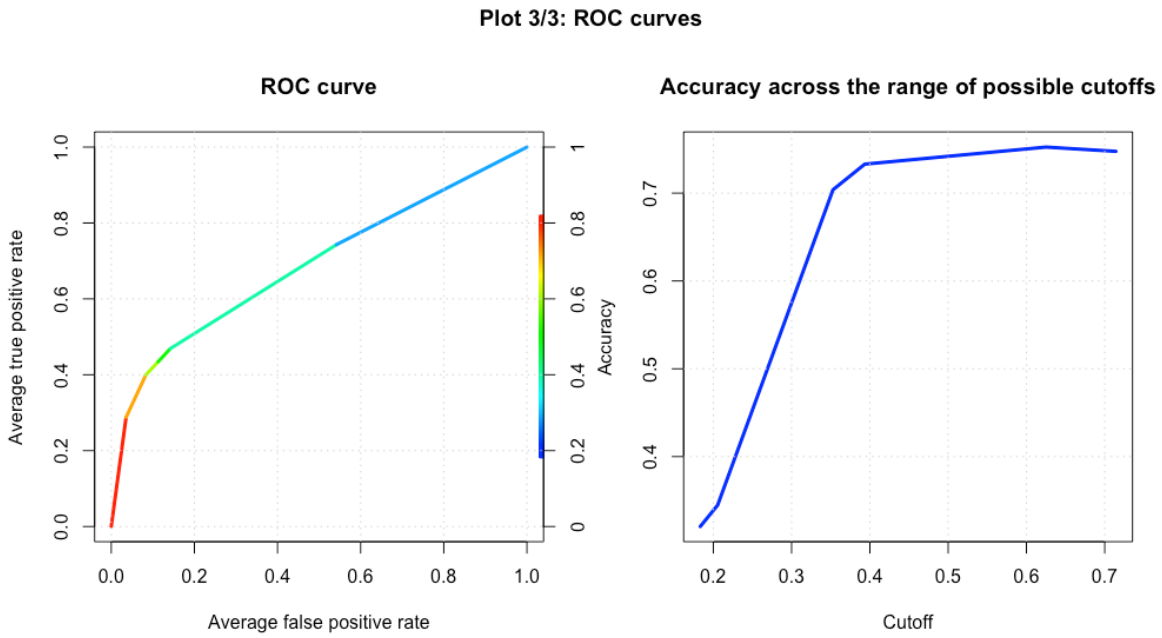
Este modelo presentó una sensibilidad de 0.3939 y una especificidad de 0.9214. A la luz de estas métricas, podemos determinar que el modelo no es adecuado debido a que su capacidad de predecir un impago es relativamente baja. Se probaron otras combinaciones de parámetros, a fin de mejorar la sensibilidad del modelo, sin embargo los resultados fueron similares. A pesar de ello, al volver a ejecutar el mismo código en un nuevo proyecto, se logró una sensibilidad de 0.5455, y una especificidad de 0.8357. Ello puede explicarse debido al fenómeno de aleatoriedad en la constitución del *training set* mencionado en diversas ocasiones a lo largo del trabajo. En este caso, dada la mayor proporción de la clase "NO", la división aleatoria de los datos en entrenamiento y prueba tiene una gran repercusión en la efectividad del modelo. Como se ha ido explicando a lo largo del trabajo, a esta problemática la denominamos varianza, y para mitigarla se empleó un modelo de *Random Forest*, que se explicará posteriormente.

En Machine Learning, es importante estudiar el rendimiento de un modelo. En los métodos clasificatorios, se suele utilizar lo que denominamos *Area Under the Curve* (AUC)[19], en relación con la curva *Receiver Operating Characteristics* (ROC)[20]. Como podemos apreciar en la Figura 31, la curva ROC mide el rendimiento de un modelo para distintos puntos de corte. En otras palabras, indica cómo de capaz es el modelo de distinguir entre clases. Cuanto mayor sea el AUC (más próximo a 1), mejor será el modelo prediciendo

[19] Representa el grado o medida de separabilidad entre clases.
[20] Se trata de una curva de probabilidad.

si el cliente va a incurrir en impago o no. Cuando el AUC es aproximadamente 0.5 en un problema de clasificación binaria, ello implicará que el modelo no tiene capacidad discriminatoria para distinguir entre clases positivas y negativas [Narkhede (2018)]. En el caso del presente modelo, el AUC fue 0.6931, lo cual indica que tiene una probabilidad del 69.31% de distinguir entre ambas clases.

Figura 31. Curva ROC del árbol clasificatorio y precisión para distintas probabilidades de corte en validación.



Fuente: Elaboración propia.

Comprobamos que fijando la probabilidad de corte[21] en un valor superior, aumenta la precisión. Por ello, se varió el criterio clasificatorio, haciendo que las observaciones fueran clasificadas como "YES" (usando el árbol previamente entrenado) sólo cuando la predicción de sus probabilidades[22] fuera superior a 0.62. Sin embargo, tal y como he mencionado anteriormente, el objetivo del modelo es predecir los clientes "*Default*" (que incurren en impago), y esta nueva clasificación resultó en una reducción de observaciones positivas, y por ende una reducción de sensibilidad.

[21] Punto de corte a partir del cual se clasificará una observación de forma positiva (en este caso, como impago).
[22] El modelo asigna a cada observación una probabilidad clasificatoria. Un modelo perfectamente robusto asignaría sólo probabilidades 1 (si predice el impago) o 0 (si predice el pago).

6.4. RANDOM FOREST

Dada la alta varianza a la que se pueden ver sometidos los árboles de decisión (aún más en este caso, que cuenta con una base de datos con una proporción sustancialmente mayor de una de las clases), variando el *training set* pueden obtenerse modelos completamente dispares. Es por ello que nace el método *Random Forest*. Tal y como se explicó anteriormente, éste hace uso de la técnica de *bagging*[23], implementando una serie de mejoras.

De esta forma, los resultados obtenidos siguiendo esta metodología fueron sustancialmente mejores que los logrados con el árbol de decisión. A pesar de que la precisión disminuyó a 0.733, la sensibilidad del modelo en el conjunto de prueba resultó ser notablemente superior (como puede apreciarse en la Figura 32), aumentando hasta 0.5152 (mientras que la especificidad se reducía a 0.8357). Estos datos se obtuvieron con un *Random Forest* con $n = 100$[24] y $m = 4$[25]. Debe notarse que, incrementando la complejidad del modelo (m), se lograba incrementar la precisión, en perjuicio de la sensibilidad. Sin embargo, como se debe priorizar la detección de la clase "YES", se opta por escoger el modelo de $m = 100$ frente a las otras alternativas probadas ($m = 200$ o $m = 1000$).

Figura 32. Matriz de confusión para el conjunto de prueba.

Validation Confusion Matrix

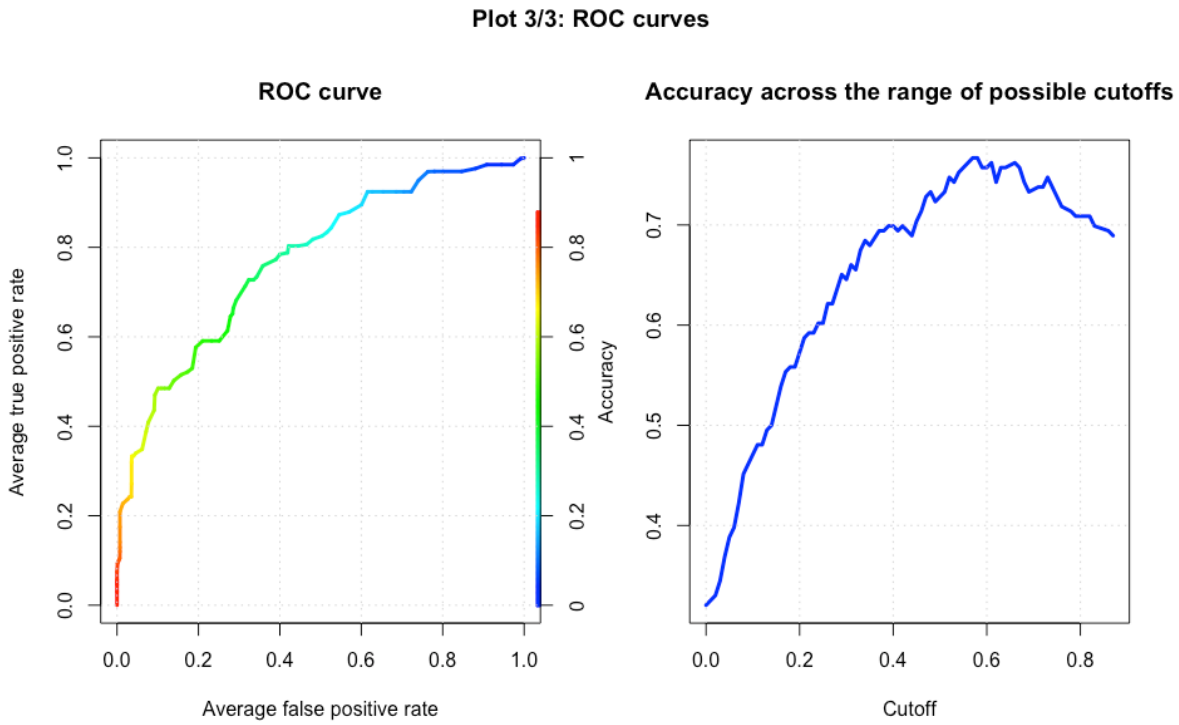
		Reference	
Prediction		NO	YES
	NO	117	32
	YES	23	34

Fuente: Elaboración propia.

En cuanto al rendimiento del modelo, se puede apreciar en la Figura 33 una mejoría atendiendo al AUC de la curva ROC para el conjunto de validación, que aumentó a 0.7690.

[23] Recordemos que su objetivo es entrenar diversos modelos mediante *subsets* obtenidos de la base de datos principal, usando para el modelo final una media de todos ellos, en una especie de votación.
[24] Siendo n el número de árboles que integran el modelo.
[25] Siendo m el número de variables máximo (escogidas aleatoriamente en cada nodo) que el modelo puede usar para dividir un nodo.

Figura 33. Curva ROC del *Random Forest* y precisión para distintas probabilidades de corte en validación.



Fuente: Elaboración propia.

6.5. COMPARACIÓN DE AMBOS MODELOS

Una vez se han entrenado ambos modelos y se ha hallado una configuración paramétrica óptima para cada uno, se recogen los resultados de precisión para cada uno de ellos. Cada modelo tiene 30 resultados (3 repeticiones de *10-fold cross validation*). Ello permite comparar los resultados de las distribuciones de precisión de ambos algoritmos. Tal y como podemos observar en las Figuras 34 y 35, tanto la precisión como el coeficiente de kappa[26] en el *Random Forest* es superior al del árbol clasificatorio, demostrando que el primero presenta una mayor robustez que el segundo.

En cuanto a cómo podría mejorarse el rendimiento en ambos modelos, dada la dispar proporción de ambas clases en el *data-set*, probablemente el método de *Cross-Validation*

[26] El coeficiente de kappa fue introducido por Jacob Cohen en 1960. Se trata de una medida estadística, considerada más robusta que la precisión, debido a que tiene en cuenta la probabilidad de que el acierto tenga lugar de forma aleatoria [Cohen (1960)].

Stratified K-Fold habría sido más efectivo para entrenar el modelo. Éste habría incluido en cada uno de los *folds* un número proporcional de clases "YES" y "NO", lo cual podría haber contribuido a una mejora en la sensibilidad de ambos algoritmos.

Figura 34. Comparativa de la precisión y el coeficiente *kappa* en ambos modelos.

Accuracy

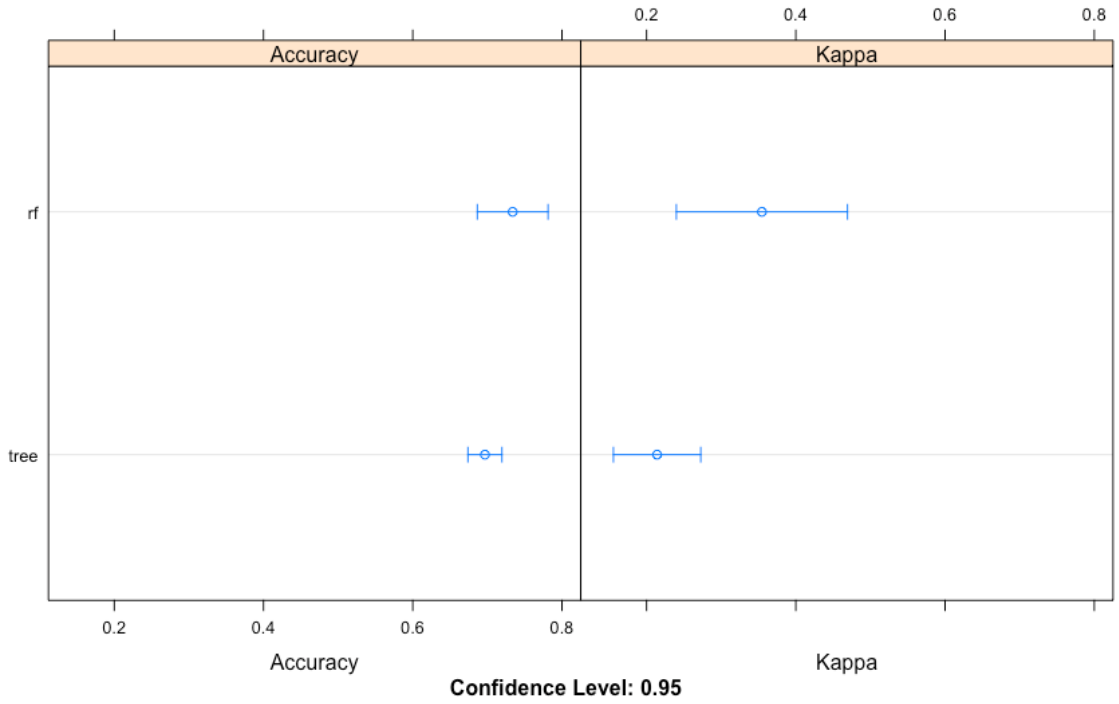
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA´s
Tree	0.6506	0.6777	0.6928	0.6967	0.7052	0.7500	0
Random Forest	0.6265	0.6846	0.7544	0.7339	0.7724	0.8214	0

Kappa

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA´s
Tree	0.0824	0.1683	0.1861	0.2141	0.2690	0.3415	0
Random Forest	0.0879	0.2228	0.4134	0.3546	0.4470	0.5783	0

Fuente: Elaboración propia.

Figura 35. Diagrama de caja comparativo de la precisión y la *kappa* de ambos modelos.



Fuente: Elaboración propia.

7. ANÁLISIS DE LAS COTIZACIONES DEL IBEX 35 EN FUNCIÓN DE SU RENDIMIENTO MEDIO Y SU VOLATILIDAD

El objetivo de esta sección es poner en práctica la teoría introducida en el apartado 4.3, dedicado al *clustering*, a través de un experimento programado en lenguaje R. Se pretende, por un lado, analizar la eficiencia del mercado a través del estudio de la rentabilidad media y la volatilidad de las empresas del IBEX 35; y por otro analizar potenciales *clusters* dentro de dicho índice, estudiando si se corresponden con las industrias a las que pertenecen. Como ya se avanzó, se hará uso de los algoritmos *k-means* y *hierarchical clustering*.

7.1. INTRODUCCIÓN

En este experimento, se analizaron 33[27] de las empresas que componen el IBEX 35. Para ello, se obtuvieron los datos de cotización de dichas compañías desde el 6 de mayo de 2015 hasta el 5 de junio de 2019, con frecuencia intradiaria.

En primer lugar, se han dividido las compañías en función del sector al que pertenecen, siguiendo para ello la distinción efectuada por la Bolsa de Madrid, quedando de la siguiente manera:

- **Basic Materials, Industry and Construction**: Acciona, Acerinox, ACS Group, ArcelorMittal, CIE Automotive, Ferrovial, Siemens Gamesa y Técnicas Reunidas.
- **Consumer Goods**: Grifols, Inditex y Viscofán.
- **Consumer Services**: Aena, IAG, Mediaset España y Meliá Hotels.
- **Financial Services**: Banco Sabadell, Banco Santander, Bankia, Bankinter, BBVA, CaixaBank y Mapfre.
- **Petrol and Power**: Enagás, Endesa, Iberdrola, Eléctrica de España y Repsol.
- **Technology and Telecommunications**: Amadeus IT Group, Cellnex Telecom, Indra y Telefónica.

[27] Debido a que la fecha de salida a bolsa entre las empresas del IBEX 35 es muy dispar, y que se pretendía trabajar con un horizonte temporal relativamente amplio, se excluyeron de este análisis ENCE (en Yahoo Finance, fuente de la cual se han obtenido los datos de cotización, no figuraban todas las cotizaciones históricas de la compañía) y Naturgy (comienza a cotizar en junio de 2018, tras cambiar su denominación Gas Natural Fenosa).

- **Real Estate Services**: Inmobiliaria Colonial y Merlin Properties.

En segundo lugar, se ha procedido al cálculo del rendimiento medio diario[28] de las acciones de cada empresa. Para ello se ha hecho uso de la siguiente fórmula:

$$Daily\ Return\ on\ Stock\ (R_t) = \frac{P_1 - P_0}{P_0}$$

$$Average\ Return\ (R_m) = \sum_{t=m}^{m-n} R_t * \frac{1}{n}$$

Siendo P_0 el precio de cierre del día anterior, y P_1 el precio de cierre del día actual.

En tercer lugar, se ha calculado la volatilidad[29] de dichas compañías, mediante el siguiente procedimiento:

$$\sigma = \sqrt{\sum_{t=m}^{m-n} (R_t - R_m)^2 * \frac{1}{n-1}}$$

7.2. ANÁLISIS EXPLORATORIO

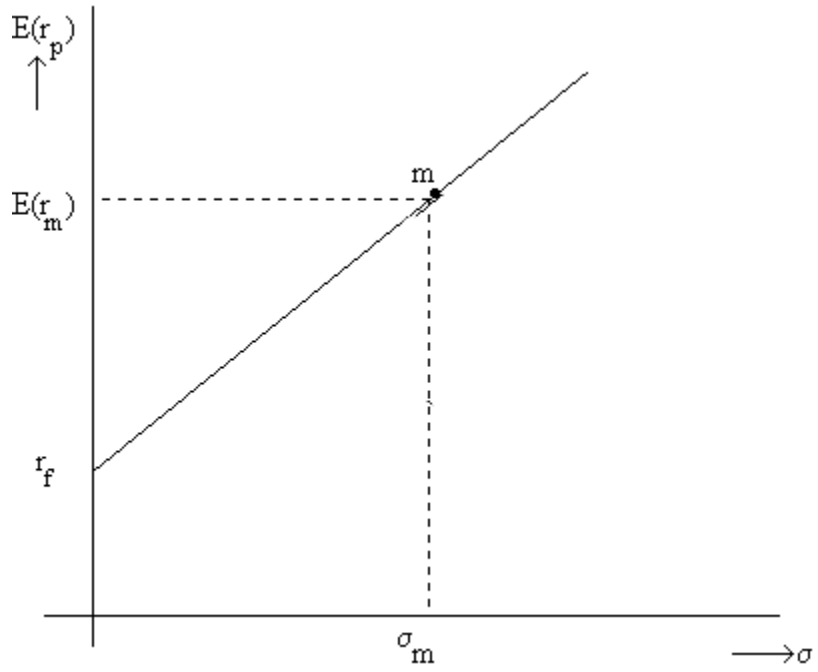
Al comparar las empresas del IBEX 35 en función de las mencionadas métricas, se puede analizar su semejanza con la Línea del Mercado de Capitales (LMC), representada en la Figura 36. Ésta muestra gráficamente el binomio rentabilidad-riesgo según el modelo *Capital Asset Pricing Model* (CAPM). Se trata de un concepto teórico que representa aquellas acciones que optimizan la relación entre rentabilidad y volatilidad, maximizando de esta manera su rendimiento. Asumiendo que el mercado es perfectamente eficiente, al representar gráficamente las empresas, éstas deberían posicionarse relativamente próximas a dicha curva.

Asimismo, será interesante la comparación entre empresas del mismo sector, debido a que su posición a lo largo de la curva CAPM determinará su asignación a uno u otro *cluster* (y de esta forma se podrá comprobar si las empresas se comportan de manera homogénea

[28] Mide el incremento medio diario del precio de una acción a lo largo del periodo de estudio.
[29] Se trata de una medida estadística que mide la dispersión de los rendimientos de una acción (cuanto menor es la dispersión, menor volatilidad tendrá una acción).

dentro de cada sector[30], o si cobra más sentido su comparación con las de su respectivo *cluster*).

Figura 36. Representación de la Línea del Mercado de Capitales.



Fuente: De Bedoya (2013).

La Figura 37 muestra la representación gráfica de las empresas estudiadas en función de su volatilidad y rendimiento medio. Como podemos apreciar, sigue una distribución relativamente parecida a la LCM, aunque quedan latentes diversas ineficiencias. Como podemos comprobar hay acciones que ofrecen mayor rentabilidad que otras, a un nivel parejo de riesgo (i.e. ANA[31] y GRF); e incluso hay otras que ofrecen una rentabilidad notoriamente superior, con una volatilidad mucho menor (SGRE y TEF).

De esta misma manera, queda latente la falta de homogeneidad entre empresas pertenecientes al mismo sector (*i.e.* FER, ACS frente a ACX y TRE), en parte explicable porque pertenecen a subsectores distintos (TRE lleva a cabo actividades de ingeniería y

[30] Debemos ser conscientes de que los sectores pueden, a su vez, dividirse (*i.e.* Bankinter forma parte del sub-sector bancario, mientras que Mapfre se integra en el de seguros).
[31] La tabla del Anexo 1 contiene todos los *tickers* de las empresas del IBEX 35.

derivados, mientras que FER opera en el sector de la construcción) pero también atribuible a la situación particular de cada entidad, que conlleva una reacción específica por parte del mercado.

Figura 37. Comparación de las empresas del IBEX 35 atendiendo al binomio rentabilidad-riesgo.

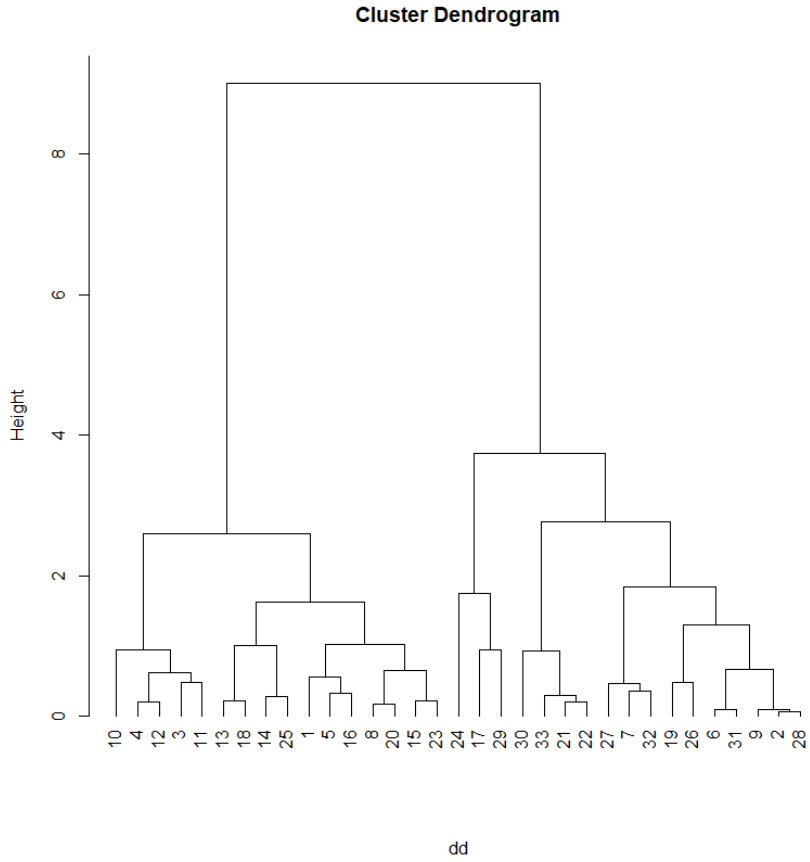
Fuente: Elaboración propia.

Tras haber realizado un análisis exploratorio de los datos, se procederá a someterlos a los modelos de *hierarchical* y *k-means clustering*.

7.3. HIERARCHICAL CLUSTERING

Tal y como se explicó en el apartado 4.3, se parten de 33 *clusters* iniciales que se unen atendiendo al criterio de proximidad (en este caso se ha aplicado el criterio *euclidean*) en el *data space*, hasta converger en uno.

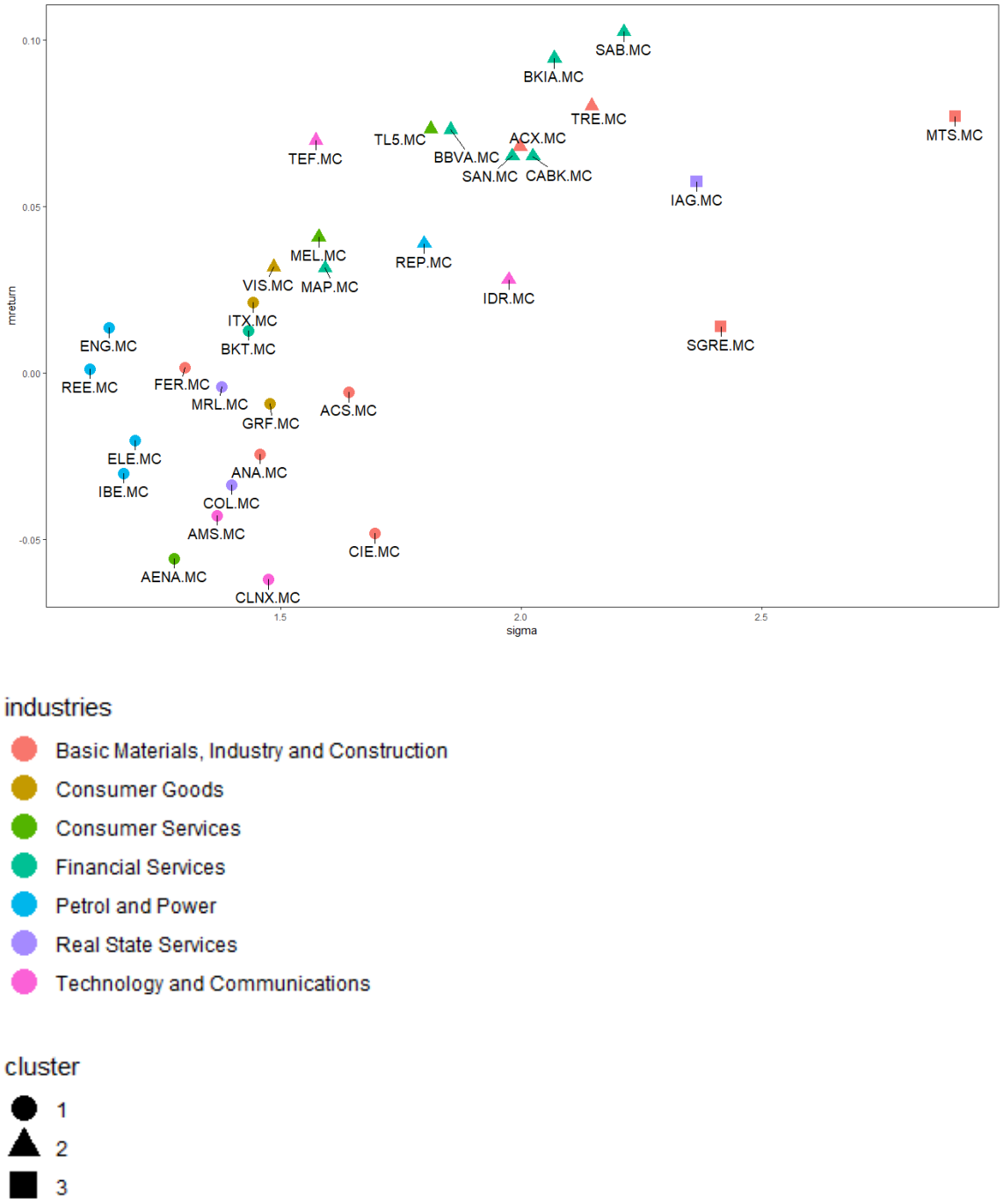
Figura 38. Dendrograma en *hierarchical clustering*.



Fuente: Elaboración propia.

En función del dendrograma representado en la Figura 38, parece bastante evidente que el número óptimo de *clusters* es 2. Sin embargo, si a su vez se tiene en cuenta la localización de los *data points* a lo largo del *data space*, podría argumentarse la existencia de un tercer *cluster*. De esta manera, se decide fijar el número de *clusters* en 3, obteniendo así el resultado contenido en la Figura 39.

Figura 39. Representación del modelo *hierarchical clustering*.



Fuente: Elaboración propia.

De esta forma, pueden observarse una serie de patrones en el *data-set*. En primer lugar, se debe mencionar la población por grupo, homogénea en los *clusters* 1 y 2 (que cuentan con 16 y 14 *data points*, respectivamente), y muy inferior en el número 3. Ello

demuestra que el número óptimo debe fijarse en 2, a pesar de que es cierto que el grupo formado por IAG, SGRE y MTS parece situarse notablemente lejos del resto de puntos (presentando una volatilidad acusada en comparación con las demás compañías).

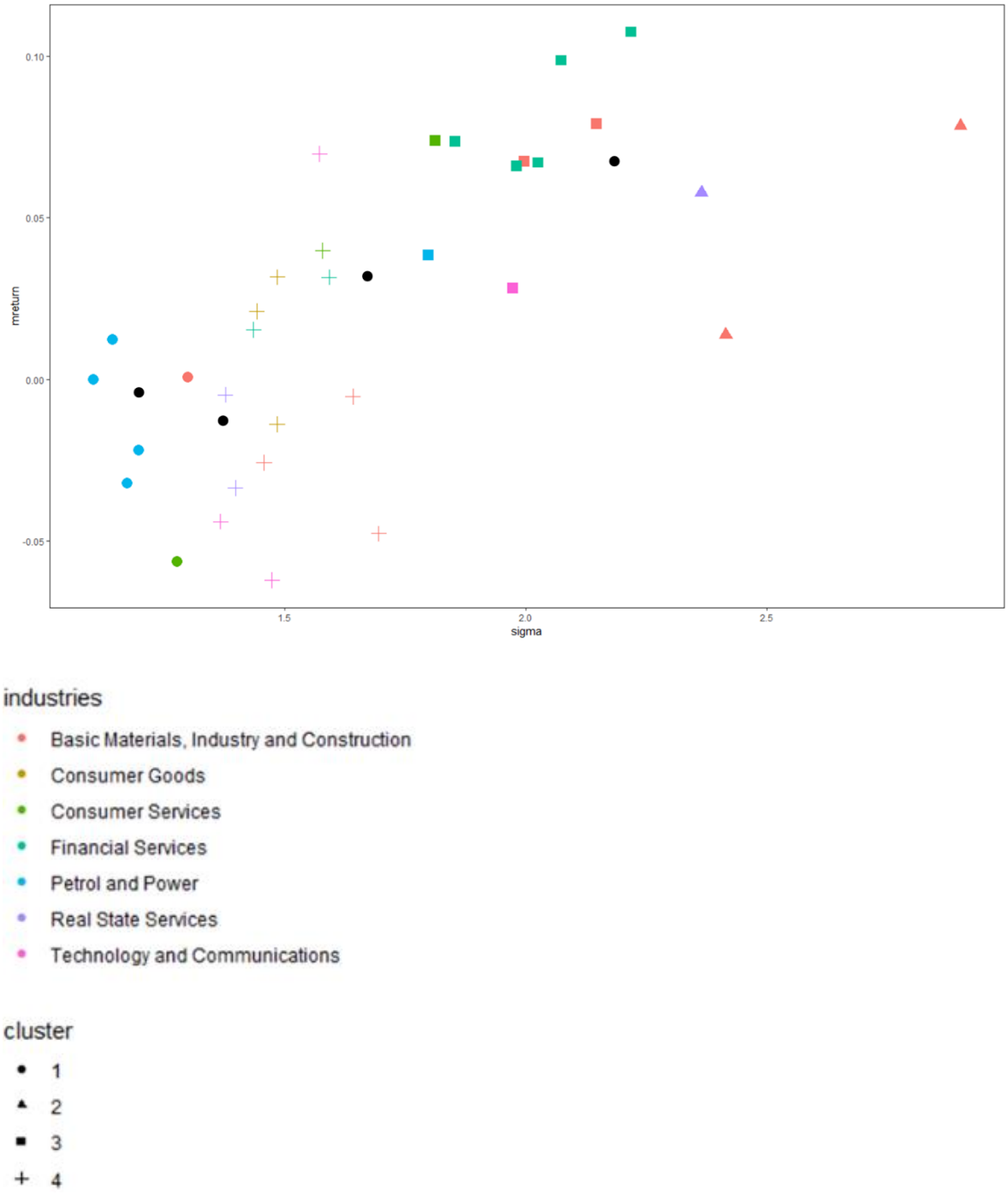
En segundo lugar, se puede apreciar que, para llevar a cabo la división, el algoritmo se ha basado en la métrica de rentabilidad media, fijando la línea divisoria en torno a 0.02. Ello podría indicar que la relación entre los *data points* se explica de mejor manera mediante esta medida.

En tercer lugar, tal y como se comentó anteriormente, es interesante comprobar como los *clusters* no coinciden necesariamente con la división por sectores. Puede deducirse que el sector con mayor robustez es el de *Financial Services*, y en concreto el subsector bancario, ya que sus *data points* presentan la mayor proximidad (salvo por BKT, agrupado en el *cluster* número 2). Esto nos indica que el mercado puntúa de manera similar a este grupo de empresas. El sector de *Basic Materials, Industry and Construction* tiene una dispersión verdaderamente acusada (con al menos un miembro en cada *cluster*). Sin embargo, hay ciertas diferencias entre subsectores. El de construcción (ACS, FER y ANA) pertenece al mismo *cluster* (1), con *data points* próximos, mientras que el de metales (CIE y ACX) pertenecen cada uno a un *cluster* distinto (1 y 2, respectivamente), separados por una amplia distancia.

7.4. K-MEANS CLUSTERING

Tal y como se mencionó en el apartado 4.3, para llevar construir un modelo *k-means* se debe comenzar por fijar un número de centroides (punto ubicado en el centro del *cluster* en torno al cual se agruparan los *data points*), es decir, el número de agrupaciones que se desea obtener. Tras llevar a cabo los procesos iterativos descritos previamente, el modelo es capaz obtener el número de *clusters* predefinidos, cuyos puntos se congregan en torno a sus respectivos centroides.

Figura 40. Representación del modelo *k-means clustering*.



Fuente: Elaboración propia.

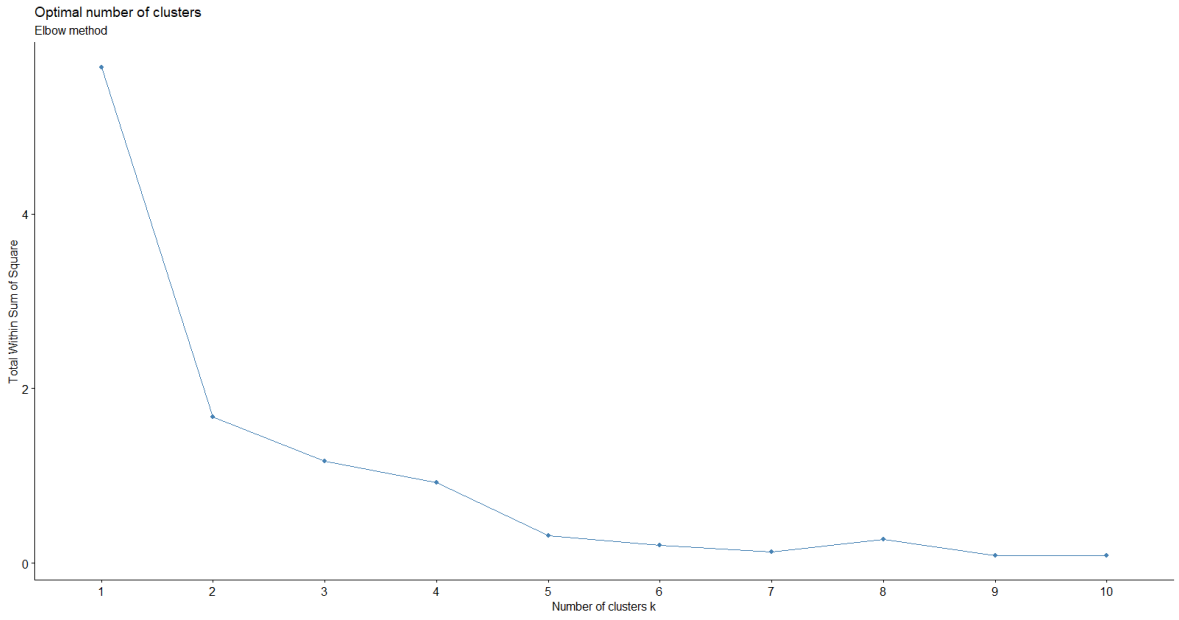
En este caso, el algoritmo parece haber hecho una partición fundamentándose en la volatilidad de las compañías. En este caso, la población de los patrones es más homogénea,

componiéndose de 10 *data points* el primer *cluster*, de 8 el segundo, de 9 el tercero y de 6 el cuarto.

En cuanto al primer *cluster* (el cuarto también parece verse afectado por este mismo problema), presenta una dispersión relativamente alta, donde algunos puntos están ubicados a gran distancia del centroide (representados por los puntos negros). Ello hace que deba plantearse la robustez o fiabilidad de dicho grupo, ya que esta falta de proximidad pone en entredicho su validez. De nuevo, el sector que presenta una mayor robustez es el de *Financial Services*, en concreto el subsector bancario. En cuanto al subsector de construcción, que en el ejercicio anterior fue clasificado en su totalidad en un *cluster*, en este caso, y a pesar de la proximidad entre sus puntos, cada empresa ha sido incluida en un grupo distinto.

El siguiente proceso es el de hacer uso de las distintas técnicas de elección del número óptimo de *clusters*. En este caso, se han empleado 3. En primer lugar, la Figura 41 representa el método *elbow* (codo), cuyo objetivo es minimizar la variación intra-*cluster*, minimizando para ello la suma de cuadrados dentro del mismo. Según este método, el número óptimo de *clusters* se encontraría dentro del rango de 3 a 5.

Figura 41. Selección del número de *clusters* óptimo vía *elbow method*.



Fuente: Elaboración propia.

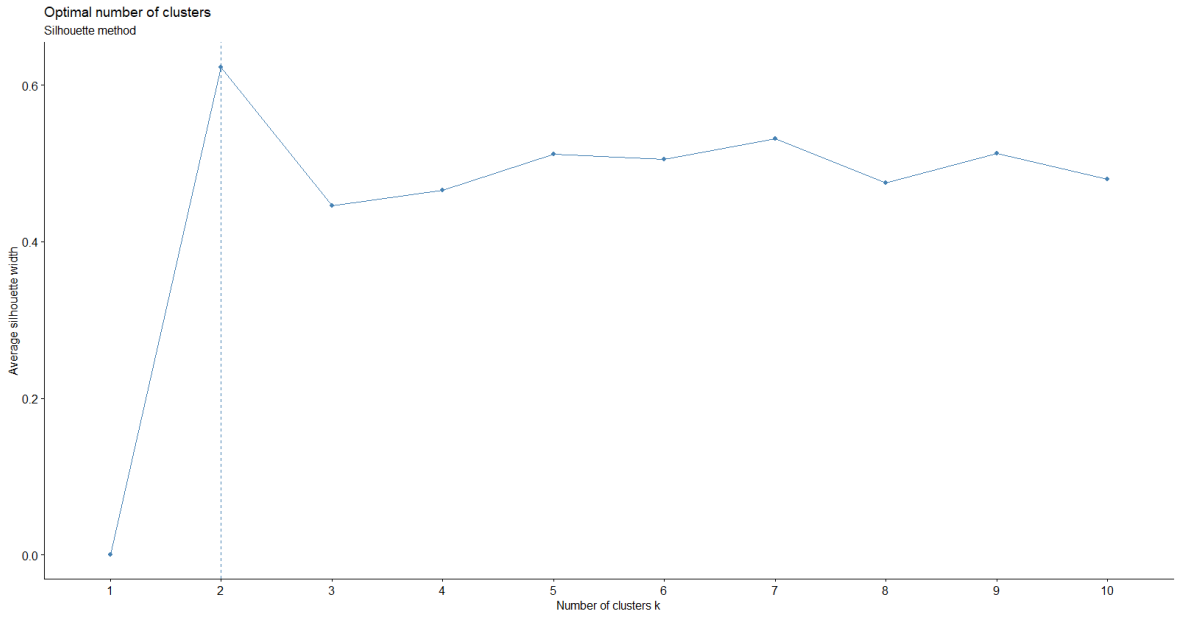
En segundo lugar, se encuentra el método *silhouette* (silueta), que mide cómo de bien se ha incluido en un *cluster* una observación, y estima la distancia media entre *clusters*. Se calcula de la siguiente manera:

$$S_i = \frac{(b_i - a_i)}{\max((b_i, a_i)}$$

Donde a_i representa la dispersión media i y todos los puntos del *cluster* a los que pertenece, y b_i representa la dispersión con el *cluster* más cercano del que no forma parte.

En la Figura 42 se puede apreciar que el número óptimo de grupos siguiendo esta metodología se sitúa en 2, aunque no presenta una cifra demasiado alta (0.62), luego no podemos afirmar que el modelo sea especialmente preciso.

Figura 42. Selección del número de *clusters* óptimo vía *silhouette method*.

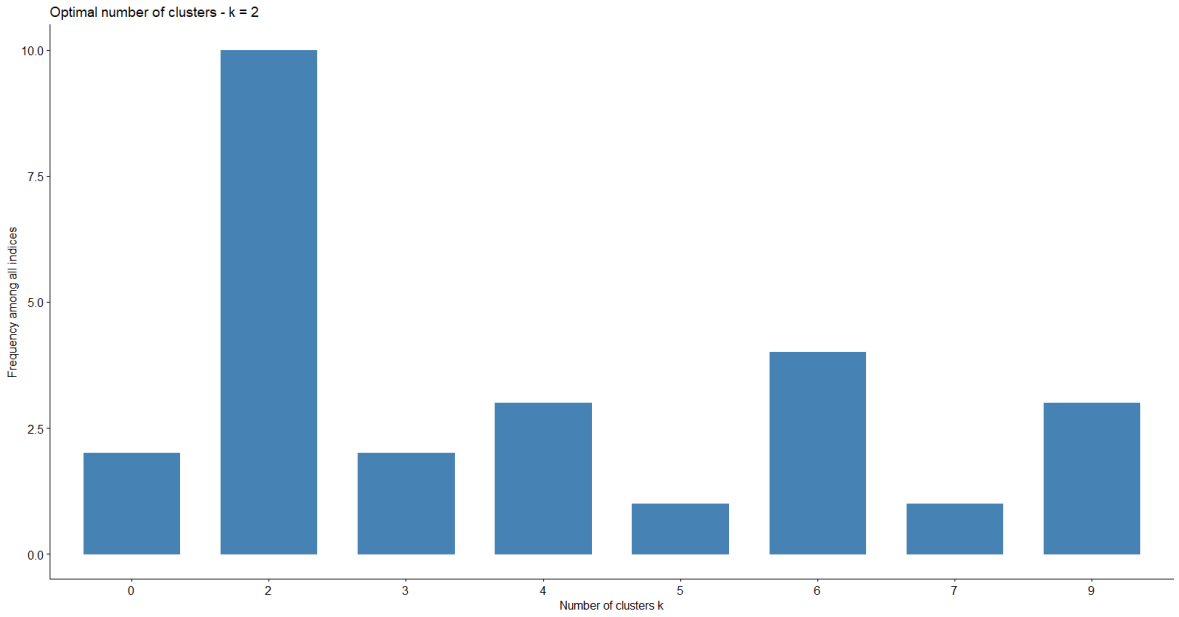


Fuente: Elaboración propia.

Por último lugar, encontramos la función *NbClustI()*, que prueba 30 índices para escoger, por mayoría, el número de *clusters* óptimo (fijando el mínimo en 2 y el máximo en 9). En la Figura 43, se puede apreciar que la mayoría de índices seleccionaron el número 2 como el óptimo. Este sistema suele ser el más preciso, ya que propone al usuario el mejor

esquema de *clustering* obtenido de diferentes resultados mediante la variación de todas las combinaciones de números de *clusters*, medidas de dispersión y métodos de *clustering*.

Figura 43. Selección del número de clusters óptimo vía *NbClust()*.



Fuente: Elaboración propia.

7.5. ANÁLISIS DE LOS RESULTADOS

Como se ha podido comprobar, el método *clustering* es útil para detectar ciertos patrones en un *data-set* no etiquetado. Si bien es cierto que, dependiendo del algoritmo concreto que utilicemos, será necesario un conocimiento previo de los datos, en aras de fijar un número de *clusters* que se acerque al óptimo lo máximo posible, hay una amplia flexibilidad que permite al usuario construir un modelo en función de lo que estime adecuado.

En este caso en concreto, ambos modelos han coincidido en que la manera óptima de agrupar los datos era en 2 grandes grupos, aunque cada uno se ha basado en una métrica distinta a la hora de llevar a cabo estas divisiones.

Este tipo de UL es útil debido a que permite al usuario percatarse de determinados patrones en los datos. Por ejemplo, hemos podido comprobar que algunos subsectores, como el bancario, reciben un tratamiento homogéneo por parte del mercado, exceptuando BKT, que se asemeja más al subsector de seguros, con un posicionamiento similar al de MAP.

Desde el punto de vista de un inversor, es interesante profundizar en esta relación, a fin de comprender los motivos por los cuales el mercado penaliza a algunos miembros concretos de un sector o subsector, a pesar de compartir características con sus en teoría homólogos. Ello podría llevar a agrupar compañías, no en base a sus atributos teóricos (empleados para definir una industria), sino a otros aspectos definidos por el propio inversor, sobre los cuales quiere construir su estrategia.

Así, se abren al usuario una infinidad de oportunidades para construir estrategias de inversión a medida, que se diferencien del resto de mercado, pudiendo así beneficiarse de patrones intrínsecos del mercado, en un principio indetectados. Además, también pueden emplearse estos modelos no supervisados, para entrenar modelos supervisados y mejorar su rendimiento. Por ejemplo, podría aplicarse la técnica de *clustering* sobre el data-set de prestatarios mencionado en la sección 6, para identificar aquellos grupos con mayor riesgo crediticio, posteriormente entrenando el modelo clasificatorio con esta información.

8. CONCLUSIÓN

Vivimos en un mundo en el cual la generación y análisis de datos cada vez cobra una mayor relevancia. Las empresas que no sean capaces de anticiparse a las demandas de un futuro, eminentemente digital, desaparecerán como consecuencia de una selección natural darwiniana.

En la industria financiera, en concreto, este fenómeno contribuye a la reducción de los costes operacionales gracias a la automatización de procesos. Si nos remitimos a la sección 6 de este trabajo, el hecho de que se pueda confeccionar un modelo que sea capaz de predecir el riesgo de impago de un potencial cliente, convierte en redundante la labor de un gran número de trabajadores en las sucursales bancarias de un banco. Asimismo, contribuye a homogeneizar el marco regulatorio y a reforzar la seguridad bancaria. Por ejemplo, una de las principales causas de la crisis económica del 2007 fue la falta de rigor de un gran número de entidades bancarias a la hora de garantizar préstamos. El panorama hipotecario a nivel mundial era realmente alarmante, y había sido generado por el incentivo que tenían los trabajadores de un banco en conceder un préstamo (ya que ello implicaba recibir una comisión a cambio). Si aplicáramos un modelo de riesgo crediticio, mediante el cual se restringiera la posibilidad de garantizar un préstamo a aquellos clientes con una puntuación determinada, se reduciría el riesgo al que se expondría un banco (y se eliminaría el sesgo humano).

Si bien es cierto que aún queda mucho por recorrer hasta alcanzar la AGI o la ASI, es indiscutible que los avances en materia de Machine Learning son exponenciales. Siendo el incremento en la generación de datos y la capacidad computacional, junto con la reducción del coste de almacenamiento, los pilares sobre los cuales se fundamenta el auge de este fenómeno, es inevitable pensar que un futuro próximo esta tecnología será no sólo una piedra angular de la industria financiera, sino de la sociedad en general.

9. REFERENCIAS BIBLIOGRÁFICAS

Agarwal, S. (2018). CIS 520: Machine Learning: *Understanding Generalization Error – Bounds and Decompositions*, *Lecture 11* [Slides de PowerPoint]. Obtenido el 20/03/2019 de http://www.shivani-agarwal.net/Teaching/CIS-520/Spring-2018/Lectures/Reading/error-bounds-decompositions.pdf.

Álvarez-Teleña, S. (2017, 31 de octubre). *Nuestros avatares, los mejores aliados con la IA*. Obtenido el 05/03/2019 de https://www.youtube.com/watch?v=WlP9FKeTsac.

Anyoha, R. (2017, 28 de agosto). *The History of Artificial Intelligence*. Obtenido el 07/03/2019 de http://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/.

Bostrom, N. (2006). How long before superintelligence? *Linguistic and Philosophical Investigations*, 5 (1), 11-30.

Breazeal, C. (1998). *Kismet*. Obtenido el 09/03/2019 de http://www.ai.mit.edu/projects/humanoid-robotics-group/kismet/kismet.html.

Breiman, L. (1996). Bagging Predictors. Machine Learning, 24, 123–140.

Brownlee, J. (2016, 22 de abril). *Bagging and Random Forest Ensemble Algorithms for* Machine Learning. Obtenido el 24/04/2019 en https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, XX (1).

Data Never Sleeps 6.0. (2018). Obtenido el 10/03/2019 de https://www.domo.com/learn/data-never-sleeps-6.

De Bedoya, D. (2013, 12 de enero). *¿Existen activos con ß=0? ¿y con ß negativa?*. Obtenido el 22/04/2019 de http://queaprendemoshoy.com/existen-activos-con-s0-y-con-s-negativa/.

Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, 1-15.

Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *Communications of the ACM*, 55 (10), 78-87.

Donges, N. (2018, 7 de marzo). Gradient Descent in a Nutshell. Obtenido el 13/03/2019 de https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0.

Faggella, D. (2019a, 4 de febrero). *Artificial Intelligence's double-edged role in Cyber Security – with Dr. Roman V. Yampolskiy*. Obtenido el 11/03/2019 de https://emerj.com/ai-podcast-interviews/artificial-intelligences-double-edged-role-in-cyber-security-with-dr-roman-v-yampolskiy/.

Faggella, D. (2019b, 19 de febrero). *The rise of Neural Networks and Deep Learning in our everyday lives – A conversation with Yoshua Bengio*. Obtenido el 11/03/2019 de https://emerj.com/ai-podcast-interviews/the-rise-of-neural-networks-and-deep-learning-in-our-everyday-lives-a-conversation-with-yoshua-bengio/.

Freund, Y., y Schapire, R. E. (1999). A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14 (5), 771-780.

Fumo, D. (2017, 15 de junio). *Types of* Machine Learning *algorithms you should know*. Obtenido el 12/03/2019 de https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.

Gates, M. (2016, 17 de mayo). The Bill & Melinda Gates Foundation announces $80 Million commitment to close gender data gaps and accelerate progress for women and girls. *Bill & Melinda Gates Foundation*. Obtenido el 06/03/2019 de https://www.gatesfoundation.org/Media-Center/Press-Releases/2016/05/Gates-Foundation-Announces-80-Mill-Doll-Comm-Closing-Gender-Data-Gaps-Acc-Progress-for-Women-Girls.

Grover, P. (2018, 5 de junio). *5 Regression Loss Functions All Machine Learners Should Know: Choosing the right loss function for fitting a model*. Obtenido el 12/03/2019 de https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0.

Guide to the IoT: How billions of online objects are making the web wiser. (2015). Obtenido el 10/03/2019 de https://www.intel.com/content/dam/www/public/us/en/images/iot/guide-to-iot-infographic.png.

Guillén, B. (2016, 04 de septiembre). *El verdadero padre de la inteligencia artificial*. Obtenido el 28/05/2019 de https://www.bbvaopenmind.com/tecnologia/inteligencia-artificial/el-verdadero-padre-de-la-inteligencia-artificial/.

Gupta, P. (2017, 17 de mayo). Decision Trees in Machine Learning. Obtenido el 13/04/2019 de https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

James, G., Witten, D., Hastie, T. y Tinshirani, R. (2013). *An Introduction to Statistical Learning*. Nueva York: Springer.

Kam Ho, T. (1995). Random decision forests. *Proceedings of the Third International Conference on Document Analysis and Recognition* (1), 278.

Kasparov vs. Deep Blue: The match that changed history. (2018, 12 de octubre). Obtenido el 09/03/2019 de https://www.chess.com/article/view/deep-blue-kasparov-chess.

Kaushik, S. (2016, 3 de noviembre). *An Introduction to Clustering and different methods of clustering*. Obtenido el 24/04/2019 de https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/.

K-Means Clustering – What it is and How it Works. (sin fecha). Obtenido el 23/04/2019 de http://www.learnbymarketing.com/methods/k-means-clustering/.

Kolanovic, M. y Krishnamachari, R. T. (2017). *Big Data and AI Strategies:* Machine Learning *and Alternative Data Approach to Investing*. New York: J.P. Morgan. Obtenido de https://www.cfasociety.org/cleveland/Lists/Events%20Calendar/Attachments/1045/BIG-Data_AI-JPMmay2017.pdf.

Kulkarni, M. (2017, 7 de septiembre). Decision Trees for Classification: A Machine Learning Algorithm. Obtenido el 12/04/2019 de https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html.

Lavrenko, V. (2014, 19 de enero). *Overfitting 2: training vs. future error*. Obtenido el 18/03/2019 de https://www.youtube.com/watch?v=Xeo6LGcsxkg.

Machine Learning. (2019). Obtenido el 11/03/2019 de https://www.coursera.org/learn/machine-learning.

Mueller, J. P. y Massaron, L. (sin fecha). Resorting to Cross-Validation in Machine Learning. Obtenido el 2/04/2019 de https://www.dummies.com/programming/big-data/data-science/resorting-cross-validation-machine-learning/.

Muñoz, A., Sánchez, E., y Portela, J. (2018). Machine Learning: Chapter 2 – Classification II [Slides de PowerPoint].

Narkhede, S. (2018, 26 de junio). Understanding AUC - ROC Curve. Obtenido el 15/04/2019 de https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

Number of smartphones sold to end users worldwide from 2007 to 2018 (in million units). (2019). Obtenido el 10/03/2019 de https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/.

Pita, S., y Pértegas, S. (2003). Pruebas diagnósticas: Sensibilidad y especificidad. *Cadernos de Atención Primaria*, 10, 120-124.

Regression Trees (sin fecha). Obtenido el 26/04/2019 de http://uc-r.github.io/regression_trees.

Schapire, R. E. (2013). Explaining AdaBoost. *Empirical Inference*, 37-52.

Shah, T. (2017, 6 de diciembre). *About Train, Validation and Test Sets in* Machine Learning. Obtenido el 14/03/2019 de https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

Shubham, J. (2018, 3 de julio). *Ensemble Learning—Bagging and Boosting*. Obtenido el 28/04/2019 de https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e.

Singh, S. (2018, 21 de mayo). Understanding the Bias-Variance Tradeoff. Obtenido el 04/04/2019 de https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229.

Tartar, A. y Qiu, Y. (2018, 26 de julio). *The new rockets racing to make space affordable*. Obtenido el 11/03/2019 de https://www.bloomberg.com/graphics/2018-rocket-cost/.

The digital universe of opportunities: Rich data and the increasing value of the internet of things. (2014, abril). Obtenido el 10/05/2019 de https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm.

Thompson, W., Li, H., Bolen, A. (2019). *Artificial intelligence,* Machine Learning, *deep learning and beyond: Understanding AI technologies and how they lead to smart applications*. Obtenido el 05/03/2019 de https://www.sas.com/en_us/insights/articles/big-data/artificial-intelligence-machine-learning-deep-learning-and-beyond.html#/.

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, 49: 433-460. Obtenido de https://www.csee.umbc.edu/courses/471/papers/turing.pdf.

Vryniotis, V. (2013, 27 de octubre). *Tuning the learning rate in Gradient Descent*. Obtenido el 12/03/2019 de http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/.

Winans, M. (2016). *10 Key Marketing Trends for 2017 and Ideas for Exceeding Customer Expectations*. New York: IBM. Obtenido de http://comsense.consulting/wp-content/uploads/2017/03/10_Key_Marketing_Trends_for_2017_and_Ideas_for_Exceeding_Customer_Expectations.pdf.

Zivot, E., y Wang, J. (2006). *Modeling Financial Time Series with S_PLUS®*. Nueva York: Springer.

10. ANEXOS

Anexo 1. Descripción de las compañías que integran el IBEX 35.

Company	Ticker	Sector
Acciona	ANA	Basic Materials, Industry and Construction
Acerinox	ACX	Basic Materials, Industry and Construction
ACS Group	ACS	Basic Materials, Industry and Construction
Aena	AENA	Consumer Services
Amadeus IT Group	AMS	Technology and Telecommunications
ArcelorMittal	MTS	Basic Materials, Industry and Construction
Banco Sabadell	SAB	Financial Services
Banco Santander	SAN	Financial Services
Bankia	BKIA	Financial Services
Bankinter	BKT	Financial Services
BBVA	BBVA	Financial Services
CaixaBank	CABK	Financial Services
Cellnex Telecom	CLNX	Technology and Telecommunications
CIE Automotive	CIE	Basic Materials, Industry and Construction
Enagás	ENG	Petrol and Power
ENCE	ENC	Consumer Goods
Endesa	ELE	Petrol and Power
Ferrovial	FER	Basic Materials, Industry and Construction
Grifols	GRF	Consumer Goods
IAG	IAG	Consumer Services
Iberdrola	IBE	Petrol and Power
Inditex	ITX	Consumer Goods
Indra	IDR	Technology and Telecommunications
Inmobiliaria Colonial	COL	Real Estate Services
Mapfre	MAP	Financial Services
Mediaset España	TL5	Consumer Services
Meliá Hotels	MEL	Consumer Services
Merlin Properties	MRL	Real Estate Services
Naturgy	NTGY	Petrol and Power
Red Eléctrica de España	REE	Petrol and Power
Repsol	REP	Petrol and Power
Siemens Gamesa	SGRE	Basic Materials, Industry and Construction
Técnicas Reunidas	TRE	Basic Materials, Industry and Construction
Telefónica	TEF	Technology and Telecommunications
Viscofan	VIS	Consumer Goods