UNIVERSIDAD PONTIFICIA DE SALAMANCA
TELEFÓNICA I+D
FACULTAD DE INFORMÁTICA
Experto en Big Data

Trabajo Fin de Experto

Haciendo Data Science a League of Legends

Jorge de Andrés González

Directores:
D. Francisco Javier Escudero Martín
Dr. D. Manuel Martín – Merino Acera

Salamanca, Junio de 2019

Agradecimientos

Durante la elaboración de este trabajo no he estado sólo mientras que lo desarrollaba, por lo que considero que hay varias personas que merecen un agradecimiento especial.

En primer lugar, me gustaría dedicar este trabajo a mis padres. Siempre han estado ahí cuando lo necesitaba, y han sido un pilar fundamental para mi formación, tanto en el Grado en Ingeniería Informática como en el Experto en Big Data.

Por otra parte, me gustaría agradecer a mi tutor de la facultad, D. Manuel Martín-Merino por la enseñanza que me ha ofrecido sobre lo que espero que sea mi futuro laboral: La Inteligencia Artificial y la Ciencia de Datos.

También quiero hacer mención especial a D. Francisco Javier Escudero Martín, quién me ha supervisado el trabajo exhaustivamente con gran paciencia y ha sido un gran apoyo para poder realizar este proyecto.

Finalmente me gustaría hacer mención al resto de profesores que he tenido en el Experto. La mayoría de ellos me han enseñado diferentes conceptos y tecnologías que han aportado granitos de arena para la construcción de este proyecto, por lo que también les doy las gracias.

Jorge

Resumen

El League of Legends es un videojuego jugado por millones de personas en todo el mundo, y es considerado el videojuego más jugado mensualmente que existe desde hace años.

Todos los años, Riot Games ofrece a cada jugador, tras un proceso de exploración de datos, las estadísticas básicas de su último año jugando. Motivado por ello, este trabajo versa sobre el análisis del videojuego pero desde otro enfoque: Las partidas en sí.

De este modo, en este trabajo se puede observar un análisis descriptivo completo en el que se detallan minuciosamente los datos referentes a las partidas, y se resuelven mitos del juego, así como claves para obtener la victoria en la partida.

Para conseguir este objetivo, se ha desarrollado un proceso completo de Data Science tanto en R como en Python, dividido en el ya mencionado análisis descriptivo y exploratorio, y en un análisis completo de técnicas de machine learning avanzadas para poder predecir el ganador de la partida.

Junto con ello, se ha desarrollado un backend con tecnología NodeJS y una base de datos MongoDB, desde los que se pueden obtener los datos si así se desea.

Según los resultados obtenidos, se puede predecir con bastante fiabilidad quién es el ganador de la partida, obteniendo errores inferiores al 4%, por lo que se puede considerar que este proceso puede ser usado para pronosticar el ganador.

Abstract

League of Legends is a videogame played by millions of people worldwide, and it is considered the monthly most played videogame.

Every year, Riot Games shows to each player, after a data exploration process, their basic statistics of their last year playing. Motivated by that, this sheets are about the analysis of the videogame but using another perspective: The matches.

So, in this work it is possible to observe a complete descriptive analysis where data referring to the matches are detailed meticulously, and some myths around the game are solved, as well as keys to winning the matches.

To get to the goal, a complete Data Science process has been developed, both in R and in Python, divided by the already mentioned descriptive and exploratory analysis, and in a complete review of advanced machine learning techniques to be able to predict the winner of the match.

Alongside, a backend with NodeJS technology has been developed, with a MongoDB database. From this system data can be received, if it is wanted.

According to the results obtained, it is possible to predict with high accuracy which team wins the match, obtaining errors that are down to 4%, so it is possible to consider that this process can be used to forecast the winner.

Descriptores

Small Data, Data science, Inteligencia Artificial, League of Legends, MOBA

Índice

Índice de Figuras

Introducción

Presentación de este Trabajo

El presente trabajo versa sobre el análisis de partidas clasificatorias de uno de los videojuegos más importantes de la historia, el League of Legends. Para conseguir este objetivo, se hará un análisis exhaustivo de las variables que puedan tener que ver con conseguir la victoria de la partida, y posteriormente otro análisis de las diferentes técnicas y pasos del proceso de data science que hay que llevar a cabo para conseguir unos resultados óptimos en clasificación.

Se considera interesante este problema debido a que consiste en la resolución de un problema que se da en la actualidad, puesto que todos los jugadores de League of Legends acaban preguntándose si son ciertos los mitos que tienen que ver con el devenir de las partidas, aspecto que se trabajará en profundidad en este proyecto. Además, los datos de las partidas son totalmente reales, por lo que se podrán comprobar de forma fehaciente estos interrogantes. Es interesante destacar que Riot Games (la empresa creadora del videojuego) suele sacar un resumen de datos al final del año, por lo que este trabajo es como un "trabajo de Data Science" en Riot Games.

Se puede afirmar que es un problema complicado debido a la altísima dimensión a la que se someten los datos si se añaden los campeones y las spells, además de que, aunque sea una clasificación a dos, el objetivo del trabajo es obtener resultados realmente satisfactorios y cercanos al 100% de acierto, lo cual conlleva una gran afinación de los algoritmos, además de un business understanding muy profundo, el cual no es sencillo como se puede comprobar en el siguiente capítulo.

También, es un problema motivante debido a la actualidad del tema, puesto que el League of Legends lleva siendo el videojuego más importante de los e-sports prácticamente desde su aparición, llevando 9 años en la élite de los videojuegos y llegando a repartir premios millonarios en el campeonato del mundo que se celebra cada año. Millones de jugadores de todas las regiones del mundo lo juegan.

En resumidas cuentas, debido a todo esto, creo que el problema que se plantea es un reto importante debido a las dificultades técnicas del mismo para entender y afinar extremadamente los modelos, pero a la vez muy interesante dado el tema, su actualidad y la conexión que se va a establecer entre el mismo y el proceso de Data Science que se va a llevar a cabo.

Estado del Arte y Objetivos

El objetivo del trabajo es la construcción de un sistema automático de predicción de victorias de partidas en el videojuego League of Legends en base a una serie de datos de partidas reales, obtenidas a través de la famosa web "Kaggle". Este problema, como se ha comentado anteriormente, es tratado por Riot Games todos los años de forma privada, pero aparte de alguna que otra pequeña aproximación no hay demasiado desarrollado sobre este tema, y menos en la profundidad en la que se va a tratar en este trabajo.

Junto con este primer objetivo de la construcción del sistema, también se desarrollará un análisis íntegro de los datos para su correcta interpretación, además de la obtención de características atractivas de los mismos que puedan estar ocultas. La metodología que se seguirá será, en primera instancia, la obtención de los datos de Kaggle, con la búsqueda también de otros datasets que puedan complementar los datos obtenidos. Posteriormente, se hará un estudio exhaustivo de las distintas técnicas que se pueden utilizar para el análisis, clasificación y predicción de los datos, dejando estos conocimientos plasmados en la memoria y posteriormente implementados en código.

Finalmente, se hará un estudio de las diversas técnicas de visualización de los datos, y se procederá a la visualización de características interesantes de los mismos.

Como forma común a todo el trabajo, para el control de todas las labores a realizar, se usará un sistema de control de tareas y errores implementado por GitHub llamado Projects, ofrecido gratuitamente y que está basado en una Kanban Board. Además, todo el código y la memoria estarán siendo versionados mediante un VCS (GitHub en este caso) para el control del trabajo y de los cambios, así como la sincronización entre los ordenadores en los que se pueda trabajar y como medida de copia de seguridad, puesto que durante el avance de este trabajo, este permanecerá como repositorio privado.

Ante el análisis de los resultados esperados, es trascendente tener en cuenta de que este no es un "problema de laboratorio" con unos datos controlados, sino que es un problema real con datos reales. De este modo, espero que los resultados estén cercanos al pleno de acierto debido a que únicamente hay dos grupos clasificables, aunque bien es cierto que muy probablemente haya que afinar mucho los algoritmos para conseguirlo, puesto que hay partidas atípicas y estas pueden dar errores en los algoritmos, a la vez que haya que controlar problemas como el overfitting.

Si se atiende a la organización de la memoria, esta dará comienzo con un primer capítulo introductorio de League of Legends y Riot Games. En este, se analizará todo lo que concierne al juego, para que el lector tenga un buen entendimiento del negocio y pueda comprender los resultados que se obtienen, ya que las partidas tienen una dinámica, unos objetivos… que si no se comprenden no se llega a comprender en profundidad las preguntas planteadas y los resultados obtenidos.

Posteriormente, habrá un amplio capítulo sobre data science, en el cual se abordarán de una manera muy pormenorizada todos los pasos que se deben de tener en cuenta para un análisis de datos completo, riguroso y fiable, haciendo especial hincapié en la zona de algoritmos de machine learning, que fundamentará el núcleo del trabajo. También se podrá observar un apartado de visualización.

En el último capítulo, se hará un análisis de todos los resultados obtenidos mediante las diferentes técnicas, de tal manera que se pueda comparar la eficacia de los algoritmos, e incluso de un mismo algoritmo con diferentes parámetros.

1. Introducción a League of Legends

League of Legends es un videojuego MOBA (Multiplayer Online Battle Arena) consistente en la lucha de dos equipos formados por 5 jugadores. El objetivo de los equipos es el derribo de objetivos para conseguir derribar el nexo enemigo, último objetivo de la partida. Para conseguir esto, necesitan conseguir otros objetivos obligatorios previamente. También existen objetivos opcionales, que pueden mejorar la fuerza de los campeones con los que se juega.

Actualmente, y desde hace tiempo, es el videojuego más jugado del mundo, con más de cien millones de jugadores activos mensualmente. Además, profesionalmente es el videojuego más seguido del mundo, llegando a tal punto que el campeonato del mundo se retransmite por internet con cientos de miles de visitas en cada partida, y las finales se juegan en estadios de fútbol llenos de gente. Los premios para los campeones son millonarios.

1.1 ¿Cómo se juega? ¿Qué se debe hacer?

El objetivo del juego es, como se comentó anteriormente, el derribo del nexo contrario, que se encuentra en el extremo contrario del mapa respecto a donde empieza el equipo. Pero antes de llegar a esta fase, muchas otras etapas deben de ser pasadas previamente.

Primero hay una preparación de la partida. En esta preparación, cada jugador del equipo “banea” o prohibe un campeón que desea que no participe en la partida, de tal manera que puede haber hasta 10 campeones que no sean elegibles por ningún jugador de la partida. Una vez que este proceso está completo, los 10 jugadores que conforman la partida eligen al campeón que quieren jugar. Junto con ello, eligen sus maestrías y sus “spells” o hechizos.

Una vez completados estos pasos, da comienzo la partida. Para ganarla, hay que conseguir al menos derribar las tres torres de una línea del mapa, el inhibidor de dicha línea, las dos torres que protegen al nexo y el nexo. Esta es la fórmula mínima y prácticamente nunca se realiza, puesto que las partidas son bastante largas y algunas torres o líneas más suelen ser derribadas.

Todo esto puede ser visualizado de una forma clara en la siguiente figura:

1-1. Mapa "Summoner's Rift"

Como se puede observar en la figura, el mapa está dividido en dos zonas claramente diferenciadas por el llamado “río”. En la zona superior comienza el equipo rojo, también conocido como equipo 2, mientras que en la zona inferior izquierda comienza el equipo azul, también conocido como equipo 1.

Se puede apreciar que el mapa está dividido en 5 zonas claramente diferenciables: TOP, MID, BOT, Jungla de Arriba y Jungla de Abajo. Los jugadores se distribuyen en estas zonas con la siguiente distribución:

- TOP: 1 jugador por equipo
- MID: 1 jugador por equipo
- BOT: 2 jugadores por equipo
- Jungla: 1 Jugador por equipo

En cada línea se generan minions, también conocidos como súbditos, que ayudan a la tarea de derribar las torres y eliminar a los enemigos. Estos minions son en principio unos aliados débiles que poseen ambos equipos, pero pueden llegar a ser muy importantes en etapas tardías de la partida.

Una vez conocidos estos elementos, es hora de explicar con mayor detalle estos objetivos, por qué es necesario conseguirlos y qué beneficios aportan.

1.2 Objetivos y victoria

1.2.1 Torres

Las torres, marcadas con flechas naranjas en la imagen, son el objetivo más básico y obligatorio del juego para conseguir la victoria. Como se explicó anteriormente, se deben de derribar todas las torres de una línea para acceder al inhibidor, marcado con flecha amarilla. Una vez derribado este también, se deben de derribar las dos torres de nexo, que se encuentran justo delante del nexo, marcado con flecha roja.

1-2. Torre

Además de servir como principal impedimento hacia la victoria, conseguir la primera torre es un objetivo importante, puesto que otorga extra de 300 de oro a cada persona del equipo que derribe esta primera torre, lo que aporta una ventaja en la partida.

Además, por cada torre derribada el equipo gana también oro, por lo que las ganancias aumentan conforme más torres se derriben y más cerca se esté de la victoria.

Esto genera un fenómeno llamado “efecto snowball” o “efecto bola de nieve”, consistente en que el equipo que consigue una ventaja suele ampliarla, debido a que posee más fuerza, por lo que la partida suele estar más desequilibrada conforme pasa el tiempo.

1.2.2 Dragones

Los dragones, al contrario que las torres, son objetivos opcionales en la partida. Es decir, se puede ganar una partida sin derribar ningún dragón, puesto que “sólo” aporta mejoras en las habilidades de los campeones del equipo que lo haya derribado.

Existen 5 tipos de dragones en el juego, que en resumidas cuentas son los siguientes:

- Dragón de fuego: Aporta mejora en las habilidades
- Dragón de agua: Aporta mejor regeneración de vida
- Dragón de aire: Aporta mejora en la movilidad
- Dragón de montaña: Aporta mejora en derribo de objetivos
- Dragón anciano: Mejora las habilidades conseguidas anteriormente con otros dragones

1-3. Dragón Anciano

Por lo tanto, la obtención de dragones, aunque no obligatoria, es muy recomendable, ya que el hecho de haber conseguido derrotar a un dragón aporta mejoras sustanciales a la hora de entrar en combate contra el otro equipo o de derribar objetivos.

1.2.3 Barones y heraldo

El heraldo y los barones, al igual que los dragones, son objetivos opcionales para el desarrollo de la partida, puesto que se puede ganar la misma sin conseguirlos, aunque estos tienen, en principio, una importancia capital en el devenir de la partida puesto que aportan mejoras mucho más sustanciales que los dragones.

1-4. Heraldo

1-5. Barón

El heraldo es un monstruo que aparece sólo durante los primeros 20 minutos de la partida. Si se consigue derrotar, se obtiene su poder, que es excelente para derribar torres si se utiliza adecuadamente.

El barón aparece a partir del minuto 20, y podría decirse que es un monstruo con una gran vida y una gran capacidad de ataque. En caso de obtenerse, el equipo consigue mejora en los minions (o súbditos) que les acompañan, mejorando tanto su vida como su daño.

1.2.4 Jungla

La jungla es una posición del videojuego, consistente en dos zonas entre las líneas donde se encuentran los monstruos de jungla. Una persona, denominada “jungler” está a cargo de “hacerse la jungla” consistente en matar los monstruos de jungla, intentar entrar en la jungla enemiga para matar sus monstruos y, también, “gankear” las líneas. Un “gankeo” consiste en un movimiento por el cual una persona que no se encuentra en una línea ayuda a alguien que si se encuentra en la misma, entrando de sorpresa y generando una superioridad numérica para conseguir derrotar al enemigo.

1.2.5 Inhibidores

Los inhibidores son el objetivo final de cada línea para los dos equipos. Consisten en una estructura circular y pequeña, la cual se puede atacar una vez se hayan derrotado las tres torres de una determinada línea. Una vez derribado el inhibidor, se obtienen dos ventajas:

1) Acceso a las torres que protegen al nexo, y por lo tanto a finalizar la partida
2) Generación de superminions, que poseen características mejoradas de los minions como mayor vida y mayor daño.

En caso de derrotar todas las líneas, y tener derrotados todos los inhibidores, se generará un

1-6. Inhibidor

mayor número de superminions, lo cual suele conllevar el final de la partida porque el equipo que va perdiendo perderá mucho tiempo y recursos en derrotar a estos superminions, mientras que el equipo contrario puede finalizar la partida con cierta calma.

1.2.6 Nexos

El nexo, como se ha comentado ya varias veces, es el objetivo final de la partida. El equipo que derribe el nexo enemigo conseguirá la victoria.

Llegar hasta el nexo no es tarea fácil, porque aparte de todas las peripecias que hay que lograr para llegar hasta él, está al lado de la zona de respawn enemiga, lo que significa que los enemigos, con la salud totalmente regenerada, atacarán al equipo atacante para evitar la derrota.

1-7. Nexo

Como se puede observar, el aspecto del nexo es muy parecido al de un inhibidor, con la única diferencia de que es más grande e importante.

2. El proceso de Data Science

2.1 Data Science, Big Data, Machine Learning, Data Mining

Cuando se menciona Big Data o Data Science, hay varios conceptos que se pueden venir a la mente, pero dos de ellos sin duda son el “data mining” y el “machine learning” (aprendizaje automático).

Las diferencias entre estos términos son ignoradas por la gran mayoría de personas, pero a continuación se hace hincapié en ellas para tener los términos bien separados, y a partir de ahora saber a qué se hace referencia:

2.1.1 Data Mining

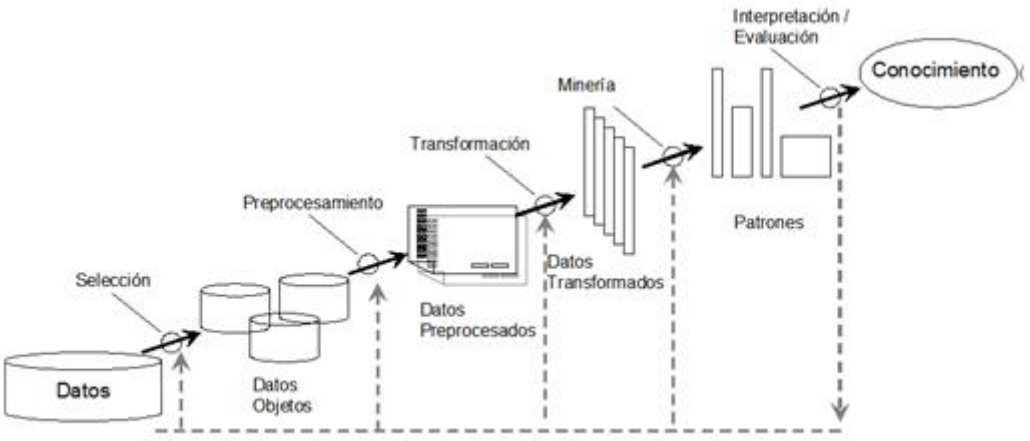
Los principios del data mining proceden de la dificultad de poder manipular diferentes tipos de datos con las herramientas existentes. Este trabajo desencadenó ideas que se tomaron prestadas de otros campos, como la estimación o el muestreo tomados de la estadística, o los algoritmos y las técnicas de machine learning provenientes de la inteligencia artificial. También otras áreas poseen un papel esencial en todo lo que rodea al data mining, como es el área de visualización, de bases de datos o de cómputo.

Así, data mining se puede definir como el proceso de revelar información que pueda ser ventajosa, a través del estudio de grandes repositorios de datos. De esta forma, con la minería de datos se intentan encontrar esquemas, patrones y soluciones de cuestiones que, de otro modo, estarían escondidas entre los datos.

Es vital distinguir entre data mining y la recogida de información. Mientras que data mining utiliza técnicas estadísticas y matemáticas para la captación de información dentro de un dataset, la recogida de información consistiría en, por ejemplo, una búsqueda en una base de datos para un elemento concreto. A pesar de centrarse los dos en los datos, son técnicas distintas y, debido a ello, deberán de mantenerse separadas.

El descubrimiento de conocimiento es el último objetivo de la minería de datos. Conocido en la comunidad de habla inglesa como KDD (Knowledge Discovery in Databases*)*, se podría decir que consiste en el proceso total de convertir los datos puros de la base de datos en una información útil. Es decir, el descubrimiento de conocimiento es el concepto por el que,

mediante el uso de data mining, se obtiene conocimientos útiles de una enorme cantidad de datos que, a priori, no revelaba información a simple vista.



2-1 Diagrama de KDD

Este descubrimiento de información consta de una serie de pasos, que discurren desde un pre-procesamiento de los datos para su preparación, hasta un post-procesamiento para su posterior obtención de información. Observemos este trayecto con más pausa:

1) Pre-procesamiento de los datos:

Una vez importado el dataset, o el conjunto de datasets con los que se va a trabajar, se debe de hacer este pre-procesamiento de los datos para prepararlos de cara al data mining. De esta manera, acciones como la unión de tablas, la reducción de la cantidad de variables (también conocido como reducción de la dimensionalidad), o la obtención de subgrupos de datos, serán pasos muy importantes de cara a preparar los datos para los próximos pasos.

Normalmente, este pre-procesamiento suele ser la parte que más tiempo consume en el proceso de la minería de datos, debido a que es muy manual y laboriosa.

2) Data Mining:

En este paso, usaremos las numerosas técnicas estadísticas y matemáticas que conforman el data mining, como pueden ser la unión por grupos, el estudio de la variabilidad, el estudio de las relaciones entre las observaciones o el estudio de la frecuencia entre muchas otras. Este conjunto de tareas recibe el nombre de “tareas descriptivas”, ya que el objetivo de las mismas es obtener patrones que resuman las relaciones que haya por debajo en los datos.

Además, junto con las tareas descriptivas podremos hacer un análisis predictivo para poder predecir ciertas características de futuras observaciones. Esto se analizará más adelante.

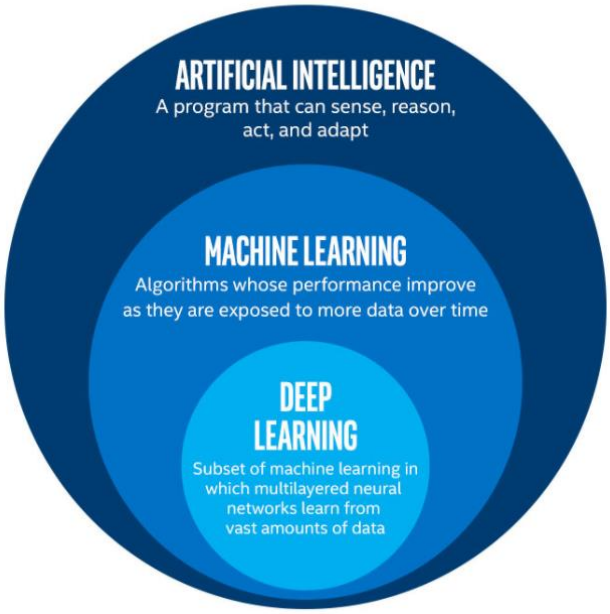
3) Post-Procesamiento de los datos:

Con el post-procesamiento de los datos, se da referencia esencialmente a la quizás necesaria trasformación final de los datos de cara a la siempre necesaria visualización. Fuera de data mining, junto con esta visualización, siempre se debe hacer un análisis e interpretación de los datos obtenidos, de cara a la aclaración de los mismos y la finalización del proceso, obteniendo información útil.

2.1.2 Machine Learning

Si se hace referencia a machine learning, se habla de un subconjunto de la inteligencia artificial que, aparte de usar los principios de data mining, es capaz de buscar correlaciones de una manera mecánica, y también es capaz de aprender de los datos que se poseen y se poseerán, de tal manera que el modelo pueda seguir mejorando.

El uso que se hace del machine learning por cualquier persona es diario, ya es usado en numerosos ámbitos, tales como la publicidad, predicción del tiempo atmosférico, optimización de la duración de baterías, sistemas de recomendaciones (Netflix, por ejemplo)…



2-2. Inteligencia Artificial & Subespacios

Las tareas más relacionadas con machine learning se denominan tareas predictivas, ya que están basadas en predecir un caso particular a través de los valores de una serie de atributos. Desde ahora, una de las formas con las que se podrá referirse a estas tareas será como “modelos predictivos”.

2.1.3 Data Science

Así, si se poseen unos algoritmos que permiten obtener información, y otros algoritmos que mejoran con el paso de los datos y obtienen una predicción de la información, ¿qué espacio queda para data science?

Con data science se hace referencia al término que cobija a data mining y a machine learning. Data science no es más que un término genérico que aúna un conjunto de técnicas o subdisciplinas, como data mining, machine learning y visualización de datos, entre otras, para obtener un conjunto de informaciones (insights) que sean útiles al usuario final, como puede ser una empresa (Business Intelligence) o cualquier otro usuario interesado.

De una forma más espectral, se podría afirmar que data science consiste en la mezcla de una serie de procedimientos matemáticos que, junto con conocimientos del problema que se trata y de tecnología especializada, consiguen obtener conclusiones fácilmente entendibles para el usuario final.

2.1.4 Big Data

Con Big Data se referencia un término muy de moda en los últimos tiempos. “Big Data” define simplemente a la disciplina que trabaja con unas enormes cantidades de datos. Es una disciplina que, al igual que “medium data” y “small data”, están muy presentes en todo proyecto de data science.

1) Small Data o “Pequeños Datos”

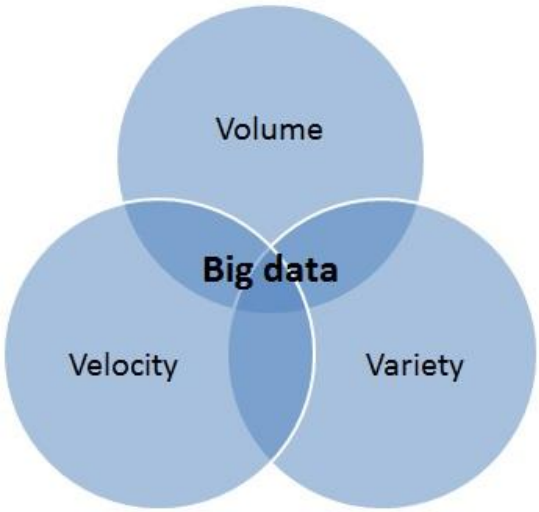
   Con small data, se habla de proyectos en los cuales los datos están en un formato CSV/TSV, una base de datos pequeña o incluso en un Microsoft Excel. En este tipo de proyectos, se suele trabajar con un ordenador estándar, y los datos pueden ser cargados perfectamente en memoria RAM.
   En el caso de este trabajo, se usará small data debido a la pequeña cantidad de observaciones y variables que se poseen a analizar.

2) Medium Data o “Datos Medianos”

   Los proyectos que usan medium data son aquellos que utilizan una cantidad de datos mayor que los de small data, y, a pesar de que los datos se pueden albergar normalmente en un ordenador, las técnicas de extracción de los mismos son distintas, ya que no se puede solicitar toda la información de golpe a riesgo de bloquear el ordenador provocando un overflow de la RAM.

3) Big Data o “Grandes Datos”

   Finalmente, los proyectos de big data son aquellos en los que se requieren varios ordenadores en cluster, o un servidor de grandes dimensiones, para poder almacenar y procesar la información. Estos proyectos son enormemente más complejos, ya que suelen necesitar técnicas de sincronización entre ordenadores, procesamiento en grid y técnicas similares.



2-3. V’s del Big Data

De este modo, se puede afirmar que “Small Data”, “Medium Data” o “Big Data” son un contexto, un “*framework*” en los que se mueven los proyectos de data science, y que

dependiendo del que sea necesario, se usarán las tecnologías del que sea conveniente para el tratamiento de los datos.

También, otro método más informal pero muy efectivo para concer bajo que paradigma se va a trabajar es el de tener en cuenta las 3 V's del Big Data: Velocity, Volume, Variety. Antes de entrar en profundidad con ellas, es importante aclarar que hay más V's, donde algunos expertos afirman que existen hasta 12. En este trabajo se repasarán las 3 centrales, las que más aceptadas están por todos los expertos en la materia.

- Velocity

Velocity se refiere a la velocidad de la creación de nuevos datos. Si se crea una gran cantidad de datos en breve periodo de tiempo, se puede afirmar decir que podríamos estar en un problema de Big Data y por lo tanto una base de datos no convencional (como una NoSQL) es una opción a valorar.

- Volume

Quizás el parámetro más fácil, la cantidad de datos con los que se va a trabajar influye en gran medida en la manera de encarar el problema. No hay reglas fijas, pero un problema de Big Data se supone que no se puede solucionar en un ordenador de casa, ni se pueden almacenar los datos en el mismo. De tal manera, los problemas que se encuadrarán dentro de Big Data serán problemas de cientos de terabytes, petabytes o incluso mayores.

- Variety

La variedad de los datos, debido a la gran cantidad de estructuras de datos y fuentes, es otro de los indicadores principales de que existe un problema que debe ser resuelto mediante técnicas de Big Data. No es el requerimiento más importante a la hora de determinar bajo que paradigma se resolverá el problema (los dos anteriores se antojan vitales), pero sí es necesario tener en cuenta que en Big Data los datos de diferentes fuentes y diferentes formatos son muy comunes.

2.2 Previamente a aplicar Data Mining

Un paso esencial en el momento de afrontar un problema con datos es la obtención de dichos datos. Estos pueden ser obtenidos de numerosas fuentes, y más cuando el problema se afronta bajo big data o medium data, que, como se ha explicado previamente, los datos pueden estar distribuidos entre numerosos ordenadores. Además, las técnicas de petición de los datos son más complejas que una simple carga en memoria RAM.

Así, el conocimiento de tecnologías como Apache Spark, para el control del flujo de datos a memoria y computación Map-Reduce, se antoja esencial en aquellos proyectos que no pertenezcan al grupo de small data.

2.2.1 En este Proyecto: Obtención de los datos

En mi caso, se podría considerar el problema de "small data", y el dataset se ha obtenido de la famosa plataforma Kaggle, al igual que los datos de mejora del mismo, que se han obtenido de otro repositorio de la misma plataforma.

También, se considera importante comentar que la variable más importante se encuentra al final de cada fila del dataset, y corresponde al equipo que ganó la partida. Estos grupos son una variable discreta que versa del 1 al 2, y se corresponde de la forma siguiente:

1. El equipo 1 ganó la partida
2. El equipo 2 ganó la partida

2.3 Data Mining

Como se ha visto anteriormente, con data mining se nos referimos al proceso de conseguir una información rentable y entendible a partir de un agregado de datos. Así, este apartado está dividido en la explicación de los diferentes procesos que se han llevado a cabo para conseguir aplicar data mining sobre este problema, y la explicación de cómo han sido realizados.

2.3.1 Primeros pasos a realizar y la preparación de los datos

Según IBM, empresa líder mundial en ventas de máquinas para negocios, cuando se preparen los datos antes de un proceso de minería de los mismos es importante seguir tres pasos:

1) Entendimiento del negocio (Business Understanding)
2) Entendimiento de los datos (Data Understanding)
3) Preparación de los datos en sí
    a. Discretización y binarización
    b. Reducción de la dimensionalidad del dataset

2.3.2 En este Proyecto: Preparación de los datos

Siguiendo las bases teóricas expuestas en el punto anterior, en este trabajo los datos han sido preparados de una manera premeditada para evitar cualquier problema. De este modo, primeramente, se ha conseguido un entendimiento del negocio y de los datos a través del juego de partidas y búsqueda de información que pudiera resultar confusa.

Para confeccionar el dataset final, se han tenido en cuenta las necesidades posteriores de los algoritmos a usar. Por ello, una serie de cambios han sido realizados:

1) Eliminación de columnas innecesarias
    a. SeasonId: Todas las partidas pertenecen a la misma temporada.
    b. Summoner Level: No es necesario saber el nivel que se necesita para cada spell, puesto que lo importante es si el jugador la lleva o no.
    c. Description: Las descripciones no son necesarias.
2) Comprobación de Na's
3) Cambio de Timestamp de Unix a formato legible de fecha
4) Unión de datasets a través de left joins

2.3.3 Análisis Exploratorio o Descriptivo

El análisis exploratorio de datos está formado por un grupo de técnicas estadísticas y de visualización de datos para resumir en primera instancia la realidad que se tiene en los datos, así como intentar encontrar patrones y relaciones entre los mismos, de tal manera que se pueda revelar la respuesta de alguna pregunta que anteriormente no se podría con una mirada escueta hacia los datos. Además, la visualización de datos históricos (especialmente en las series

temporales) puede aportar mucha información de la posición en la que se encuentra el problema en el momento. Esta técnica fue inventada por el estadista John Turkey en la década de 1970.

Además de encontrar patrones y relaciones, el análisis exploratorio de los datos es un proceso que se antoja imprescindible a aplicar antes del machine learning, debido a que en la mayor parte de los datasets hay datos outliers, que están faltos o que son inconsistentes. Por ello, realizar un análisis exploratorio profundo, demostrar si las relaciones que existen cuadran con la realidad (mediante Data Understanding), y eliminar variables outliers o que no aportan nueva información debido a su varianza muy próxima al cero es realmente requerido para obtener unos modelos posteriores de machine learning que sean rápidos, eficaces y exactos.

Como causa de lo anterior, primero se deben hacer unas pequeñas comprobaciones para comprobar los datos de los que se dispone, que se dividen en la comprobación de los tipos de datos y la comprobación de la calidad de los mismos.

Una vez que ambos han sido comprobados, se puede empezar con la parte estadística del análisis exploratorio. Hay varios pasos que se pueden cubrir, que se irán viendo en los siguientes apartados.

2.3.3.1 Resumen de las estadísticas del Dataset

El resumen de las estadísticas del dataset no consiste en más que un conjunto de números indicando varias características estadísticas de un dataset. Dichas estadísticas están conformadas por un conjunto de apartados que se detallan a continuación:

a. Frecuencia y moda

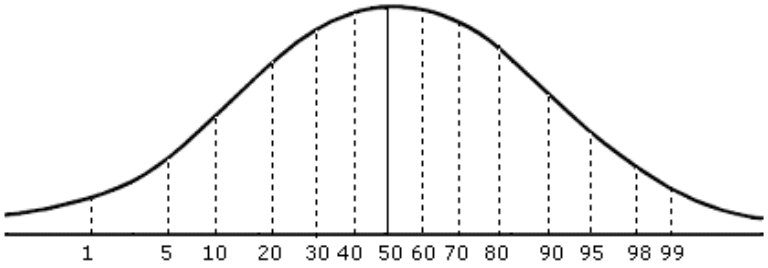
La frecuencia de una variable consiste en un número continuo, con un rango entre 0 y 1, que indica el tanto por uno de ocurrencias de dicho valor de la variable en una lista de m objetos. Por ello, sigue la siguiente fórmula:

$$f(x) = \frac{nº\ objetos\ con\ valor\ x}{nº\ objetos\ totales}$$

Respecto a la moda, simplemente se hace referencia al valor de cierta variable que tiene una frecuencia mayor que los otros.

b. Percentiles

Para una serie de datos ordenados, el percentil de un conjunto de valores es capaz de aportar una gran cantidad de información. Dado un número p que oscila entre 0 y 100, el p-esimo percentil es un valor de x donde el p% de los datos totales son menores a ese valor p-esimo. Así, se puede obtener qué valores destacan por encima del resto en un determinado porcentaje, como se puede observar en la figura 2-4.



2-4. Distribución de los percentiles sobre una curva normal

c. Media y mediana

En caso de tener datos continuos, dos de las estadísticas más básicas y a la vez más solicitadas son la media y la mediana.

Se puede definir la mediana como el valor que está en el medio de un conjunto de datos ordenados. Si hubiera un número par de datos, sería la media de los valores del centro.

La media se puede definir mediante la siguiente función:

$$\bar{X} = media(x) = \frac{\Sigma\, Xi}{n^{o}\, valores\ total}$$

La media suele conllevar problemas de medición, debido a que puede estar realizada incluyendo valores outliers, o incluso que sin ser outliers la distorsionen en cierto modo. Por ello fue inventado el concepto de "media recortada", que radica en coger un porcentaje, que suele variar entre el 1% y el 5%, y desechar ese porcentaje de datos tanto de la parte superior como de la inferior del dataset ordenado, aplicando la media al nuevo dataset ya recortado.

d. Rango y varianza

El rango y la varianza son las llamadas "medidas de dispersión", ya que miden el dominio en el que se encuentran los datos y la diferencia entre los valores.

La más sencilla es el rango, que se puede medir tanto con la resta del valor más alto entre el valor más bajo, como con un intervalo cerrado donde el primer valor del mismo sea el más pequeño, y el segundo el más grande.

La varianza, aunque más complicada, es el valor preferido a la hora de calcular la dispersión de los datos, y se suele representar como s^2. La fórmula a la que se acoge, siendo m el total de datos, es la siguiente:

$$s^2 = \frac{\sum(xi - \bar{x})^2}{m - 1}$$

La desviación típica, representada con s, responde a:

$$s = \sqrt{\frac{\sum(xi - \bar{x})^2}{m - 1}}$$

e. Resumen de estadísticas multivariable

Las estadísticas multivariable son aquellas que poseen más de un atributo. Para calcular las estadísticas en este caso, se debe de calcular la media y la mediana de cada una de las variables.

En caso de la dispersión, también puede ser calculada de manera separada para cada variable, aunque en este caso pueden ser comparadas unas medidas con otras mediante la matriz de covarianza. Ésta es una matriz bidimensional (comparando variables dos a dos), y los valores que se muestran en cada celda son la covarianza de las que forman la fila y la columna.

Dadas las variables X_i y X_j, y la cantidad total de variables m, se puede calcular la covarianza de la forma siguiente:

$$cov(Xi, Xj) = \frac{\sum_{k=1}^{m}(Xki - \overline{Xi}) * (Xkj - \overline{Xj})}{m - 1}$$

También es muy utilizada en data science la llamada matriz de correlación. Esta es otra matriz bidimensional que representa la fuerza de relación lineal entre todas las variables que conforman un dataset, y en cada celda se encuentra mostrada la correlación entre las dos variables afectadas. Esta matriz es de gran interés, ya que es simétrica respecto a la diagonal, y además la diagonal consiste en las correlaciones de cada variable consigo misma, lo cual otorga un resultado siempre de 1 sobre 1.

La correlación se representa mediante la siguiente fórmula:

$$Rij = corr(Xi, Xj) = \frac{cov(Xi, Xj)}{si * sj}$$

2.3.4 Machine Learning

En este apartado se desarrolla el núcleo de este proyecto: El machine learning. Este apartado es el centro ya que las máquinas pueden aprender de los datos que se les brindan, y con ello predecir o clasificar otros datos de los cuales se les puede aportar o no la solución. Por esto, si los algoritmos de machine learning se encuentran correctamente entrenados se abre un gran abanico de posibilidades ante las cuestiones que se planteen.

La clasificación es definible como la tarea de asignar un objeto o un conjunto de ellos a una categoría, que se puede encontrar predefinida anteriormente. Algunos ejemplos de uso de clasificación en machine learning son la clasificación de correos para la detección de spam o la clasificación de tumores a partir de imágenes, entre otras muchas utilidades.

Dentro de machine learning se puede hacer una pequeña distinción con los algoritmos predictivos, puesto que estos algoritmos predicen un valor de observaciones desconocidas y continuas, recibiendo el nombre de “regresiones”.

Las técnicas de clasificación suelen funcionar bien con cualquier conjunto de datos, pero siempre se debe ir con cautela, puesto que los datos ordinales y las jerarquías no son tipos de datos con los que los algoritmos de clasificación funcionen especialmente bien. Debido a ello, es interesante conocer la naturaleza de los datos antes de usar este tipo de algoritmos, tal y como se explicó anteriormente que se debía hacer antes del análisis exploratorio de los datos.

Tras conocerse estos pequeños detalles, a continuación se dará un paso más para irse acercando a los algoritmos de clasificación.

2.3.4.1 ¿Cómo trabaja un algoritmo de clasificación? ¿Cómo se mide su eficacia?

Un algoritmo de clasificación en machine learning sigue unos pasos definidos, y que son en cierto modo sencillos.

El primer detalle a tener en cuenta es que los datos deben de tener una dimensión donde se indique la categoría a la que pertenecen, en caso de que se quiera realizar un aprendizaje supervisado. Tras asegurarse la obtención de esta columna, se debe de partir el dataset en dos conjuntos, siendo el primero de ellos para el entrenamiento del modelo y el segundo para el test. Es importante tener en cuenta que esta división no debe de ser igual, sino que el conjunto de entrenamiento debe de ser muy significativamente mayor al de test, con un ratio de aproximadamente 8 a 2, aunque esto depende de los datos, llegando a obtener resultados muy variados dependiendo de esta agrupación, como se verá posteriormente.

Una vez que se poseen el conjunto de entrenamiento y de test preparados, es posible aplicar este grupo de datos de entrenamiento al algoritmo de clasificación que se esté usando (SVM, K-NN…). La asociación del algoritmo y los datos es lo que a partir de ahora se denominará "modelo", y en este caso será el modelo de entrenamiento.

Este modelo de entrenamiento se deberá poner a prueba con el conjunto de test, previamente preparado. Tras obtener la matriz de confusión y calcular el número de aciertos en la diagonal de la misma respecto al número de fallos, es posible calcular la eficacia del modelo y, si es necesario, hacer posteriores ajustes para obtener mejores resultados en este test.

Así, la matriz de confusión es una compañera ideal en los métodos de clasificación y predicción para comprobar la eficacia del modelo. Se puede calcular el acierto y el error en tanto por uno siguiendo las siguientes fórmulas:

$$Acierto = \frac{Predicciones\ correctas}{Total\ de\ Predicciones}$$

$$Error = 1 - Acierto$$

2.3.4.2 Problemas y soluciones con los clasificadores

Los errores que se suelen consumar en un problema de clasificación pueden ser divididos en dos tipos: Errores de generalización y errores de entrenamiento. El primer tipo es posible definirlo como el error del modelo cuando se le aplican datos no vistos anteriormente, mientras que el segundo podría expresarse como el número de elementos clasificados erróneamente en el entrenamiento. Es importante destacar que un error de entrenamiento superior al esperado no implica imperiosamente un error en test alto, pues el algoritmo es posible que haya generalizado de una forma correcta y obtenga resultados de test positivos, mientras que un error de entrenamiento muy bajo puede ser debido a que se haya caído en overfitting.

Overfitting es definible como un sobreajuste del modelo, de tal forma que no generaliza de una manera óptima, sino que está demasiado ajustado hacia el conjunto de entrenamiento, lo que implica una mayor ratio de fallos a la hora de clasificar otros datos no vistos anteriormente.

De la misma forma, se puede definir underfitting como lo opuesto al overfitting, que consistiría en una generalización demasiado simple debido a que el algoritmo no ha conseguido entender bien la estructura y relaciones de los datos.

De estos dos inconvenientes, el más común es el overfitting, puesto que se tiende siempre a intentar mejorar el modelo y a veces esta mejora lo único que implica es el empeoramiento del mismo. Por ello, a continuación se analizan algunas razones por las que puede haber overfitting en un modelo clasificatorio:

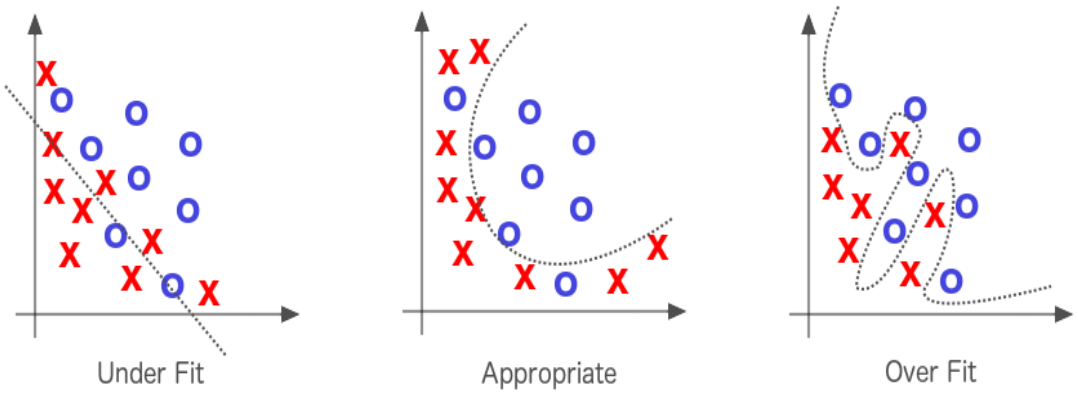
a. Debido a ruido en los datos

La presencia de errores en los datos provoca que los resultados obtenidos se desvirtúen respecto de la realidad. Así, cuando unos datos mal clasificados se introduzcan para entrenar un modelo, ese modelo tanto para esos casos como para casos muy similares obtendrá unos resultados erróneos cuando se prediga con él.

Esta problemática, si se pone en contexto de un árbol de decisión, dará como resultado un aumento del número de bifurcaciones en el árbol, y con ello se obtendrá el overfitting que se intenta evitar. De este modo, y de una forma más acusada en datasets de grandes dimensiones, se suele tomar como éxito un error ciertamente pequeño, puesto que estos errores rara vez son evitables cuando se poseen grandes cantidades de datos.

b. Debido a la falta de muestras realmente representativas en los datos

Cuando se realizan clasificaciones con una cantidad de muestras muy pequeña, la probabilidad de que aparezca overfitting es muy alta. La falta de elementos similares con características parecidas que pertenezcan al mismo grupo provoca que el modelo realice una peor generalización, con su correspondiente aumento del error de test.



2-5. InfraAjuste, Óptimo y SobreAjuste

Como respuesta a esta problemática, se deben de buscar soluciones para evitar caer en underfitting y overfitting en cualquier clasificador. Para ello existen 3 métodos ampliamente usados, que se exponen a continuación:

1. El Método Holdout

El Método Holdout tiene su comienzo dividiendo los datos en dos conjuntos disjuntos, denominados “conjunto de entrenamiento” y “conjunto de test”. Una vez realizado este paso, se crea el modelo con el primer conjunto y se prueba su eficacia con el segundo.

Este método, como el lector ha podido comprobar en estos momentos, no es en sí un método para mejorar los errores de generalización, pero como se comentó previamente hay que tener en cuenta la proporción en la que se dividan los datos, y esto queda a decisión del experto. Si se utilizan demasiados datos para el entrenamiento, se obtienen demasiados pocos registros para el test y el acierto no ser del todo preciso, mientras que si se obtienen pocos datos de entrenamiento se puede caer en el apartado b anterior: Debido a la falta de muestras representativas en los datos.

Por lo tanto, el Método Holdout es normalmente el primer paso en la creación de un modelo de inteligencia artificial, tanto predictivo como de clasificación. Por ello, es un método necesario y de altísima importancia, y se debe tener en cuenta de una forma especial.

2. Método Random Subsampling

El método de Random Subsampling simplemente consiste en la reproducción del Método Holdout n veces para probar las mejoras que se pueden dar en el rendimiento del clasificador.

Random Subsampling encuentra dificultades respecto al Método Holdout, puesto que este método no utiliza todos los datos disponibles para el entrenamiento, además de que no posee control alguno de cuantas veces se utiliza una observación para el entrenamiento o el test, por lo que algunas se utlizarán para entrenar más veces que otros, y esto puede resultar en un modelo desigual y no entrenado de forma óptima. Para la solución de este problema, se desarrolló el método Cross-Validation.

3. Método Cross-Validation

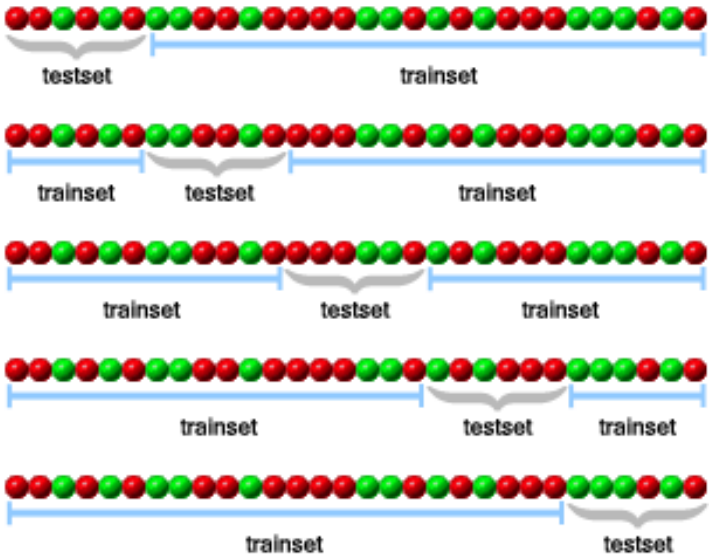
Como se ha comentado, ante los problemas de los métodos anteriores surge cross-validation (también conocido como X-Validation, o en castellano validación cruzada) y consiste en el uso de cada uno de los registros el mismo número de veces para el entrenamiento, y solo en una ocasión para test, por lo que se obtiene un modelo mucho más ajustado y balanceado.

Seguidamente, debido a la gran importancia de este método, se explica detenidamente paso por paso como funciona este algoritmo:

El primer paso es la aplicación del Método Holdout; es decir, la partición de los datos en un grupo de entrenamiento y un grupo de test. Se entrena el modelo y entonces se cambian los roles de los grupos, siendo el de test el que en este caso hará el entrenamiento y el de entrenamiento el que hará el test. Esto es lo que se conoce como “2 Fold Cross-Validation”, puesto que se han utilizado dos grupos.

Si este método se realiza N veces, se estarán dividiendo los datos en N grupos, y mediante el cambio de los roles de dichos grupos se entrena el modelo. Así, cada uno de los grupos será en un único entrenamiento grupo de test, mientras que será en N-1 entrenamientos grupo de

entrenamiento. Es importante destacar que todos los grupos de entrenamiento se unen como grupo único a la hora de entrenar un algoritmo en cada una de las iteraciones.



2-6. Ejemplo de Cross Validation con 5 Folds

En la figura 2-6 se puede apreciar un ejemplo de 5 Fold Cross-Validation, siguiendo la metodología expuesta anteriormente:

Un caso especial de Cross-Validation es en el que el número de grupos coincide numéricamente con la cantidad de datos que se poseen en el set de datos, y este método es conocido como "leave one out": Deja uno fuera, haciendo referencia al que se usará de test.

Este método posee una gran desventaja, que es que computacionalmente es extremadamente costoso, especialmente en datasets de numerosas observaciones, y por si no fuera suficiente, la varianza en las métricas de cada entrenamiento y test será extrema, ya que en algunos tests se obtendrá un acierto del 100% y en otros del 0%. Debido a esto, este método deberá de ser usado como complementario a una validación cruzada con un N más pequeño y siempre que se posea un dataset de pocas observaciones.

Ahora que se han repasado los principales problemas, es el momento de hacer una clasificación en profundidad de los algoritmos de machine learning que existen. En la literatura hay reflejados diferentes métodos de clasificación, puesto que se puede atender al modo en el que aprenden, la estructura que siguen… pero la clasificación que se seguirá en este trabajo será la siguiente debido a las grandes diferencias que existen entre los siguientes grupos:

- Algoritmos supervisados: Supervised Learning
- Algoritmos no supervisados: Unsupervised Learning
- Algoritmos semi-supervisados: Semi-Supervised Learning
- Algoritmos de aprendizaje por refuerzo: Reinforcement Learning
- Algoritmos de redes neuronales y deep learning: Neural Networks and Deep Learning.

2.3.4.3 Algoritmos Supervisados: Supervised Learning

El aprendizaje supervisado tiene su comienzo con un conjunto determinado de datos, y un entendimiento ciertamente profundo de la estructura de los datos. Este tipo de aprendizaje busca encontrar patrones en los datos, de tal forma que se puedan realizar procesos analíticos sobre unos datos ya etiquetados.

Estos algoritmos se entrenan usando ejemplos procesados anteriormente, y su precisión se mide con un conjunto de test que es, como se ha visto anteriormente, totalmente excluyente respecto al conjunto de entrenamiento.

Este tipo de algoritmos se utilizan en numerosos ámbitos, como la detección de fraudes bancarios, análisis de todo tipo de riesgos, algoritmos de recomendación…

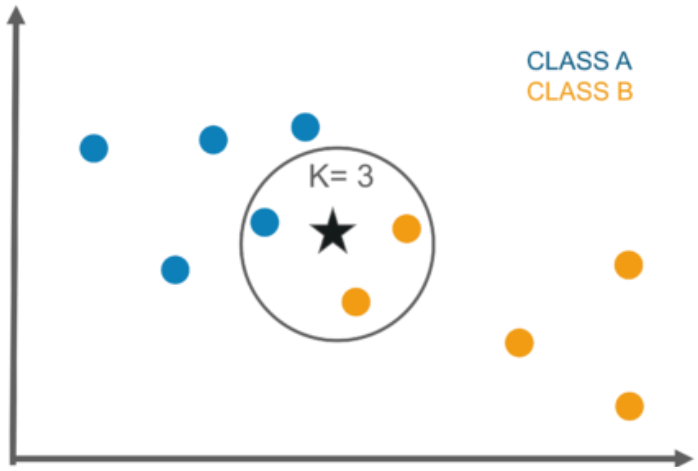
Algunos de los algoritmos más importantes que se encuentran en este grupo son:

- K Nearest Neighbors (KNN)
- Árboles de Decisión
- Regresión Linear
- Support Vector Machines (SVM)

2.3.4.3.1 K Nearest Neighbors (KNN)

El algoritmo "KNN", traducido como "los K vecinos más cercanos", es aquel que se basa en la búsqueda de atributos similares dentro del conjunto de los datos basándose en distancias, y así ser capaz de predecir la clase a la que pertenece dicho atributo. Llevado hacia un razonamiento más humano, se podría explicar de la siguiente manera: "Si se parece a un coche, es tan grande como un coche, va por carretera y pueden ir personas dentro, entonces es un coche".

Este algoritmo destaca por tener la K delante, que viene a indicar el número de vecinos con los que se va a comparar la observación determinada para obtener cuál es su clase. Así, si K es igual a 1, el elemento que esté más cerca de la observación sobre la que se quiere saber la clase será la que determine la clase de la misma. En caso de que el número K sea un número mayor, la clase que posea más elementos cerca de la observación será la que determine el tipo de la observación. En caso de que haya dos o más clases en estado de empate, se resolverá de forma arbitraria.



2-7. Ejemplo de KNN con K = 3

Es importante incidir en que la búsqueda de estos K vecinos se realiza de una forma circular o radial mediante distancias de Gauss, de tal forma que, a mayor valor de K, mayor será el círculo que se formará en torno a la observación determinada para buscar los más próximos a la misma.

Como se puede apreciar en la figura 2-7, el elemento sobre el que se quiere predecir la clase, marcado con una estrella, ha buscado a los 3 elementos más cercanos y los ha introducido dentro de un círculo. Como es obvio, es mayor el número de elementos naranjas a los de azul dentro de este círculo, de tal manera que el elemento estrellado tendrá inferida la clase B, correspondiente al grupo naranja.

El algoritmo funciona usando la distancia como elemento de similitud, puesto que dos elementos cercanos se supondrán parecidos. Así, de la observación de la que se quiere conocer el grupo se obtiene una lista de elementos próximos, y usando modernas técnicas de indexación las computaciones que se deben de hacer para obtener esta lista son menores.

Una vez que se ha obtenido la lista, se realiza la clasificación en función del grupo que posea la mayoría de los casos, donde todas las observaciones de la lista poseen el mismo peso.

El algoritmo KNN posee una serie de peculiaridades propias que establecen cuando debe ser utilizado y los “peligros” que entraña, explicándose todo ello a continuación:

1. Este algoritmo no requiere de la construcción del modelo

Como se explicó unas páginas antes, un modelo es la unión de un algoritmo con una serie de datos. El algoritmo KNN no crea un modelo, de tal manera que no se pierde en dicha tarea, pero a la hora de computarlo del algoritmo es bastante lento debido a la necesidad de calcular las distancias de todos los elementos a determinar con el resto de observaciones.

2. Es un algoritmo que toma decisiones de forma local, no de forma global

Como se puede deducir, al observar únicamente a los elementos más cercanos se incurre en que se elige una clase de forma local, al contrario de otros algoritmos como los árboles de decisión (se verán posteriormente), que toman decisiones de forma global. De esta manera, se debe tener especial cautela con los valores de K, puesto que un valor pequeño es propenso al ruido que pueda generar una zona y, por lo tanto, obtener un valor erróneo en la predicción.

Además, este modelo suele sufrir de underfitting y overfitting muy fácilmente si no se obtiene un valor K certero. En caso de elegir un valor demasiado pequeño, debido al ruido se tenderá al overfitting, puesto que solo se observarán los elementos más cercanos. Si el valor de K, por el contrario, es demasiado elevado, el algoritmo puede sufrir underfitting y el modelo se volverá demasiado simple. De esta manera, como aproximación se suele aceptar que el valor de K sea la raíz cuadrada del número total de elementos a determinar, aunque este valor siempre debe de ser corroborado, puesto que depende totalmente de los datos del problema.

3. KNN obtiene predicciones erróneas si no se realiza un preprocesamiento adecuado

KNN es un algoritmo muy delicado conforme a lo que el pre-procesamiento se refiere, puesto que al trabajar con distancias son importantes los cambios que se puedan hacer a las medidas del dataset. Por ejemplo, si hay numerosas dimensiones con valores cercanos a cero, y también

hay otra dimensión con una variabilidad enorme en sus valores, el algoritmo no funcionará correctamente puesto que esta última será la más influyente de todas, lo que no tiene por qué ser correcto.

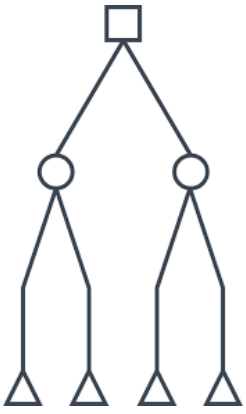
De esta forma, para trabajar con este algoritmo un pre-procesamiento a base de centrado y escalado, y la eliminación de las columnas menos importantes escogidas con un PCA sería un proceso óptimo antes de encararlo.

2.3.4.3.2 Árboles de Decisión

Una técnica muy usada en la clasificación que requiere una pequeña distinción es la utilización de los llamados árboles de decisión. Esta técnica consta de gran simpleza, pero a la vez de una gran eficacia en una enorme cantidad de problemas de data science.

Los árboles de decisión están situados dentro de las técnicas de clasificación, y por lo tanto desean hallar una respuesta a partir de unos datos previamente etiquetados. De este modo, un árbol de clasificación parte de un nodo denominado raíz, que no posee entrada alguna, pero posee salidas. En este nodo raíz se realiza una pregunta, y según la respuesta que obtenga se irán desarrollando caminos. En cada uno de estos caminos se añadirán nodos, donde se seguirá haciendo preguntas, y seguirá el árbol bifurcándose por cada respuesta hasta que sea capaz de llegar a una decisión última en cada rama. Estos nodos que se han ido formando a base de preguntas a partir del nodo raíz son denominados nodos intermedios, y las respuestas finales se denominan hojas, desde las que no saldrá ningún camino. Por cada nodo por el que pasen los datos se va haciendo una criba, de tal forma en el final de cada rama (en las hojas) solo resta un pequeño grupo de datos que poseen numerosas características similares.

La construcción de estos árboles, como es apreciable, no es extremadamente sencilla a simple vista, debido a que las preguntas correctas deben ser hechas en el momento preciso, y en un dataset de alta dimensionalidad el gran número de preguntas que se pueden hacer hace que el número de árboles construibles tienda a infinito. Por ello, algunos algoritmos que construyen árboles de decisión han sido creados para computar en tiempos razonables, como el de Hunt.



2-8. Distribución básica de un árbol de decisión

Además, en la construcción se plantean otros interrogantes, como la elección de la pregunta adecuada, las respuestas adecuadas (datos continuos) o la condición de parada del algoritmo.

Respecto al primer aspecto, el algoritmo que se use deberá de tener un sistema implementado para la evaluación de la aportación de cada pregunta hacia el propio algoritmo, de cara a entender si la pregunta ha sido beneficiosa o no.

Respecto a la condición de parada del algoritmo, es entendible que es algo necesario ya que, en caso de no existir, el algoritmo se seguiría ejecutando hasta que se acabaran las dimensiones sobre las que formular preguntas, y eso no siempre es algo positivo de cara al resultado final. Habitualmente se usan criterios tales como que todos los elementos restantes tras las preguntas posean el mismo valor, y ese valor será el que se usará como hoja final de esa rama y como condición de parada al mismo tiempo.

A continuación, se realiza una explicación de cómo se puede controlar y solucionar el overfitting en los árboles, puesto que consiste en el problema más común, de tal manera que se puedan mejorar dichos árboles de clasificación y obtener resultados óptimos.

1. Método de la Pre-poda

En el caso del uso de este método, el algoritmo no deja crecer el árbol para formar el árbol completo que encajaría perfectamente con todos los datos de entrenamiento.

Para hacer esto, se debe de poner una condición muy restrictiva para dar por finalizado el algoritmo, como el incremento de alguna impureza o especialmente el error en la generalización.

El problema de esta solución es que, si la restricción es demasiado prohibitiva, el modelo quedará en underfitting y por lo tanto será poco certero, mientras que si la restricción es demasiado ligera el modelo empeorará por overfitting y por lo tanto generalizará también mal.

2. Método de la Post-poda

En el caso del uso de este método, el primer paso es dejar al algoritmo crecer libre hasta su máxima extensión, y tras la finalización del algoritmo comienza la poda. Esta se suele hacer obteniendo subárboles, y cambiando estos subárboles por una hoja final perteneciente al grupo que posee la mayoría de los individuos en ese subárbol.

Este método es el más usado debido a que obtiene mejores resultados, fruto de una poda posterior donde las decisiones de donde recortar provienen de un árbol formado totalmente.

2.3.4.3.3 Regresiones

La regresión es un método predictiva para estimar variables continuas. Algunos ejemplos de predicción a través del uso de regresión son el precio de acciones en la bolsa, predicción de ventas en relación con gasto en publicidad…

Una definición más formal de regresión sería: “Tarea de aprender una función objetivo de tal manera que al aplicar un valor x sobre la misma se obtenga correctamente un valor continuo y”.

De este modo, el objetivo que tienen los métodos de regresión es encontrar esta función objetivo que posea el mínimo error en la predicción de los datos. La función de error de la misma se suele dar de dos maneras:

$$Error\ Absoluto = \sum_i |\ yi - f(xi)\ |$$

$$Error\ cuadrático = \sum_i \big(yi - f(xi)\big)^2$$

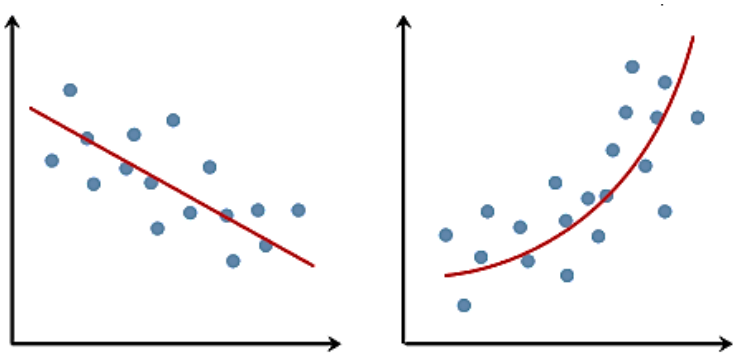
Con estas ecuaciones se puede observar el sumatorio de la diferencia entre el valor real y el valor esperado aplicando x sobre la función objetivo. En el primer caso, ya que se pueden dar restas negativas debido a que el valor de la función objetivo sea superior que el real, a cada iteración del sumatorio le es aplicado el valor absoluto, puesto que lo que resulta de interés es la distancia del fallo.

En el caso de la segunda ecuación, al estar calculando el error cuadrático no es necesario aplicar el valor absoluto, puesto que, matemáticamente, cualquier valor que sea elevado al cuadrado será positivo. En esta ecuación, los errores más grandes son penalizados exponencialmente, mientras que los pequeños (inferiores a 1) son minimizados.

Para ajustar la función objetivo al máximo a los datos, normalmente se suele utilizar un método, conocido en la comunidad anglosajona como "Least Square Method". Se supone que se debe calcular para una variable la función objetivo que llevará la siguiente estructura:

$$f(x) = a * x + b$$

Donde a y b son los llamados "coeficientes de regresión". Usando LSM, se deben de hallar a y b para que la suma de los errores cuadrados sea mínima, y por lo tanto la función objetivo elegida sea la más cercana a la óptima. Sobre esta función objetivo posteriormente se harán las predicciones de los valores.



2-9. Ejemplos de Regresiones

Es importante destacar que no todas las regresiones son lineales, como es el caso de la figura 2-9. También existen regresiones no lineales, que son aquellas en las que se ajusta el modelo con una ecuación con exponente 2 o superior, de tal forma que la recta de regresión deja de ser recta en pos de ser curva. Estas ecuaciones poseen la siguiente forma:

$$f(x) = h * x^k + \cdots + i * x^2 + j * x + k$$

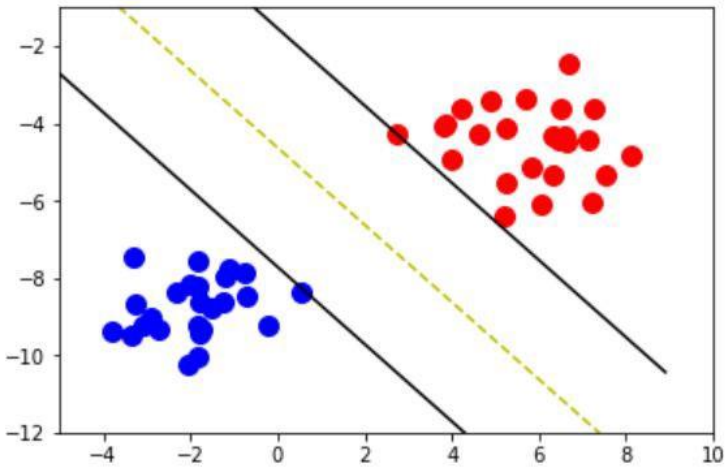
2.3.4.3.4 Support Vector Machines (SVM)

El algoritmo SVM es uno de los algoritmos más usados en el machine learning, puesto que posee fuertes bases estadísticas y matemáticas y ha demostrado gran solvencia en aplicaciones prácticas, como por ejemplo en el reconocimiento de caracteres manuscritos. Para comprender correctamente cómo funciona SVM, es importante entender los hiperplanos, puesto que son el fundamento de esta técnica de inteligencia artificial.

Los hiperplanos son planos infinitos en un espacio. En SVM, estos hiperplanos son denominados "hiperplanos de margen máximo". Son usados como soporte de las fronteras para separar dos o más grupos de datos por el grupo al que pertenecen, de tal forma que se consiga

la separación más óptima. Cada frontera, al ser infinita en una cierta dirección, solo tendrá dos hiperplanos que la soporten, que se colocarán a cada uno de los lados de la misma.

En el caso de que los datos sean linealmente separables, existirán infinitas fronteras que los separen, puesto que por un punto pueden pasar infinitas rectas, y por lo tanto planos o fronteras. Pero aquí se plantea un problema, y es que en el caso de que exista un nuevo dato, se tendrá que saber cuál de todas esas infinitas fronteras será la que discierna mejor a qué grupo pertenecerá dicho punto. Aquí es donde los hiperplanos cobran sentido. Como se ha visto, cada frontera posee dos hiperplanos, que serán paralelos a la frontera, y su ubicación estará determinada por el elemento más cercano que se pueda encontrar desde la frontera hacia cada una de las clases de una forma ortogonal. De este modo, la frontera ideal será la que posea una mayor distancia con sus hiperplanos de soporte, puesto que será la que posea un mayor margen con cada una de las clases, y por lo tanto una menor probabilidad de errar, puesto que se realiza una mejor generalización. Todo ello es fácilmente apreciable en la siguiente imagen:



2-10. Ejemplo de SVM Linealmente Separable

En caso de que la frontera sea muy pequeña, normalmente se obtiene en un caso de overfitting, puesto que el margen de error para tomar las decisiones es pequeño y la frontera deberá de ser más ajustada para poder tener unos hiperplanos de soporte lo más amplios posible.

Es vital a partir de ahora destacar que existen dos tipos de SVM, que son el lineal y el no lineal. En el caso del primero, se busca la frontera con el máximo margen con sus hiperplanos de apoyo, lo que hace que se le conozca también con el nombre de “clasificador de máximo margen". En caso de dar referencia a SVM no lineal, la técnica consiste en la transformación del espacio en el que se encuentran los datos, de tal forma que se pueda aplicar una frontera lineal para separar las clases del problema. A continuación, se expondrán estos diferentes casos que se pueden dar de acuerdo con esta clasificación, y se detallarán con profundidad:

1. SVM Lineal: Caso Separable

En caso de tener un caso separable en un SVM lineal, como se ha comentado anteriormente se está en un caso de clasificador de márgenes máximos. Por ello, la frontera será una recta, y cumplirá la siguiente ecuación:

$$ax + b = 0$$

De este modo, el aprendizaje de este algoritmo será la determinación de los valores a y b de la ecuación anterior mediante los datos de entrenamiento.

2. SVM Lineal: Caso No Separable

El caso de tener unos datos no separables linealmente hace que haya que tener mucho más cuidado a la hora de elegir la frontera, puesto que a veces muchas fronteras que no incurren prácticamente en errores en el entrenamiento tienen unos márgenes muy pequeños, y generalizan francamente mal. Por ello, en estos casos se debe hacer una aproximación llamada "soft margin", que consiste en la búsqueda de un equilibrio entre los márgenes de la frontera y el número de clasificaciones erróneas que se dan en el entrenamiento. Esto es reducible en cierto modo en la etapa previa al algoritmo, puesto que si se dividen los datos por su grupo real creando subgrupos, y se detectan los outliers, se puede simplificar este proceso de "soft margin".

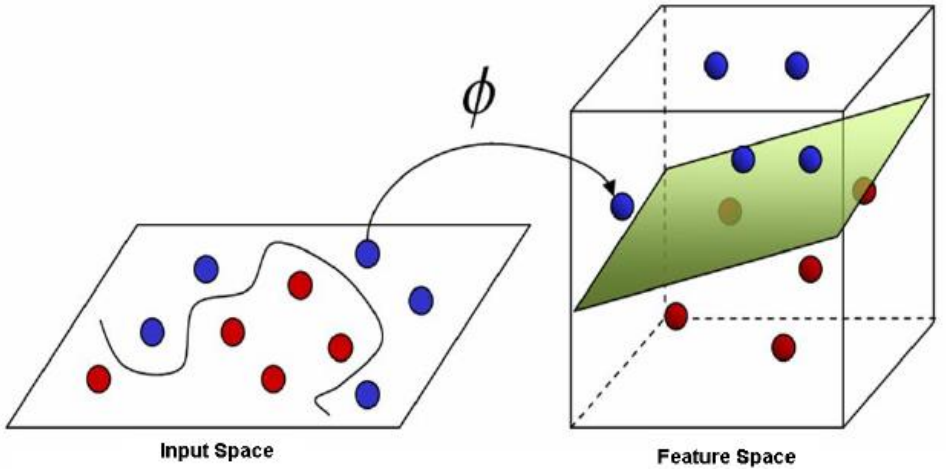
3. SVM No Lineal

En el caso de haber un problema de kernel no lineal, se debe hacer la transformación del espacio tal como se explicó anteriormente y como se muestra simplificadamente en la figura 2-13.

Así, tras la transformación del espacio, se habrá convertido un problema no lineal en un problema lineal, y se podrán aplicar las técnicas anteriormente descritas para resolver el problema.

Algunas de las características principales de SVM son:

1. Los problemas planteados con SVM son problemas de optimización, que por regla general llegan a la solución encontrando un mínimo global, mientras que otros algoritmos como las redes neuronales artificiales suelen caer en mínimos locales.



2-11. Transformación de SVM No Lineal

2. Al afrontar un problema con SVM, es importante anotar el tipo de kernel que se usará (lineal o radial), y controlar también la función de coste C.
3. SVM también funciona con datos categóricos, pero para ello hay que aplicar una binarización, que se ha visto anteriormente.

2.3.4.4 Algoritmos no Supervisados: Unsupervised Learning

El aprendizaje no supervisado se caracteriza por hacerse sobre un conjunto de datos sin etiquetas de grupo. Así, estos algoritmos tendrán que encontrar patrones en los datos y clasificarlos respecto a estos patrones sin ninguna intervención humana.

Estos algoritmos se utilizan en ámbitos muy diversos, como los sistemas anti-spam de los correos, reconocimiento de imágenes, obtención de información de redes sociales…

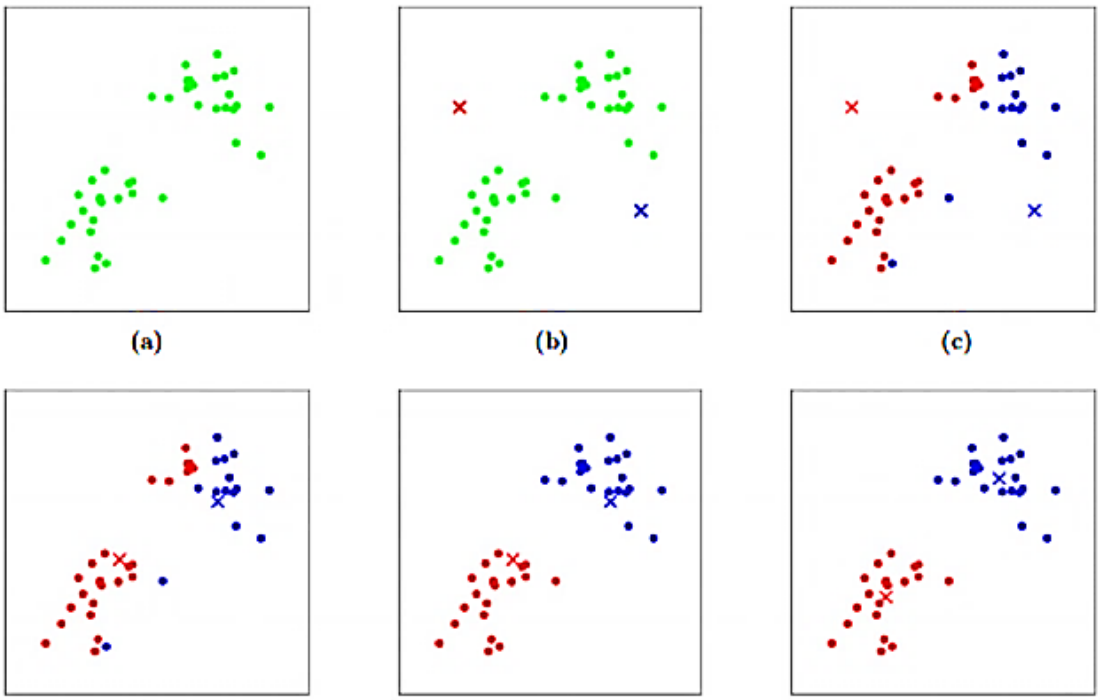
Algunos de los algoritmos más importantes que se encuentran encuadrados en este grupo son:

- K-Means
- Reglas de Asociación

2.3.4.4.1 K Means

K-Means es una técnica de clustering de datos que se basa en la creación de un centroide, normalmente creado como la media de un grupo de objetos, que se aplica a objetos en un espacio n-dimensional.

El funcionamiento de K-Means es simple: Se empieza con la elección de K centroides, siendo K el número de grupos que se pretende discernir. Entonces, a base de distancias gaussianas, se obtienen las distancias de cada elemento del dataset con los centroides y se va asignando cada elemento a un cierto grupo. Una vez finalizado este primer paso, se redefine el centroide de cada grupo en función de los elementos que estén formando en ese momento el cluster, y se vuelve a empezar. La condición de parada es la falta de elementos que cambien cualquier objeto de un cluster a otro, y por lo tanto que ninguno de los centroides tenga que cambiar su posición. Para evitar costes computacionales innecesarios, en datasets extremadamente grandes se puede aplicar una parada cuando menos de un 1% de los puntos hagan cambios, de tal manera



2-12. Explicación de K-Means paso a paso

que no se tengan que recalcular ni las distancias ni los centroides y, por lo tanto, evitar iteraciones no necesarias en el algoritmo.

Otra visión que se puede tener del algoritmo es la de que es un algoritmo de optimización, donde la función objetivo debe minimizar las distancias de los puntos con el centroide más cercano.

Es importante destacar que K-Means, como se ha comentado anteriormente, utiliza distancias euclídeas, pero estas no son las únicas distancias que existen. Otra distancia que también sería compatible con este algoritmo sería la distancia de Manhattan, aunque con esta distancia en vez de utilizar la media para calcular los centroides se utilizaría la mediana. Por otra parte, la distancia de Jaccard es una distancia que se suele usar más en el análisis de documentos y la similitud entre los mismos y, por lo tanto no es la más indicada para este algoritmo ni para este problema.

Una vez que se tiene clara la distancia a usar en este algoritmo, es interesante la explicación de la "suma del error cuadrado". Esta suma, de forma similar a lo visto previamente en la regresión, consiste en la adición de las distancias de todos los puntos de un cluster con su centroide más cercano. Si esto se realiza para varios clusters, el cluster más acertado será el que posea una suma del error cuadrado menor. Al igual, si se tienen varios sets de clusters distintos, la mejor elección será la que posea la suma del error cuadrado más pequeña. La elección que se haga de los centroides al principio es vital para la suma del error cuadrado final.

La elección de los centroides iniciales es de gran importancia a la hora de iniciar el algoritmo de K-Means, puesto que las diferentes elecciones que se puedan hacer producen diferentes resultados y variar la suma del error cuadrado. Por ello, existen diferentes técnicas para la inicialización de estos centroides:

1. Inicialización de forma aleatoria

La inicialización de los centroides en un punto aleatorio del espacio conlleva que se pueda encontrar un mínimo local que pueda parecer óptimo, pero rara vez se consigue un mínimo global que sea la mejor solución del problema.

2. Sucesión de inicializaciones aleatorias

Una técnica que se suele utilizar es la inicialización del algoritmo N veces de forma aleatoria, llegando hasta el final y seleccionando los clusters con menor suma de error cuadrado. Esta técnica presenta numerosos problemas, puesto que por una parte es muy costosa computacionalmente, pero además de ello, la re-inicialización del algoritmo sobre los mismos datos conlleva que muchos intentos sean fallidos. Por ejemplo, si se pasa como parámetro al algoritmo K = 4 y existen 4 grupos bien diferenciados, pero tres de los centroides comienzan en uno de los grupos, dicho grupo acabará siendo dividido y por lo tanto la formación de los clusters será errónea.

K-Means suele tener otros problemas, además de la elección del centroide inicial. En las siguientes líneas se analizan estos problemas y las posibles soluciones que se pueden dar, o las recomendaciones a seguir a la hora de aplicar esta técnica:

a. Manejo de clusters vacíos

Uno de los problemas con los algoritmos de K-Means básicos es que se puede dar que ningún punto sea asignado a un centroide y, por lo tanto, se obtenga un cluster totalmente vacío en las etapas de asignación de puntos a centroides anteriormente vistas. Por esta razón, los algoritmos de K-Means deberán de tener una serie de políticas de reemplazamiento de centroides por otros en caso de que esto pase, porque en caso contrario la suma del error cuadrado será demasiado alta debido a las grandes distancias que se pueden acabar formando en el resto de grupos debido a las clasificaciones erróneas.

Una aproximación que suelen hacer estos algoritmos consiste en coger el punto más alejado, y que por tanto más suma al error cuadrado, y eliminarlo, de tal manera que ningún centroide pueda establecer allí su primera base y por lo tanto no haya posibilidad de obtener grupos vacíos. Si se analiza esta acción con otra perspectiva, se puede observar que este método está realizando una eliminación de un outlier.

b. Problemas con los outliers

De forma obvia se puede inferir que si se usa el error cuadrado, el hecho de que haya elementos outliers influirá de gran manera al resultado final. Esto se debe a que los centroides, tras la última iteración del algoritmo, es improbable que estén situados en el punto óptimo donde deberían de estar, sino demasiado influenciados por los outliers, de tal manera que la suma del error cuadrado aumentará.

Por lo tanto, uno de los mayores problemas a enfrentarse a la hora de hacer clustering, y en especial con K-Means, es el problema de los outliers y su identificación. Existen numerosas aproximaciones para identificar outliers, pero una de las más sencillas es la eliminación de puntos que presenten un error mucho más alto que los compañeros del cluster. También, en el caso de la existencia de clusters especialmente pequeños, es interesante una valoración especial de si el cluster es válido, puesto que puede ser simplemente un grupo de outliers y no un grupo real válido.

Debido a estos dos problemas que se han encontrado, la búsqueda de soluciones para mejorar los algoritmos de clustering es obligatoria. A consecuencia de esto, técnicas como el post-procesado del clustering para reducir la suma del error cuadrado se plantean como técnicas interesantes a emplear, además de otras estrategias que se presentarán a continuación:

Normalmente se puede mejorar el error obtenido aumentando la K, puesto que al haber más centroides, si se inicializan de una manera correcta, los puntos estarán más cercanos a ellos y por lo tanto las distancias disminuirán. Pero normalmente no se pretende aumentar el número de grupos que existen, por lo que el analista debe de empezar a pensar de una manera más global. Si no se está conforme con el clustering realizado, es posible que K-Means haya caído en un mínimo local, lo que significa que hay una solución mejor pero que no ha sido capaz de llegar a ella. La repetición del algoritmo puede llevar hacia un mínimo global.

En caso de que estas técnicas anteriores no sean satisfactorias, se deberá de pensar en hacer un post-procesado del clustering. Hay dos métodos:

1. Incremento del número de clusters

En caso de que un mínimo local no haya sido encontrado y se quiera aumentar la precisión del algoritmo, el incremento del número de clusters, como se ha comentado anteriormente, es una solución factible. Así, existen dos opciones a tener en cuenta:

a. División de un cluster

En caso de la elección de la división de un cluster, se deberá de elegir el que posea un mayor error. También, como opción alternativa, aquel que tenga una mayor varianza puede ser el que sea elegido. Una vez que se tenga elegido el cluster en cuestión, se procederá a la división del mismo en dos o más grupos, de tal manera que se obtengan grupos mucho más cohesionados y cercanos.

b. Introducción de un nuevo centroide al problema

En caso de elegir la introducción de un nuevo centroide, la técnica que se suele escoger es la de la elección del punto más alejado de cualquier centroide de los clusters, y la introducción de un nuevo centroide en ese punto.

Debido a elementos explicados anteriormente, esta estrategia tiene dos problemas: El primero es el gran coste computacional que conlleva el cómputo del elemento más alejado de un dataset, y el segundo es la gran posibilidad de obtener un cluster muy reducido con el punto outlier que se elija, de tal manera que se debería de pensar en la eliminación de ese punto en caso de que esto pasara.

2. Decrementar el número de clusters

La reducción del número de clusters, mientras que se intenta minimizar el aumento en el error, puede seguir otras dos estrategias, que serán explicadas a continuación:

a. Dispersión de un cluster

La dispersión de un cluster consiste en la eliminación del centroide del cluster en cuestión, y la reasignación de los puntos de ese antiguo cluster a los restantes, siguiendo el mismo procedimiento que se sigue en el algoritmo mediante negociación por distancias euclídeas.

De una forma ideal, el cluster que debe de ser dispersado será aquél que aumenta el error total de una forma mínima.

b. Fusión de dos clusters

La unión de dos clusters es una opción muy interesante en caso de tener en el problema dos clusters que sean muy unidos, puesto que al aplicar esta técnica aumentará de una manera ligera el error total. También se puede realizar una computación que, aunque costosa, permite determinar qué dos clusters al unirse producirán un aumento del error total más pequeño y actuar en consecuencia.

A continuación se explican algunas de las fortalezas y debilidades que K-Means posee, puesto que al ser un algoritmo tan utilizado y conocido deberán de ser expuestas claramente:

K-means tiene grandes dificultades para obtener los clusters “naturales” que se pueden dar en la realidad, puesto que se centra principalmente, debido a sus distancias euclídeas, en clusters con un tamaño similar y una forma esférica. Además, suele tener problemas en clusters donde la densidad varía, puesto que suele poner un centroide donde haya una mayor densidad, a pesar de que ahí pueda haber dos o más grupos.

Por otra parte, K-means es un algoritmo que se puede utilizar para numerosos tipos de datos (pero no todos), además de que es bastante eficiente, especialmente cuando el número de datos a clasificar no es extremadamente alto. De esta manera, se pueden realizar varias ejecuciones del algoritmo para poder encontrar el mínimo global. Además, como se ha visto anteriormente, los errores por outliers son fácilmente identificables y solucionables.

2.3.4.4.2 Reglas de Asociación

Los clasificadores basados en reglas de asociación son aquellos que se basan en la clasificación de elementos usando reglas condicionales, de tal manera que siguen la estructura básica de programación if-then, que se puede ver en la forma siguiente:

```
If
    …
Then
    …
```

Las reglas, una vez formadas, se expresan mediante condiciones disjuntas donde el operador ∧, equivalente a la conjunción "y", delimita las reglas. Finalmente, las reglas acaban con una flecha con dirección hacia la derecha donde se muestra el resultado de la regla. En la tabla a continuación se puede ver gráficamente este formato de presentación de las reglas, donde se hace presentación de 3 reglas distintas:

Tabla 2-2-1. Ejemplo Reglas de Asociación Finales

Condición 1	Operador	Condición 2	Dirección	Consecuente
Es moto = T	∧	Es eléctrica = F	→	Contaminante
Es moto = T	∧	Es eléctrica = T	→	No Contaminante
Es moto = F	-	-	→	No es moto

Como se puede ver en la tabla superior, si un elemento es una moto y es eléctrica las reglas lo incluirán dentro de la categoría de no contaminante, mientras que, si es una moto y no es eléctrica, se introducirá dentro de la categoría de contaminante. En cambio, si no es una moto directamente se introducirá dentro de una tercera categoría "no es moto".

Es importante remarcar que las reglas que están más a la izquierda son más determinantes a la hora de tomar las decisiones, puesto que son unos "antecedentes" o "precondiciones", por lo que los elementos que pasen las reglas que estén situadas más a la izquierda serán más similares. Este sistema se parece en gran medida a los árboles de decisión (ver figura 2-10). También es importante comentar que se pueden poner tantas condiciones como dimensiones tenga el problema, o incluso más, puesto que, por ejemplo, en los elementos numéricos pueden darse dos condiciones sobre una misma variable en caso de querer acotarla tanto superior como inferiormente.

Es importante definir dos términos a la hora de hablar de reglas, que son cobertura y precisión:

Cobertura se define como la fracción de los elementos en un dataset que activan una determinada regla.

Precisión, también conocido como factor de confianza, se define como la fracción de los elementos activados por una determinada regla cuyas clases sean iguales a las predichas.

En términos matemáticos, para un total D de elementos en un dataset, con una regla R, un número de elementos C que satisfacen las condiciones, y un grupo real G, se podrían definir en los siguientes términos:

$$Cobertura(R) = \frac{C}{D}$$

$$Precisión(R) = \frac{C \cap G}{C}$$

Calculemos como ejemplo la cobertura y la precisión de una determinada regla sobre la ilustración 2-15. Pongamos como ejemplo la siguiente regla

$$(Outlook = Rainy) \wedge (\text{Temp} = \text{Cool}) \rightarrow No$$

en la cual se está diciendo que si el tiempo atmosférico está lluvioso y la temperatura es fría no se jugará. Para la simplificación del ejemplo, no se tendrán en cuenta las condiciones de humedad ni de viento.

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

2-13. Ejemplo de tabla de reglas

En este caso, lo primero que se deberá de calcular a la cobertura, es decir, cuantas reglas respecto del total cumplen estas dos condiciones. Como se puede observar en las líneas 5 y 6, esas dos suposiciones cumplen la regla ejemplo respecto del total de 14, por lo que la cobertura será la siguiente:

$$Cobertura = \frac{2}{14} = \frac{1}{7} = 0.142 = 14{,}2\%$$

A continuación, habrá que investigar la precisión de la regla, es decir, de estas dos reglas qué porcentaje han cumplido la predicción

$$Precisión = \frac{1}{2} = 0.5 = 50\%$$

ya que la línea 5 afirma que se puede jugar, mientras que la línea 6 lo desmiente. Ya que mi regla de ejemplo lo desmentía, podemos solo tener un 50% de acierto.

Un clasificador en base a reglas clasifica en función de las reglas que haya cumplido un cierto elemento. De esta manera, el elemento central en estos clasificadores obviamente son las reglas, que pueden ser de dos tipos:

- Reglas mutuamente excluyentes

En un conjunto de reglas, dos reglas serán mutuamente exclusivas si no hay dos reglas en todo el conjunto que puedan ser cumplidas por el mismo elemento del dataset.

- Reglas exhaustivas

Un conjunto de reglas es exhaustivo si hay una regla por cada combinación que se pueda dar de los atributos o dimensiones de los elementos del dataset. Eso asegura que absolutamente todos los elementos del dataset estarán cubiertos por al menos una regla.

Si se junta la propiedad de la mutua exclusividad y la exhaustividad, se obtiene un conjunto de reglas donde cada elemento del dataset está cubierto siempre y sólo por una regla. Desafortunadamente, prácticamente todas las clasificaciones por reglas no cumplen esta situación idílica.

Hay algunas características peculiares de los sistemas clasificadores basados en reglas que son importantes tener en cuenta antes de la ejecución de un modelo de este tipo:

1. Los clasificadores basados en reglas ofrecen modelos descriptivos mucho más fáciles de interpretar que otros métodos de machine learning, pero no tan buen rendimiento, quedándose en un lugar similar al árbol de decisión.
2. El coste computacional de crear un conjunto de reglas para un dataset es muy comparable con el coste que sería el hacer un árbol de decisión del mismo dataset, puesto que un árbol de decisión, como se comentó anteriormente, está compuesto de reglas exhaustivas y exclusivas entre ellas. El problema que poseen los conjuntos de reglas es que raramente se obtienen reglas exhaustivas y exclusivas, por lo que se necesitan rehacer los parámetros de las reglas (y con ello cambiar las fronteras de decisión) para que el algoritmo funcione de una manera óptima.

2.3.4.5 Algoritmos Semi-Supervisados: Semi-Supervised Learning

Los algoritmos semi supervisados son un tipo de algoritmos que están cogiendo una gran fuerza hoy en día. Aunque se pueden hacer diversas aproximaciones al grupo, son ampliamente conocidos como unos algoritmos en los que solo una parte de los datos están etiquetados, mientras que otra parte, que suele ser la mayoría, no poseen etiqueta.

El procedimiento que realizan es aprender a partir de los algoritmos etiquetados, de tal manera que al pasarle los elementos no etiquetados el modelo pueda hacer predicciones.

Un ejemplo de un ámbito en el que se suele utilizar este algoritmo es un “call center” o centro de llamadas, donde se puede hacer análisis a la voz de la gente y obtener datos como su estado de ánimo o el género, e intentar inferir qué tipo de problema tiene y por lo tanto con qué interlocutor debe de ser redirigido de una manera automática.

Muchos algoritmos supervisados ya explicados pueden hacer la función como algoritmos semi-supervisados, entre los que se incluyen Support Vector Machines (SVM) o Random Forest.

2.3.4.6 Algoritmos de Aprendizaje por Refuerzo: Reinforcement Learning

Los algoritmos de aprendizaje por refuerzo se basan en un cambio de comportamiento y filosofía respecto a los anteriores. En estos, el algoritmo recibe un feedback desde la parte de analítica de datos, de tal forma que se le va guiando a la mejor solución. Como se puede observar, estos algoritmos no están entrenados a la hora de que el usuario lo use, sino que van aprendiendo a base de prueba y error. Esto conlleva que una serie de errores harán al algoritmo aprender, mientras que una serie de aciertos le aplicarán un refuerzo que le acerquen a la solución.

Como se puede observar, este tipo de algoritmos son muy parecidos a la anteriormente explicada economía de fichas, puesto que una serie de errores (castigos) harán que el algoritmo no siga dicho camino, mientras que una serie de aciertos (premios) harán que el algoritmo siga por esa vía, puesto que está llevando un buen camino de cara al futuro.

Este tipo de algoritmos se utilizan especialmente en robótica y en los personajes de los videojuegos. Un ejemplo del segundo caso es en el que se lucha contra el personaje controlado por inteligencia artificial de cara a un objetivo, y el personaje aprende de los movimientos del jugador que le perjudican para mejorar y poder conseguir el objetivo de una manera más óptima.

También se utiliza este algoritmo para los coches de conducción autónoma. En este caso, el uso del algoritmo es de una dificultad extrema, puesto que la cantidad de obstáculos que puede haber en la carretera, así como imprevistos, es altísimo. Si todos los coches fueran autónomos, mediante comunicación entre ellos resultaría más sencillo, pero en la vida real con conducción realizada por humanos el movimiento de los coches es impredecible.

Este tipo de algoritmos no son demasiado interesantes hacia este trabajo, puesto que como se ha comentado, son mucho más usados en temas de robótica. Debido a esto, simplemente se hará comentario teórico de los mismos a continuación.

2.3.4.7 Redes Neuronales y Deep Learning: Neural Networks & Deep Learning

Deep Learning, o aprendizaje profundo, es una técnica de machine learning que recrea una red neuronal artificial formada por una serie de capas ocultas, de tal manera que el algoritmo pueda aprender de una manera iterativa. Estos algoritmos han sido clasificados en un grupo distinto debido a que pueden ser tanto supervisados como no supervisados, por lo que poseen un trato distinto.

Las redes neuronales artificiales, especialmente al principio, surgieron como un intento de copia de las redes neuronales biológicas para poder trabajar con abstracciones, al igual que la mente humana. Una red neuronal artificial consiste en una red de nodos, llamados neuronas, que se distribuyen normalmente en un mínimo de tres capas:

1. Capa de entrada: Capa que recibe los datos introducidos a la red.
2. Capa oculta: Capa con un número muy variable de neuronas donde los datos se modifican para el entrenamiento de la red. Esta capa es opcional, y puede haber más de una, lo que marca una de las diferencias entre una red neuronal o una red deep learning.
3. Capa de salida: Capa en la que también se modifican los datos y finalmente se ofrece un resultado. Esta capa suele tener funciones de activación diferentes al resto.

Con Deep Learning se hace referencia a una técnica dentro de machine learning en la que se usan redes neuronales de una forma jerárquica, donde cada red neuronal puede tener hasta millones de nodos densamente interconectados. Además, como se indicó previamente, se diferencia también de las redes neuronales tradicionales en que suele tener más de una capa oculta.

Las redes neuronales se suelen utilizar en ámbitos como el reconocimiento de imágenes y la visión artificial, aunque también pueden actuar como algoritmos de regresión y clasificación.

2.3.4.7.1 Redes Neuronales

Como se ha comentado anteriormente, las redes neuronales artificiales surgieron con la intención de simular redes neuronales humanas. El cerebro humano consiste principalmente en neuronas que se intercomunican con otras neuronas mediante unos extremos llamados axones, que producen y reciben impulsos eléctricos. De una manera simplificada, podríamos decir que las neuronas humanas están conectadas a los axones de otras neuronas mediante las dendritas, que son extensiones del cuerpo de la neurona. La acción de enviar información por una dendrita hacia el axón de otra neurona es conocido como sinapsis.

De forma análoga, las redes neuronales artificiales tienen una serie de nodos interconectados entre sí que pasan una cierta información a otros nodos, de tal manera que se acaba llegando a una solución final.

La red neuronal más sencilla es el perceptrón. Debido a su sencillez, a continuación se explicará el funcionamiento del mismo, así cómo los modelos pueden ser entrenados para resolver problemas de clasificación.

El perceptrón consiste en un tipo de red neuronal artificial que posee dos tipos de nodos, siendo unos de entrada y otro de salida. Los nodos de entrada sirven para representar la entrada de datos, y habrá un nodo por cada dimensión del problema. El nodo de salida obtendrá y sacará el resultado del problema. Estos nodos son los llamados neuronas.

En los perceptrones, los nodos están directamente conectados al nodo de salida mediante una especie de enlace con un peso determinado, que se podría comparar análogamente con la sinapsis. Este peso es el elemento más importante de la conexión, puesto que controla la fuerza de dicha conexión y por lo tanto su importancia en el resultado final, puesto que el nodo de salida hace la suma de cada dato de cada neurona, pero multiplicado por el peso de dicha conexión. Cuando un modelo de perceptrón se entrena, se calculan estos pesos hasta que el resultado final coincide con el que debería de obtenerse. De este modo, al introducir nuevos datos, las relaciones y sus pesos estarán ya establecidos y por lo tanto el resultado de este nuevo dato será, en condiciones óptimas, correcto. Los pesos, al iniciar el entrenamiento de la red neuronal normalmente son inicializados de manera aleatoria.

Es importante destacar que en el perceptrón no hay ninguna capa intermedia u oculta, ya que si no estaríamos entrando en el campo del deep learning. Estas capas ocultas son otras capas que se pueden añadir a la red neuronal entre la capa de entrada y la capa de salida, y permiten hacer más operaciones de cara a la obtención del resultado final, complicando más la red y pudiendo obtener resultados óptimos de relaciones más complejas. Además, es también importante destacar que el número de funciones de activación de los nodos es muy reducido, lo cual también contribuye a que los resultados obtenidos por el perceptrón sólo sean buenos en problemas de bajísima complejidad.

En el proceso de aprendizaje o entrenamiento del perceptrón se tiene en cuenta un factor muy importante, llamado “ratio de aprendizaje”. Se podría definir como un hiperparámetro de

las redes neuronales en el que se controla cuanto se están cambiando los pesos de una red neuronal en función del descenso del *gradiente*. En el caso de que este ratio sea pequeño, la red neuronal avanzará poco a poco, lo cual es un elemento a favor ya que se buscará con cautela un mínimo local donde converger, pero por otra parte cuanto más pequeño sea el ratio de aprendizaje más costoso computacionalmente será entrenar a la red. El ratio de aprendizaje también es ampliamente conocido con otro nombre: Decay.

Por lo tanto, en resumen, se puede afirmar que el cálculo de los nuevos pesos de la red neuronal atiende a la siguiente fórmula:

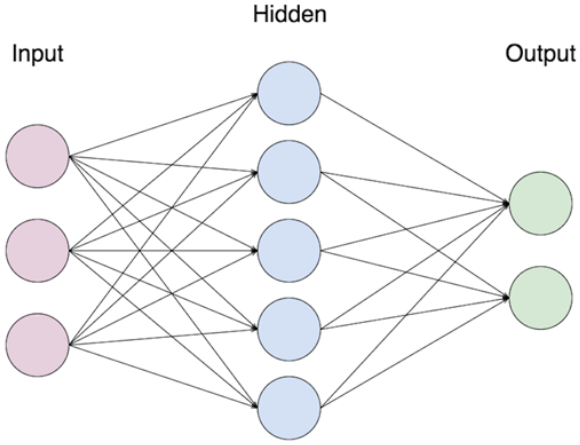
$$newPeso = oldPeso - rApren * gradiente$$

Continuando con la explicación, ahora el análisis se deberá de hacer sobre las redes neuronales artificiales multicapa. Estas redes son la evolución del perceptrón, debido a la falta de potencia del mismo ya que no es capaz de separar dos grupos de datos que no sean separables mediante un hiperplano.

Las redes neuronales multicapa tienen una serie de características propias:

1) La existencia de capas ocultas con nodos ocultos en ellas, al contrario del perceptrón.
2) El número de funciones de activación en este caso no se reduce sólo al signo, de tal manera que se pueden producir resultados no lineales.

Estos elementos hacen que estas redes neuronales más complejas permitan obtener resultados positivos sobre datos que poseen relaciones más complejas entre ellos, aunque



2-14. Red Neuronal Multicapa Feed Forward

computacionalmente sean más costosas. Es importante destacar que, cuantas más neuronas haya en la capa oculta, la red neuronal va a poder resolver problemas más complejos, pero a la vez hay mayores posibilidades de caer en overfitting. Debido a ello, el control del mismo con técnicas como el decay es algo que siempre se debe de tener en cuenta.

Además del hiperparámetro decay, es interesante el uso de otro hiperparámetro llamado softmax. Esta *función de activación*, de una manera simple, lo que hace es la transformación de los números que llegan a una cierta capa (normalmente la de salida) de la red neuronal en probabilidades, cuya suma total es igual a uno. Con ella, se consigue suavizar la posibilidad de que dos elementos puedan tener el mismo peso o pesos demasiado diferenciados, donde en este segundo en caso de equivocación el error tiende a infinito.

Suele usarse en clasificaciones, aunque uno de sus mayores problemas es el hecho de que no tiene en cuenta la posibilidad de que un elemento pueda pertenecer a dos categorías. En caso de que en el problema esto no sea así, siempre es bueno intentar resolver el problema con una red neuronal donde se tenga softmax activado para intentar mejorar los resultados en la capa de salida, puesto que con este hiperparámetro conseguimos probabilidades de pertenencia a cada grupo en las neuronas de salida.

Dentro de las redes neuronales multicapa, es interesante distinguir entre dos tipos:

1) Redes Neuronales Feed Forward

En las redes neuronales feed forward, los nodos de una capa están conectados únicamente a los nodos de la siguiente capa. Esta red se puede ver de forma esquematizada en la figura 2-17.

2) Redes Neuronales Recurrentes

En las redes neuronales recurrentes, los nodos pueden estar conectados a cualquier nodo, incluyendo los de la siguiente capa, los de capas anteriores o incluso a nodos en la misma capa.

Para entender cómo funciona una red neuronal multicapa, es importante entender cómo funciona el perceptrón, puesto que el fundamento es el mismo: El cálculo de los pesos de las conexiones entre nodos para poder reducir el error al clasificar al máximo.

En la mayoría de ocasiones, el output que ofrece una red neuronal multicapa no es una función lineal, y esto es debido a las funciones de activación que se hayan elegido. Esto lo que hace es que las soluciones que se puedan obtener no tengan por qué ser las globalmente óptimas, y algunas soluciones como el gradiente descendente han sido codificadas para intentar mejorar este problema de optimización.

El método del gradiente descendente puede ser usado para aprender los pesos de las capas intermedias y de salida de la red neuronal. Para los nodos que estén en la capa oculta, la computación a realizar no es para nada trivial, puesto que es difícil saber el error que generan esos nodos sin saber cual es el valor real que deberían de tener, debido a que no son la capa final. Ante este problema, se construyó una solución llamada backpropagation (en español, retro propagación), en la que se distinguen dos fases claramente diferenciadas:

1) Hacia delante

En esta primera fase, los pesos obtenidos en la iteración anterior son usados para computar el valor de salida de cada neurona en la red. Esta computación se da en orden, empezando por las neuronas de entrada y siguiendo un estricto orden hasta las neuronas de salida.

2) Hacia atrás:

En la segunda fase, y siguiendo un estricto orden desde las neuronas de salida hasta las de entrada, la fórmula de actualización de pesos se vuelve a ejecutar. Esta segunda pasada, elemento que aporta la retro propagación, ayuda a usar los errores de las neuronas en la capa n+1 para estimar los errores de las neuronas anteriores, de la capa n.

A continuación se hace un análisis de las características de las redes neuronales, así como sus ventajas y desventajas:

1) Las redes neuronales con una capa oculta al menos son unos aproximadores universales, lo que significa que se pueden utilizar para aproximar cualquier función a su óptimo. Esto las hace realmente útiles para hacer una primera aproximación a cualquier problema, pero por otra parte es frecuente caer en overfitting con ellas debido al intento de mejorar en exceso el problema.
2) Las redes neuronales artificiales son esencialmente buenas cuando se tenga la duda de si alguna variable no es demasiado interesante, puesto que ellas mismas reajustarán los pesos en función de la relación que tengan los datos, por lo que no hay que preocuparse de ello. Sin embargo, esto no indica que no se deba de hacer alguna técnica de reducción de la dimensionalidad previamente, puesto que a mayor dimensionalidad del problema, más neuronas en la capa de entrada tendrá que haber y mayor coste computacional habrá en el problema.
3) Las redes neuronales son muy sensibles hacia la presencia de ruido, especialmente en el set de datos de entrenamiento. Debido a esto, el planteamiento de soluciones es algo importante, donde se puede introducir un decay para reducir el overfitting o usar un set de test para comprobar el estado de la red neuronal tras el entrenamiento.
4) El método del gradiente descendente normalmente dirige el modelo hacia un mínimo local. Esto deberá ser tomado en cuenta siempre, y entrenar el algoritmo sucesivas veces comprobando el mejor resultado en el test siempre es una buena práctica en caso de ser computacionalmente posible.
5) En relación con el apartado anterior, el entrenamiento de una red neuronal artificial puede conllevar un gran tiempo de procesamiento, especialmente cuando la cantidad de datos y/o dimensiones es muy elevada. Esto también puede verse agravado por el número de nodos intermedios, puesto que se tendrán que hacer más cálculos por cada pasada en la red neuronal.

2.3.4.7.2 Deep Learning

El deep learning, o aprendizaje profundo, consiste en una técnica de aprendizaje automático que permite a los modelos neuronales multicapa de machine learning aprender representaciones de datos con un nivel de abstracción mucho mayor que con las redes neuronales convencionales. Las dos principales diferencias con las redes neuronales multicapa son las siguientes:

1) Las redes de Deep Learning suelen tener varias capas ocultas, debido a que cada capa se suele especializar en la obtención o transformación de una cierta información.
2) Las redes de Deep Learning son redes que se basan en la extracción de características, ideología contraria a las redes neuronales multicapa que se basan en la transformación matemática de datos mediante funciones.

Su principal baza, siguiendo con el punto 2, es que la extracción de características del dataset es más potente que la extracción con métodos tradicionales, incluso cuando estos han sido refinados para el problema concreto con expertos humanos. Los modelos de deep learning son utilizados hoy en día en numerosos ámbitos, tan diversos como el reconocimiento de voz, reconocimiento visual, genómica e ingeniería genética o creación de imágenes.

Para la creación y el correcto funcionamiento de un modelo de deep learning hace falta un dataset ciertamente grande, donde la red neuronal profunda pueda aprender la estructura del mismo y las relaciones entre las observaciones. Esto se suele realizar mediante una técnica que

se ha visto anteriormente, llamada backpropagation. Otra razón por la que las redes deep learning deben usar grandes datasets es debido a que normalmente estas redes tienen una gran cantidad de neuronas divididas en más de una capa oculta para conseguir funciones complejas que permitan obtener resultados precisos y evitar el ruido, pero para ello hace falta hacer numerosas actualizaciones de los pesos de la red.

2.3.5 Visualización

La visualización es una de las partes fundamentales de cualquier proyecto relacionado con datos, puesto que el entendimiento de las conclusiones depende mayoritariamente de los gráficos que se muestren al público interesado. De este modo, las visualizaciones pueden ser una fuente de profundo entendimiento, pero también pueden ser una fuente de confusión.

Uno de los elementos más importantes que hay que tener en cuenta es el público al que se dirige la presentación, tanto por el registro lingüístico a usar como por los gráficos a utilizar, siendo ambos factores fundamentales.

Se debe distinguir entre gráficos para presentación y gráficos para exploración de datos:

- Gráficos para presentación

Estos gráficos suelen ser en su mayoría estáticos y únicos. Normalmente deberán de tener una alta calidad de imagen, y es recomendable que haya una leyenda que explique las variables para la total comprensión del oyente.

Estos gráficos deben de ofrecer una visión convincente de los resultados a los que se ha llegado, e ir fuertemente correlacionados al resto de la presentación para que el oyente no se pierda.

- Gráficos para la exploración

Estos gráficos se usan para la búsqueda rápida de resultados. En ellos prima la rapidez con la que se obtengan para ver los resultados, y no tanto que sean perfectamente precisos ni de alta calidad. Si el analista comprende en profundidad las variables, no es necesario que incluyan leyenda ni ningún tipo de elemento aclaratorio, puesto que su ciclo de vida será extremadamente corto.

Es importante tener también en cuenta que no todos los tipos de gráficos, ni los colores, ni incluso las dimensiones son elementos que se tengan que elegir al azar. Hay una serie de parámetros y decisiones que se deben de tener siempre en cuenta a la hora de hacer un gráfico, y son las siguientes:

1) Escalas
2) Ordenamiento
3) Color
4) Tamaño y Ratio
5) Tipo de gráfico

A continuación, se explican detalladamente cada uno de estos elementos:

- Escalas

Escoger la escala a utilizar en un gráfico cuando las variables son categóricas es una decisión complicada. Tanto, que incluso software de alta calidad a veces falla a la hora de elegir la escala de visualización, por lo que esta tarea, aunque sea a veces fácil de discernir, no es siempre sencilla. En caso de que las variables sean continuas, la decisión se complica exponencialmente, puesto que habrá que elegir además unas ciertas divisiones y finalizaciones.

Una práctica muy extendida es la de coger como escala los extremos de los datos, es decir, el mínimo que obtienen y el máximo, pero esto es una práctica incorrecta, puesto que algunos puntos, barras o líneas estarán sobre los ejes y no se apreciarán correctamente. Además, si una serie de elementos no tienen como mínimo el cero, se puede estar incurriendo en un caso de falseo visual de los datos al no poner un mínimo absoluto como referencia. Todos estos elementos se pueden ver en la figura a continuación:

2-15. Mismos datos, diferentes escalas

De esta manera, a la hora de escoger escalas es importante no engañar con los datos, puesto que como se puede observar en la figura anterior, en la presentación de la izquierda parece un aumento de los tipos de interés extremo, cuando puesto en perspectiva respecto al cero podemos ver que el aumento no es para nada preocupante.

- Ordenamiento

Cuando más de una variable es representada, caso que se da en la gran mayoría de las ocasiones, la forma en la que se ordenen estas variables puede marcar una gran diferencia. Debido a esto, y para no alterar los datos, se suelen ordenar los mismos mediante criterio alfabético, o si corresponde, mediante criterio geográfico o agrupaciones relevantes.

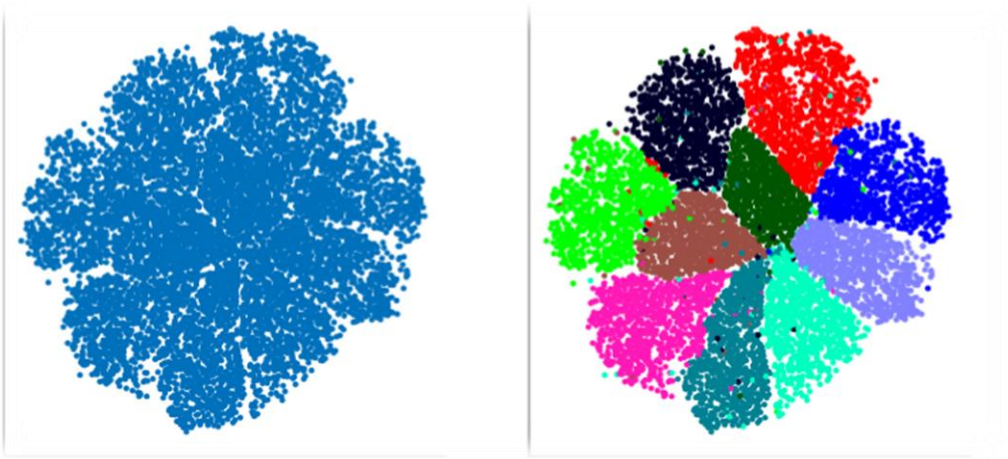
- Color

El color, aunque parezca una banalidad, es uno de los elementos más importantes a decidir. Y esto es porque no todos los colores se perciben igual en el cerebro, no todos son igualmente fáciles de ver, especialmente en conjunción con otros (como, por ejemplo, el de fondo), y

también hay que tener en cuenta que puede haber personas con enfermedades relacionadas con la percepción del color, como el daltonismo.

Además, se ha demostrado científicamente que los gráficos cuyos elementos más importantes son el color y el tamaño son de los más complicados de interpretar por la mente humana, especialmente si los tamaños o los colores son ciertamente similares entre sí.

Ante todos estos problemas, el color destaca por que, si se consigue acertar en su elección, puede ser un elemento diferencial a la hora de crear gráficos. Esto destaca especialmente en clusters, donde todos los datos suelen ser representados con la misma forma y tamaño, y el color se plantea como elemento diferencial entre el entendimiento y la ignorancia del significado del mismo. Esto se puede apreciar en la figura siguiente:



2-16. Clustering con Colores vs Sin Colores

En resumen, a la hora de crear gráficos uno de los elementos más a tener en cuenta es el color, puesto que puede marcar la diferencia a la hora de entender o no entender los datos. Además del color, habrá que tener en cuenta el tamaño del elemento representado, puesto que si dos elementos distintos poseen el mismo color y un tamaño similar muchos oyentes interpretarán que pertenecen al mismo grupo o clase.

- Tamaño y Ratio

Los gráficos deben de ser lo suficientemente grandes como para que el oyente pueda observarlos sin problema ninguno, a la vez que no deben pasarse de grandes debido a que se desaprovecha espacio.

Si se quieren añadir marcos a los gráficos, hay que tener en cuenta de que también ocupan espacio de la visualización, por lo que la recomendación general suele ser añadirlos en caso de querer hacer una separación de las representaciones.

Finalmente, el ratio es quizás el elemento más complicado de este apartado, puesto que tiene un gran impacto en cómo se visualizará. De este modo, un aumento del eje y conllevará una dramatización de cualquier cambio, mientras que una elongación del eje x muestra un cambio más gradual en las series temporales. Sea como fuere, el ratio es un parámetro muy delicado y se debe de actuar con cautela a la hora de elegirlo.

- Tipo de gráfico

Una vez se han tenido las recomendaciones anteriores en cuenta se tiene que elegir el gráfico que representará la información deseada. A continuación, se explican algunos de los más utilizados:

1) Gráfico de Barras

El gráfico de barras es el gráfico más conocido y usado mundialmente. Este destaca por representar los valores de los datos a través de longitudes de barras, dando igual la anchura de las mismas. Puede ser mejorado mediante la aplicación de diferentes colores a cada barra, o aplicando colores según variables que pertenezcan a un grupo superior.

Este gráfico es ideal para cuando se quieren representar datos con variables discretas, como por ejemplo con frecuencias en las que se da un determinado evento. Es muy importante tener en cuenta en este gráfico el parámetro del ratio, puesto que este es de los gráficos más sensibles al cambio si las barras se sitúan en posición vertical y se amplía la componente y.

2) Gráfico de sectores

El gráfico de sectores es un gráfico con forma circular donde los datos se dividen proporcionalmente en formas triangulares, por lo que recibe informalmente nombres como “gráfico de quesitos” o “gráfico de tarta”.

El gráfico de sectores tiene una fuerte oposición entre los expertos, debido a que incumple el parámetro del color-tamaño que se expuso previamente. Dos franjas separadas pueden tener tamaños distintos pero similares, y no se apreciará bien esta diferencia. Debido a ello, para subsanarlo, normalmente se añade el porcentaje o el valor del elemento que representa cada división.

Estos gráficos son esencialmente usados a la hora de comparar porcentajes o tamaños sin querer obtener demasiada precisión, sino pudiendo ver similitudes y grandes diferencias.

3) Histograma

Usado ampliamente en estadística, el histograma es un gráfico de barras que se usa para representar mediante las mismas una serie de valores, pero al contrario que en el gráfico de barras no presenta la frecuencia de una categoría, sino de un intervalo de valores.

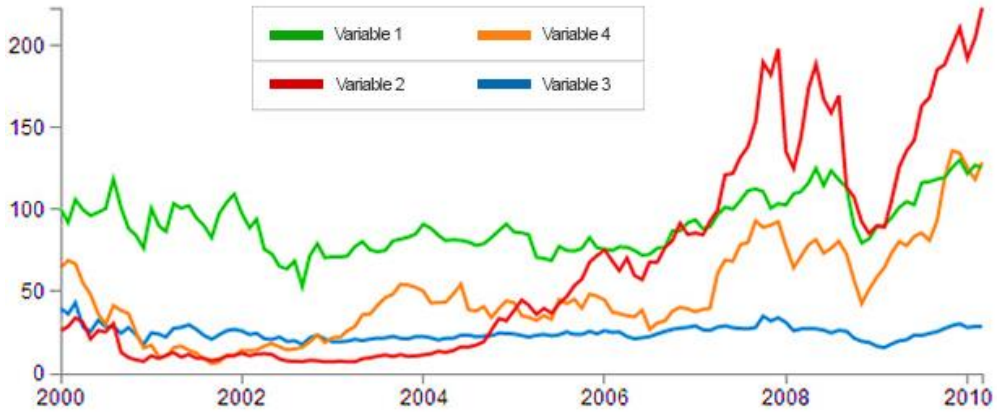
Uno de los elementos que se deben de evitar en los histogramas es el representar intervalos de diferente tamaño, a excepción de los que van hasta infinito en los extremos. Esto es debido a que puede llevar a confusión y es una alteración de la representación de los datos y del tamaño de los grupos.

A veces, este gráfico suele incorporar una línea ascendente sobre él que indica la frecuencia acumulada en cada intervalo.

4) Gráfico de líneas

Los gráficos de líneas son representaciones que sirven para observar la evolución de una o más variables conforme al paso del tiempo. Para hacer esto, en el eje horizontal se muestran una serie de fechas, y en el eje vertical se muestran los valores que pueden tomar las variables.

Con esta disposición, se puntúa cada uno de los valores de cada variable para cada fecha, y se juntan con una línea los puntos que pertenecen a cada variable.



2-17. Ejemplo de Gráfico de Líneas

Este gráfico tiene como principal ventaja su facilidad de interpretación, puesto que muestra a lo largo del tiempo las mejoras y empeoramientos de cada variable. Además, si la última fecha pertenece a la actualidad, se puede observar qué variables están por encima del resto para el valor elegido en el eje vertical.

Una de sus principales desventajas radica en la posibilidad de que haya demasiadas variables. En ese caso, el gráfico puede no quedar claro a simple vista para el lector u oyente.

5) Gráfico de dispersión o scatterplot

El scatterplot es un gráfico conocido como representador de todos los elementos de un dataset en los ejes x e y.

De esta forma, el scatterplot es un gráfico muy interesante para la búsqueda de correlaciones entre variables, puesto que se puede ver para dos dimensiones la tendencia que poseen, en caso de que posean alguna. Sobre este gráfico suele incluirse la recta de correlación, junto con la variable R^2 para mostrar el índice de correlación, que fluctuará entre 0 y 1.

También se puede observar el nivel de dispersión de los datos con este gráfico.

Una de las principales limitaciones que tiene esta representación es la dificultad de interpretación en caso de llevarlo a tres dimensiones, y la imposibilidad de visualización en caso de llegar a 4 dimensiones o más. Debido a esto, este gráfico suele estar limitado a la búsqueda de correlaciones entre variables de dos a dos.

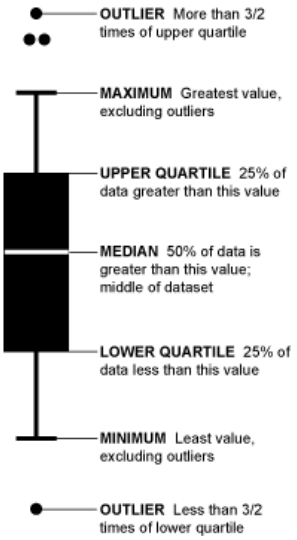
6) Gráfico de cajas o boxplot

El gráfico de cajas, también conocido como boxplot, es otro sistema de visualización usado muy frecuentemente en ámbitos estadísticos. Este gráfico permite hacer una representación de los estadísticos principales de cada variable, de tal manera que se pueden llegar a representar:

- Media
- Mediana
- Cuartiles
- Bigotes

Especial mención merece este último elemento, pues consiste en la representación del valor al que pueden llegar los datos de dicha variable para no considerarse outliers, tanto por encima de la media como por debajo.

Esta representación es realmente útil a la hora de comparar valores y dispersión de variables, y suele ser bastante usada en los análisis exploratorios.



2-18. Explicación de BoxPlot

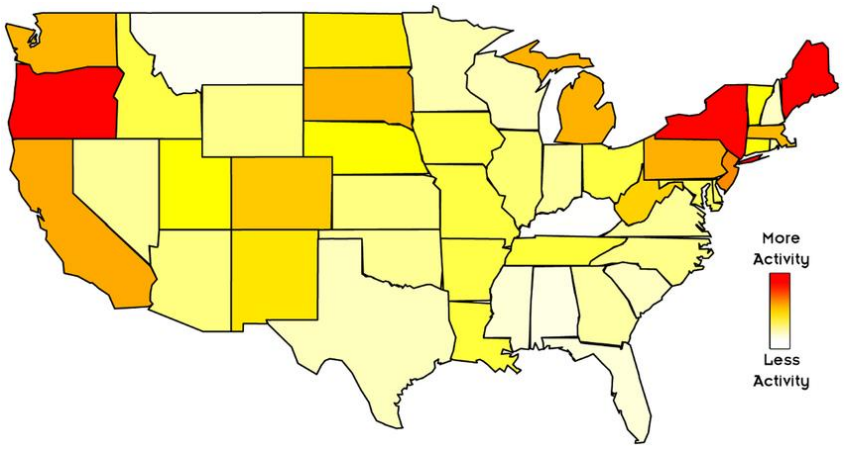
La desventaja que posee esta representación es que la persona a la que se le explica debe de tener una formación estadística para poder entenderlo y comprenderlo en profundidad, por lo que suele ser más un gráfico de exploración que de presentación.

7) Cartograma

El cartograma es un tipo de representación de datos basado en mapa, donde cada país o región de interés posee un valor, normalmente representado con escalas de color acompañadas de una leyenda para su aclaración.

Una de las ventajas de estos cartogramas es que, además de jugar con el color, se puede trabajar con la forma de las regiones deseadas, de tal forma que se puede distorsionar el tamaño de las mismas para representar una cierta variable. Aunque esta práctica es ampliamente usada, hay que tener cuidado porque muchas veces desvirtuará en exceso el mapa e incluso algunas regiones pueden acabar desaparecidas, por lo que no se verá el color de la variable principal, además de que dificultan el entendimiento inmediato de la representación.

Una de las principales desventajas que tiene este tipo de representaciones es la de que suelen usar escalas de color, y los seres humanos, como se ha comentado anteriormente, somos seres que nos cuesta distinguir entre colores parecidos. De esta forma, dos países con valores similares para una cierta variable serán difícilmente diferenciables. Esta desventaja desaparece si se representa un número de colores pequeño relacionados con valores discretos.



2-19. Ejemplo de Cartograma

La conclusión de este apartado de visualización radica en darse cuenta de que la elección de la representación de los datos no es banal, puesto que son el método de exposición de los mismos hacia el oyente o lector interesado. Así, se deben tener en cuenta numerosos factores que pueden dificultar la interpretación de los gráficos, como el ratio o el color, y elegir correctamente la representación para poder alcanzar un nivel de entendimiento y precisión adecuados, ya que no sirve de nada obtener conclusiones importantes si no se transmiten de una manera correcta.

3. REST

Roy Fielding, uno de los padres de la arquitectura HTTP, inventó esta arquitectura en el año 2000, en su libro “Architectural Styles and the Design of Network Based Software Architectures”. Actualmente, REST es ampliamente usado en la construcción de todo tipo de aplicaciones.

Es muy raro llegar a ver un proyecto donde no haya al menos una API REST para el intercambio de datos con el servidor. Es una arquitectura del lado del servidor, y se podría definir como una interfaz, totalmente compatible dentro del protocolo HTTP, que se utiliza para obtener, modificar, eliminar o introducir datos, o incluso hacer operaciones con dichos datos. Normalmente, REST trabaja con dos tipos de estructuras de datos: XML y JSON.

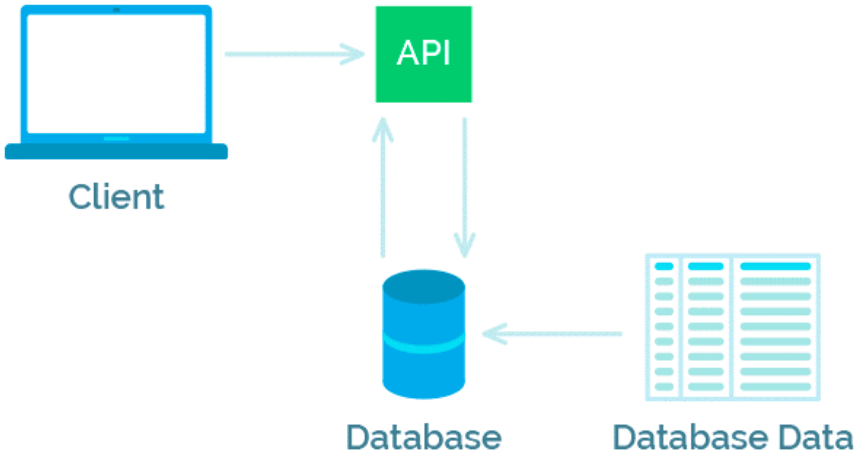
REST tiene una serie de características que lo hacen único, que se pueden ver a continuación:

1) Sin Estado: No guarda el estado ni el contexto de la operación, sino que cada petición lleva unas cabeceras y un cuerpo con toda la información necesaria para hacer la petición.

2) Operaciones HTTP: Como se dijo anteriormente, REST se basa en el protocolo HTTP, de tal manera que las operaciones que se pueden hacer con esta arquitectura son las mismas:
    a. POST: Creación
    b. GET: Obtención
    c. PUT: Edición
    d. DELETE: Eliminación

    Aunque existan las cuatro, las dos más utilizadas son GET y POST, debido a que con esas dos suele ser suficiente para hacer las cuatro operaciones. En caso de que pueda haber sobrecarga de estas dos peticiones, es una buena práctica usar las cuatro.

3) URI como entrada: Para hacer llamadas a una API REST, es necesario siempre llamar a una URI. Cada recurso (operación) de la API REST está identificada con una URI, que debe ser única dentro del conjunto de recursos de la misma operación HTTP, y mediante el envío de la cabecera y el cuerpo a la misma se efectuarán las operaciones.



3-1. Estructura de una Petición GET en API REST

A continuación, se explicará el funcionamiento básico de la estructura REST:

Como se puede apreciar en la imagen 3-1, el cliente envía una petición al servidor que contiene la API REST. Esta petición, tal como se explicó anteriormente, puede ser enviada o en XML o en JSON. Una vez recibida la petición, que en este caso es de GET, la API se pone en contacto con la base de datos mediante el protocolo HTTP, que puede estar en el mismo servidor u en otro. En ese momento, la base de datos ejecuta la petición deseada, y devuelve el resultado a la API REST. La API lo parsea, y lo devuelve en formato XML o JSON al cliente, donde el frontend lo modificará para enseñárselo al usuario de una manera "amigable".

En este trabajo, la base de datos a la que se va a hacer la petición es una de las bases de datos No SQL más famosas del mundo: MongoDB. Para la creación de la API REST, se usará un framework basado en JavaScript bastante moderno, llamado NodeJS. A continuación, estas dos tecnologías serán desarrolladas en mayor profundidad para la comprensión de las mismas.

3.1 MongoDB

El nombre de MongoDB proviene del término anglosajón "humongous", que tiene como traducción "enorme". Este término viene a referirse a las grandes cantidades de datos que esta base de datos puede almacenar.

MongoDB es una base de datos No SQL, orientada a documentos. Esto significa que los datos se guardan en una serie de "líneas" consistentes en un conjunto de clave – valor, conocidas como documentos. Los documentos se guardan en una serie de "tablas" de documentos similares, que se conocen como colecciones. Una base de datos será un conjunto de una o más colecciones.

Debido a estas características, las bases de datos No SQL, y entre ellas MongoDB, son unas bases de datos muy cómodas a la hora de hacer cambios, puesto que no es necesario remodelar la base de datos entera para añadir el soporte de nuevos campos.

Otras características muy interesantes de MongoDB son la escalabilidad y la alta disponibilidad de la que dispone. Es fácilmente auto escalable y replicable en servidor, de tal manera que la tolerancia a fallos es muy alta.

3-2. Logo de MongoDB actual

Algunas de sus características más importantes son:

1) Esquemas dinámicos: Como se ha comentado anteriormente, MongoDB es muy cómoda a la hora de hacer cambios, puesto que no afecta a la estructura de la base de datos ni al resto de los datos. Debido a ello, es una base de datos ideal cuando los requerimientos de una aplicación pueden cambiar.

2) Inteligencia operacional: MongoDB posee un sistema interno de agregación y Map Reduce que permite obtener conocimiento en tiempo real para las aplicaciones, por lo que en algunos aspectos mejora a otras tecnologías como Hadoop o las aplicaciones antiguas y tradicionales de Business Intelligence.

3) Flexibilidad en la implementación: MongoDB fue concebida para ser usada en arquitecturas Cloud especialmente. Las peticiones a la base de datos son robustas y aseguran un buen rendimiento.

4) Escalado simple: Otra de las características para las que se concibió MongoDB es para ser escalada en múltiples servidores sin demasiadas trabas. Así, si los datos crecen las organizaciones pueden añadir más nodos a otros clusters, y MongoDB balanceará los datos de forma nativa entre todos ellos.

¿Cuándo se debe usar MongoDB?

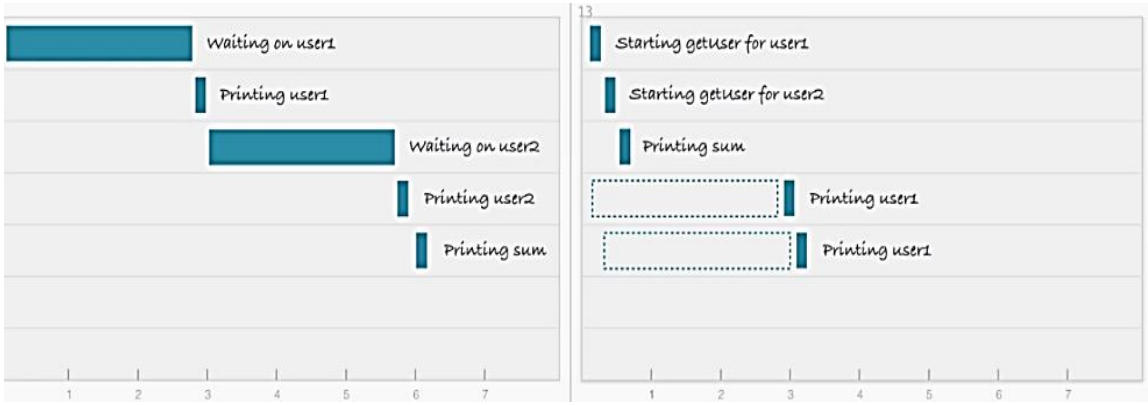
1) Cuando se desee tener analíticas en tiempo real, o se tenga esquemas complejos.
2) Cuando se necesite una latencia pequeña, alta disponibilidad y posibilidad de escalado.
3) Cuando se deseen poder hacer pequeños pero importantes cambios en la base de datos sin necesidad de cambiar toda su estructura.

3.2 NodeJS

Se podría definir NodeJS como un entorno de JavaScript, lo que significa que incluye todos los elementos necesarios para poder ejecutar un programa escrito en dicho lenguaje.

NodeJS fue inventado cuando se quiso cambiar la filosofía de ejecución de JavaScript. De este modo, NodeJS se concibió para poderse ejecutar en una máquina como una aplicación en sí en vez de en un buscador web, como ocurre con JavaScript.

Como se comentó anteriormente, una API REST tiene que ser un programa conductor de entradas y salidas. NodeJS cumple dicha función, puesto que es capaz de cumplir las peticiones HTTP que se le hagan con órdenes tanto de input como de output. Uno de los elementos más importantes que posee NodeJS respecto a otros lenguajes de APIs es el hecho de que su entrada salida es no bloqueante a pesar de ser de procesamiento mononúcleo. Esto lo que significa es que, aunque dos usuarios hagan petición, las dos peticiones se pueden iniciar al mismo tiempo, haciendo que una petición no tenga que esperar a la finalización de una petición anterior. Esto se puede observar claramente en la imagen siguiente:



3-3. Entrada/Salida Bloqueante Vs Entrada Salida no Bloqueante

Como se puede apreciar, en la imagen de la izquierda hasta que el usuario 1 no ha recibido la respuesta a su petición el usuario 2 no inicia su petición al sistema. En cambio, en la imagen de la derecha, la entrada/salida no es bloqueante, lo que significa que ante dos peticiones muy juntas, el sistema las ejecutará "en paralelo" y según se vayan teniendo las salidas se irán entregando a los dispositivos correspondientes. Esta es una de las principales ventajas que aporta NodeJS sobre otros lenguajes y frameworks de desarrollo backend.

Finalmente, es importante destacar el papel que hace NPM en el desarrollo de las aplicaciones NodeJS.

NPM, siglas de "Node Package Manager", es un manejador de un conjunto de librerías creadas por la comunidad que son capaces de resolver la mayoría de los problemas que se pueden presentar a la hora de desarrollar una aplicación NodeJS. Los comandos que se ejecuten con este manejador siempre empezarán por "npm", seguido de un verbo que indicará la acción a ejecutar. Posteriormente, podrán ir una serie de parámetros.

4. Resultados Obtenidos y Conclusiones Finales

4.1 Resultados Obtenidos

En las siguientes páginas se exponen los resultados obtenidos tras la realización del proceso de Data Science, así como las conclusiones a las que se ha llegado en este trabajo.

- **Análisis Exploratorio y Estadístico**

  En el análisis exploratorio se han examinado numerosos ámbitos del extenso dataset. Vayamos analizándolos por orden:

  En primer lugar se han obtenido todas las estadísticas del dataset. Debido a la alta dimensionalidad del mismo, en la siguiente figura se muestra un extracto de los datos obtenidos:

```
t2_inhibitorKills t2_baronKills    t2_dragonKills  t2_riftHeraldKills
Min.   : 0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
Median : 0.0000   Median :0.0000   Median :1.000   Median :0.0000
Mean   : 0.9851   Mean   :0.4145   Mean   :1.404   Mean   :0.2401
3rd Qu.: 2.0000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:0.0000
Max.   :10.0000   Max.   :4.0000   Max.   :6.000   Max.   :1.0000
```

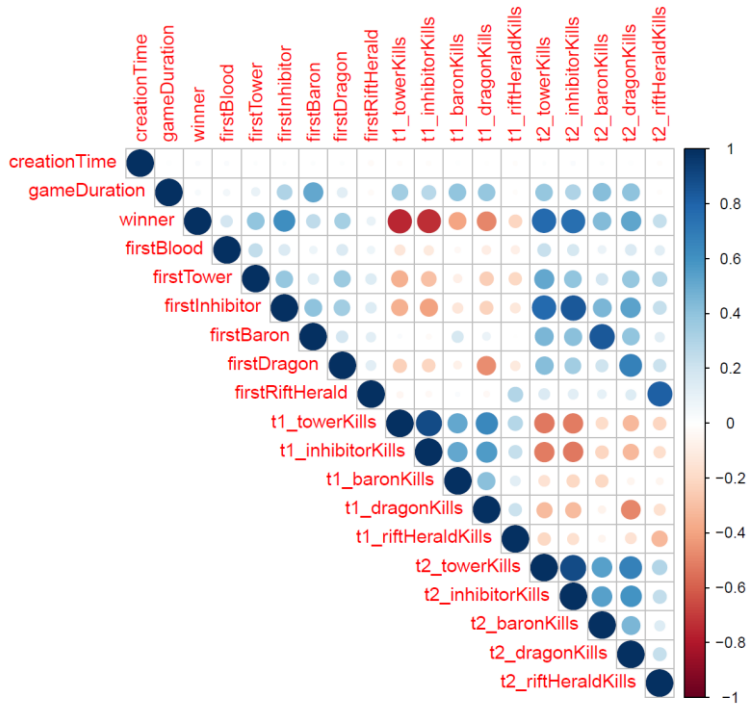
4-1. Extracto Estadísticas Obtenidas

  Analicemos este pequeño extracto:

  - Inhibidores: El número de inhibidores mínimo que ha derribado el equipo 2 ha sido de 0, lo cual significa que no han derribado ninguno. Como máximo 10, lo que significa que la partida se ha alargado hasta la regeneración de inhibidores varias veces. La mediana está en cero, por lo que existen más partidas en las que no se haya derribado inhibidor que en las que sí, lo cual se sigue corroborando con la media.

- Baron Kills: La partida en la que menos barones se han derrotado ha sido con resultado 0, mientras que en la que más se han derrotado hasta 4 barones. La media y la mediana dan conclusiones similares a los inhibidores.
- Dragon Kills: En esta variable es interesante que la mediana se encuentra en 1, por lo que al menos el punto medio de las variables ordenadas es que en dicha partida se derrotó a un dragón. El tercer cuartil (75% de las partidas) han derrotado dos dragones o menos.
- Heraldo: Como se verá en estadísticas posteriores, el heraldo prácticamente no es derrotado en la partida y las estadísticas lo corroboran. Además, demuestro que los datos no son incorrectos, porque el máximo de heraldos derrotados es de 1, cuando el límite lógico de la partida corresponde a dicho valor.

Si se continúa con la estadística, se ha obtenido la matriz de correlación de las variables, con el siguiente resultado:



4-2. CorrPlot de las Variables

En esta matriz de correlación se pueden observar algunas muy destacadas:

- Podemos apreciar las siguientes correlaciones sobre destruir torres:
  • Destruir más torres tiene una relación directa con destruir inhibidores
  • Destruir más torres tiene una cierta relación directa con matar dragones
  • Destruir más torres tiene una cierta relación inversa con que el equipo contrario destruya tus torres
  • Destruir más torres tiene una cierta relación con que el equipo contrario mate menos dragones

- Respecto a ganar, obtenemos las siguientes correlaciones:

• Muchas veces, quien destruye el primer inhibidor es el equipo que acaba ganando la partida

• Un aumento en que el equipo 1 destruya más torres tiene que ver con que tengan más posibilidades de victoria. Esto se da con una relación inversa, ya que la victoria del equipo 1 se marca con un 1, la del 2 con un dos, y el aumento de towerkills de t1 implica una bajada en win (1 en vez de dos)

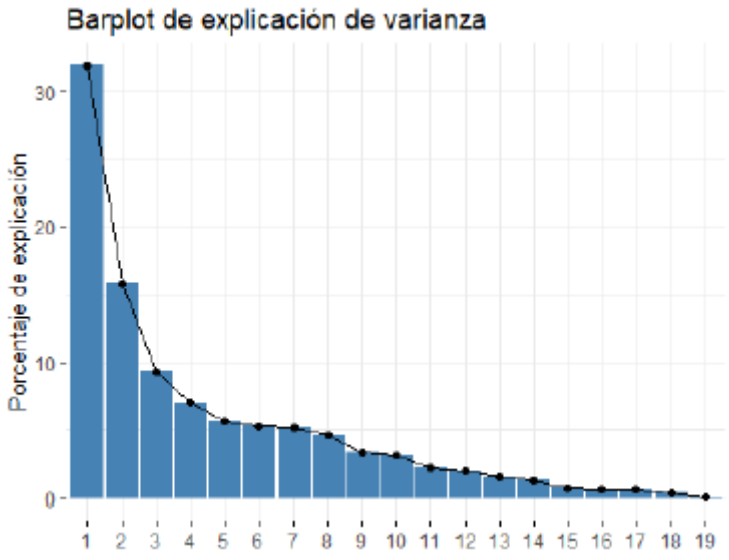
• Lo mismo pasa con la cantidad de inhibidores destruidos, donde es también una relación muy fuerte la que hay.

• Con el equipo 2 pasa lo mismo, lo que pasa es que, por lo explicado anteriormente, en este caso se muestra como relación directa, y de este modo podemos ver las mismas correlaciones de una parte que de otra.

- Respecto a dragones, podemos ver:
  • Es más importante para el equipo 2 hacerse el primer dragón de cara a hacerse más que el que el equipo 1 haga el mismo.

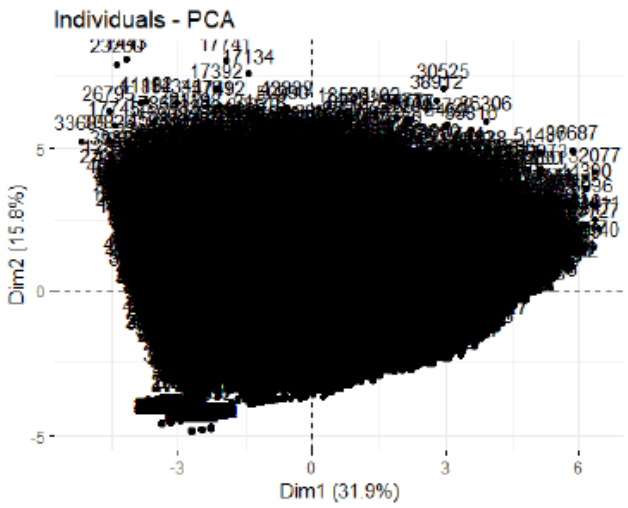
- Finalmente, respecto al momento de la creación de la partida, podemos observar:
  • No tiene ninguna relación el paso del tiempo con el aumento o disminución de ninguna de las variables.

Si se continúa, el screeplot de PCA arroja los siguientes resultados:

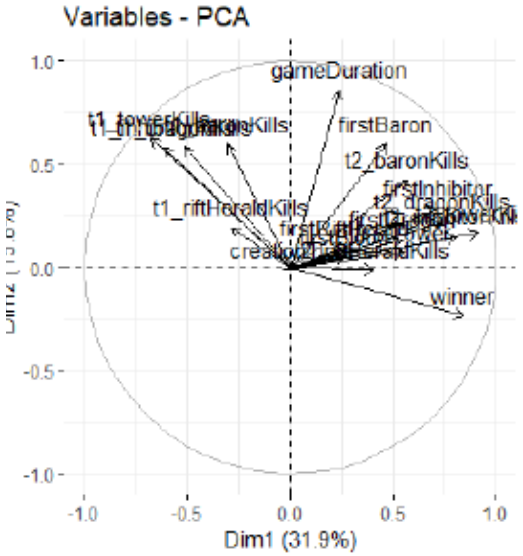


4-3. ScreePlot

Donde se puede observar como una variable posee una grandísima explicación. Este gráfico viene bien entendido junto con las siguientes figuras:



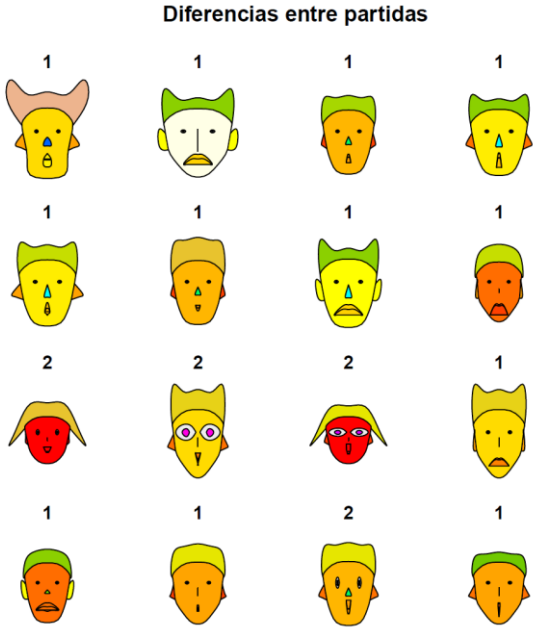
4-5. Observaciones sobre las dos componentes principales



4-4. Variables sobre las dos componentes principales

Como se puede ver, sobre las dos componentes principales (acumulan un 45.5% de explicación) se puede ver como todas las observaciones forman una gran masa de la que, sin colores, es complicado distinguir los grupos, mientras que en la segunda figura se observan las dimensiones del problema, también reflejadas sobre estas dos componentes principales.

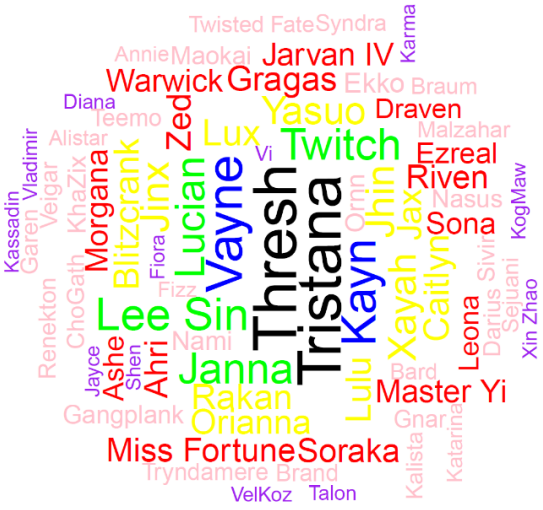
Siguiendo con un análisis exploratorio, las caras de Chernoff son una manera informal de observar si los grupos poseen grandes diferencias. Aquí se plantean las 16 primeras partidas convertidas en caras de Chernoff:



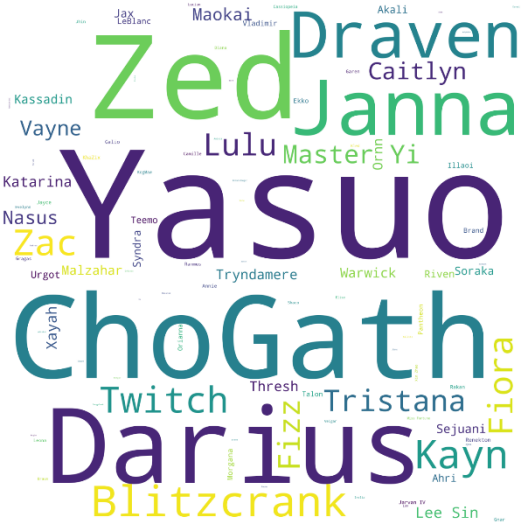
4-6. Caras de Chernoff

Como se puede observar, las partidas ganadas por el equipo 1 son parecidas (colores claros y pelo hacia arriba), mientras que las del equipo 2 por regla general son con colores oscuros y pelo hacia abajo. Se puede comprobar como también existen caras intermedias, por lo que la diferenciación entre partidas se complica.

Posteriormente, se analizaron cuáles son los campeones más jugados y más baneados, puesto que esto tiene que ver con el devenir de la partida. Se obtuvieron los siguientes wordclouds:



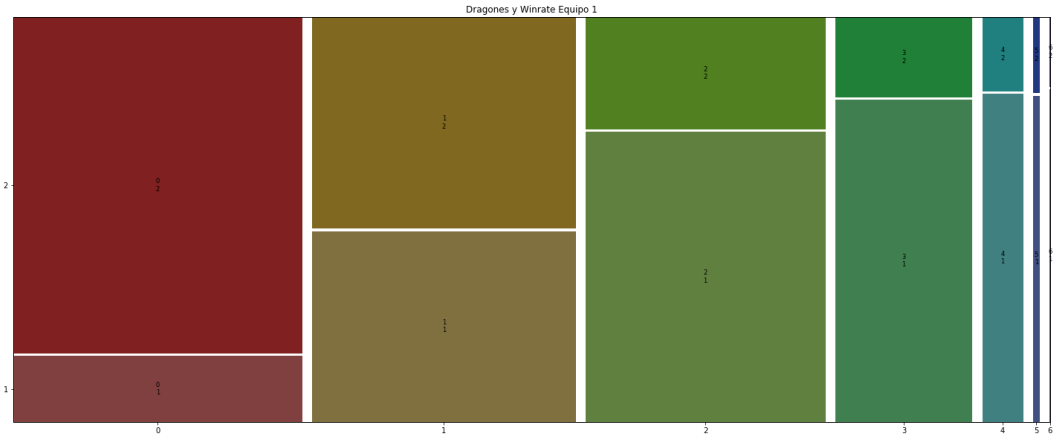
4-8. WordCloud Picks



4-7. Wordcloud Bans

En los wordcloud se puede observar como Thresh y Tristana son los dos campeones más escogidos, mientras que los dos campeones más baneados son Yasuo y Chogath. Esto tiene sentido, puesto que los más jugados lo son porque se les permite jugar.

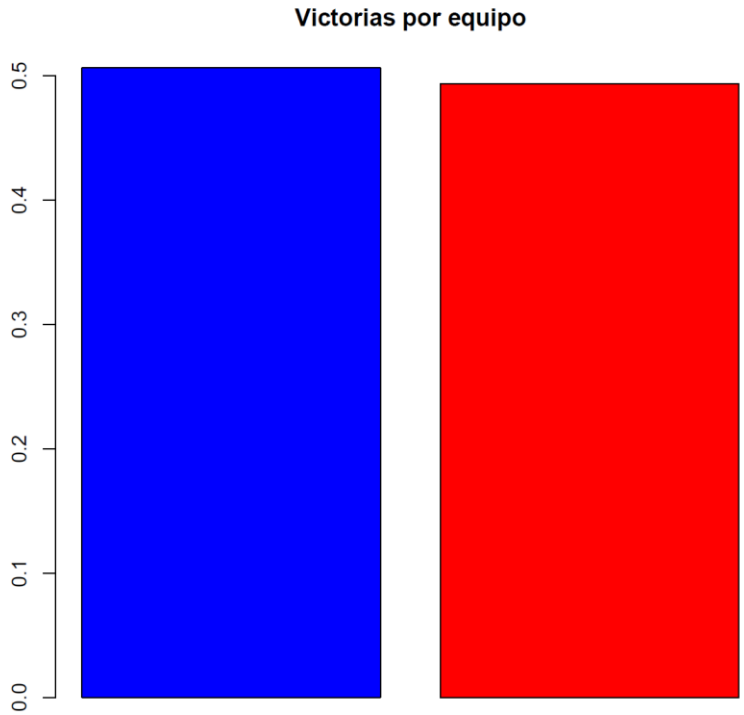
Siguiendo con el análisis, se estudiaron la cantidad de dragones que debe de asesinar un equipo para ganar las partidas, y se obtuvo el siguiente resultado para el equipo 1:



4-9. MosaicPlot Drakes T1

Con este Mosaicplot, que se puede ver a mayor calidad en el repositorio de GitHub, se muestra como a mayor número de dragones matados por el equipo 1, mayor porcentaje de victorias consigue. Se obtuvo también para el equipo 2, donde el gráfico resulta inverso respecto al eje y.

Uno de los mayores mitos en el juego es comentar que jugar en el lado de arriba (el del equipo 2), afecta en el devenir de la partida y se pierden más partidas hacia ese lado. Esto se estudió, obteniendo el siguiente resultado:



4-10. Winrate Por Equipo

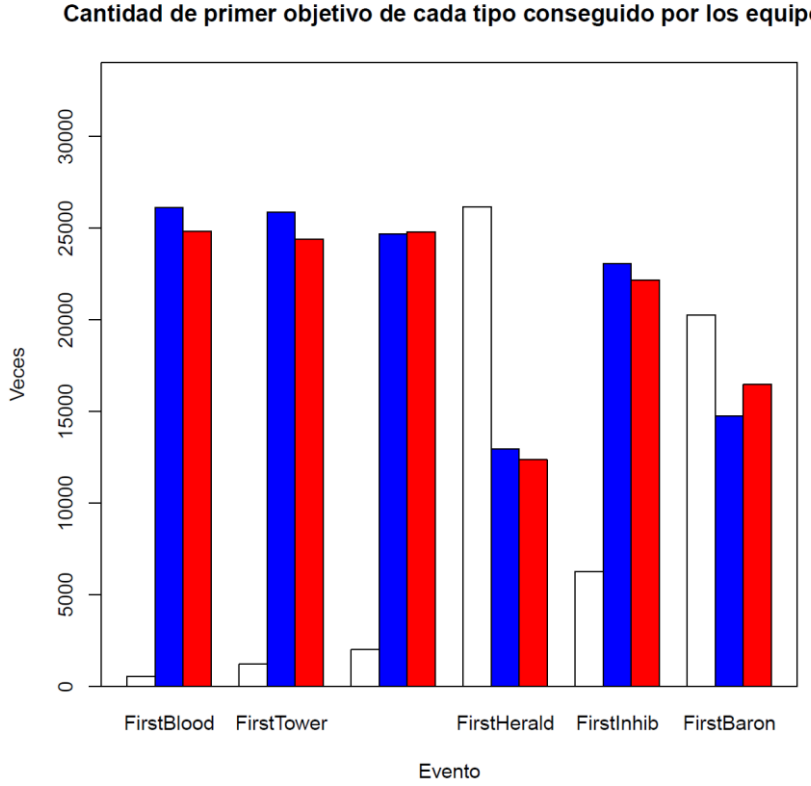
Donde se aprecia que la diferencia es mínima, aunque es cierto que el equipo azul posee algunas victorias más que el equipo rojo.

El estudio de quién conseguía los primeros objetivos también fue objeto de estudio en este trabajo, puesto que quizás algunos objetivos, por su colocación en el mapa o por el devenir de las partidas, están más a favor de un equipo u otro. Se observa en la siguiente figura, que resume todos los objetivos:

Como se puede observar, en los primeros objetivos (firstBlood y firstTower) hay una cierta igualdad, aunque se decanta hacia el lado azul. Posteriormente, FirstDragon suele ser más del equipo rojo, FirstHerald suele no hacerse por ningún equipo, el FirstInhibitor suele caer más del

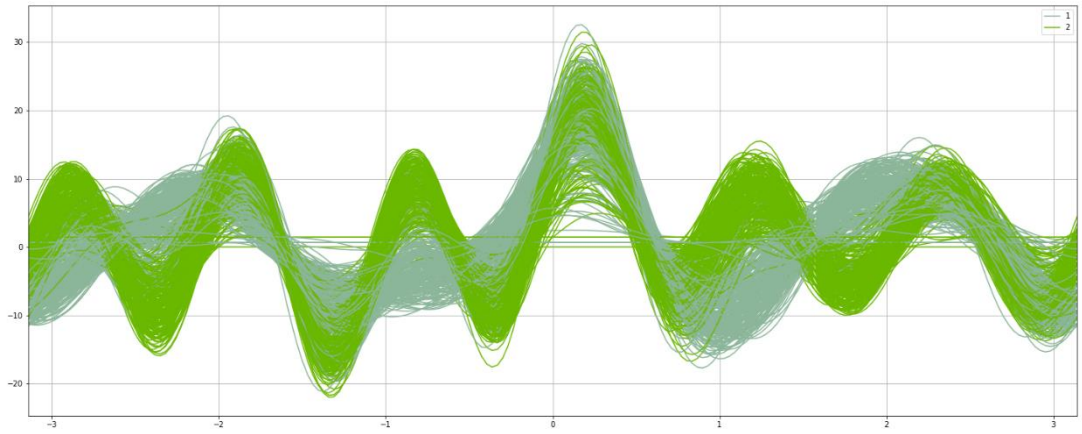
lado azul y el FirstBaron del lado rojo, aunque hay muchas partidas en las que no se asesina al barón.

Esto, en resumen, viene a decir que la partida es un “caos” y que los objetivos, da igual el momento, pueden caer de cualquier lado, estando bastante igualados.



4-11. Cantidad de Primeros Objetivos

Finalmente, para acabar el análisis exploratorio, se realizaron unas curvas de Andrew para observar la complicación que se tendrá a la hora de predecir con Machine Learning, puesto que parece ser por los análisis anteriores que así será:



4-12. Curvas de Andrews

Como se puede apreciar, los dos tipos de partida (ganadas por el T1 y por el T2) son bastante parecidas, con pequeñas diferencias, por lo que la predicción puede resultar un poco complicada. Esta la analizaremos con profundidad a continuación.

- **Métodos de Inteligencia Artificial**

Se han utilizado diversos métodos de inteligencia artificial para la clasificación de las partidas en sus respectivos grupos, obteniendo resultados bastante satisfactorios por regla general tras pasar los datos por un proceso de centrado, escalado, reducción de la dimensionalidad / dispersión de matriz y método holdout.

En los resultados se analizará primero los resultados con el dataset enriquecido con la binarización de los campeones, y posteriormente los resultados obtenidos sin los campeones, solo con el dataset original.

Veamos a ver los resultados obtenidos:

- Redes Neuronales

Las redes neuronales fueron desarrolladas tanto en lenguaje R como en lenguaje Python, obteniendo los siguientes resultados:

Tabla 4-1. Resultado de Redes Neuronales

Neuronas	1	1	2	2	3	**3**	10	10	20	50
Softmax	No	Sí	No	Sí	No	**Sí**	No	No	No	No
Decay	-	-	-	-	-	**-**	0,01	-	0,01	0,001
Resultado Test	96,3%	96,1%	96,2%	96,7%	96,9%	**97%**	90,4%	91,2%	91,4%	91,7%

Como se puede apreciar, el mejor resultado se ha conseguido con 3 neuronas y con softmax activado, con un 97% de acierto total. El aumento de neuronas no ha servido para mejorar el modelo, aunque sí se nota una pequeña mejoría de 10 a 50 neuronas, seguramente a cambio de cometer overfitting en el entrenamiento.

- KNN

Este método ha sido desarrollado, al igual que todos los siguientes, usando el paquete de inteligencia artificial de Python3: Scikit Learn, también conocido como SKLearn.

Con KNN se han obtenido resultados diversos, puesto que se ha probado primero determinando el K a mano y posteriormente realizando una malla. Los resultados son los siguientes:

Tabla 4-2. Resultados KNN con Campeones

K	3	5	7	**Grid (K=5, CV = 3)**
Resultado Test	94,6%	94,8%	94,9%	**95,1%**

Este método también se ha probado eliminando los campeones y las spells, con lo que se reduce la dimensionalidad y se obtienen un resultado de K = 9 con hasta un **95,5% de acierto de media**.

- SVM

Support Vector Machines es el siguiente algoritmo que se probó. Continuando con la estrategia de la malla, se probó con los 3 kernels posibles (lineal, RBF y polinómico), y posteriormente se probó en malla. Los resultados son los siguientes:

Tabla 4-3. Resultados SVM con Campeones

Kernel	Lineal	**RBF**	Polinómico	Malla (RBF)
Resultado Test	95,9%	**97%**	96,8%	97%

- Árbol de Decisión

Se probó la técnica del árbol de decisión antes de Random Forest para poder comparar ambos algoritmos, y los resultados son los siguientes:

Tabla 4-4. Resultados Árbol de Decisión

	Con CV(3)	Sin CV
Con Campeones	96,5%	94,3%
Sin Campeones	**96,6%**	-

- Random Forest

Con el algoritmo de Random Forest se siguió una estrategia similar, primero probando a mano y posteriormente creando una malla de valores para obtener el mejor. Se obtuvieron los siguientes resultados:

Tabla 4-5. Resultados Random Forest

	Con CV (3)
Con Campeones	95,5%
Sin Campeones	**97%**

Como se puede apreciar, es ligeramente mejor el resultado obtenido con Random Forest que con el árbol de decisión.

- Descenso del Gradiente Estocástico

Con este método se obtuvieron los siguientes resultados:

Tabla 4-6. Resultados SGD

	Resultado de Test
Con Campeones	92%
Sin Campeones	**95,9%**

- Naive Bayes

Con el método Naive Bayes (de la probabilidad condicionada), se obtienen estos resultados:

Tabla 4-7. Resultados Naive Bayes

	Resultado de Test
Con Campeones	76,8%
Sin Campeones	**93,6%**

- AdaBoost

Los resultados obtenidos con el método AdaBoost son los siguientes:

Tabla 4-8. Resultados AdaBoost

	Resultado de Test
Con Campeones	91,2%
Sin Campeones	**96,4%**

- Gradient Tree Boosting

Estos son los resultados que fueron obtenidos con Gradient Tree Boosting:

Tabla 4-9. Resultados Gradient Tree Boosting

	Resultado de Test
Con Campeones	**97,4%**
Sin Campeones	97,3%

Este es el único algoritmo en el que mejora la calidad de la predicción teniendo campeones en el dataset.

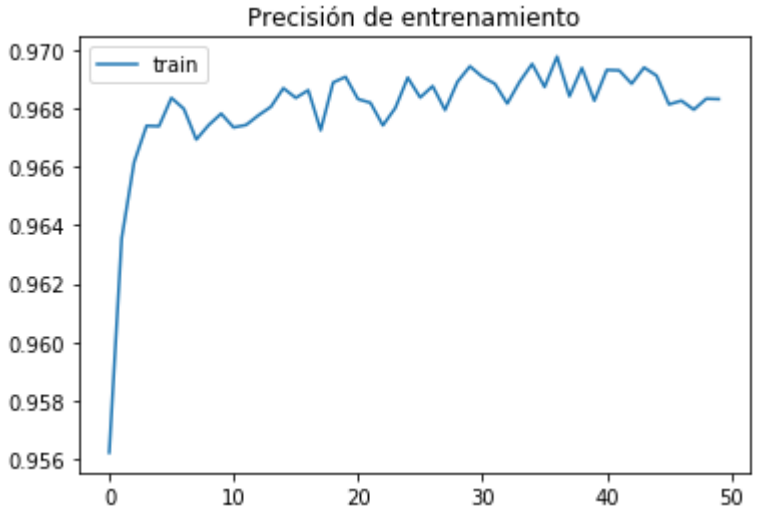
- Deep Learning

El modelo de deep learning ha sido entrenado de una manera supervisada, ya que se poseen las etiquetas del mismo.

Se ha entrenado primero el modelo sin campeones, con una única capa oculta de 12 neuronas, obteniendo un resultado del 96,7% de acierto.

Posteriormente, siguiendo los últimos estudios en Deep Learning, se ha entrenado el modelo con varias capas ocultas para favorecer la "especialización" de cada una de las capas en la extracción de algunas características. Esto se ha hecho creando un modelo con 3 capas ocultas de 8 neuronas cada una, obteniendo un resultado de **96,8% de acierto**, una ligera mejoría.

La precisión del entrenamiento en este último discurrió de la siguiente manera:



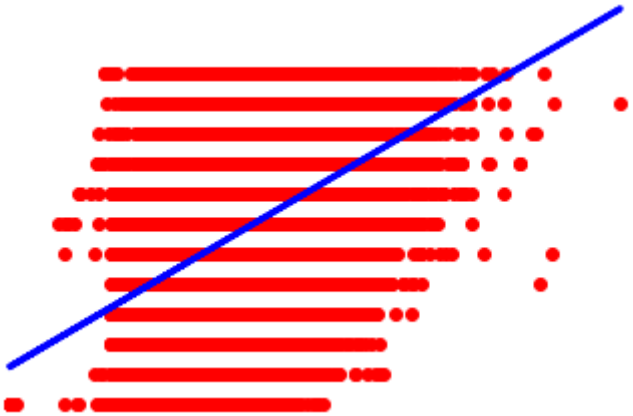
4-13. Precisión de Entrenamiento Deep Learning

Finalmente, se entrenó el modelo con los campeones, obteniendo un paupérrimo resultado de un 50,4% de acierto.

- Regresiones

Para no convertir todo el machine learning en clasificación, se han realizado dos regresiones con muy distintos resultados:

La primera consiste en la regresión entre la duración de la partida y la cantidad de torres derribadas por el equipo 1, obteniendo un valor de R cuadrado de 0,002, lo cual indica que no existe correlación.



4-14. Correlación Torres T1 y Duración Partida

La segunda correlación consiste en la que se obtiene entre las torres derribadas entre uno y otro equipo. En este caso, la correlación es de -0,53, un valor bastante interesante ya que es una correlación inversa (si un equipo tira muchas torres el otro, en condiciones normales, tirará menos), por lo que es bastante probable que sea correcta desde el entendimiento del juego.

Ante los resultados obtenidos, la siguiente tabla muestra la comparación de los resultados de test para su visualización final:

Tabla 4-4-1. Mejores Resultados Finales

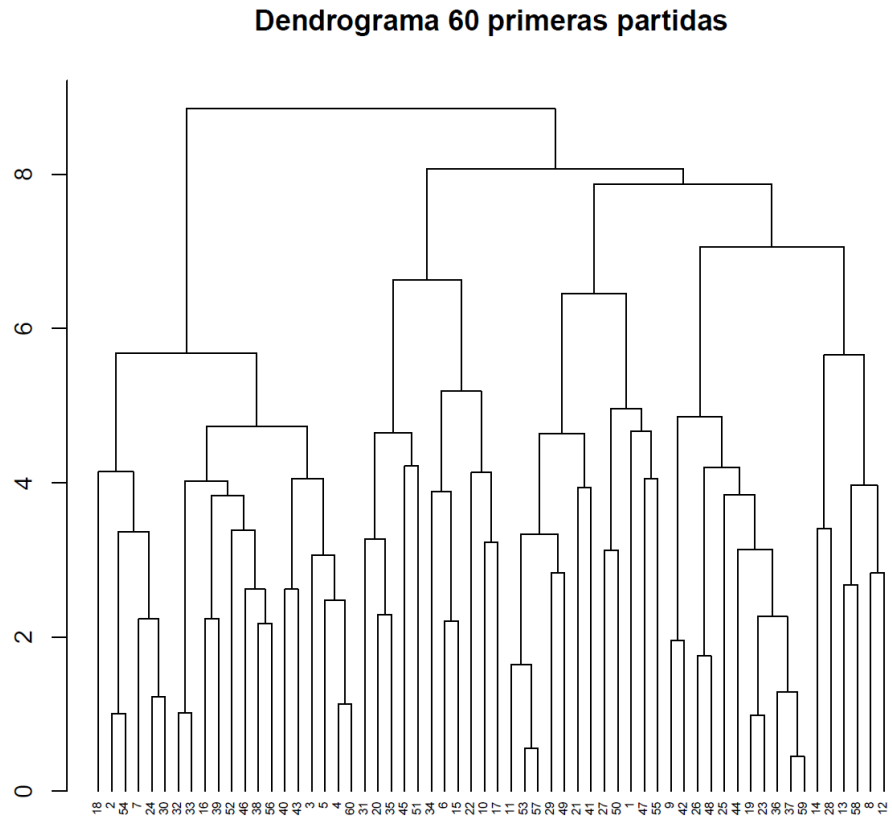
	Sin Campeones	Con Campeones
KNN	95,5%	95,1%
Random Forest	95,5%	**97%**
SVM	**97%**	-
Árbol de Decisión	96,6%	96,5%
SGD	95,9%	92%
Naive Bayes	93,6%	76,8%
AdaBoost	96,4%	91,2%
Gradient Tree Boosting	97,3%	**97,4%**
Redes Neuronales	**97%**	-
Deep Learning	50,4%	96,8%

Como se puede comprobar, en prácticamente todos los algoritmos existe una mejora si no se posee a los campeones binarizados. Esto se puede deber a numerosas causas, entre las que destaco el que la diferencia de Winrate entre campeones es muy baja y por lo tanto seguramente sea mal predictor.

Respecto al algoritmo no supervisado, no se puede obtener una valoración objetiva sobre él ya que funcionan sin la etiqueta “grupo”, de tal manera que los resultados obtenidos no son comparables con la realidad. Teniéndolo en cuenta, sus resultados se publicarán a continuación:

- Dendrograma

Con el dendrograma se ha obtenido el siguiente resultado:



4-15. Dendrograma

Donde se observa como las partidas se dividen en dos grupos, y dentro de uno de los dos grupos hay grandes subdivisiones, mientras que en el otro son más pequeñas.

4.2 Conclusiones

De este trabajo se pueden obtener numerosas conclusiones, las cuales se expondrán a continuación:

1) El League of Legends es uno de los videojuegos más jugados desde hace años en el panorama internacional de los e-sports, y por lo tanto un análisis de sus partidas siempre es una tarea interesante para poder predecir quién será el ganador.

2) Las partidas del League of Legends PUEDEN ser predichas mediante inteligencia artificial a partir de los datos de las mismas, con eficacias que se elevan hasta el 97,4% de acierto.

3) Un problema real de ciencia de datos, y por tanto de machine learning, va a poseer un dataset idílico sobre el que realizar las operaciones, por lo que los cambios sobre el mismo son siempre necesarios. En este caso, aparte de alguna que otra limpieza, se hicieron joins de tablas para enriquecer el dataset y mejorarlo, binarizaciones…

4) En trabajos muy especializados como este, el business understanding es una etapa fundamental puesto que el conocimiento de las variables y sus relaciones es el causante de buscar en la dirección correcta e interpretar de una manera fiel los datos obtenidos.

5) Si el dataset está balanceado y limpio, los algoritmos supervisados más comunes como K Nearest Neighbors, Support Vector Machines y Random Forest (Tree) son algoritmos que funcionan muy bien con este problema, con especial mención a Boosting que obtiene los mejores resultados de todos los algoritmos probados.

4.3 Líneas Futuras, Ampliaciones y Entornos de Aplicación

El actual trabajo tiene una filosofía puramente investigadora y académica, por lo que su aplicación al mundo real es estudiable posteriormente.

En caso de aplicarse, se podrían hacer diversas modificaciones para que, mediante un sistema como Kafka, se enviaran los datos de partidas en directo y se pudiera predecir quién sería el ganador (incluso con porcentajes), de tal manera que se pudiera desarrollar un sistema de predicción en directo de ganadores de las partidas.

Esto se podría usar en apuestas de e-sports o entornos similares, además de como elemento informativo para los espectadores de eventos de League of Legends donde jueguen equipos profesionales.

Otra forma de implementarlo sería como sistema automático de dar información a los jugadores sobre las partidas de los profesionales, o incluso de partidas amateur, de tal manera que los jugadores puedan comprobar las estadísticas fundamentales de las partidas.

Bibliografía

1. **Tan, P., Steinbach, M. y Kumar, V.** (2006). *Introduction to Data Mining.* Boston: Pearson.

2. **Hurwitz, J. y Kirsch, D.** (2018). *Machine Learning.* Hoboken: John Wiley and Sons.

3. **LeCun, Y., Bengio, Y. y Hinton, G**. (28 de Mayo de 2015). Deep Learning. *Nature*. *521*, 436-444.

4. **Chun-Houh, C., Härdle, W. y Unwin, A**. (2008). *Handbook of data visualization.* Leizpig: Springer.

5. Inside Big Data [En Línea]. Obtenido de: https://insidebigdata.com/2014/11/09/ask-data-scientist-importance-exploratory-data-analysis/

6. Aukera [En Línea]. Obtenido de: https://aukera.es/blog/data-science-que-es-y-que-no-es/

7. IBM [En Línea]. Obtenido de: https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.datatools.datamining.doc/c_dp_datapreparationoverview.html

8. Medium.com [En Línea]. Obtenido de: https://medium.freecodecamp.org/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc

9. Medium.com [En Línea]. Obtenido de: https://medium.com/@violante.andre/simple-reinforcement-learning-temporal-difference-learning-e883ea0d65b0

10. BBVA [En Línea]. Obtenido de: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

11. Credera [En Línea]. Obtenido de: https://www.credera.com/blog/technology-insights/java/mongodb-explained-5-minutes-less/

12. Medium [En Línea]. Obtenido de: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5

13. Movistar e-sports [En Línea]. Obtenido de: https://esports.as.com/league-of-legends/League-of-Legends-conceptos-roles-competiciones_0_1121887801.html