

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

[Agradecimientos iii](#_Toc10744922)

[Introducción 1](#_Toc10744923)

[Presentación y Motivación del Trabajo 1](#_Toc10744924)

[Estado del Arte 2](#_Toc10744925)

[1. Capítulo 1: Introducción a League of Legends 3](#_Toc10744926)

[1.1 ¿Cómo se juega? ¿Qué se debe hacer? 3](#_Toc10744927)

[1.2 Objetivos y victoria 5](#_Toc10744928)

[1.2.1 Torres 5](#_Toc10744929)

[1.2.2 Dragones 5](#_Toc10744930)

[1.2.3 Barones y heraldo 6](#_Toc10744931)

[1.2.4 Jungla 7](#_Toc10744932)

[1.2.5 Inhibidores 7](#_Toc10744933)

[1.2.6 Nexos 8](#_Toc10744934)

[2. Capítulo 2: El proceso de Data Science 9](#_Toc10744935)

[2.1 Data Mining Vs Machine Learning Vs Data Science Vs Big Data 9](#_Toc10744936)

[2.1.1 Data Mining 9](#_Toc10744937)

[2.1.2 Machine Learning 11](#_Toc10744938)

[2.1.3 Data Science 11](#_Toc10744939)

[2.1.4 Big Data 12](#_Toc10744940)

[2.2 Antes de hacer Data Mining 14](#_Toc10744941)

[2.2.1 En este trabajo: Obtención de los datos 14](#_Toc10744942)

[2.3 Data Mining 14](#_Toc10744943)

[2.3.1 Pasos previos y preparación de los datos 14](#_Toc10744944)

[2.3.2 En este trabajo: Preparación de los datos 14](#_Toc10744945)

[2.3.3 Análisis Exploratorio o Descriptivo 15](#_Toc10744946)

[2.3.3.1 Resumen de las estadísticas del Dataset 15](#_Toc10744947)

[2.3.4 Machine Learning 18](#_Toc10744948)

[2.3.4.1 ¿Cómo funciona un algoritmo de clasificación en machine learning? 18](#_Toc10744949)

[2.3.4.2 Problemas y soluciones con los clasificadores 19](#_Toc10744950)

[2.3.4.3 Algoritmos Supervisados 22](#_Toc10744951)

[2.3.4.3.1 K Nearest Neighbours (KNN) 22](#_Toc10744952)

[2.3.4.3.2 Árboles de Decisión 24](#_Toc10744953)

[2.3.4.3.3 Regresión 26](#_Toc10744954)

[2.3.4.3.4 Support Vector Machines (SVM) 27](#_Toc10744955)

[2.3.4.4 Algoritmos no Supervisados 29](#_Toc10744956)

[2.3.4.4.1 K Means 30](#_Toc10744957)

[2.3.4.4.2 Reglas de Asociación 34](#_Toc10744958)

[2.3.4.5 Algoritmos Semi-Supervisados 37](#_Toc10744959)

[2.3.4.6 Algoritmos de Aprendizaje por Refuerzo 37](#_Toc10744960)

[2.3.4.7 Algoritmos de Redes Neuronales y Deep Learning 38](#_Toc10744961)

[2.3.4.7.1 Redes Neuronales 39](#_Toc10744962)

[2.3.4.7.2 Deep Learning 42](#_Toc10744963)

[2.3.5 Visualización 43](#_Toc10744964)

[3. Capítulo 3: REST 52](#_Toc10744965)

[3.1 MongoDB 53](#_Toc10744966)

[3.2 NodeJS 55](#_Toc10744967)

[4. Capítulo 4: Resultados Obtenidos y Conclusiones Finales 56](#_Toc10744968)

[4.1 Resultados Obtenidos 56](#_Toc10744969)

[4.2 Conclusiones 69](#_Toc10744970)

[4.3 Líneas Futuras, Ampliaciones y Entornos de Aplicación 70](#_Toc10744971)

[**Bibliografía 71**](#_Toc10744972)

Índice de Figuras

[1‑1. Mapa "Summoner's Rift" 4](#_Toc10743129)

[1‑2. Torre 5](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743130)

[1‑5. Barón 6](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743131)

[2‑1 Proceso KDD detallado 10](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743132)

[2‑2. Subdivisiones de la Inteligencia Artificial 11](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743133)

[2‑3. Las 3 V's del Big Data 13](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743134)

[2‑5. Percentiles sobre una normal 16](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743135)

[2‑6. Underfitting, Óptimo y Overfitting 20](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743136)

[2‑7. Ejemplo 5-fold Cross Validation 21](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743137)

[2‑8. KNN 23](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743138)

[2‑9. Estructura básica de un árbol de decisión 25](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743139)

[2‑10. Regresiones 27](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743140)

[2‑11. SVM Linealmente Separable 28](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743141)

[2‑12. SVM No Lineal 29](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743142)

[2‑13. K-Means paso a paso 30](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743143)

[2‑14. Ejemplo en inglés de tabla de reglas 36](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743144)

[2‑15. Red Neuronal Multicapa Feed Forward 40](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743145)

[2‑16. Mismos datos, diferentes escalas 44](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743146)

[2‑17. Clustering Colores 45](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743147)

[2‑18. Gráfico de Líneas 47](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743148)

[2‑19. Explicación de BoxPlot 48](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743149)

[2‑20. Ejemplo Cartograma 49](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743150)

[3‑1. Petición GET en API REST 53](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743151)

[3‑2. Logo de MongoDB 54](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743152)

[3‑3. Entrada/Salida Bloqueante Vs Entrada Salida no Bloqueante 55](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743153)

[4‑1. Wordcloud Nombres 56](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743154)

[4‑2. Scatterplot Edad - Sexo 57](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743155)

[4‑3. Inhibido Vs Impulsivo 57](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743156)

[4‑4. Matriz de correlación 58](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743157)

[4‑5. Estadísticos principales del Dataset 59](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743158)

[4‑6. Distribución Pacientes Componentes Principales 61](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743159)

[4‑7. Precisión de Entrenamiento Deep Learning 67](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743160)

[4‑8. Dendrograma Final 68](file:///D:\Desktop\TFE\MEMORIA\Haciendo%20Data%20Science%20a%20League%20of%20Legends.docx#_Toc10743161)

Índice de Tablas

[Tabla 2‑1. Ejemplos Reglas Finales 34](#_Toc8751764)

[Tabla 4‑1. Resultado de Redes Neuronales 62](#_Toc8751765)

[Tabla 4‑2. Matriz Confusión Test KNN 63](#_Toc8751766)

[Tabla 4‑3. Matriz Confusión Random Forest 63](#_Toc8751767)

[Tabla 4‑4. Matriz Confusión SVM 64](#_Toc8751768)

[Tabla 4‑5.Matriz Confusión KNN Balanceado 65](#_Toc8751769)

[Tabla 4‑6.Matriz Confusión Random Forest Balanceado 66](#_Toc8751770)

[Tabla 4‑7.Matriz Confusión SVM Balanceado 67](#_Toc8751771)

[Tabla 4‑8. Mejores Resultados Finales 68](#_Toc8751772)

# 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.

# 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.

## ¿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.

## Objetivos y victoria

### 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.

### 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.

### 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.

1‑5. Barón

1‑4. Heraldo

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.

### 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.

### 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.

1‑6. Inhibidor

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

### 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.

# El proceso de Data Science

## 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).

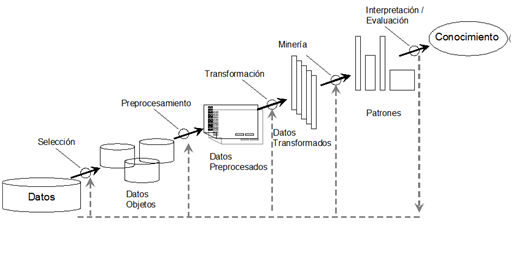
Las diferencias entre estos términos son ignoradas por la gran mayoría de personas, pero a continuación se hace hincapié en ellas para tener los términos bien separados, y a partir de ahora saber a qué se hace referencia:

### 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.

1. 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.

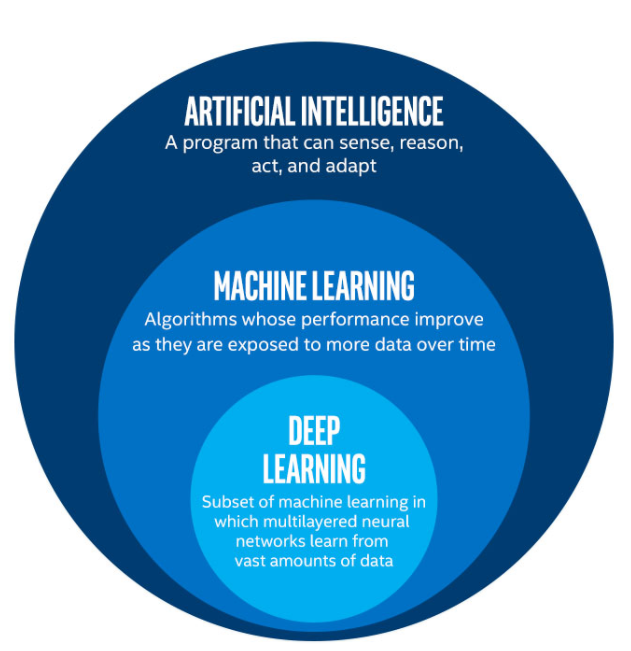
Además, junto con las tareas descriptivas podremos hacer un análisis predictivo para poder predecir ciertas características de futuras observaciones. Esto se analizará más adelante.

1. 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.

### 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”.

### 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.

### 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.

1. 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.

1. 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.

## 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.

### 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

## 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.

### 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
   1. Discretización y binarización
   2. Reducción de la dimensionalidad del dataset

### 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
   1. SeasonId: Todas las partidas pertenecen a la misma temporada.
   2. 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.
   3. 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

### 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.

#### 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:

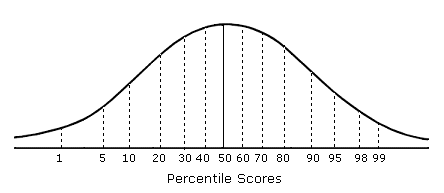
1. Frecuencia y moda

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:

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

1. 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

1. 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:

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.

1. 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 . La fórmula a la que se acoge, siendo m el total de datos, es la siguiente:

La desviación típica, representada con , responde a:

1. 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 Xi y Xj, y la cantidad total de variables m, se puede calcular la covarianza de la forma siguiente:

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:

### Machine Learning

A continuación se expone el núcleo de data science y de este trabajo: El aprendizaje automático o machine learning. Este apartado es el centro debido a que las máquinas pueden aprender de los datos que se les brindan, y con ello predecir o clasificar otros datos de los cuales no se les aporta la solución. Debido a ello, si los algoritmos de machine learning están bien entrenados se abre un gran abanico de posibilidades y respuestas ante las preguntas que se puedan plantear.

La clasificación se podría definir como la tarea de asignar un objeto o un conjunto de ellos a una categoría, normalmente 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.

Dentro de la clasificación podemos hacer una pequeña distinción con los algoritmos predictivos, puesto que estos algoritmos predicen un valor de observaciones desconocidas y continuas, recibiendo también el nombre de regresiones. Los clasificadores trabajan prediciendo grupos en variables discretas.

Las técnicas de clasificación funcionan bien con prácticamente cualquier conjunto de datos, pero siempre se debe tener cuidado, puesto que con los datos ordinales y con las jerarquías los algoritmos de clasificación no funcionan óptimamente. Debido a ello, es importante conocer la naturaleza de los datos antes de empezar con este tipo de algoritmos, tal y como se explicó anteriormente que se debía hacer antes del análisis exploratorio de los datos.

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

#### ¿Cómo funciona un algoritmo de clasificación en machine learning?

Un algoritmo de clasificación en machine learning normalmente sigue unos pasos definidos, y que son ciertamente simples.

El primer elemento a tener en cuenta es que los datos deben de tener una dimensión (es decir, una columna) donde se indique la categoría real a la que pertenecen los mismos, en caso de querer realizar aprendizaje supervisado. Tras la obtención de esta columna, se debe de dividir el dataset en dos grupos, siendo el primer grupo para el entrenamiento del modelo y el segundo para el test del mismo. 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 tienen el conjunto de entrenamiento y de test preparados, se puede aplicar este nuevo conjunto de datos de entrenamiento al algoritmo de clasificación que se esté usando (SVM, K-NN…). La unión del algoritmo y los datos es lo que se conocerá a partir de ahora como modelo, y en este caso será el modelo de entrenamiento.

Este modelo de entrenamiento se deberá poner a prueba ahora con el conjunto de test. Tras obtener la matriz de confusión y calcular el número de aciertos en la diagonal de la misma respecto al número de fallos, se puede calcular la eficacia del modelo y, si es necesario, ajustarlo más para obtener mejores resultados en este test.

Así, la matriz de confusión es una gran aliada en los métodos de clasificación y predicción para ver la eficacia del modelo. Se puede calcular el acierto y el error en tanto por uno de la siguiente manera:

#### Problemas y soluciones con los clasificadores

Los errores que se suelen cometer en un problema de clasificación se pueden dividir en dos tipos: Errores de generalización y errores de entrenamiento. El primer tipo se puede definir 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 alto no conlleva obligatoriamente un error de test alto, pues puede haber generalizado de una forma correcta y obtener resultados de test aceptables, mientras que un error de entrenamiento muy bajo puede ser debido a que se haya caído en overfitting o underfitting.

Se puede definir overfitting 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 conlleva una mayor tasa de fallos a la hora de clasificar otros datos no vistos previamente.

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

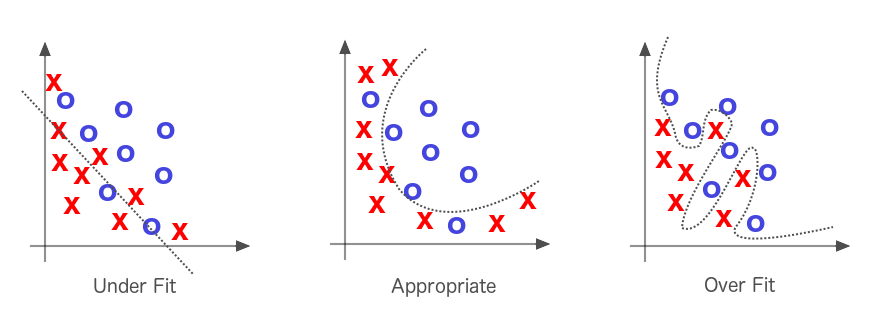
De estos dos problemas, el más común es el overfitting, puesto que se tiende siempre a intentar mejorar el modelo y a veces esta mejora lo único que conlleva es el empeoramiento del mismo. Debido a ello, a continuación se analizarán algunas de las causas por las que puede haber overfitting en un modelo clasificatorio:

1. Debido a la presencia de ruido

La presencia de errores en el dataset hace que los resultados obtenidos de un árbol de clasificación se desvirtúen. Así, cuando unos datos mal clasificados se introduzcan para entrenar un modelo, ese modelo tanto para esos casos como para casos muy similares brindará unos resultados erróneos cuando se prediga con él.

Esto, si lo ponemos 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 más en datasets de gran tamaño, se suele tomar como éxito un error ciertamente pequeño, puesto que estos errores rara vez son evitables cuando se tienen grandes cantidades de datos.

1. Debido a la falta de muestras realmente representativas

Cuando se hacen clasificaciones con un número de muestras muy pequeño (como en el caso de este proyecto), la probabilidad de que aparezca overfitting es muy alta. La falta de elementos similares con características parecidas que pertenezcan al mismo grupo hace que el modelo generalice peor, con su correspondiente aumento del error de test.

2‑5. InfraAjuste, Óptimo y SobreAjuste

Como respuesta a este problema, se deben de buscar soluciones para evitar caer en underfitting y overfitting en cualquier clasificador. Para ello hay 3 métodos ampliamente usados, que se exponen seguidamente:

1. Método Holdout

El método holdout comienza dividiendo los datos en dos conjuntos disjuntos, llamados “conjunto de entrenamiento” y “conjunto de test”. Una vez hecho esto, se crea el modelo con el conjunto de entrenamiento y se prueba su eficacia con el conjunto de test.

Este método, como el lector ha podido comprobar a estas alturas, no es en sí un método para mejorar los errores de generalización, pero como se comentó anteriormente hay que tener en cuenta la proporción en la que se dividan los datos, y esto queda a juicio del experto. Si se utilizan demasiados datos para el entrenamiento, se pueden tener 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: Falta de muestras representativas en el entrenamiento.

Por lo tanto, el método holdout es el primer paso en cualquier creación de modelo de inteligencia artificial. Es un método necesario y de alta importancia, y por ello hay que tenerlo siempre en cuenta.

1. Método Random Subsampling

El método de random subsampling (en castellano, submuestras aleatorias) simplemente consiste en la repetición del método holdout n veces para probar las mejoras que se pueden dar en el rendimiento del clasificador.

Pero random subsampling encuentra problemas respecto al método holdout, puesto que este método no utiliza todos los datos disponibles para el entrenamiento, además de que no tiene ningún control de cuantas veces se utiliza una observación para el entrenamiento o el test, por lo que algunos se usarán para entrenar más veces que otros, y esto puede hacer que quede un modelo desigual y no entrenado óptimamente. Para solucionar este problema, se desarrolló el método Cross-Validation, que se expone a continuación.

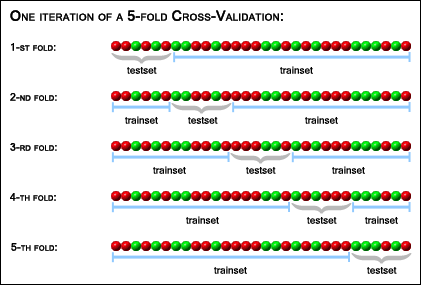
1. Método Cross Validation

Ante los problemas de los métodos anteriores surge cross-validation (también conocido como X-Validation, y en castellano validación cruzada) y consiste en la utilización de cada uno de los registros el mismo número de veces para el entrenamiento, y solo una vez para test, por lo que se obtiene un modelo mucho más ajustado.

A continuación, debido a la importancia de este método, se procede a explicar detenidamente paso por paso como este algoritmo funciona:

El primer paso es aplicar el método holdout; es decir, particionar 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 hará el entrenamiento y el de entrenamiento el que hará de test. Esto es lo que se conoce como “2 fold cross validation”, puesto que se han utilizado dos grupos.

Si este método se realiza k veces, se estarán partiendo los datos en k grupos, y mediante el cambio de los roles de todos los grupos se entrena el modelo. Así, cada uno de los grupos será en un único entrenamiento grupo de test, mientras que será en k-1 entrenamientos grupo de train. Es importante destacar que todos los grupos de entrenamiento se unen a la hora de entrenar un algoritmo en cada una de las iteraciones, formando “un único grupo de entrenamiento”.

En la figura siguiente se puede apreciar un ejemplo de 5 fold cross validation, siguiendo la metodología expuesta anteriormente:

2‑6. Ejemplo de Cross Validation con 5 Folds

Un caso especial de cross validation es en el que el número de grupos coincide numéricamente con la cantidad de datos que se tienen en el dataset, y este método es conocido como “leave one out”, que significa dejar uno fuera.

Este método posee una gran desventaja en que computacionalmente es extremadamente costoso, especialmente en datasets ciertamente grandes, y por si no fuera suficiente, la varianza en las métricas de cada entrenamiento y test será enorme, ya que en algunos tests se obtendrá un acierto del 100% y en otros del 0%. Debido a ello, este método deberá de ser usado como complementario a una validación cruzada con un k más pequeño, y sólo en datasets pequeños.

Ahora que se han visto los principales problemas, es hora de hacer una clasificación en profundidad de los algoritmos de machine learning que existen. Hay 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
* Algoritmos no supervisados
* Algoritmos de aprendizaje por refuerzo
* Algoritmos de redes neuronales y deep learning.

#### Algoritmos Supervisados

El aprendizaje supervisado comienza con un conjunto determinado de datos, y un entendimiento ciertamente profundo de la estructura de los datos. Este tipo de aprendizaje lo que busca es encontrar patrones en los datos, de tal manera que se puedan hacer procesos analíticos sobre unos datos ya etiquetados.

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

Este tipo de algoritmos se utilizan en numerosos ámbitos, como la detección de fraudes, análisis de riesgos, algoritmos de recomendación o incluso el reconocimiento de la voz.

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

* KNN
* Árboles de Decisión
* Regresión Linear
* SVM

##### K Nearest Neighbours (KNN)

El algoritmo “k nearest neighbours”, traducido como “los k vecinos más cercanos”, es aquel que se basa en la búsqueda de atributos similares dentro del conjunto de los datos, y así poder predecir la clase a la que pertenece dicho atributo. Llevado hacia un razonamiento más humano, se podría explicar así: “Si se parece a un avión, es tan grande como un avión, vuela y van personas dentro, entonces es un avión”.

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.

Es importante incidir en que la búsqueda de estos K vecinos se realiza de una forma radial mediante distancias gaussianas, de tal manera que, cuanto mayor sea K, más grande será el círculo que se formará alrededor de la observación determinada para buscar los vecinos de la misma.

2‑7. Ejemplo de KNN con K = 3

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

El algoritmo funciona usando la distancia como elemento de similitud, puesto que dos elementos muy cercanos se supone que serán muy parecidos. Así, del elemento del que se quiere saber el grupo se obtiene una lista de elementos cercanos, y usando modernas técnicas de indexación las computaciones que se deben de hacer para obtener esta lista son mucho menores.

Una vez que se tiene la lista, se clasifica en función del grupo que posea la mayoría, donde todas las observaciones de la lista poseen el mismo peso.

El algoritmo KNN alberga una serie de características propias que determinan cuando debe ser utilizado y los “peligros” que entraña, exponiéndose todo ello a continuación:

1. Este algoritmo no requiere de la construcción de un modelo

Como se explicó anteriormente, un modelo es el conjunto de un algoritmo con una serie de datos. El algoritmo KNN no crea modelo, de tal manera que no se pierde tiempo a la hora de la creación del mismo, pero a la hora de la computación del algoritmo es bastante costoso debido a la necesidad de calcular las distancias de todos los elementos a determinar con el resto de elementos.

1. Es un algoritmo que toma decisiones en zonas locales, no globalmente

Como se puede deducir, al mirar sólo a los elementos más cercanos se incurre en que se toma una decisión a nivel local, no como otros algoritmos como los árboles de decisión (que serán vistos posteriormente), que las toman a nivel global. De esta forma, se deberá tener especial cuidado con los valores de K, puesto que un valor pequeño es susceptible al ruido que pueda generar una zona y, por lo tanto, brindar un valor erróneo.

Además, este modelo sufre de underfitting y overfitting muy fácilmente si no se obtiene un valor de K preciso. En caso de obtener un valor demasiado pequeño, debido al ruido se caerá en overfitting, puesto que solo miraremos el elemento más cercano. Si el valor de K, por el contrario, es demasiado grande, se puede sufrir de underfitting y el modelo se volverá demasiado simple. De este modo, 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 de los datos del problema.

1. KNN produce predicciones erróneas si no se hace un pre-procesamiento correcto

KNN es un algoritmo muy delicado en términos de pre-procesamiento, puesto que al trabajar con distancias son importante los cambios que se hagan a las medidas de los datos. Por ejemplo, si hay numerosas dimensiones cercanas a un número, y también hay otra dimensión con una variabilidad enorme, el algoritmo no funcionará bien puesto que esta última será la más influyente de todas.

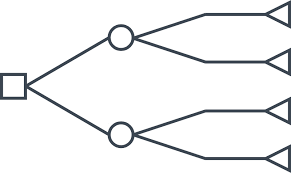
De este modo, para trabajar con este algoritmo un pre-procesamiento a base de centrado y escalado, y una eliminación de las columnas menos importantes escogidas con un PCA sería óptimo.

##### Árboles de Decisión

Una técnica muy utilizada en la clasificación que requiere una pequeña mención aparte es la utilización de los llamados árboles de decisión. Esta técnica es de una simpleza extrema, pero a la vez de una gran eficacia en una gran cantidad de problemas.

Los árboles de decisión recordemos que entran dentro de las técnicas de clasificación, y por lo tanto quieren hallar una respuesta a partir de unos datos previos. De este modo, un árbol de clasificación parte de un nodo raíz, que no posee ninguna entrada, pero tiene salidas. Este nodo raíz se formula una pregunta, y según la respuesta que obtenga se irán desarrollando caminos. En cada uno de estos caminos se plantarán nodos, donde se seguirá haciendo preguntas, y seguirá bifurcándose por cada respuesta hasta que sea capaz de llegar a una decisión final en cada camino. Estos nodos que se han ido formando a base de preguntas a partir del raíz serán denominados nodos intermedios, y las respuestas finales se denominarán como hojas, de las que por supuesto no saldrá ningún camino. Por cada nodo por el que pasen los datos, se va haciendo una criba, de tal manera que al final de cada rama, en las hojas, solo queda un pequeño grupo de datos que poseen numerosas características en común.

La construcción de los árboles de decisión, como se puede apreciar, no es demasiado sencilla a simple vista, debido a que se deben hacer las preguntas adecuadas en el momento adecuado, 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, se han creado algunos algoritmos que construyen árboles de decisión dentro de un espacio óptimo en tiempos razonables, como el de Hunt.

Además, en la construcción se plantean otros interrogantes, como la elección de la pregunta adecuada o la condición de parada del algoritmo.

2‑8. Distribución básica de un árbol de decisión

Respecto a la primera, el algoritmo que se use deberá de tener un sistema implementado para la evaluación de la bonanza de cada pregunta hacia el propio algoritmo, de cara a aprender si la pregunta ha sido buena o no.

Respecto a la condición de parada del algoritmo, es obvio que es algo obligatorio ya que, en caso contrario, el algoritmo seguiría ejecutándose hasta que se acabaran las dimensiones sobre las que preguntar, y eso no siempre es algo positivo de cara al resultado final. Normalmente se usan criterios tales como que todos los elementos restantes tras las preguntas tengan el mismo valor, y ese valor será el que se utilizará como hoja final de esa rama y como condición de parada al mismo tiempo.

A continuación, se procede a hacer una explicación de cómo se puede controlar y solucionar el overfitting, puesto que es el problema más común, de tal manera que se puedan mejorar estos árboles de clasificación y obtener resultados más certeros.

1. Método de la Pre-poda

En el caso de usar este método, el algoritmo que hace crecer el árbol para antes de 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 aumento de una cierta impureza o esencialmente en el error de la generalización.

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

2. Método de la Post-poda

En caso de decantarse por este método, el primer paso es dejar al algoritmo crecer 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 tiene a la mayoría de los individuos en ese subárbol.

Este método es el más utilizado debido a que da mejores resultados, fruto de una poda posterior donde las decisiones de donde recortar vienen dadas de un árbol completamente formado.

##### Regresiones

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

Una definición más formal de regresión podría darse de la siguiente forma: 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 esta manera, 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:

Con estas ecuaciones se aprecia 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 sustracciones negativas debido a que el valor de la función objetivo sea mayor que el real, a cada iteración del sumatorio se le aplica el valor absoluto, puesto que lo que es interesante 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 cualquier valor elevado al cuadrado será positivo. En esta ecuación, los errores grandes son penalizados exponencialmente, mientras que los pequeños 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”, y en la hispanoblante como “Método del Mínimo Cuadrado”.

Supongamos que se tiene que calcular la función objetivo para una serie de puntos que llevará el siguiente patrón:

Donde a y b son los llamados coeficientes de regresión. Usando el método del mínimo cuadrado, se debe 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 ideal. Sobre esta función objetivo posteriormente se podrán hacer las predicciones de valores futuros.

Es importante destacar que no todas las regresiones son lineales, como en el caso de la imagen anterior. También existen regresiones no lineales, que son aquellas en las que se ajusta el modelo con una ecuación con coeficiente 2 o superior, de tal forma que la recta de regresión no es recta sino curva. Estas ecuaciones siguen el siguiente formato:

2‑9. Ejemplos de Regresiones

##### Support Vector Machines (SVM)

El algoritmo SVM es uno de los algoritmos más utilizados dentro del machine learning, puesto que tiene unas fuertes bases estadísticas y matemáticas y ha demostrado solvencia en muchísimas aplicaciones prácticas, como por ejemplo el reconocimiento de caracteres manuscritos. Para comprender cómo funciona SVM, es importante entender la idea de los hiperplanos, puesto que es el fundamento de esta técnica.

Los hiperplanos son planos infinitos en un determinado espacio. En el caso de SVM, estos hiperplanos se denominan “hiperplanos de margen máximo”. Estos se utilizan como soporte de las fronteras para separar dos o más muestras de datos por el grupo al que pertenecen, de tal manera que se pueda conseguir la mejor separación posible. Cada frontera, al ser infinita en una dirección, solo tendrá dos hiperplanos que la soporten, uno a cada lado.

Si los datos son linealmente separables, habrá infinitas fronteras que los separen, puesto que por un punto pueden pasar infinitas rectas, y por lo tanto fronteras. Pero aquí se plantea un problema, y es que en el caso de que haya 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 entran en acción los hiperplanos. 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 tenga una mayor distancia con sus hiperplanos de soporte, puesto que será la que tenga un mayor margen con cada una de las clases, y por lo tanto un menor índice de error, puesto que se hace una mejor generalización. Todo ello es fácilmente apreciable en la siguiente imagen:

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

2‑10. Ejemplo de SVM Linealmente Separable

Es importante a partir de ahora destacar que hay 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 en profundidad:

1. SVM Lineal: Caso Separable

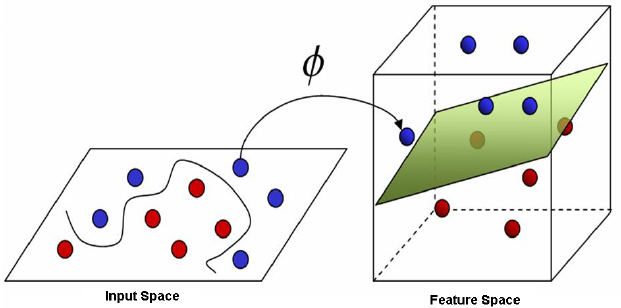
En caso de tener un caso separable en un SVM lineal, como se ha comentado anteriormente se está en un caso de clasificador de márgenes máximos. Por ello, la frontera será una recta, y cumplirá la siguiente ecuación:

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.

1. 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”.

1. 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.

2‑11. Transformación de SVM No Lineal

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. 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.

#### Algoritmos no Supervisados

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.

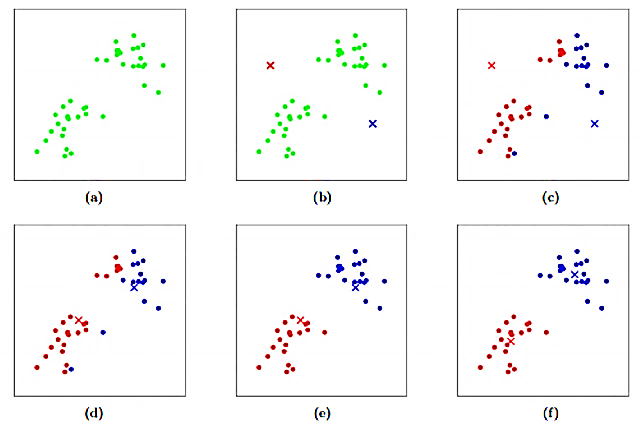
Estos algoritmos se utilizan en ámbitos muy diversos, como los sistemas anti-spam de los correos, reconocimiento de imágenes, obtención de información de redes sociales…

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

* KMeans
* Reglas de Asociación

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

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

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.

1. 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:

1. 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.

1. 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 no se haya encontrado un mínimo global y se quiera aumentar la precisión del algoritmo, el aumento del número de clusters, como se ha comentado anteriormente, es una solución. Así, se tienen dos alternativas:

1. División de clusters

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.

1. Introducción de un nuevo centroide

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.

1. 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:

1. 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.

1. 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.

##### 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 ᴧ, 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 2‑1. Ejemplo Reglas de Asociación Finales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Condición 1 | Operador | Condición 2 | Dirección | Consecuente |
| Es coche = T | ᴧ | Es eléctrico = F | 🡪 | Contaminante |
| Es coche = T | ᴧ | Es eléctrico = T | 🡪 | No Contaminante |
| Es coche = F |  |  | 🡪 | No es coche |

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

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:

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

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.

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:

‑. Ejemplo de tabla de reglas

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

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.

#### Algoritmos Semi-Supervisados

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.

#### Algoritmos de Aprendizaje por Refuerzo

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.

#### Algoritmos de Redes Neuronales y 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.

##### 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.

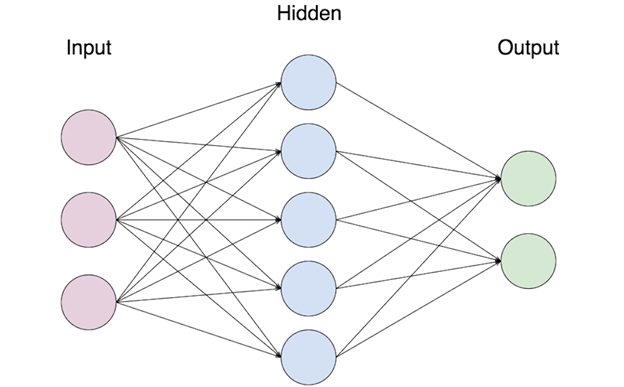
En el proceso de aprendizaje o entrenamiento del perceptrón se tiene en cuenta un factor muy importante, llamado “ratio de aprendizaje”. Se podría definir como un hiperparámetro de las redes neuronales en el que se controla cuanto se están cambiando los pesos de una red neuronal en función del descenso 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:

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 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.

2‑14. Red Neuronal Multicapa Feed Forward

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.

1. 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.

1. 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.

##### 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.

### 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.

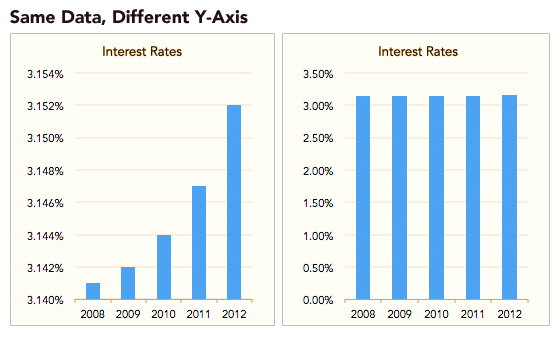
Es importante tener también en cuenta que no todos los tipos de gráficos, ni los colores, ni incluso las dimensiones son elementos que se tengan que elegir al azar. Hay una serie de parámetros y decisiones que se deben de tener siempre en cuenta a la hora de hacer un gráfico, y son las siguientes:

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.

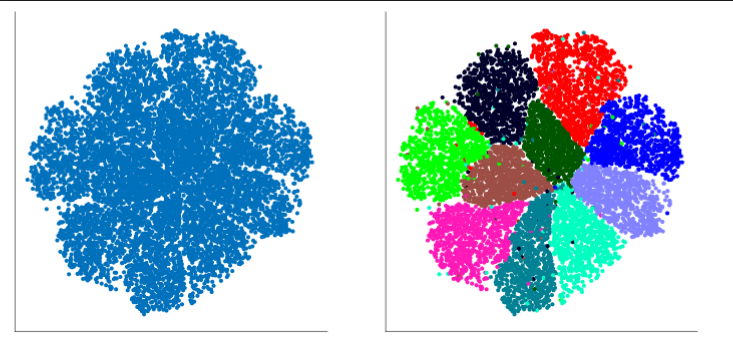
* Ordenamiento

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:

‑. 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.

1. 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.

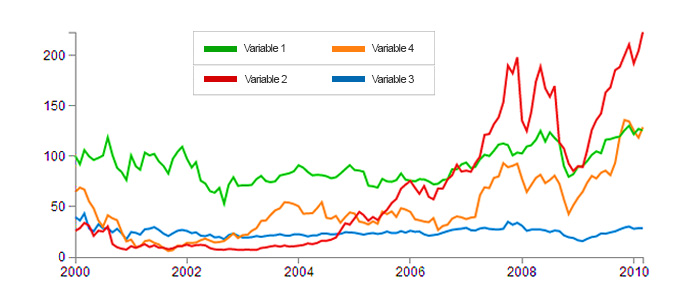
1. Histograma

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.

1. 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.

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.

2‑17. Ejemplo de Gráfico de Líneas

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.

1. 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 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.

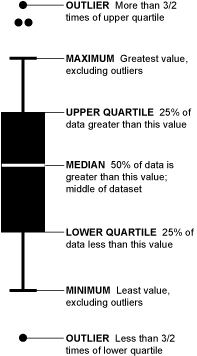
Una de las principales limitaciones que tiene esta representación es la dificultad de interpretación en caso de llevarlo a tres dimensiones, y la imposibilidad de visualización en caso de llegar a 4 dimensiones o más. Debido a esto, este gráfico suele estar limitado a la búsqueda de correlaciones entre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

1. 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

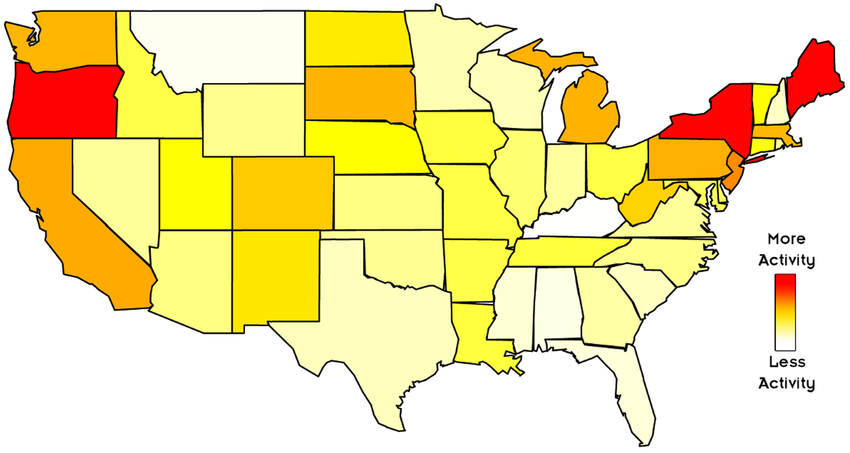
La desventaja que posee esta representación es que la persona a la que se le explica debe de tener una formación estadística para poder entenderlo y comprenderlo en profundidad, por lo que suele ser más un gráfico de exploración que de presentación.

1. 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.

# 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.

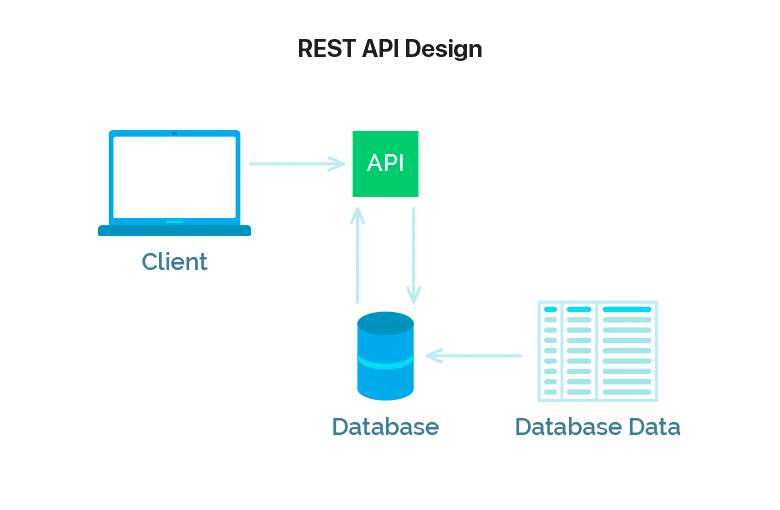
Es muy raro llegar a ver un proyecto donde no haya al menos una API REST para el intercambio de datos con el servidor. Es una arquitectura del lado del servidor, y se podría definir como una interfaz, totalmente compatible dentro del protocolo HTTP, que se utiliza para obtener, modificar, eliminar o 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:
   1. POST: Creación
   2. GET: Obtención
   3. PUT: Edición
   4. 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.

1. 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.

## 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 quiera 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 quieran poder hacer pequeños pero importantes cambios en la base de datos sin necesidad de cambiar toda su estructura.

## 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.

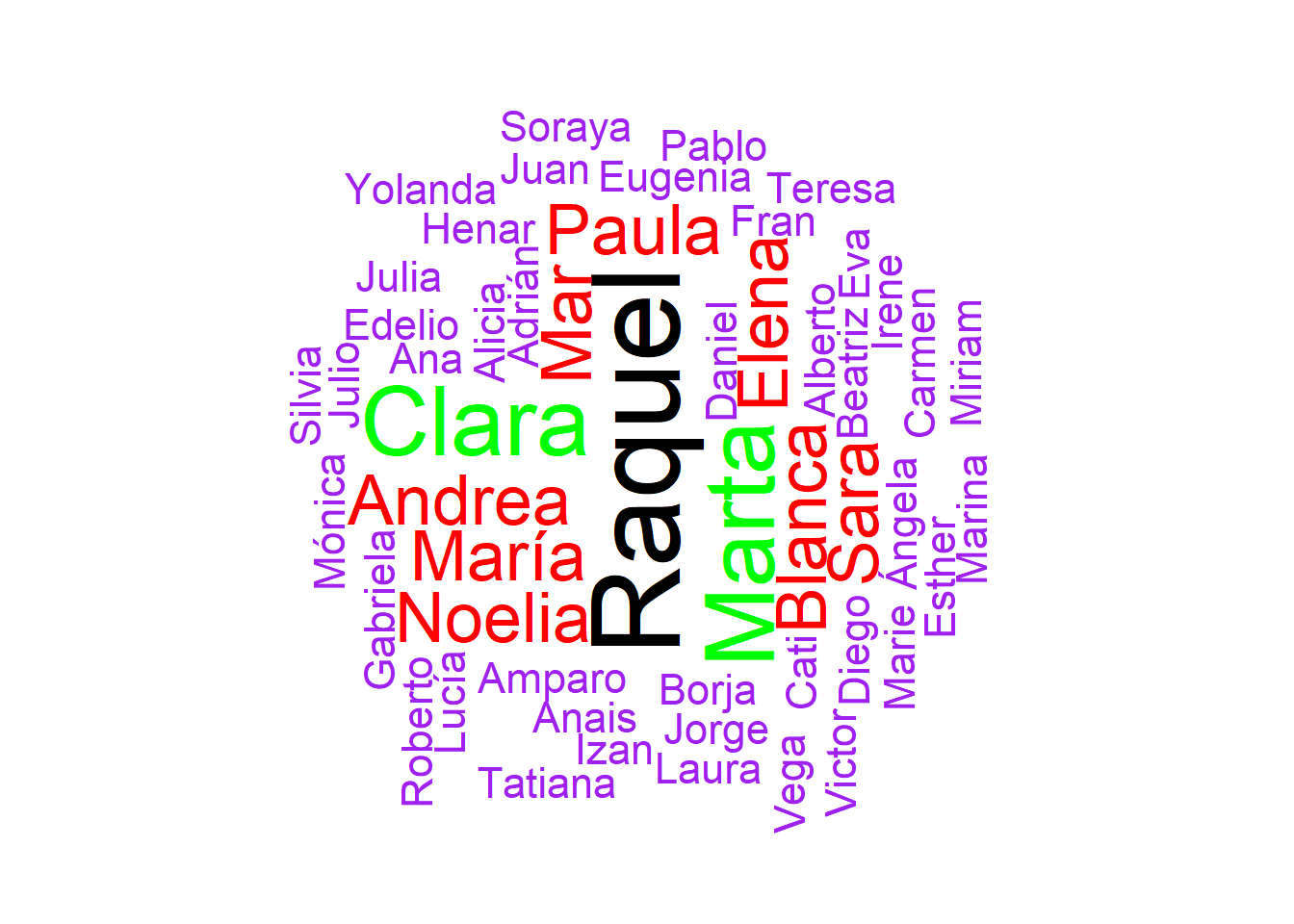
NPM, siglas de “Node Package Manager”, es un manejador de un conjunto de librerías creadas por la comunidad que son capaces de resolver la mayoría de los problemas que se pueden presentar a la hora de desarrollar una aplicación NodeJS. Los comandos que se ejecuten con este manejador siempre empezarán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

# Resultados Obtenidos y Conclusiones Finales

## Resultados Obtenidos

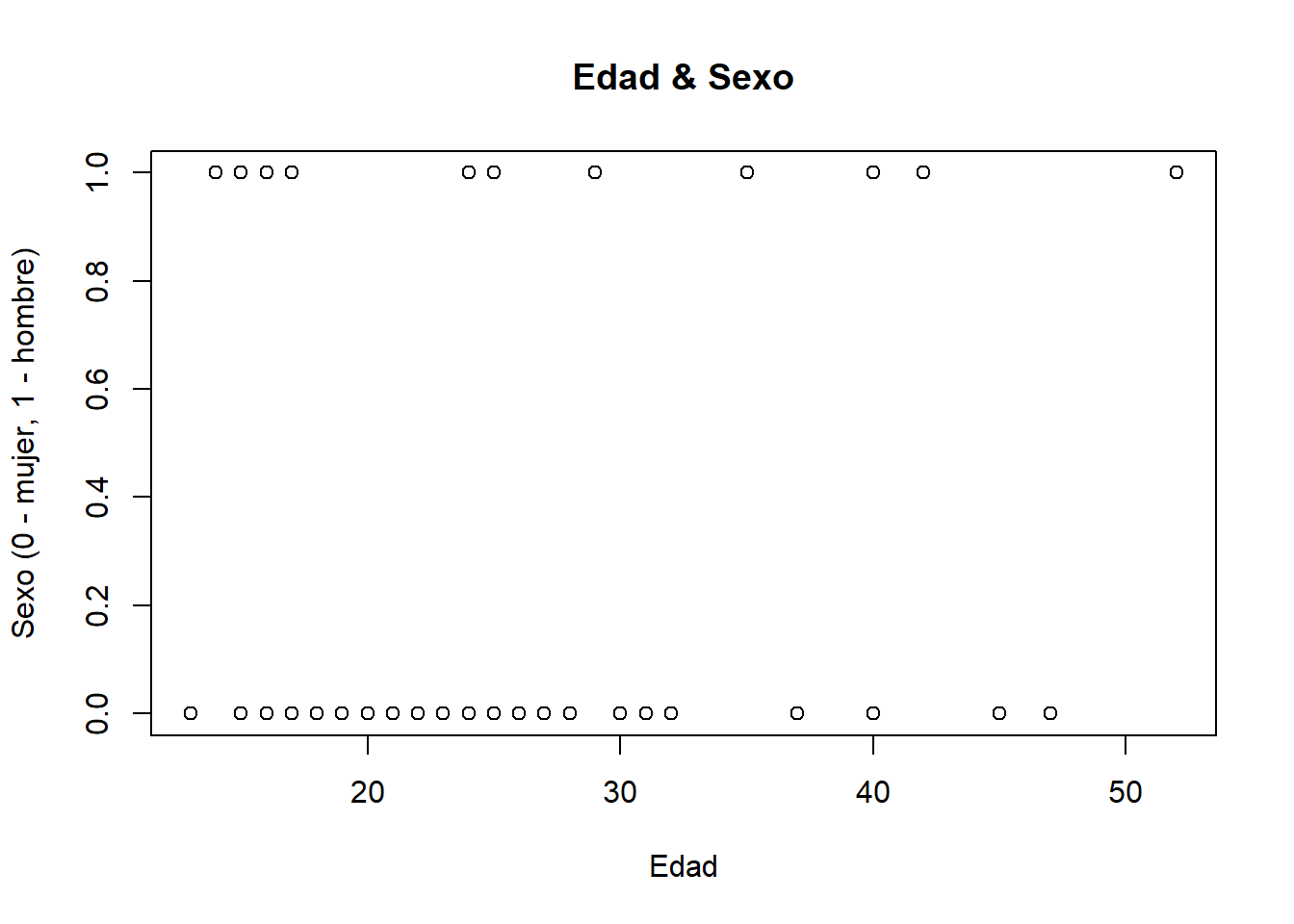
A continuación se exponen los resultados obtenidos tanto en el análisis exploratorio de los datos como con las técnicas de machine learning.

* Análisis Exploratorio

En el análisis exploratorio se han explorado numerosos ámbitos del dataset. Uno, por ejemplo, es la búsqueda de qué nombre se repite más frecuentemente. Esto se ha solventado mediante un wordcloud, también conocida como nube de palabras, que se puede observar en la figura siguiente:

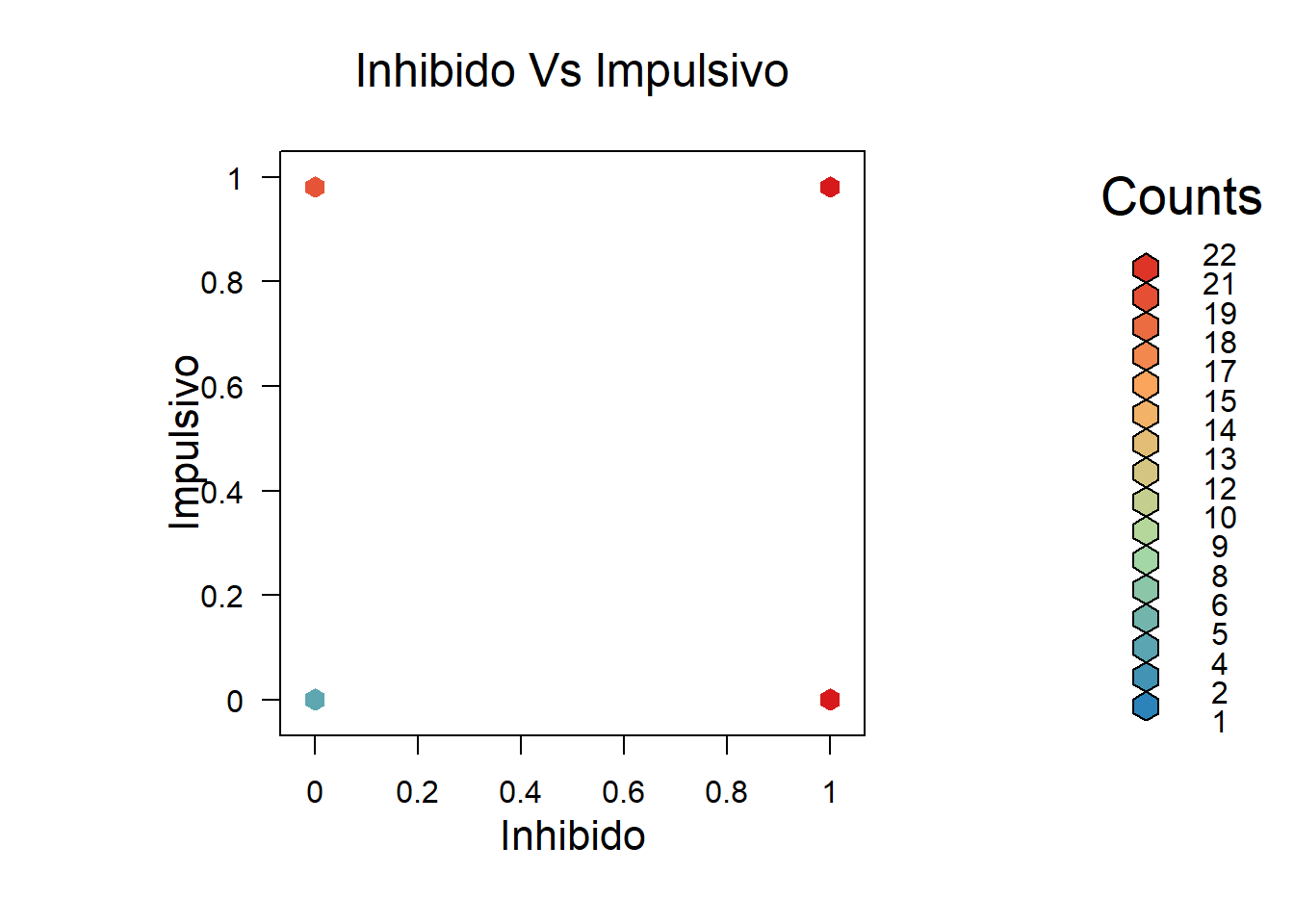
4‑1. Wordcloud Nombres

Como se puede analizar, el nombre más común entre los pacientes analizados es el de Raquel, mientras que otros nombres como Clara o Marta también son bastante destacados.

Posteriormente, se ha hecho un análisis de la relación entre la edad y el sexo de los pacientes. Al tener poca muestra se ha considerado que la creación de un histograma sería menos fácilmente apreciable que un scatterplot, cuyo resultado se muestra a continuación:

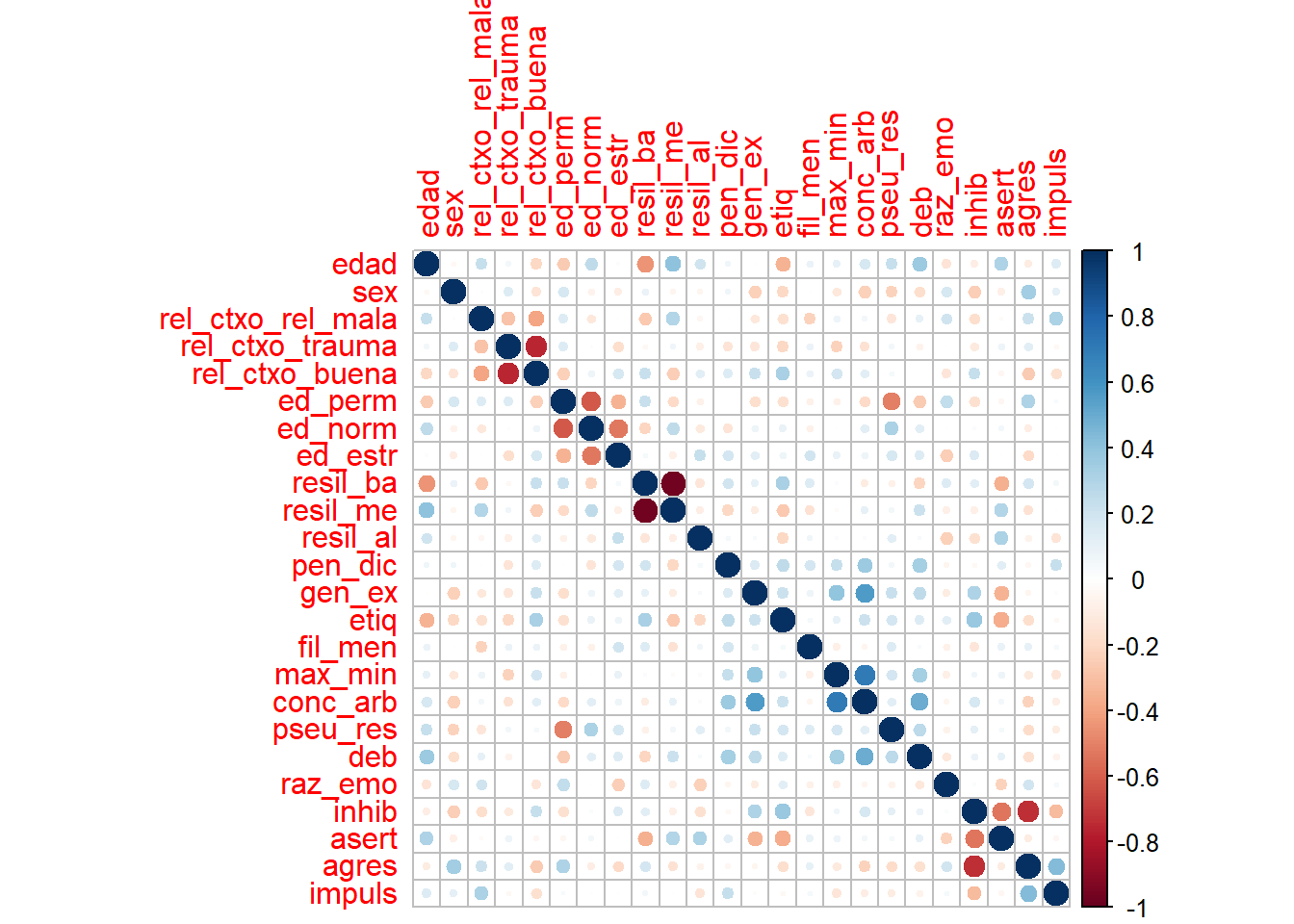
4‑2. Scatterplot Edad - Sexo

Como se puede apreciar en la figura obtenida, las mujeres que acuden a la consulta suelen ser más jóvenes, mientras que los hombres tienen un rango mucho más esparcido en la edad.

Posteriormente se analizaron varias relaciones entre variables, también con scatterplots basados en colores para comprobar la cantidad de individuos residentes en cada caso. Un ejemplo de este estudio es el siguiente, que analiza la inhibición y la impulsividad:

4‑3. Inhibido Vs Impulsivo

Como se puede observar en la figura 4-3, hay numerosos pacientes que son inhibidos e impulsivos, pero también hay bastantes que son inhibidos y no impulsivos. Si nos fijamos en los pacientes no inhibidos, podemos observar que hay más que son impulsivos que los que no lo son.

Posteriormente, siguiendo una metodología científica y estadística más rigurosa se obtuvo una matriz de correlación con las variables significativas del problema, habiendo quitado variables que no aportan información como el nombre previamente.

4‑4. Matriz de correlación

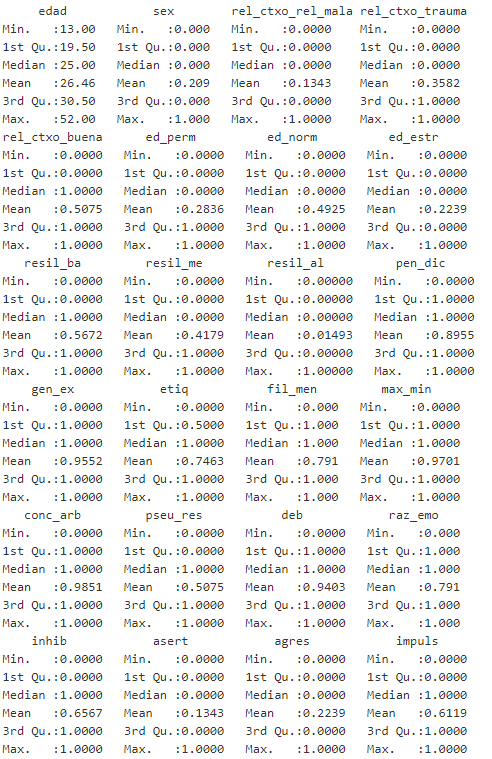
En esta matriz de correlación se pueden observar varias correlaciones obvias, como por ejemplo que una relación con el contexto buena es fuertemente inversamente correlacionada con una relación con el contexto de trauma, o que la resiliencia baja y media también tienen una fuerte correlación inversa. También es destacable que la inhibición está inversamente correlacionada con la agresividad.

Posteriormente se pueden observar otras correlaciones menos conocidas, como que la pseudo-resiliencia está inversamente relacionada con una educación permisiva. Esto tiene cierto sentido, porque ante una educación permisiva, donde la persona puede hacer lo que quiera desde niño sin consecuencias, esta persona tendrá menos experiencia en soportar los aspectos negativos o de presión que puede traer la vida, y por lo tanto aparecerá la pseudo-resiliencia.

Respecto a relaciones directas, es interesante destacar la que existe entre las distorsiones cognitivas de las conclusiones arbitrarias y la maximización y minimización. Esta relación afirma que las personas que tienden a exagerar lo malo y minimizar lo bueno, suelen también obtener conclusiones falsas sin argumentos fehacientes.

También existe relación directa entre las conclusiones arbitrarias y la generalización excesiva. Esta relación tiene mucho sentido debido a que si una persona saca conclusiones arbitrarias, tiene sentido que también generalice y saque conclusiones que de cosas que le han ocurrido en el pasado, aunque sea solo una vez, le vuelvan a ocurrir. Como se puede apreciar, en ambas distorsiones cognitivas se tiene poca o nula prueba de lo que puede pasar, y aun así se dicta lo que se cree que va a pasar con total seguridad.

También la agresividad y la impulsividad tienen una cierta correlación, la cual, según estudios recientes como el hecho por Vigil-Colet, Morales-Vives y Tous en 2008, afirman que una gran cantidad de jóvenes agresivos son impulsivos, pero que con el paso del tiempo, a la llegada de la edad adulta esto cambia, ya que los comportamientos son más controlados. Si se vuelve a mirar la figura 4-2, se puede observar como la gran mayoría de los pacientes que se tienen en el dataset son pacientes jóvenes, por lo que esta correlación está llena de sentido.

Siguiendo con la estadística, se obtuvieron los siguientes resultados estadísticos del dataset:

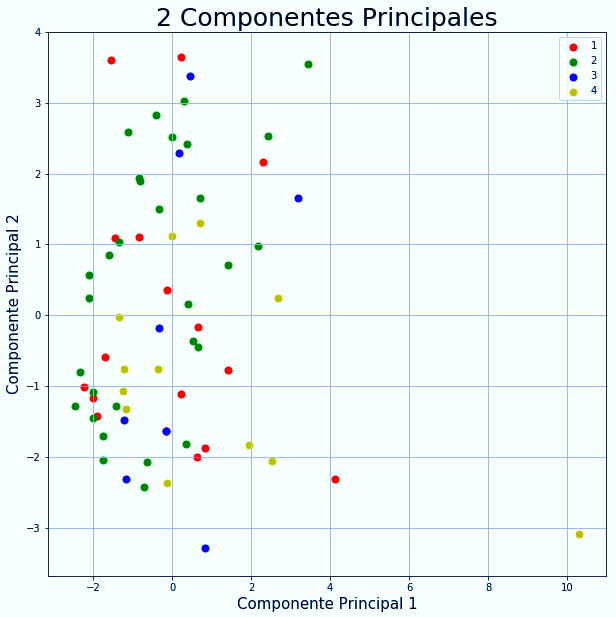
4‑5. Estadísticos principales del Dataset

Se pueden hacer algunas valoraciones con estos datos:

1. Edad: La media de la muestra que acude a consulta tiene 26 años y medio, estando el primer cuartil en 19,5 años y el tercer cuartil en 30,5 años. Esto quiere decir que el 50% de los pacientes son veinteañeros. También se puede observar que la edad mínima de un paciente en la muestra es de 13 años.
2. Sexo: A pesar de ser una variable categórica, se pueden sacar conclusiones ya que es binaria a partir de la media y los cuartiles. Esta variable nos indica que hay muchas más mujeres que hombres en la muestra (media de 0,20 y tercer cuartil de 0).
3. Relación con el contexto: Respecto a esta variable es interesante fijarse en la media, ya que como la anterior, es binaria, donde se puede observar que la mayoría de gente tiene una relación con el contexto buena, seguido de trauma y finalmente de mala.
4. Educación: La mayor parte de la gente posee una educación normal, siendo la permisiva la siguiente.
5. Resiliencia: Se puede observar que en la única en la que está la mediana con un valor distinto a cero es en la resiliencia baja, lo cual nos indica que esa variable está “activada” numerosas veces. Además, esto también se puede observar fácilmente por la media y el tercer cuartil.
6. Distorsiones cognitivas: Se aprecia que las medias son bastante cercanas a 1, por lo que se aprecia que están muy presentes en los pacientes en su mayoría. Es interesante también observar que todas, a excepción de la pseudo resiliencia ya obtienen el valor 1 en el primer cuartil.
7. Habilidades sociales: Observando la media y la mediana, la mayoría de personas son inhibidas, siguiendo por la agresividad y acabando por la asertividad.
8. Impulsividad: La media reside en 0.61, y la mediana en 1 (lógico debido a que es una variable categórica de 2 categorías), lo que quiere decir que hay una mayoría de personas impulsivas en la consulta.

* Métodos de Inteligencia Artificial

Se han utilizado diversos métodos de inteligencia artificial para la clasificación de los pacientes en sus respectivos grupos, obteniendo resultados no extremadamente satisfactorios con el dataset original, aunque bastante diversos en su calidad. Posteriormente, tras la igualación de los grupos se han obtenido resultados mucho más satisfactorios.

Es importante destacar la distribución de los pacientes en una gráfica, para ver si son fácilmente diferenciables o no. Para esto, se ha reducido el problema a las dos componentes principales (es decir, las dos dimensiones más significativas) y se ha obtenido una gráfica donde se pueden ver los pacientes diferenciados por colores según el trastorno.

‑. Distribución Pacientes Componentes Principales

Como se puede observar, bajo las dos componentes que más información aportan al problema la clusterización de los pacientes se antoja como una tarea dificilísima, puesto que están realmente mezclados entre sí y bajo estas dimensiones es muy complicado separarlos. Debido a esto, es probable que muchos métodos acaben teniendo problemas a la hora de clasificar, en especial los métodos basados en distancias, ya que al aumentar el número de componentes para intentar visualizar más diferencias se pueden encontrar con la maldición de la dimensionalidad.

Veamos a ver los resultados obtenidos sobre el dataset original:

* + Redes Neuronales

Las redes neuronales fueron desarrolladas en lenguaje R, donde se probaron numerosos parámetros para afinarlas. La siguiente tabla muestra los mejores resultados obtenidos con los mejores parámetros para dicha configuración, tras 50 entrenamientos y tests:

Tabla 4‑1. Resultado de Redes Neuronales

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Neuronas | 1 | 1 | 2 | 2 | **3** | 3 | 3 | 3 | 5 | 5 | 5 | 5 |
| Softmax | No | Sí | No | Sí | **No** | Sí | No | Sí | No | Sí | No | Sí |
| Decay | - | - | - | - | **-** | - | 0,2 | 0,03 | - | - | 0,1 | 0,05 |
| Resultado Test | 66,6% | 66,6% | 66,6% | 75% | **75%** | 75% | 75% | 66,6% | 75% | 66,6% | 75% | 50% |

Como se comentó en la parte teórica, estos resultados no son totalmente correctos, pues hay que tener en cuenta dos elementos:

1. Las personas que tienen un TOC tienen un trastorno de ansiedad, por lo que una clasificación del grupo 1 en el 2 sería técnicamente correcta, y esto no se está teniendo en cuenta a la hora de valorar las redes neuronales.
2. Softmax no tiene en cuenta que un elemento pueda pertenecer a varias categorías, por lo que sus resultados deben de ser rechazados en este estudio.

Debido a esto, y de un modo estricto (sin tener en cuenta la condición multigrupo del TOC), la configuración de perceptrón multicapa idónea para este problema será la de 3 neuronas sin softmax ni decay, debido a que con menor número de neuronas (y por lo tanto menor overfitting) se ha obtenido un resultado máximo, con un valor de acierto del 75%.

* + KNN

Este método, al igual que todos los siguientes, han sido desarrollados en Python 3, usando el paquete de inteligencia artificial del mismo: Scikit Learn, también conocido como SKLearn.

Con K Nearest Neighbors se han obtenido resultados no tan positivos como con las redes neuronales. Para evitar la gran ineficiencia de ir probando los métodos cambiando pequeños parámetros, se ha formado una malla de parámetros para que el algoritmo determine cuál es la óptima, y así poder trabajar con los mejores. También se debe remarcar que se ha aplicado una cross-validation de 3 folds.

Ante los resultados obtenidos, el algoritmo determinó que los mejores parámetros se daban usando la distancia de Manhattan y con un K igual a 12.

La puntuación máxima obtenida, tanto en entrenamiento como en test es del 50%, obteniendo la siguiente matriz de confusión:

Tabla 4‑2. Matriz Confusión Test KNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 0 | 1 | G1 |
| 1 | 7 | 0 | 0 | G2 |
| 0 | 1 | 0 | 0 | G3 |
| 3 | 1 | 0 | 0 | G4 |
| G1 | G2 | G3 | G4 | Predicción\Real |

Como se puede apreciar en la tabla, de los 18 pacientes a clasificar 9 los ha clasificado correctamente en los grupos 1 y 2 (TOC y ansiedad), confundiendo 1 paciente de TOC por ansiedad, lo que elevaría la cifra de acierto hasta el **55%**.

También se puede observar que ha clasificado muy mal los pacientes correspondientes a los grupos 3 y 4, no habiendo acertado ninguno. Esto se debe a la falta de muestras recogidas de estos grupos (en parte por la escasez de los pacientes con estos cuadros en el gabinete en el momento de la entrevista), por lo que se supone que el entrenamiento ha sido poco fiable hacia estos grupos con su correspondiente consecuencia en el test.

* + Random Forest

Con el método Random Forest se ha seguido una estrategia similar. Se ha creado una malla con parámetros para el entrenamiento del algoritmo, se ha aplicado una cross-validation de 5 folds y se ha obtenido el mejor modelo posible, con el que se han obtenido los siguientes resultados:

Parámetros óptimos:

* Criterio de aceptación: Gini
* Profundidad máxima del árbol: 5 unidades
* Máximas Características: Automático
* Número de estimadores: 100

Con estos parámetros se ha conseguido un acierto de entrenamiento del 92%, mientras que el acierto de test se ha quedado en un 47,1%.

La matriz de confusión de este método se queda de la forma siguiente:

Tabla 4‑3. Matriz Confusión Random Forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicción \ Real | G1 | G2 | G3 | G4 |
| G1 | 1 | 2 | 0 | 1 |
| G2 | 2 | 6 | 0 | 0 |
| G3 | 0 | 1 | 0 | 0 |
| G4 | 1 | 2 | 0 | 1 |

Como se puede observar, especialmente del grupo 2 ha obtenido muy buenos resultados este modelo, especialmente si tenemos en cuenta las dos clasificaciones de TOC en ansiedad, que también se darán como válidas. Respecto a los grupos 3 y 4 vuelve a haber gran cantidad de errores.

El resultado final, teniendo en cuenta estas clasificaciones sería de un **55,5%** de acierto en test.

* + 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 el modelo declaró que los mejores parámetros, para una cross-validation de 3, eran los siguientes:

* Kernel: RBF
* C: 2
* Grado: 2

Hay que tener en cuenta que el grado solo se utiliza si el kernel es polinómico, puesto que sería el grado de la ecuación del polinomio. Al usar kernel RBF, este parámetro no deberá ser tenido en cuenta.

Los resultados obtenidos datan de un 88% de acierto en el entrenamiento, pero de un 41,2% de acierto en el test, siguiendo con la dinámica de los modelos anteriores.

Tabla 4‑4. Matriz Confusión SVM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicción \ Real | G1 | G2 | G3 | G4 |
| G1 | 1 | 2 | 0 | 1 |
| G2 | 2 | 6 | 0 | 0 |
| G3 | 0 | 1 | 0 | 0 |
| G4 | 1 | 3 | 0 | 0 |

Como se puede apreciar, la matriz de confusión obtenida en SVM se parece mucho a la matriz de confusión obtenida en Random Forest, a excepción de la predicción del grupo 4 donde obtiene un error más. Debido a esto, los resultados serán similares.

Teniendo en cuenta las dos clasificaciones correctas hacia la relación entre TOC y Ansiedad, se puede afirmar que el resultado final es de un **50%** de acierto, habiendo acertado 9 de 18 casos.

Ante estos resultados pobres, se tuvo que analizar la situación, llegando a la conclusión de que las clases poco balanceadas podían afectar fuertemente en la obtención de resultados positivos, puesto que las matrices de confusión mostraban resultados extremadamente negativos en los grupos 3 y 4 en la mayoría de los algoritmos. Debido a ello, se nivelaron los grupos 1, 3 y 4 haciendo que estuvieran mucho más cerca del número de observaciones del grupo 2, mediante técnica de resampling.

De este modo, se volvieron a ejecutar los algoritmos con sus respectivas mallas de valores, obteniendo los siguientes resultados:

* + KNN Balanceado

El KNN Balanceado obtuvo de la malla los siguientes valores como óptimos:

* Métrica: Manhattan
* K = 1

A pesar de que este K = 1 lleva a overfitting (puntuación de train de 100%), el resultado de test mejoró considerablemente, llegando al **77,7%** de acierto. Es también importante el que se destaque que la cross validation aplicada en este caso ha sido de 4.

La matriz de confusión se muestra a continuación:

Tabla 4‑5.Matriz Confusión KNN Balanceado

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicción \ Real | G1 | G2 | G3 | G4 |
| G1 | 4 | 0 | 0 | 2 |
| G2 | 0 | 5 | 1 | 2 |
| G3 | 1 | 0 | 8 | 0 |
| G4 | 0 | 0 | 0 | 4 |

Como se puede observar, en todas las clases el acierto es muchísimo mayor que el error, siendo la clase 2 la menos acertada con 5 aciertos de 8, un 62,5%. Es interesante observar como las clases 3 y 4, que anteriormente eran clases donde apenas se acertaba una observación como máximo, actualmente aciertan prácticamente la totalidad de sus observaciones.

Para la interpretación de estos resultados es importante destacar que se ha hecho sobre el conjunto de test que tiene el resample, de tal manera que hay más observaciones que en los anteriores, pero ello no distorsiona el resultado final, que se medirá siempre en porcentaje.

* + Random Forest Balanceado

En esta aplicación de Random Forest se ha vuelto a utilizar un cross validation de 3, y con ello se han obtenido los siguientes valores como óptimos:

* Criterio: Entropía
* Máxima Profundidad del Árbol: 7
* Máximas características: Automático
* Número de estimadores: 50

Con estas características, se han obtenido en entrenamiento un resultado del 100%, mientras que en test el resultado obtenido en este caso es del **77,7%** de acierto.

La matriz de confusión se muestra a continuación:

Tabla 4‑6.Matriz Confusión Random Forest Balanceado

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicción \ Real | G1 | G2 | G3 | G4 |
| G1 | 4 | 0 | 0 | 2 |
| G2 | 0 | 5 | 1 | 2 |
| G3 | 1 | 0 | 8 | 0 |
| G4 | 0 | 0 | 0 | 4 |

Como se puede apreciar, la matriz de confusión del Random Forest es la misma que la obtenida con KNN, por lo que ambos métodos resultan igual de efectivos a la hora de resolver el problema.

* + SVM Balanceado

En el caso del algoritmo de Support Vector Machines, se ha mantenido la cross validation de 3 para comparar resultados con los no balanceados, y con esta premisa se han obtenido los siguientes valores de la malla como óptimos:

* C = 1
* Grado = 2 (Para Kernel Polinómico)
* Kernel = RBF

La puntuación obtenida es similar a la de los métodos anteriores, donde en este caso el entrenamiento ha tenido un valor del 92,3% de acierto y el test se ha quedado en un **74,1%.**

La matriz de confusión de este SVM se muestra a continuación:

Tabla 4‑7.Matriz Confusión SVM Balanceado

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predicción \ Real | G1 | G2 | G3 | G4 |
| G1 | 5 | 0 | 0 | 1 |
| G2 | 0 | 4 | 1 | 3 |
| G3 | 1 | 0 | 8 | 0 |
| G4 | 1 | 0 | 0 | 3 |

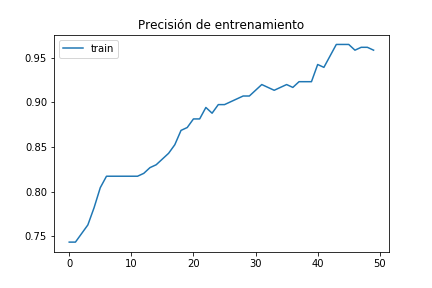
La matriz de confusión en este caso de SVM es menos precisa tanto en el grupo 2 como en el grupo 4, es igual de precisa en el grupo 3 y el único grupo que aumenta la precisión con SVM es el grupo 1.

* + Deep Learning

El modelo de deep learning ha sido entrenado de una manera supervisada, ya que se poseen las etiquetas del mismo. Además, ya que se conoce que los modelos pueden aprender mejor con clases balanceadas, se ha entrenado con estos datos obteniendo un resultado del **93,5%** de acierto máximo en test, con un 96,4% de acierto en train.

Para mejorar el modelo de deep learning, la estrategia ha sido la creación de 3 capas ocultas con pocas neuronas cada una (6, 5, 5), de tal manera que cada capa pueda “especializarse” en una extracción de características determinada y puedan funcionar mejor de manera conjunta.

Además, para hacer deep learning se ha tenido que codificar como variables dummies los grupos, y así tener 4 neuronas de salida, una por grupo. Junto con ello, hay una capa de entrada de 24 neuronas, con un tamaño de input de 24 dimensiones.

A continuación se expone una imagen del aumento del acierto de entrenamiento en el modelo de deep learning con el paso de las iteraciones:

‑. Precisión de Entrenamiento Deep Learning

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

Tabla 4‑8. Mejores Resultados Finales

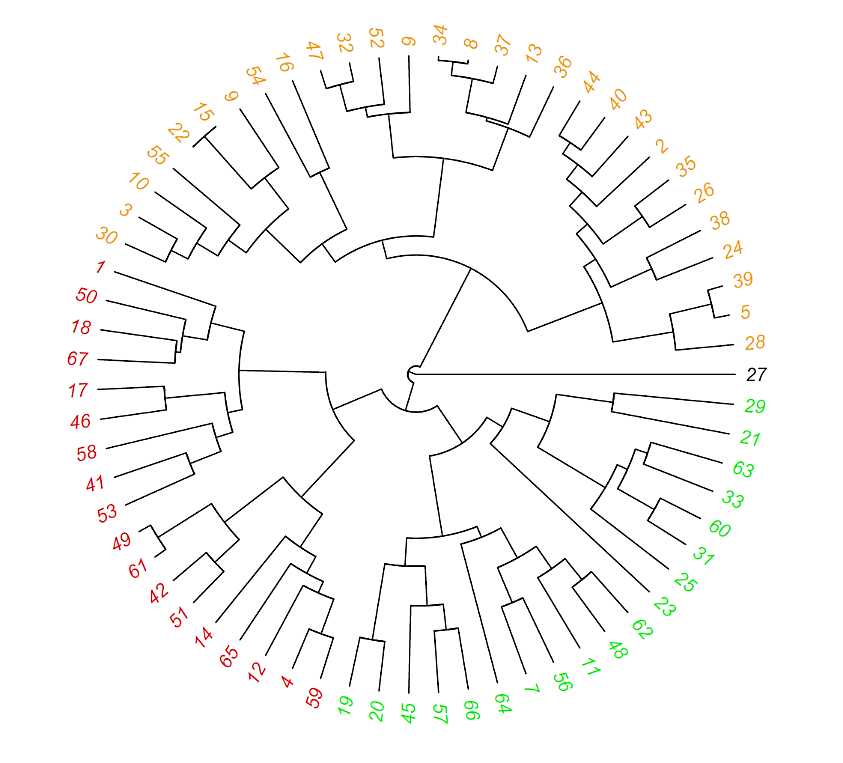
|  |  |  |
| --- | --- | --- |
|  | No Balanceado, CV = 3 | Balanceado, CV = 3 |
| KNN | 55,5% | 77,7% |
| Random Forest | 55,5% | 77,7% |
| SVM | 50% | 74,1% |
|  | No Balanceado | Balanceado |
| Redes Neuronales | 75% | - |
| Deep Learning | - | 93,5% |

Como se puede observar, hay una clara mejoría de los algoritmos supervisados ante el balanceo de los grupos. Estos resultados es posible que se puedan mejorar con un cross validation mayor, pero a efectos de investigación se acaba de demostrar que estos resultados mejoran, bajo las mismas condiciones, si los grupos se balancean.

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:



‑. Dendrograma Final

Como se puede observar, el elemento outlier se ha obtenido como un grupo aparte, mientras que el resto de pacientes han sido clasificados en tres grupos. Se ha dejado este elemento outlier como prueba, debido a que la psicóloga ha afirmado que es un chico sin ninguna patología especial, por lo que resulta extremadamente interesante como el único paciente sin patología ha sido determinado como grupo único.

## Conclusiones

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

1. Los trastornos psicológicos son un elemento a la orden del día, cuyo diagnóstico y tratamiento está enfocado en el “framework” de la psicología cognitivo-conductual. La mayor parte de los psicólogos actualmente trabajan bajo este marco, y este trabajo ha sido enfocado de la misma manera, obteniendo resultados satisfactorios según lo establecido.
2. Estos trastornos psicológicos PUEDEN ser predichos en términos generales, con grupos generales, en acierto cercano al 80%, a través de las distorsiones cognitivas y otros factores, como la edad, el sexo o las habilidades sociales de la persona. Este trabajo no intenta demostrar que una máquina puede predecir mejor que una persona los trastornos, puesto que aparte de que ha quedado demostrado que la fiabilidad no se acerca al 100% (excepto en Deep Learning), hay numerosos factores que pueden hacer que una persona posea varios trastornos al mismo tiempo, y es bastante complicado que una máquina llegue a dichas conclusiones.
3. Un problema real de ciencia de datos, y por ende de machine learning, nunca va a tener un dataset idílico sobre el que trabajar, por lo que sobre el mismo se deben de hacer numerosas transformaciones para su posterior uso en machine learning. En este caso, aparte de la ligera limpieza que hubo que hacer (se hizo en la recogida), es muy interesante el hecho de haber clases poco balanceadas en la muestra, por lo que hubo que balancearlas. Además, otras técnicas como el PCA tuvieron que ser aplicadas.
4. Los algoritmos de machine learning no trabajan bien con pocas observaciones (pobre generalización), lo cual probablemente ha hecho que no se puedan afinar más los resultados, ni tampoco con demasiadas dimensiones (maldición de la dimensionalidad), fenómeno que no ha llegado a ocurrir en este trabajo. Tampoco se deben reducir demasiado las dimensiones del dataset (ver figura 2-4), debido a que cuantas menos dimensiones haya menor explicación habrá, y más difícil será una correcta clasificación de los datos. En este trabajo, ha sido realmente importante la elección de las variables, y por ende el business understanding, para obtener buenos resultados, puesto que la cantidad de observaciones era extremadamente pequeña para un problema de este tipo.
5. Si el dataset está balanceado, 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. En el caso de KNN, se ha observado que consigue buenos datos haciendo overfitting, pudiendo ser por la escasez de los pacientes, en SVM se ha comprobado que el mejor kernel es RBF, lo que indica que no existe una buena separación lineal, y el algoritmo de Tree Random Forest consigue resultados bastante buenos en base a la diferenciación con preguntas.

## Líneas Futuras, Ampliaciones y Entornos de Aplicación

El actual trabajo está hecho con una intencionalidad puramente investigadora y académica, no teniendo una finalidad de despliegue comercial. Debido a ello, una línea futura podría ser el despliegue de este sistema aplicando alguno de los algoritmos más exitosos sobre una plataforma web, o una aplicación móvil, de tal manera que al introducir las variables del paciente se pudiera visualizar la predicción del trastorno que sufre.

Junto con ello, se debería aumentar el espectro de los trastornos psicológicos, puesto que en este trabajo se están clasificando los cuatro tipos más comunes, pero se podría ampliar hacia el resto de los trastornos.

Como está indicado en las conclusiones, un sistema de esta índole nunca deberá de suplir al experto humano, puesto que muchas otras variables y sensaciones en la entrevista personal son importantes a la hora de diagnosticar un trastorno, pero ello no impide que un sistema de este tipo sirva de confirmación hacia las sospechas que un psicólogo pueda tener hacia el trastorno de un paciente.

Otra línea de ampliación consistiría en la prueba de otros algoritmos, como AdaBoosting, Extreme Gradient Boosting, Spectral Clustering o incluso Bayes, ya que son otros métodos de clasificación que podrían dar buenos resultados, aunque personalmente no creo que fueran muy superiores a los actualmente obtenidos.

Finalmente, una línea de ampliación muy interesante, para hacer más útil la primera ampliación, sería la creación de un sistema de envío de datos de pacientes totalmente anonimizado, de tal manera que la base de datos de pacientes aumentara de los 67 actuales a numerosos pacientes más, de tal forma que los algoritmos puedan ir mejorando con el paso de los mismos.

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