UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO ORIENTADAS AL DIAGNÓSTICO MÉDICO

Pablo Nocedal González

Tutor: Rafael Llobet Azpitarte

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2019-20

Valencia, 25 de noviembre de 2019

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Agradecimientos

Primero de todo me gustaría expresar mi más sincero agradecimiento a todos mis seres queridos, entre ellos mis padres, mi hermano Óscar, mi pareja y mis amigos, por su ayuda y apoyo incondicional a lo largo de este intenso proceso. También me gustaría hacer una especial mención a mis compañeros del grupo ARA, donde he compartido experiencias inolvidables, en el ámbito estudiantil y en experiencias fuera del mismo. Por último, también me gustaría agradecer al tutor de este trabajo, Rafael Llobet Azpitarte, por el grandísimo trabajo que ha realizado conmigo ya que sin él nada de esto hubiera sido posible.

Acknowledgments

First, I would like to express my sincere gratefulness to my parents, my brother Oscar, my girlfriend and my friends, for their unconditional support and help throughout this intense process. I would also like to make a special mention to my colleagues in the ARA group, where we have shared unforgettable experiences, not only in the student field, but also out of it. Finally, I would also like to thank my supervisor, Rafael Llobet Azpitarte, for the great work he has done with me, without him none of this would have been possible.

Resumen

Cada vez más se generan grandes volúmenes de datos de los que es posible extraer información de gran relevancia, y el área de la salud no es una excepción en este sentido. Estos datos permiten emplear técnicas de aprendizaje automático supervisado en las que la máquina trata de aprender a predecir el resultado de alguna variable de salida (por ejemplo, un diagnóstico sobre cierta enfermedad) a partir de un conjunto de variables de entrada (por ejemplo, datos clínicos, fisiológicos, etc.). En este TFG se propone aplicar técnicas de aprendizaje automático orientadas a la predicción, diagnóstico y pronóstico de enfermedades, abordando para ello todas las fases del proceso, a excepción de la adquisición: preparación de los datos, visualización, análisis descriptivo, aprendizaje de un modelo predictivo y evaluación de resultados. Se utilizará para ello el lenguaje de programación Python junto con algunas de sus librerías orientadas a tal fin.

Resum

Cada vegada s'obtenen volums més grans de dades dels quals és possible obtindre informació de gran rellevància i, en aquest sentit, l'àrea de la salut no és una excepció. Aquestes dades permeten utilitzar tècniques d'aprenentatge automàtic supervisat en las què la màquina tracta d'aprendre un predir el resultat d'alguna variable de sortida (per exemple, un diagnòstic sobre una certa malaltia) a partir d'un conjunt de variables d'entrada (per exemple, dades clíniques, fisiològiques, etc.). En aquest TFG es proposa aplicar tècniques d'aprenentatge automàtic orientades a la predicció, diagnòstic i pronòstic de malalties, abordant totes les fases del procés, a excepción de l'adquisició: preparació de les dades, visualització, análisi descriptiva, aprenentatge d'un model predictiu i avaluació dels resultats. En aquest sentit s'utilitzarà el llenguatge de programación Python juntament amb algunes de les seues llibreries orientades a aquesta finalitat.

Abstract

More and more, large volumes of data are being generated from where it is possible to obtain information of great relevance, and the area of health is not an exception in this regard. These data allow to apply supervised machine learning techniques where the machine tries to learn to predict the outcome of some variable (for example, a diagnosis of a certain disease) from a set of input variables (for example, clinical data, physiological, etc.).

In this final degree project, the proposal is to apply machine learning techniques oriented to prediction, diagnosis and prognosis of diseases, going through all phases of the process,

except from the acquisition: data preparation, visualization, descriptive analysis, learning through a predictive model and evaluating the result. Python programming language will be used for this along with some libraries designed for the purpose.

Índice

Índice de figuras

Capítulo 1. Introducción

A pesar de que el aprendizaje automático o "Machine Learning" parezca un nuevo campo en el sector tecnológico, lo cierto es que sus inicios son mucho antes de lo que se pueda imaginar.

En 1943 se dio el primer caso de redes neuronales, cuando el neurofisiólogo Warren McCulloch y el matemático Walter Pitts escribieron un artículo sobre las neuronas y su funcionamiento. Pero no es hasta 1950 cuando Alan Turing creó la famosa prueba de Turing, donde la máquina debía ser capaz de convencer a un humano de que no era una máquina, si no que era un humano.

En 2019, Victor Gonzalez Pacheco establece que, entre 1952 y 1956, Arthur Samuel escribía el primer programa informático con la capacidad de aprender. Este programa consistía en un juego de damas, donde cada partida que se jugaba, la máquina mejoraba su juego. En este período de tiempo, Martin Minsky y John McCarthy, junto con Claude Shannon y Nathan Rochester, organizaron la conferencia de Darthmouth, considerada como el evento donde nace el campo de la Inteligencia Artificial (IA) [1].

"History of Machine Learning" (s.f.) define como otro suceso temprano de red neuronal se produjo en 1959, cuando Bernard Widrow y Marcian Hoff crearon dos modelos en la universidad de Stanford. El primero se llamaba ADELINE, donde era capaz de detectar patrones binarios. La próxima generación la llamaron MADELINE, donde era capaz de eliminar el eco en las líneas telefónicas, siendo este mas relevante ya que tenía una aplicación útil para aquel entonces. Pero a pesar de estos grandes avances en el campo de la inteligencia artificial, las agencias financieras de este tipo de investigaciones decidieron no introducir mas fondos económicos debido a unas altas expectativas donde el avance era muy escaso. Por ello, en los siguientes años no hubo apenas avance en el campo, aunque si cabe destacar que se escribe el algoritmo "Nearest Neighbor", siendo este considerado como el nacimiento al campo del reconocimiento de patrones en las máquinas [2].

No fue hasta principios de los 80 cuando el interés comenzó a aumentar nuevamente, donde John Hopfield sugirió crear una red que tuviera líneas bidireccionales. Además, en esta década Japón anunció que estaba centrándose en la investigación en este sector, por lo que la financiación estadounidense se centró también en investigar en esta área. Pero tampoco se realizaron muchos avances, exceptuando de la forma en que se utiliza el aprendizaje automático, donde pasa a ser un estudio orientado al análisis de los datos para luego obtener conclusiones coherentes.

Y llega el siglo XXI, cuando las empresas se dan cuenta de que el aprendizaje automático resulta ser muy útil para el análisis de datos ya que hubo un aumento considerable en la potencia de cálculo.

1.1 Motivación

La principal motivación por la cual se ha decidido escoger esta rama de la inteligencia artificial es debido a que cada día se manejan volúmenes de datos más grandes, por lo que la explotación del análisis de estos puede llegar a ser muy beneficioso para las empresas y las futuras investigaciones. Y como ya sabemos, los datos son el alma de todos los negocios, independientemente del ámbito de este. Y ahí, el aprendizaje automático puede tener un papel fundamental para obtener un valor significativo a tal cantidad de datos, pudiendo incluso influir en decisiones que mantengan a un negocio por delante de su competidor.

Además, resulta bastante interesante el hecho de que el caso de uso del aprendizaje automático es ilimitado, ya que puede aplicarse a cualquier tipo de industria. Desde el sector de producción primario, como la agricultura, pasando por el sector de servicios e incluso por actividades relacionadas con la sanidad.

Por lo tanto, resulta bastante interesante estudiar cuales son los patrones que siguen las empresas para conocer nuestras necesidades y como estudian las posibles tendencias que nos deparan en el futuro.

1.2 Objetivos

Los principales objetivos de este trabajo son conocer los fundamentos del aprendizaje automático y como se aplican estas técnicas a un caso de estudio. Se tratará de entender qué tipos de aprendizaje automático existen y para qué se utilizan, y cómo generar un buen modelo de predicción a partir de algunas técnicas básicas del aprendizaje automático.

Para ello trataremos de obtener un conjunto de datos público (en este caso orientado al área de la salud), con el fin de poder realizar tareas de predicción dirigidas al diagnóstico médico. Realizaremos un análisis descriptivo de los datos mediante técnicas de preparación de los datos, para posteriormente poder aplicar técnicas de aprendizaje automático con las que realizar predicciones orientadas a la detección de enfermedades. Además, evaluaremos el rendimiento de algunas de las técnicas de clasificación más empleadas, como la Regresión Logística, Naive Bayes, Random Forest y KNN (*K-Nearest Neighbours*). Finalmente se compararán los resultados obtenidos por cada una de las técnicas con una serie de métricas habitualmente empleadas en el aprendizaje automático, como son la precisión, la matriz de confusión, o las curvas ROC.

1.3 Contenidos y organización del trabajo

Los contenidos que se han decidido tener en cuenta en este trabajo son:

- Estudiar los fundamentos teóricos del aprendizaje automático supervisado y no supervisado.
- Conocer los distintos tipos de tareas que existen en el aprendizaje automático.
- Aprender los conceptos básicos de como generar un algoritmo de predicción de la manera más precisa posible.
- Estudiar y aplicar técnicas comunes para:
    - Realizar un estudio previo de los datos utilizados para el modelo.
    - Generar el modelo de predicción correctamente.
    - Analizar los resultados del modelo mediante las distintas métricas en función del tipo de aprendizaje automático en el que nos encontramos.
- Aplicar y estudiar algunas de las técnicas más conocidas y empleadas en esta rama de la inteligencia artificial.

La organización del trabajo va a ser la siguiente:

1. Estudio de conceptos básicos del lenguaje de programación Python.
2. Investigación y estudio en distintas fuentes de los fundamentos teóricos del aprendizaje automático.
3. Investigación de algunos tipos de aprendizaje automático.
4. Estudio sobre como realizar un modelo de predicción óptimo, teniendo en cuenta las mejores técnicas a aplicar para poder realizarlo.
5. Investigación de las técnicas predictivas más utilizadas en el mundo del aprendizaje automático.
6. Estudio de las métricas utilizadas para analizar los datos resultantes de un modelo de predicción realizado.
7. Aplicar las técnicas aprendidas para analizar los datos de entrada, en este caso, orientados al diagnóstico médico.
8. Aplicar las técnicas aprendidas para realizar un modelo de predicción óptimo.
9. Realizar distintos modelos para algunas técnicas predictivas como:
    a. Regresión logística
    b. Naive Bayes
    c. Random Forest
    d. KNN (K-Nearest-Neighbour)

10. Aplicar las métricas utilizadas para analizar el modelo realizado.
11. Sacar conclusiones acerca de la comparación realizada de los distintos modelos predictivos con las distintas técnicas aplicadas y pensar en futuras líneas de investigación posibles.

Capítulo 2. Fundamentos del aprendizaje automático supervisado

2.1 Introducción

¿Qué es el aprendizaje automático? En 2019, "Aprendizaje Automático" establece que el aprendizaje automático es una área de las ciencias de computación y una rama de la inteligencia artificial, donde el objetivo es desarrollar técnicas que consigan que las computadoras aprendan de forma autónoma. De este modo, se buscan generar algoritmos que sean capaces de generalizar comportamientos para un conjunto de datos muy grande (2019a) [3].

El aprendizaje automático está relacionado con el reconocimiento de distintos patrones en situaciones en las que el humano no podría detectar, ya sea por estar trabajando con una cantidad de datos muy grande, o por la dificultad de dichos patrones. También se puede ver como una forma de automatización de métodos científicos, mediante métodos matemáticos.

Además, el aprendizaje automático tiene una amplia gama de aplicaciones, ya que se puede aplicar a cualquier campo del cual se quiera hacer una investigación profunda. Algunos ejemplos de aplicación, entre otros, podrían ser:

- Análisis del mercado.
- Reconocimiento de imágenes, voz, caracteres, etc.
- Motores de búsqueda.
- Diagnósticos médicos.
- Recuperación de la información.

El funcionamiento principal del aprendizaje automático es la operación de algoritmos, siendo esto una serie de secuencias compuestas de instrucciones a implementar por la máquina. Dichas instrucciones son las que permiten a la máquina tomar una serie de decisiones en función de la situación de los datos que se han introducido.

El aprendizaje automático se divide principalmente en dos tipos: aprendizaje supervisado y aprendizaje no supervisado, que los veremos con más detalle después.

2.2 Aprendizaje supervisado y no supervisado

El aprendizaje supervisado consiste en una técnica de deducción a partir de un conjunto de datos, donde se busca una correspondencia entre las entradas y las salidas deseadas. Es decir, tienes una serie de variables de entrada (X), en las cuales buscas obtener una variable de salida (Y), mediante el uso de algoritmos que permita el mapeo desde la entrada hasta la salida de los datos.

$$Y = f(x) \quad (1)$$

La finalidad principal será que, a pesar de que se introduzcan nuevos datos de entrada (x), se pueda predecir de forma eficiente la salida (Y) de dichos datos sin necesidad de modificación del algoritmo, a priori. A este tipo de aprendizaje automático se le llama supervisado ya que se estará procesando desde el punto de vista en el cual el desarrollador sabe cuál es la respuesta correcta; entonces, el algoritmo generado realizará predicciones sobre los datos de entrenamiento, y éstos serán corregidos por el desarrollador hasta que él mismo considere que el nivel de procesamiento del algoritmo sea suficiente.

Este aprendizaje se puede agrupar en dos tipos: clasificación y regresión, los cuales se definirán mediante algunos ejemplos, pero los veremos más adelante con más detalle.

La clasificación se puede dividir en dos tipos: clasificación binaria y clasificación multiclase.

Clasificación binaria: estos modelos de aprendizaje automático supervisado predicen con un resultado binario, es decir, sí o no. Algunos ejemplos:

- ¿El correo electrónico es spam o no?
- ¿Sufre enfermedad de corazón?
- ¿Comprarías este libro?

Clasificación multiclase: **e**stos modelos permiten realizar predicciones con un único resultado, o más de dos resultados. Algunos ejemplos:

- ¿Este producto es un libro, una revista o un periódico?
- ¿Cuál ha sido el resultado de los partidos? ¿Gana el equipo local, el visitante o empatan?
- ¿A qué género literario pertenecen estas novelas? ¿Fantástica, ciencia ficción o gótica?

Por otro lado, los modelos de regresión consisten en predecir un valor numérico. Algunos ejemplos:

- ¿A qué precio se venderá esta casa?
- ¿Cuál será la temperatura en España mañana?
- ¿Cuál será el índice de natalidad en 2025?

Sin embargo, en 2017, Julio Cesar Carpio Ticona establece que el aprendizaje no supervisado es diferente respecto al supervisado, por el hecho de que no hay un conocimiento acerca de ello. Es por esto qué la técnica trata los objetos de entrada como un conjunto de

variables aleatorias, es decir, sin un conocimiento previo acerca de las mismas. Por lo tanto, el algoritmo debe ser capaz de reconocer los patrones para poder etiquetar las nuevas entradas del sistema. Se diferencia entre supervisado y no supervisado ya que ahora no se esperan resultados específicos, no es posible predecir los resultados (2017a) [4].

Además, en este caso no existe la diferencia entre el conjunto de datos de entrenamiento y el de prueba. El proceso trata las variables de entrada del sistema con la finalidad de encontrar alguna relación, por lo tanto, el tener más información aportará más robustez al proceso. Entonces, podríamos diferenciarlos por dos motivos principalmente:

- No existen respuestas correctas o incorrectas.
- No existe un desarrollador que corrija el algoritmo debido a que no conoce cuál debe ser la respuesta correcta.

El aprendizaje no supervisado se puede dividir en dos conjuntos de problemas: agrupación o *clustering* y asociación.

- *Clustering:* Cuando tratas de descubrir la agrupación de los datos debido a su relación.
- Asociación: Cuando a través de un comportamiento que dictan los datos, se descubren otras reglas de comportamiento en los mismos.

Modelaremos y aplicaremos técnicas orientadas al aprendizaje supervisado debido a que el conjunto de datos de entrada aporta información etiquetada, es decir, la respuesta se encuentra dentro de los datos. Estas etiquetas indican si los pacientes sufren enfermedad de corazón o no. Por ello, veamos con más detalle los tipos de tareas en el aprendizaje supervisado: la clasificación y la regresión.

2.3 Tipos de tareas: clasificación vs regresión

Como ya sabemos, esta área debe tener la capacidad de formar sistemas que tengan la inteligencia suficiente para poder aprender de manera automática sin la necesidad de realizar una programación dedicada previamente. Es por ello por lo que, después de una fase de aprendizaje, tendremos un algoritmo desarrollado por el humano que sea capaz de proporcionar una salida como resultado de la aplicación de una función a los datos de entrada, ya sea con modelos de regresión o de clasificación. La principal diferencia entre ellos es en el tipo de objetos que se intentan predecir.

2.3.1 Clasificación

En 2017, Álex Rayón define como problema de clasificación cuando la variable a predecir es un conjunto de estados discretos o categóricos [5]. Además, cuando solo se permiten dos

clases, se denomina clasificación binaria, mientras que si se permiten más de dos clases, hablamos de clasificación multiclase.

La clasificación binaria es probablemente el problema más utilizado en el aprendizaje automático, generando así un gran número de desarrollos algorítmicos de gran calidad e importancia en el mundo de la ciencia. Si lo aplicamos a las matemáticas se podría definir mediante el siguiente ejemplo:

• Dado un patrón x, extraído de un dominio X, estimar qué valor tendrá la variable aleatoria asociada $y \in \pm 1$.

Imaginemos que una gran empresa contrata nuestros servicios para poder identificar si el correo electrónico es spam o no. Ante esta situación el valor a estimar está acotado o dos posibles valores, sí o no, es decir, uno o cero. La primera tarea sería la identificación de un patrón dentro del dominio de correos facilitados por la empresa. A partir de ahí, generar un algoritmo y poder ajustarlo para poder actuar de forma eficiente sin necesidad de programación explícita del desarrollador.

El gran nivel computacional de estas herramientas nos permite abordar este tipo de problemas básicos en una gran variedad de entornos prácticos, sin necesidad de tener un conocimiento previo en el sector obligatoriamente.

Además, existen muchas variantes para realizar las estimaciones donde es posible que:

• Nos encontremos ante una secuencia de pares (x_i, y_i) donde y_i necesita ser estimado inmediatamente de manera online. Esto es conocido como aprendizaje en línea.

• Observemos dos conjuntos de pares $X := \{x_1, \dots, x_n\}$ e $Y := \{y_1, \dots, y_n\}$donde se utilizan para realizar la estimación mediante la combinación de x_i e y_i. Esto es conocido como aprendizaje por lotes.

• Se conozca X_0 previamente a la construcción del modelo, conociéndose esto como transducción.

• Se permita elegir el conjunto X para la construcción del modelo de predicción. Esto es conocido como aprendizaje activo.

• Falten coordenadas del conjunto X, es decir, que tenemos información parcial y no completa, derivando esto al problema de estimación con variables faltantes.

• Los conjuntos X y X_0 puedan venir de distintas fuentes de datos, derivando esto al problema de corrección del cambio covariable.

- Los errores de estimación puedan ser penalizados de manera distinta en función del error.

La diferencia principal de la clasificación multiclase es, en este caso, la variable a estimar, y tendrá un dominio de $\{1, \dots, n\}$.

Imaginemos la situación de identificar el riesgo de que una persona pueda sufrir un infarto. Nuestro modelo dependerá en gran medida, de si tenemos en cuenta o no fases prematuras del infarto, considerándolas como una fase de salud óptima. También aplicándolo al otro extremo, en el que sobre clasificamos erróneamente las fases avanzadas del infarto de corazón.

2.3.2 Regresión

Se define como problema de regresión cuando la variable que queremos predecir es numérica, como las ventas de una empresa a partir de los precios a fijar (Álex Rayón, 2017). En este caso el objetivo principal es estimar el valor de y, siendo $y \in \mathbb{R}$, mediante un patrón x. Por ejemplo, tal vez queramos estimar cuál es precio de una vivienda. Para ello, como patrón tendremos las ventas de otras viviendas con características lo más similares posibles y en función de ello se podría realizar dicha estimación.

Este tipo de análisis incluye distintas formas de modelar el algoritmo, como SVM (Support Vector Machines) y la regresión lineal, siendo está una de las técnicas más utilizadas en estos tipos de análisis.

2.4 Conjuntos de datos: training/test, validación cruzada, LOO, bootstrapping

Previo a introducir la definición de los distintos conjuntos de datos y las distintas técnicas que se utilizan para realizar dichos conjuntos, cabe destacar la necesidad de obtener datos de entrenamiento etiquetados. Para ello, los datos de entrada deberán estar formados por las distintas características que lo forman y su etiqueta, es decir, a qué clase pertenece. Por ejemplo, como características predictivas se tendría la edad, el género, el colesterol en sangre y la presión arterial del paciente, mientras que la etiqueta se definiría como un uno, si el paciente sufre algún tipo de enfermedad, o cero, si el paciente resulta estar sano.

Veamos la metodología de conjuntos de entrenamiento y prueba. ¿Qué sucede si solo tenemos un conjunto de datos? Por ejemplo, extraemos datos de la distribución oculta, dividimos el conjunto grande en dos más pequeños: uno para entrenamiento y otro para prueba. Debemos mantenerlos separados. Además, realizamos una selección aleatoria para que datos del mismo tipo queden agrupados. ¿Qué porcentaje de los datos debe tener cada una de las divisiones? Esto supone un conflicto ya que, si el conjunto de datos de entrenamiento es muy grande, la capacidad de aprendizaje será mucho mayor, pero puede existir un sobreentrenamiento del modelo (problema que se definirá más adelante), y a la hora de analizar

el algoritmo con el conjunto de prueba nuestro modelo será peor de lo esperado. Si el conjunto de datos de prueba es muy grande, los intervalos de confianza a la hora de analizar los datos serían mucho más precisos, pero la falta de datos de entrenamiento puede suponer una sobregeneralización del modelo.

Si nos encontramos ante un caso donde el conjunto de datos es de millones de registros, con que el 15%-20% de los datos formen parte del conjunto de prueba es suficiente. Sin embargo, si el conjunto de datos es mucho más pequeño, debemos usar una validación combinada. Cabe mencionar que los datos de prueba no deben usarse para entrenar el modelo de predicción, ya que esto lo único que aportará serán pronósticos falsos respecto a la eficacia del algoritmo de predicción generado. Como hemos visto anteriormente, si a la hora de probar nuestro modelo con el conjunto de datos de prueba obtenemos un 100% de los resultados correctos, seguramente haya un error de sobreajuste.

Veamos las distintas técnicas que existen para una división eficiente del conjunto de datos.

2.4.1 Validación cruzada (*Cross Validation*)

Amazon Web Services (s.f.) define la validación cruzada como una técnica para evaluar modelos de aprendizaje automático mediante el entrenamiento de varios modelos en subconjuntos de los datos de entrada disponibles y evaluarlos con el subconjunto complementario de los datos. La validación cruzada se utiliza para detectar el sobreajuste, es decir, en aquellos casos en los que no se logre generalizar un patrón [6].

Hemos comprobado cómo de complicado es saber qué porcentaje aplicar al subconjunto de entrenamiento y qué porcentaje al subconjunto de prueba, ya que supone un problema de sobreajuste osobregeneralización, respectivamente. Aquí es donde se aplica una de las técnicas más utilizada en la validación cruzada, denominada validación cruzada de K iteraciones *(K-Fold Cross Validation)*.

Además, en un estudio reciente (Julio Cesar Carpio Ticona, 2017a) define que en este tipo de validación cruzada, los datos se dividen en *k* subconjuntos, donde el modelo se repetirá *k* veces de modo que cada vez sea uno de los subconjuntos el que se utiliza como subconjunto de prueba, y los otros subconjuntos k-1 se unen para formar un único subconjunto de entrenamiento. Así, el error se promedia sobre todos los *k* ensayos para conseguir una alta efectividad en el modelo [4].

Una de las mayores ventajas de esta técnica es que reduce el sesgo, ya que estamos utilizando la mayoría de los datos para el ajuste del algoritmo, pero a la vez reduce la variación, dado que al mismo tiempo son estos datos los que se utilizan en el conjunto de

validación/prueba. Es por ello por lo que intercambiar los conjuntos de entrenamiento y prueba resulta muy efectivo. Generalmentese utilizarán entre 5 y 10 iteraciones, pero no existe ninguna ley fija que nos obligue a hacer esto. No obstante, no se recomienda que sea un número de iteraciones mucho menor ya que la técnica perderá la eficacia que se consigue con ello.

Además, Prashant Gupta (2017) comenta en un estudio reciente que también existe una variante a la validación cruzada de K iteraciones, denominada validación cruzada de K iteraciones estratificada *(Stratified K-Fold Cross Validation).* Esta técnica se aplica en casos donde existe un gran desequilibro en las variables del conjunto de datos. Por ello, se realiza una variación donde cada iteración contenga aproximadamente el mismo porcentaje de muestras de cada clase para el conjunto de datos completo, o en el caso de problemas de predicción, que el valor medio de respuesta en cada una de las iteraciones sea lo más parecido posible [7].

2.4.2 LOO *(Leave-One-Out Cross Validation)*

Este tipo de validación es un caso particular de la técnica de validación LPOO *(Leave-P-Out Cross Validation)*, qué consiste en que aísla *p* puntos del subconjunto de datos de entrenamiento para utilizarlos como subconjunto de datos de validación. Es decir, si existen *x* puntos en el conjunto de datos original, los puntos *x-p* se utilizarán como subconjunto de entrenamiento, mientras que los puntos *p* restantes se utilizarán como subconjunto de validación.

Una de las particularidades de este método de validación es que es necesario entrenar y validar el conjunto de datos para todas las combinaciones posibles, donde se deberá promediar el error de cada uno de los ensayos para que sea efectivo.

Respecto a LOO, consiste en asignar a p un valor de 1. Esto quiere decir que el conjunto de datos será entrenado por todos los puntos excepto uno, que se utilizará para la validación y predicción del modelo. Como hemos comentado anteriormente, para poder obtener el error en $p \geq 2$ se debe realizar un promedio por cada punto p que aislemos del conjunto de datos de entrenamiento. El método LOO se suele utilizar más respecto a otras debido a que no sufre este tipo de cálculo intensivo, ya que el número de combinaciones posibles es igual al número de puntos de datos en la muestra original.

A priori, el cálculo de la evaluación dada por LOO puede parecer costoso; sin embargo, los modelos que tengan un buen equilibro pueden realizar predicciones LOO de la misma manera computacionalmente hablando que cualquier otro tipo de predicción. Esto quiere decir que calcular el error dado por LOO no cuesta más computacionalmente que otros tipos de técnicas y es una forma más eficiente de evaluar otros modelos.

2.4.3 Bootstrapping

Al igual que la validación cruzada, el método de *bootstrapping* también es un método de remuestreo de los datos. Una de las principales características de este método que se diferencian respecto a la validación cruzada es que el remuestreo de los datos para un conjunto de datos *n* se realiza de manera uniforme, (al igual que la validación cruzada) solo que con remplazamiento. Y, ¿qué beneficio aporta qué se realice con remplazamiento? En 2015, Alice Zheng establece que una de las principales ventajas es que produce conjuntos de datos que tienen el mismo número de registros que un conjunto de datos original. Además, al igual que la validación cruzada, también se podrán calcular elementos como la varianza e intervalos de confianza de predicción [8].

Está técnica no es más que una inspiración de otra técnica de remplazamiento denominada *jackknife*, que es muy similar a LOO.

El funcionamiento de esta técnica es la siguiente:

1. Seleccionamos un punto de datos al azar de manera uniforme.
2. Dichos puntos se agregarán al subconjunto de datos *bootstrapped.*
3. Se vuelven a introducir dichos puntos en el conjunto de datos.
4. Se repite el procedimiento.

El hecho de volver a introducir los puntos en el conjunto de datos es debido a que lo que se estaría evaluando en ese caso es una muestra real del conjunto de datos, además de ser valores reales del mismo. Esto es porque lo que en realidad obtenemos es una distribución subyacente, proporcionando una distribución fiable de los datos. Es decir, está técnica simula nuevos registros a partir de registros reales del conjunto de datos. Y, al ser una distribución subyacente y no igual, nos permite realizar dicha devolución de los datos por cada repetición del procedimiento.

Pero también existe la posibilidad de que el mismo registro aparezca múltiples veces. Teóricamente, si el sorteo aleatorio se repite *n* veces, entonces la proporción esperada de instancias únicas en el conjunto generado es de $1 - 1/e \approx 63.2\%$.

2.5 Sobreajuste

El sobreajuste, más conocido como *overfitting*, es un término muy importante y conocido en el mundo del aprendizaje automático. En 2017, Ricardo Moya establece que resulta ser un error muy común en los desarrolladores que empiezan a introducirse en esta área, ya que se tiende a buscar la perfección en el modelo de predicción, provocando un sobreentrenamiento en el

modelo. Además, Ricardo Moya menciona que el problema que origina un sobreentrenamiento de los datos de entrada es que cuando se introduzcan nuevos datos en el modelo, éste no será capaz de predecirlos correctamente, por lo que el error tenderá a ser más elevado, por lo que la predicción o el modelo no es correcto [9]. Pongamos un ejemplo para conocer el término con más detalle.

Imaginemos una situación en la que se está intentando predecir cuál va a ser el resultado entre Rafael Nadal y Marcel Granollers, número 2 y número 91 del mundo según la clasificación de la ATP, respectivamente. Sabemos que Rafael Nadal es uno de los mejores jugadores de la historia y del mundo actualmente, pero da la casualidad de que, en los datos de entrenamiento utilizados para realizar dicha predicción, sólo aparecen dos enfrentamientos entre ambos en el cual Marcel Granollers le ha ganado en ambos. Sin embargo, también tenemos en los datos que la mayoría de los partidos jugados por Rafael acaban en victoria, y que en el 95% de los casos en los que se enfrenta a un español, gana.

Aunque influyen muchos factores, viendo estos datos un humano podría predecir sin ninguna duda que, a pesar de que esos partidos contra Marcel no los ganó, si se vuelven a enfrentar, sería Rafael quien ganaría ya que es uno de los mejores jugadores del mundo actualmente. El problema es que el algoritmo de aprendizaje automático no tiene esa capacidad de razonamiento que el humano sí tiene, pero sí se espera que el resultado sea parecido y tenga cierta coherencia al igual que el de un humano. Pero ¿qué resultados obtendría el algoritmo si se produjera un sobreajuste?

En este caso, los datos de entrada muestran que Marcel ha sido quién ha ganado ambos partidos, entonces debería ser una nueva victoria por parte de Marcel Granollers. Pero también existe el concepto de *underfitting*, en el que, al generalizar demasiado, como Rafael Nadal gana en el 95% de los casos, saldría una victoria clara por parte de Rafael.

En resumen, lo que se pretende dar a entender es que el modelo no tiene por qué predecir todos los casos a evaluar, si no que la predicción sea lo suficientemente buena como para que la mayoría de los casos sean correctos, permitiendo así tener un margen de error puntual en el modelo.

Un estudio reciente ("¿Qué es el Machine Learning?", 2017) muestra estos conceptos de manera gráfica para los problemas de clasificación y regresión [9]:

- Clasificador que aproxime f: $R^2 \rightarrow \{*, x\}$:

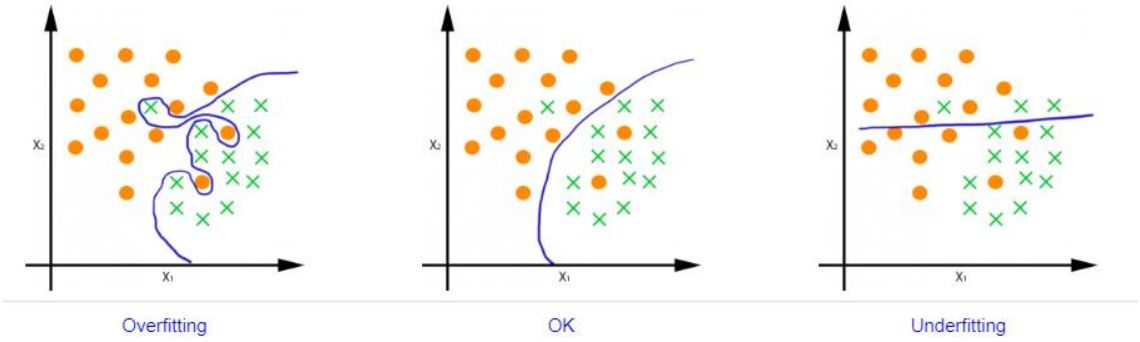

Ilustración 1. Gráficas modelo de clasificación [9].

- Modelo de regresión que aproxime f: R→R:

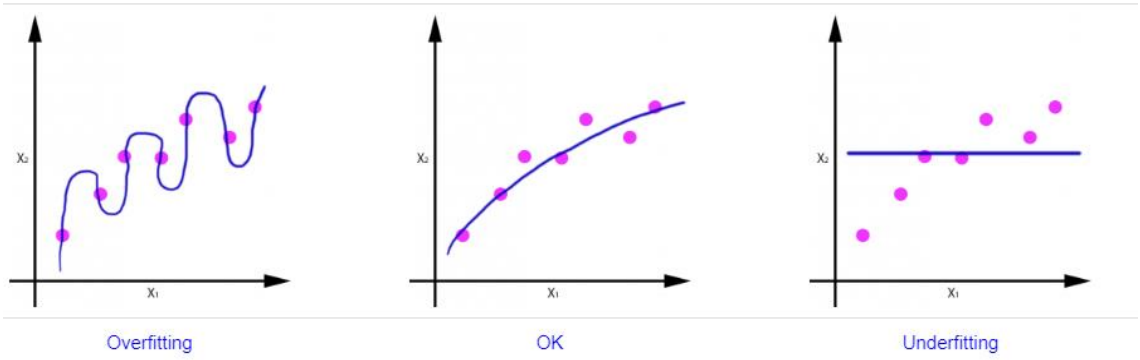

Ilustración 2. Gráficas modelo de regresión [9].

Veamos por ejemplo para el caso de la regresión, el error que se cometería si nos llegase un nuevo dato de entrada (punto rojo), para cada una de las funciones obtenidas:

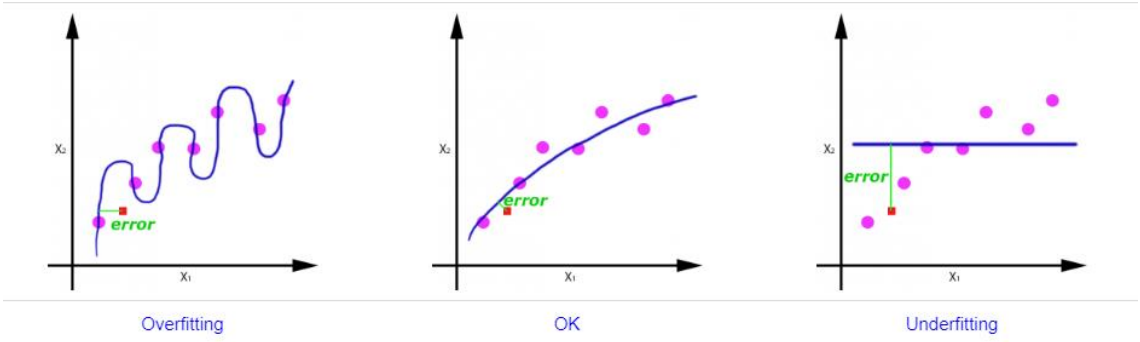

Ilustración 3. Gráficas modelo de regresión con error introducido [9].

Y, ¿cómo prevenir este sobreajuste o sobregeneralización? Una de las mejores formas de prevenir esto y por lo tanto generar el mejor algoritmo posible de predicción, es la división del conjunto de datos en subconjuntos, denominados como conjuntos de entrenamiento y de prueba.

2.6 Preparación de los datos

La primera tarea que se debe realizar para desarrollar un algoritmo de manera eficiente es la preparación de los datos, ya que nuestro conjunto de datos a analizar puede tener una serie de imperfecciones que pueden hacer que nuestro modelo muestre predicciones incorrectas.

Por ello, se deberá realizar un preprocesamiento y limpieza de los datos para que el conjunto de datos se pueda usar para el algoritmo de aprendizaje automático a desarrollar. Esto es debido a que, normalmente, al estar procesando volúmenes tan grandes de datos, pudiendo proceder de varias fuentes distintas, estos pueden contener registros ruidosos, es decir, incompletos, dañados o introducidos de forma incorrecta en el conjunto de datos, pudiendo producir resultados engañosos a la hora de realizar la predicción. En 2017, Microsoft establece cuales son las principales causas por las que un conjunto de datos puede presentar problemas [10]:

- Con ruido: los datos introducidos presentan errores o valores muy atípicos.
- Incompletos: los datos contienen registros con características incompletas.

Por lo tanto, la preparación de los datos resulta ser una tarea imprescindible previa a la realización del modelo. Existen varias formas de preparar los datos de una forma óptima:

- Detección de errores y limpieza de los datos: existen varias técnicas para detectar todos estos problemas de una manera rápida respecto a los grandes volúmenes de datos con los que se suele trabajar. La detección de *outlayers*, mediante los histogramas o los *box-plot* son algunas de las técnicas más comunes para obtener estas métricas. A partir de la detección previamente realizada, procedemos a la limpieza del conjunto de datos eliminando los valores atípicos, limpiando los datos con ruido y rellenando los valores faltantes.

- Datos faltantes: para rellenar los datos faltantes existen varias posibilidades para realizarlas de forma correcta. Por un lado, debemos analizar si es un registro que nos va a aportar la información suficiente o si simplemente se pudiera ignorar/eliminar. Por otro lado, existen formas de rellenar dichos valores obteniendo la media, la moda o el valor previsto que debería de existir en esa característica.

- Transformación de los datos y selección de las características: en muchas ocasiones puede darse el caso donde no todas las características del conjunto de datos nos den una información relevante. Es por ello por lo que puede existir la posibilidad de descartar alguna de estas características para tener un control de los datos más sencillo.

Veamos algunas de las técnicas para realizar correctamente la preparación de los datos.

2.6.1 Histogramas

En 2019, RAE – ASALE define los histogramas como una representación gráfica de una distribución de frecuencias por medio de rectángulos, cuyas anchuras representan intervalos de la clasificación y cuyas alturas representan las correspondientes frecuencias [11].

En otras palabras, no es más que una representación gráfica de la distribución de los datos. Esta técnica estadística sirve para identificar los distintos datos que resultan ser atípicos en cada una de las características del modelo. Definamos mediante la siguiente ilustración un histograma.

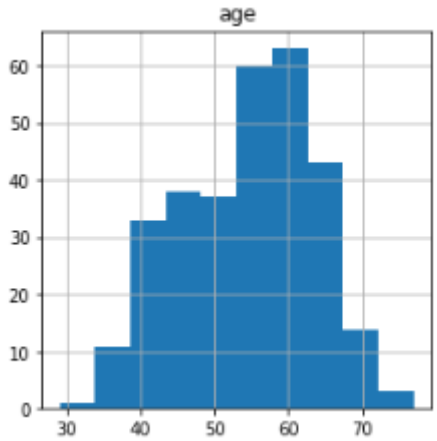

Ilustración 4. Ejemplo histograma.

Las principales características que observamos en la imagen son:

- Solamente representa una característica, en este caso la edad.
- Cada una de las barras representa los valores o intervalos de valores dentro de la característica.
- El eje Y representa la frecuencia de cada uno de los valores.
- El eje X representa el valor numérico de cada uno de los posibles valores dentro del conjunto de datos.
- Cada una de las barras tendrá la misma anchura.
- La característica que se representa en el histograma debe ser continua.

Además, dichas representaciones pueden mostrarse tanto en horizontal como en vertical, donde lo único que cambiará será el significado de los ejes *X* e *Y*. Cabe destacar la importancia de no confundir los histogramas con los gráficos de barras, ya que estos solo sirven para comparar las distintas categorías de una variable.

2.6.2 Caja de bigotes *(box-plot)*

"Diagramas de Caja y Bigotes" (s.f.) define los diagramas de caja y bigotes como una presentación visual que describe varias características importantes, al mismo tiempo, tales como la dispersión y la simetría [12] .Esta representación estadística está formada por los extremos (superiores e inferiores), los cuartiles (superiores e inferiores), la mediana y los datos atípicos.

Definamos la función de cada una de sus componentes con ayuda de la siguiente ilustración:

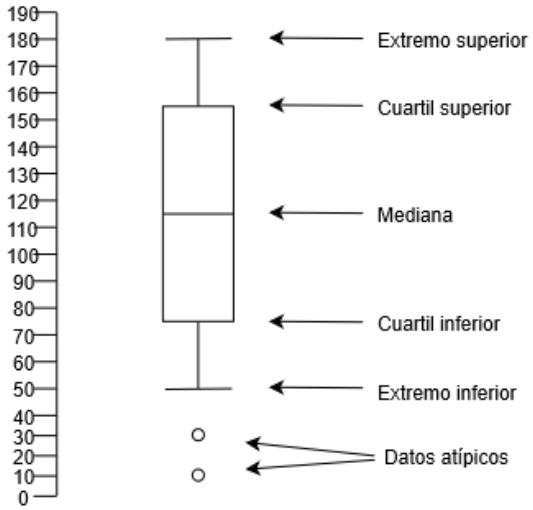

Ilustración 5. Definición y representación caja de bigotes.

- Extremo superior: valor máximo dentro de la distribución.
- Extremo inferior: valor mínimo dentro de la distribución.
- Cuartil superior: representa el valor que sobrepasa al 75% de los valores de la distribución.
- Cuartil inferior: representa el valor mayor que el 25% de los valores de la distribución.
- Mediana: representa el valor de la variable que ocupa el lugar central en un conjunto de datos ordenado.
- Datos atípicos: vualquier valor que no se encuentre dentro del rango entre el extremo superior y el extremo inferior.

2.7 Observaciones, atributos de entrada

Siempre es importante realizar un análisis previo de las distintas características o atributos de entrada a los que nos enfrentamos de los distintos conjuntos de datos. Como hemos comentado anteriormente, resulta imprescindible realizar una selección de las características del conjunto de datos, ya que simplifica en gran medida el manejo de los datos a la hora de realizar los distintos análisis. Pero para ello, previamente se debe realizar un estudio de los atributos, identificando así cuales son válidos y cuáles se pueden descartar para nuestro modelo. Además, esto supone un factor importante, debido a que de esta manera seremos capaces de entender los datos que estamos intentando analizar porque habremos realizado un análisis de cada una de las características y podremos entender los valores que estas aportan.

En el análisis práctico de nuestro conjunto de datos realizaremos un estudio en profundidad de cada una de las características, ya que, al ser un conjunto de dato orientado al diagnóstico médico, el conocimiento en esta área resulta ser insuficiente para poder entender los valores que aparecen en estos datos.

Capítulo 3. Métricas

3.1 Clasificación

Las métricas en el aprendizaje automático son técnicas que se utilizan para realizar una evaluación correcta de los datos. Pero, a pesar de que existe una gran lista de distintas métricas que se pueden aplicar, solo veremos algunas de ellas, diferenciando entre métricas para problemas de clasificación y métricas para problemas de regresión.

3.1.1 FP, FN, TP, TN, Sensibilidad y especificidad (clasificación binaria)

Previamente a conocer las distintas métricas que se utilizan, debemos definir los siguientes elementos que se utilizan para la evaluación de estas. A la hora de realizar un algoritmo con la intención de predecir los datos mediante un aprendizaje supervisado, es decir, sabiendo los resultados esperados, se pueden dar cuatro casos. En un estudio reciente ("Machine Learning: Selección Métricas de clasificación, 2019") se definen los distintos elementos [13]:

- TP (True Positive): cuando en el conjunto de datos el valor era uno, es decir, verdadero, y en la predicción también es uno.
- FN (False Negative): cuando en el conjunto de datos el valor era uno, pero en la predicción el resultado es cero.
- FP (False Positive): cuando en el conjunto de datos el valor era cero, es decir, falso, y en la predicción el resultado es uno.
- TN (True Negative): cuando en el conjunto de datos el valor era cero y en la predicción también es cero.

También existen otros parámetros como la sensibilidad y la especificidad (que son dependientes de los parámetros anteriormente definidos) dentro de la estadística, que son:

- Sensibilidad: la sensibilidad es la división de los verdaderos positivos y la suma de estos con los falsos negativos. Se define mediante la siguiente ecuación:

$$\boldsymbol{Sensibilidad} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (2)$$

Un ejemplo que definiría la sensibilidad sería la capacidad del modelo a la hora de detectar la enfermedad entre sujetos enfermos.

- Especificidad**: l**a especificidad es la división de los verdaderos negativos y la suma de estos con los falsos positivos**.** Se define mediante la siguiente ecuación:

$$\boldsymbol{Especificidad} = \frac{\boldsymbol{TN}}{\boldsymbol{TN+FP}} \quad (3)$$

Un ejemplo que definiría la especificidad sería la capacidad del modelo a la hora de detectar individuos sanos entre sujetos que no tienen ningún tipo de enfermedad.

Para entender la importancia de estas métricas, pongamos un ejemplo. Imaginemos la situación en la que, mediante un algoritmo de predicción, debemos identificar el número de pacientes que sufren una enfermedad de corazón.

En este caso, la sensibilidad se definiría como la capacidad para acertar los pacientes que sufren una enfermedad en el corazón, y la especificidad como la capacidad para predecir correctamente los pacientes sanos, es decir, que no sufren ninguna enfermedad de corazón.

Como desarrolladores, una de las validaciones que se deben realizar es comprobar que nuestro algoritmo ha realizado correctamente la predicción y, para ello, se evalúa la especificidad y la sensibilidad del modelo. Para identificar que el modelo es correcto, lo ideal sería que la sensibilidad y la especificidad tuvieran un valor igual a uno. Esto indicaría que el modelo ha identificado correctamente tanto los verdaderos positivos como los verdaderos negativos, o que no existen falsos negativos y falsos positivos. En este caso, al ser una clasificación binaria, es decir, que los valores son verdaderos o falsos, resulta más sencillo identificar estos verdaderos o falsos positivos, pero para ello se debe determinar un punto de corte.

El punto de corte, matemáticamente, se define como puntos importantes a la hora de representar gráficamente una función. En nuestro caso se definiría como un punto/línea que separa entre verdaderos y falsos positivos.

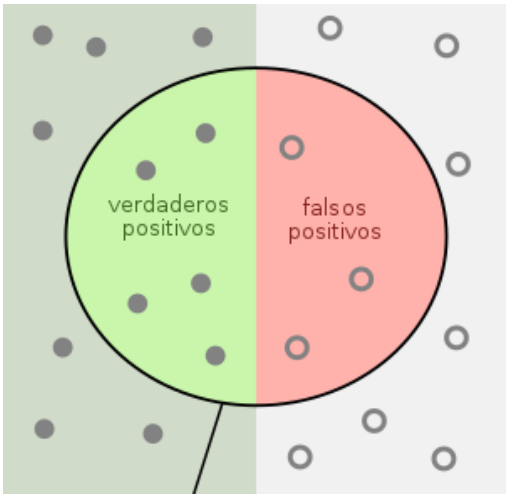

Ilustración 6. Sensibilidad, especificidad y punto de corte [14].

Este punto de corte indica a partir de qué valor de probabilidad a posteriori el registro será clasificado como clase positiva o negativa. Si este punto de corte lo ponemos para cada uno de los valores entre cero y uno, obtenemos la curva ROC (Receiver Operator Characteristic), que

definiremos más adelante. Además, una técnica eficiente para elegir el punto de corte óptimo será mediante el índice de *Youden*:

$$\boldsymbol{Y = Sensibilidad + Especificidad - 1} \quad (4)$$

donde a mayor valor de Y, mejor será el punto de corte.

3.1.2 Matriz de confusión

Una vez definidos *FP, FN, TP y TN*, podemos proceder con el análisis de la matriz de confusión. Para ello, sigamos con el ejemplo utilizado anteriormente. Como se ha comentado anteriormente, la variable de salida tiene el siguiente significado:

- 1: paciente sufre enfermedad de corazón.
- 0: paciente no sufre enfermedad de corazón.

	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

Ilustración 7. Matriz de confusión.

Como podemos observar, la matriz de confusión resulta ser una técnica muy útil para identificar rápidamente de manera visual tanto los verdaderos y falsos positivos como los verdaderos y falsos negativos. De esta manera se puede identificar fácilmente dónde nos estamos equivocando en nuestro modelo de aprendizaje supervisado.

3.1.3 Exactitud *(Accuracy)*

La exactitud se define como el porcentaje total de elementos clasificados correctamente, es decir, el número de predicciones que se han realizado correctamente dividido por el número total de predicciones.

$$\boldsymbol{Accuracy = \frac{Correct\ predictions}{Total\ predictions}} \quad (5)$$

Sin embargo, aplicado a la clasificación binaria y por lo tanto, aplicando la matriz de confusión, la exactitud también se puede calcular en función de estos términos como:

$$\boldsymbol{Accuracy = \frac{TP+TN}{TP+TN+FP+FN}} \quad (6)$$

Apliquemos está formula al ejemplo. Imaginemos que nos encontramos ante 300 muestras, donde los resultados han sido los siguientes:

TRUE POSITIVE (TP)	**FALSE POSITIVE (FP)**
Valor esperado: 1 Maligno (Sufre enfermedad)	Valor esperado: 0 Benigno (No sufre enfermedad)
Predicción del modelo: 1 (Sufre enfermedad)	Predicción del modelo: 1 (Sufre enfermedad)
Número de TP: 279	Número de FP: 16
FALSE NEGATIVE (FN)	**TRUE NEGATIVE (TN)**
Valor esperado: 1 Maligno (Sufre enfermedad)	Valor esperado: 0 Benigno (No sufre enfermedad)
Predicción del modelo: 0 (No sufre enfermedad)	Predicción del modelo: 0 (No sufre enfermedad)
Número de FN: 2	Número de TN: 3

Tabla 1. Ejemplo TP, FP, FN y TN (Matriz de confusión).

Por lo tanto, la exactitud del modelo según estos datos sería:

$$Accuracy = \frac{279 + 3}{(279 + 3 + 16 + 2)} = 0{,}94 = 94\ \%$$

Esto quiere decir que, de cada 100 predicciones, 94 de ellas serán correctas, lo cual a priori parece un resultado bastante óptimo el cual indica un correcto funcionamiento del modelo. Realicemos un análisis del resultado obtenido.

De las 300 muestras que tenemos, el modelo ha sido capaz de identificar 282 de forma correcta, mientras que solo 18 de forma incorrecta. Respecto a los pacientes que sufren una enfermedad en el corazón, este modelo resulta ser muy exacto ya que, de los 281 pacientes con enfermedad, ha sido capaz de predecir 279. Esto supone que en aproximadamente un 99% de los casos el modelo será capaz de predecir correctamente este tipo de caso.

Veamos el caso en el que el paciente no sufre una enfermedad de corazón. Aquí, observamos como en 3 pacientes que no sufren enfermedad, el modelo ha realizado la predicción correctamente, pero se ha equivocado en 16 de ellos. Esto implica que, de cada 19 pacientes, 16 van a obtener un diagnostico falso, ya que ellos no sufren ninguna enfermedad de corazón. En otras palabras, aproximadamente 8 de cada 10 pacientes sanos estarán obteniendo un resultado falso, siendo esto un porcentaje de error muy grande.

Por lo tanto, podemos llegar a la conclusión de que obteniendo únicamente la exactitud no es suficiente para determinar si nuestro modelo supervisado es bueno o no. Existen otras métricas como la precisión y la exhaustividad que, unida a la exactitud, la evaluación del algoritmo será mucho más completa.

3.2 Regresión

En problemas de regresión también existen una serie de métricas para evaluar el modelo. A continuación, definiremos algunas de ellas.

3.2.1 Error cuadrático medio (MSE)

Se define mediante la siguiente ecuación:

$$\boldsymbol{MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \quad (7)$$

Donde y_i significa el resultado esperado e $\hat{y}_i$ la predicción del modelo.

Esta métrica de regresión es muy común, ya que es capaz de medir el error promedio de la predicción realizada por nuestro modelo. Básicamente, para cada punto i dentro del conjunto de datos N, calcula la diferencia entre la predicción realizada y el resultado real, y a partir de ahí, calcula el promedio. Gráficamente calcula la distancia o error que existe en la línea de regresión y cada uno de los distintos puntos. Por lo tanto, cuanto menor sea el resultado de esta métrica, mejor será nuestro modelo, ya que implicará que el número de predicciones realizadas correctamente será elevado.

3.2.2 Raíz cuadrada MSE (RMSE)

Se define mediante la siguiente ecuación:

$$\boldsymbol{RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{MSE}} \quad (8)$$

Por norma general, a la hora de realizar las métricas de nuestro modelo de regresión, se suele utilizar *RMSE* antes que *MSE*. Esto se debe a que, gracias a la raíz cuadrada, se obtienen las mismas unidades que la cantidad de unidades que se muestran en el eje vertical de nuestro modelo de regresión.

3.2.3 Error medio absoluto (MAE)

Se define mediante la siguiente ecuación:

$$\boldsymbol{MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|} \quad (9)$$

El error se calcula como el promedio de la diferencia absoluta entre la predicción realizada y el resultado real.

La principal diferencia es que, al estar realizando el valor absoluto de cada una de las diferencias, todas estas se ponderan por igual en el promedio. Además, esta métrica sirve para identificar más fácilmente los errores más notorios, dato que no se identifica con *MSE*.

3.3 Ranking: Curvas ROC y AUC

Google Developers (s.f.-a) definen la curva ROC como un gráfico que muestra el rendimiento de un modelo de clasificación en todos los umbrales de la clasificación [15]. Esta curva representa dos parámetros:

- Tasa de verdaderos positivos (TPR), que se define según la siguiente ecuación:

$$\boldsymbol{TPR = \frac{TP}{TP+FN} = Sensibilidad} \quad \textbf{(10)}$$

- Tasa de falsos positivos (TFR), que se define según la siguiente ecuación:

$$\boldsymbol{FPR = \frac{FP}{FP+TN} = 1 - Especificidad} \quad \textbf{(11)}$$

Un ejemplo de una gráfica representando una curva ROC sería la siguiente:

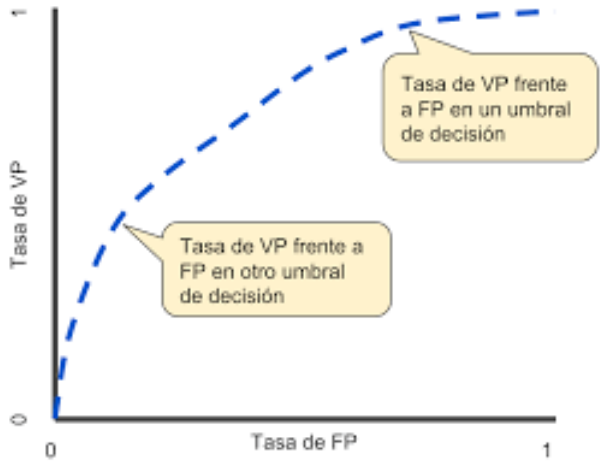

Ilustración 8. Tasa de VP frente a FP en diferentes umbrales de clasificación [15].

Como podemos observar, la curva ROC consiste en una representación de TPR frente a FPR en los distintos umbrales de decisión que se definen. Sin embargo, para calcular estos puntos deberíamos evaluar el algoritmo de aprendizaje automático realizado con distintos umbrales de decisión elegidos de forma manual.

Además, Google Developers (s.f.-a) define AUC *(Area Under the Curve)* como toda el área que se encuentra por debajo de la curva ROC trazada en función de TPR y FPR. Consiste en una métrica que considera todos los umbrales de clasificación posibles [15]. En otras palabras,

es la probabilidad de que el modelo empleado por el desarrollador al realizar una predicción positiva, dicha predicción sea realmente positiva, respecto a que la predicción sea negativa, y el modelo haya predicho un resultado positivo. A esta métrica también se le denomina AUROC *(Area Under the Receiver Operating Characteristics)*.

Esta combinación suele ser utilizada para medir el rendimiento de problemas de clasificación en varios umbrales. Básicamente indica como de capaz es el modelo de distinguir entre las distintas clases, es decir, positivos y negativos. Cuanto mayor sea el valor de AUC, mejor será el modelo. Entendamos las distintas situaciones que se pueden dar mediante las siguientes gráficas:

- **Caso 1:**

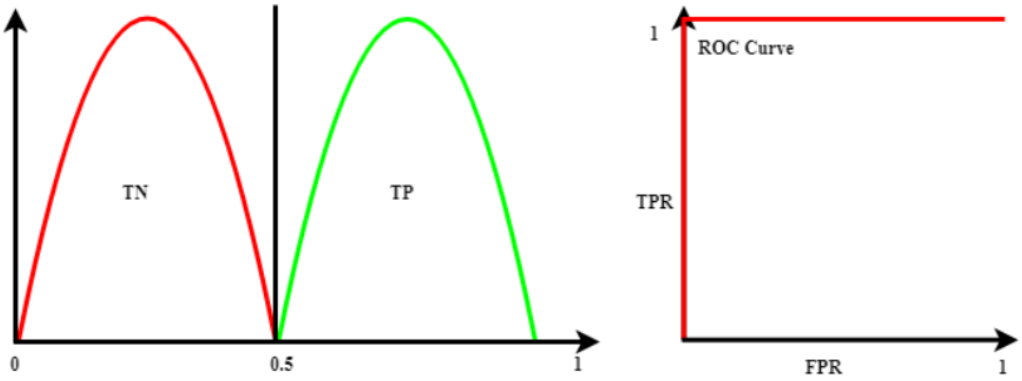

Ilustración 9. Caso ideal Curva ROC.

Este caso define claramente lo que sería un caso ideal, debido a que las curvas de verdaderos positivos y negativos no se superponen en ningún punto. Es decir, el modelo es capaz de distinguir perfectamente entre clases positivas y clases negativas. Por lo tanto, el valor de AUC en este caso sería igual a uno.

- **Caso 2:**

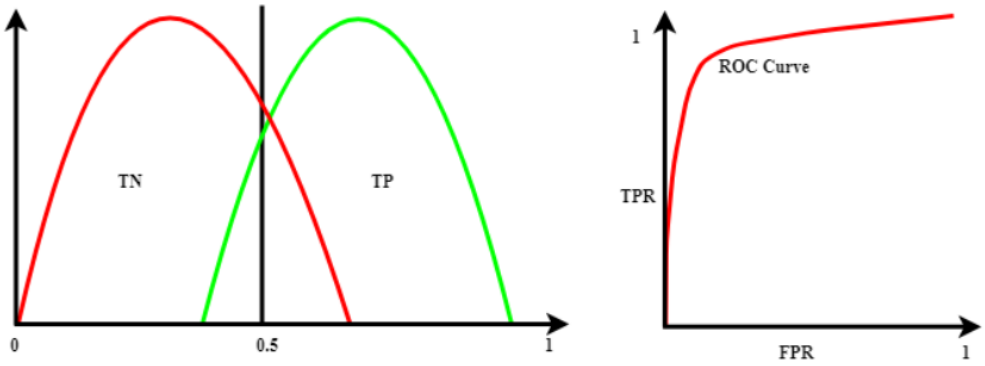

Ilustración 10. Caso no ideal, donde se introducen predicciones erróneas.

En este caso sí apreciamos cómo se superponen las dos curvas, por lo que se estarían introduciendo predicciones erróneas. Suponiendo el caso donde el valor de AUC a partir de esta representación es de 0.8, significaría que existe un 80% de posibilidades de que el modelo sea de capaz de distinguir correctamente entre clases positivas y negativas.

- **Caso 3:**

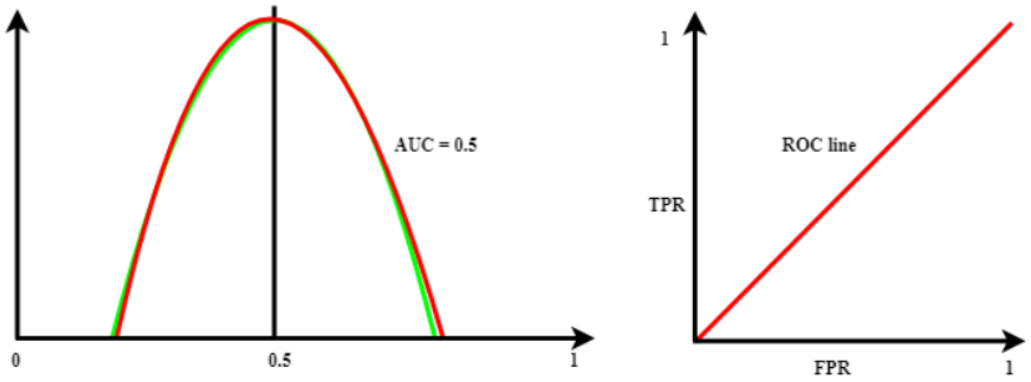

Ilustración 11. Caso de predicciones aleatorias, modelo inútil.

Este caso representa la peor situación posible, donde el modelo no tiene ningún tipo de capacidad para distinguir entre clases positivas y negativas. La representación de la curva ROC pasaría a mostrarse como una línea diagonal constante, indicando que el área AUC tiene un valor de 0.5, es decir, que cada una de las entradas por predecir tienen las mismas posibilidades de clasificarse tanto positivas como negativas.

- **Caso 4:**

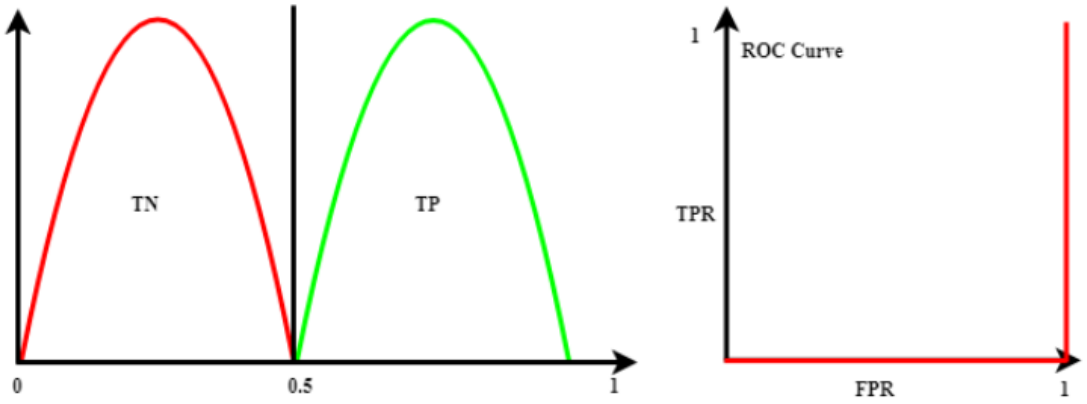

Ilustración 12. Caso donde la predicción es totalmente inversa.

Este caso representa cuando el valor de AUC es cero, ya que el modelo está realizando la predicción de la manera completamente inversa. Es decir, está realizando la predicción de las clases positivas como negativas y viceversa.

Capítulo 4. Técnicas predictivas

Existen muchas técnicas/algoritmos para afrontar los distintos problemas que se presentan en el aprendizaje automático, tanto de clasificación como de regresión. A continuación, se explicarán algunas de las técnicas básicas más utilizadas.

4.1 Regresión lineal

En un estudio reciente, Google Developers, s.f.-b, define la regresión lineal como un tipo de modelo de regresión que da como resultado un valor continuo a partir de una combinación lineal de atributos de entrada [16].

En el aprendizaje automático se trata de uno de los modelos más básicos y a la vez más utilizados en los problemas de regresión. Además, se trata de un análisis donde el número de variables independientes es uno y existe una relación lineal entre la variable independiente *x* y la variable dependiente *y*. Principalmente trata de encontrar la relación lineal que puede existir entre las variables de entrada independientes *x* respecto a las de salida y. El modelo se puede definir mediante la siguiente expresión matemática:

$$\boldsymbol{Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_i X_i} \quad \textbf{(12)}$$

donde:

- Y_i es la variable dependiente como se ha comentado anteriormente.
- β_0 se define como la ordenada al origen, indicando el valor de la variable Y_i cuando X_n es igual a cero.
- $\beta_1, \beta_2, \ldots, \beta_i$ se define como la pendiente de la recta, indicando el cambio de la variable Y_i por cada incremento en la variable independiente X. Cabe recalcar que:
  - Si $\beta_i > 0 \rightarrow$ la variable Y_i sufrirá un crecimiento por cada incremento de X.
  - Si $\beta_i < 0 \rightarrow$ la variable Y_i sufrirá un decrecimiento por cada incremento de X.
- X_1, X_2, X_n se define como cada uno de los valores de la variable independiente X.

Aplicado al aprendizaje automático, X_i consistirá en las distintas características que querremos usar para predecir algún valor de la variable dependiente Y_i. Por lo tanto, Y_i será la característica que estaremos intentando predecir.

Una de las características principales de este modelo es que tratará de predecir un valor numérico, por lo que no será válido para cualquier ejemplo donde se intente predecir una característica dependiente Y_i a partir de distintas características X_i. Por ejemplo:

Imaginemos la situación donde se quiere realizar la predicción de si el paciente sufre algún tipo de enfermedad en el corazón o no, a partir de una serie de características como la edad, el colesterol y la presión arterial del paciente. En este caso, aunque tengamos la finalidad de obtener la predicción de una característica de salida Y_i, en función de una serie de características de entrada X_i, el resultado que queremos obtener no es numérico, por lo que este caso sería un problema de clasificación multiclase.

Sin embargo, imaginemos que queremos realizar la predicción de cuál será el precio de una vivienda en función del terreno disponible, ubicación, años de antigüedad, etc. En este caso sí que estamos ante un problema de regresión, donde se busca predecir un valor numérico a partir de una serie de características de entrada independientes. Por lo tanto, este modelo de regresión lineal podría ser perfectamente aplicable a esta situación.

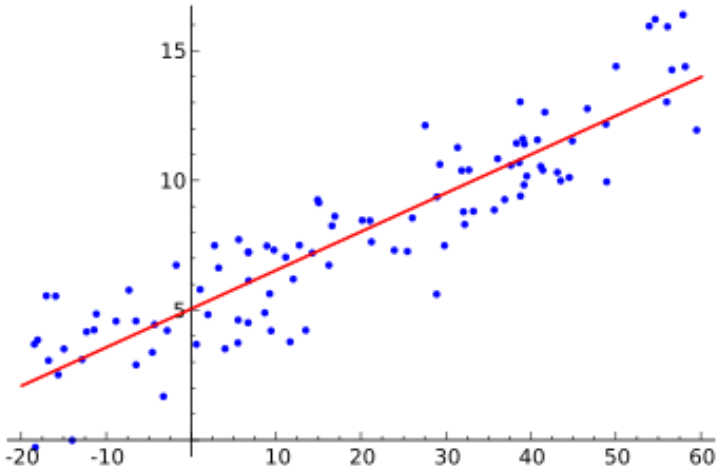

Ilustración 13. Ejemplo regresión lineal con una variable dependiente y una variable independiente [17].

La ilustración muestra una representación gráfica de la salida que se obtiene tras analizar algún problema de regresión. Como se puede observar, a partir de una serie de valores de la característica de entrada (los puntos azules), se traza una recta que define los distintos valores que tendría la característica de salida para cada uno de los valores de la característica de entrada.

A continuación, se muestra una aplicación real:

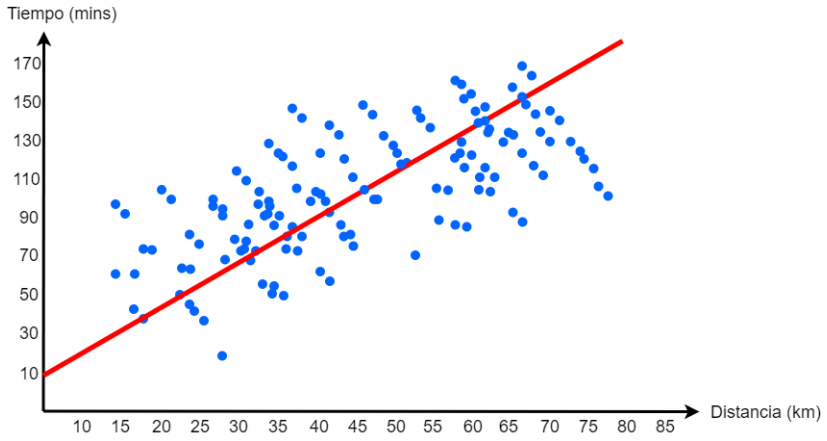

Ilustración 14. Aplicación real regresión lineal.

Esta ilustración muestra la predicción de cuál será el tiempo que se tardará en minutos en recorrer *n* kilómetros en función de un histórico de datos como entrada. Para determinar cuál es la recta que mejor se ajusta se debe aplicar la siguiente función:

$$\sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} \left(y_i - (ax_i + b)\right)^2 \quad (13)$$

donde:

- e_i^2 es el error, siendo esto la diferencia entre el valor observado y el valor ajustado obtenido a partir de la ecuación de la recta $y = ax_i + b$.

Observando la ecuación *(13)*, el criterio que se aplica para determinar la recta lineal idónea consiste en que la desviación cuadrática sea mínima [18]. Otra forma de obtener la recta que define la relación entre las variables *x* e *y* es mediante el coeficiente de correlación:

$$r = \frac{\sum(x_i - x)(y_i - y)}{N\,\alpha_x \alpha_y} \quad (14)$$

donde:

- $\sum(x_i - x)(y_i - y)$ es igual al producto de las desviaciones de los valores *X* e *Y* en función de sus valores medios.
- $N\,\alpha_x \alpha_y$ es igual a las desviaciones cuadráticas medias de *X* e *Y*, *N* veces.

4.2 Regresión logística

Google Developers (s.f.-b) define la regresión logística como un modelo que genera una probabilidad para cada valor de etiqueta discreto posible en problemas de clasificación al aplicar una función sigmoide a una predicción lineal [19]. Se puede utilizar tanto en problemas de clasificación binaria como en problemas de clasificación multiclase, donde se denomina regresión multinomial.

En un estudio reciente (Centro de Servicios Informáticos de la UNED, s.f.), define que la regresión logística (matemáticamente), se trata de una técnica analítica que nos permite relacionar funcionalmente una variable dicotómica con un conjunto de variables independientes [20]. Esta técnica resulta ser potente en ciertos ámbitos, ya que nos permite analizar las relaciones de variables categóricas entre sí. Esta técnica puede considerarse una extensión de los modelos de regresión lineal, con la particularidad de que el dominio de salida está acotado al intervalo [0,1] y que el procedimiento de estimación, en lugar de mínimos cuadrados, utiliza el procedimiento de estimación máximo-verosímil.

Por lo tanto, la principal diferencia que se encuentra en la regresión lineal y la regresión logística seria que en la regresión lineal las predicciones son de forma continua, mientras que las predicciones en la regresión logística serian de tipo discreto, donde solo ciertos valores o categorías son permitidos.

A pesar de que por su nombre pueda parecer que nos estemos refiriendo a un algoritmo de predicción, no es así: se trata de un algoritmo de tipo supervisado utilizado para clasificación.

Existen tres tipos de regresión logística:

- Regresión logística binaria (enfermo o sano).
- Regresión logística multinomial (gatos, perros, conejo, hámster).
- Regresión logística ordinal (bajo, medio, alto).

Describamos un ejemplo para entender en detalle su funcionamiento.

Pongámonos en una situación donde se nos proporcionan una serie de datos relacionados con las distintas características del paciente, y el objetivo es predecir si el paciente sufre algún tipo de enfermedad o no. En estos datos tendremos dos clases distintas: enfermo (1), sano (0). Además, tendremos dos características representativas como son la edad y el genero del paciente. Representando gráficamente estos datos de entrada, se vería de la siguiente manera:

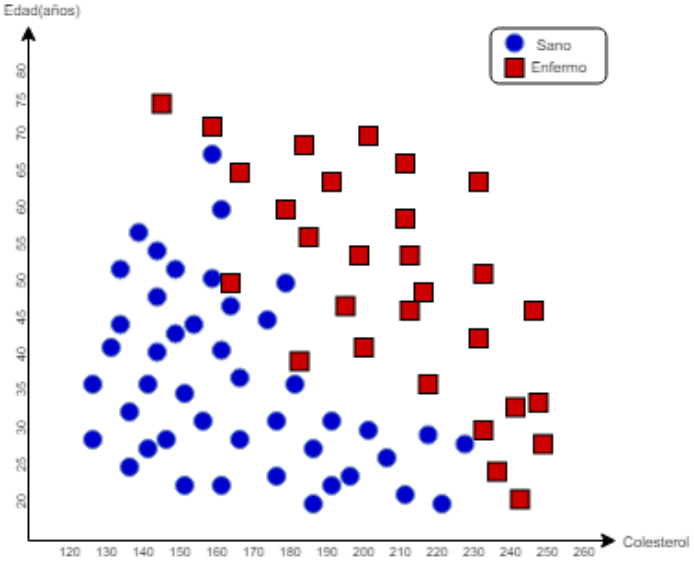

Ilustración 15. Representación de puntos para la Regresión Logística.

Por otro lado, para tratar de asignar los valores predichos a las distintas probabilidades, se hace uso de la función sigmoide. La función sigmoide o función logística es una función característica por tener forma de "S" o curva sigmoidea, que se define mediante la siguiente fórmula:

$$\mathbf{S(x)} = \frac{\mathbf{1}}{\mathbf{1}+e^{-x}} = \frac{e^{x}}{e^{x}+\mathbf{1}} \qquad \mathbf{(15)}$$

donde:

- S(x) será la salida, teniendo un valor comprendido entre cero y uno.
- x serán los datos de entrada a la función.

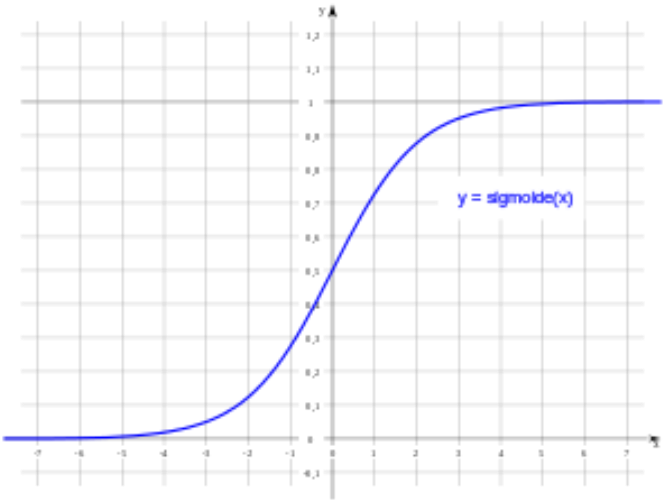

Ilustración 16. Gráfica de una función sigmoide [21].

Como se puede observar en la anterior ilustración, el resultado de la función será un valor comprendido entre cero y uno, donde, si la curva tiende a infinito positivo, la predicción del

valor de salida se convertiría en uno, en este caso, el paciente será de tipo enfermo; mientras que si la curva tiende a infinito negativo, la predicción del valor de salida se convertirá en cero, es decir, el paciente será de tipo sano. Pero también existen muchas más salidas como resultado de aplicar la función sigmoide, donde el valor estará comprendido entre cero y uno, pero no tenderá ni a infinito positivo ni a infinito negativo. Para poder mapear todos estos valores de tipo discreto habrá que fijar un límite donde el modelo clasificará el valor como cero o como uno, según dicho límite.

4.3 Naive Bayes

En 2019, "Naive Bayes classifier" define Naive Bayes como una técnica de clasificación probabilística basada en la aplicación del teorema de Bayes y algunas hipótesis simplificadoras adicionales, donde, a causa de estas simplificaciones, se suelen resumir en las hipótesis de independencia entre las variables predictoras (2019a) [22].

En 2018, José Francisco López, establece que el teorema de Bayes es utilizado para calcular la probabilidad de un suceso, teniendo información de antemano sobre ese suceso [23]. El teorema se define según la siguiente formula:

$$\boldsymbol{P(A_i|B) = \frac{P(B|A_i)\times P(A_i)}{P(B)}} \quad \textbf{(16)}$$

donde:

- $P(A_i)$ son las probabilidades a priori.
- $P(B|A_i)$ es la probabilidad de B en la hipótesis A_i.
- $P(A_i|B)$ son las probabilidades a posteriori.

Mediante este teorema, somos capaces de encontrar la probabilidad A, sabiendo que B va a suceder; es decir, B será la característica de entrada, mientras que A será la hipótesis. Por lo tanto, el teorema asume que las predicciones son independientes entre sí. Veamos un ejemplo para entender en profundidad de este clasificador.

Como se ha utilizado en otros ejemplos, la temática del conjunto de datos de entrada tendrá que ver con una serie de características médicas que definen finalmente si el paciente padece algún tipo de enfermedad o no.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Ilustración 17. Ejemplo conjunto de datos de entrada

Si se toma como ejemplo el primer registro se asume que, para una edad superior a los sesenta años, y un colesterol superior a los 230 mg/dl, el paciente sufre algún tipo de enfermedad. Pero, tal y como se ha expresado anteriormente, como las predicciones son independientes entre sí, que el paciente tenga una edad superior a los sesenta años no implica que el paciente sea del tipo enfermo. Con este ejemplo, la fórmula del teorema de Bayes se define de la siguiente forma:

$$\boldsymbol{P(y|X) = \frac{P(X|y)\times P(y)}{P(X)}} \quad \textbf{(17)}$$

donde:

- y será la variable de predicción que indica si el paciente está enfermo o no.
- $X = (x_1, x_2, \dots, x_n)$ como representación de las distintas características.

Ahora, mediante la sustitución de X y mediante la regla de la cadena obtendríamos algo similar a lo siguiente:

$$\boldsymbol{P(y|x_1, x_2, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)\times P(y)}{P(x_1)P(x_2)\dots P(x_n)}} \quad \textbf{(18)}$$

Por lo tanto, ya se podría obtener cada una de las distintas probabilidades independientes mediante la sustitución del conjunto de datos de entrada en la ecuación. En este caso, la variable de salida solo puede tener dos valores, uno o cero. Por ello, se debe encontrar la clase de máxima probabilidad.

Existen distintos tipos de clasificadores de Naive Bayes:

- Naive Bayes *multinomial*, donde la salida tendrá más posibilidades que uno o cero.
- Naive Bayes de *Bernoulli*, donde los predictores son variables booleanas.
- Naive Bayes *gaussiano*, donde los predictores son valores de tipo continuo y no discreto.

El algoritmo de Naive Bayes se sitúa como un algoritmo bastante utilizado por su sencillez y rapidez a la hora de su implementación. Sin embargo, su mayor desventaja será que las predicciones son de tipo independiente, no siendo una casuística normal en casos reales, por lo que el rendimiento del clasificador disminuye.

4.4 KNN

En 2019, "k-nearest neighbors algorithm" establece que, KNN *(K-Nearest-Neighbour)*, o algoritmo de k-vecinos más cercano, se define como un algoritmo de aprendizaje automático supervisado simple utilizado para resolver problemas de clasificación y regresión. Se trata de un método no paramétrico utilizado donde la entrada de datos consiste en las *k* muestras de entrenamiento más cercanas en el espacio de las distintas características (2019b) [24]. En un término simplificado, el algoritmo de KNN asumirá que cosas similares se encuentran próximas entre sí.

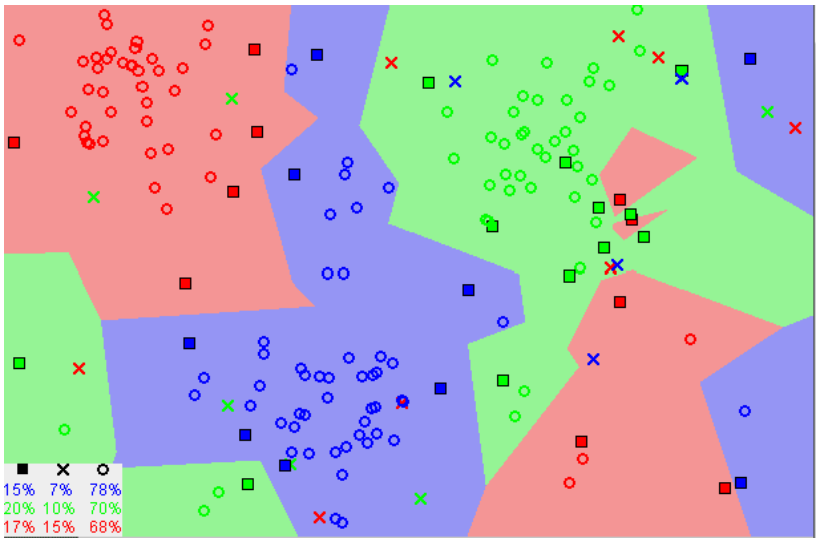

Ilustración 18. Ejemplo salida al aplicar KNN [24].

En la imagen superior se observa como, en la mayoría de los casos, los puntos que son similares entre sí están situados uno al lado del otro. La base del algoritmo de KNN es este, donde supone que los puntos más cercanos entre sí serán de la misma clase. Este cálculo matemático no es más que el calculo de la distancia o proximidad entre dos puntos. Existen muchas maneras de calcular esta distancia, pero la distancia euclidiana suele ser una elección común. En un estudio reciente (EcuRed, s.f.) define la distancia euclidiana como una función no negativa usada en diversos contextos para calcular la distancia entre dos puntos, primero en el plano y luego en el espacio [25]. En el plano cartesiano, la distancia euclidiana entre dos puntos se calcula mediante la siguiente ecuación:

$$\boldsymbol{d(A,B) = \sqrt{(X_b - X_a)^2 + (Y_b - Y_a)^2}} \quad \textbf{(19)}$$

Por lo tanto, se trata de un algoritmo de clasificación supervisado donde clasificará cada registro según sus *k* vecinos más cercanos. Para obtener dicha clasificación, el algoritmo trata básicamente en obtener la distancia que existe para todos los registros del conjunto de entrenamiento, para después ordenar estas distancias de menor a mayor. Así, el registro que se

trata de clasificar obtendrá la misma clase que sus *k* más cercanos poseen. Veamos un ejemplo sencillo.

Supongamos que nos encontramos ante la siguiente situación:

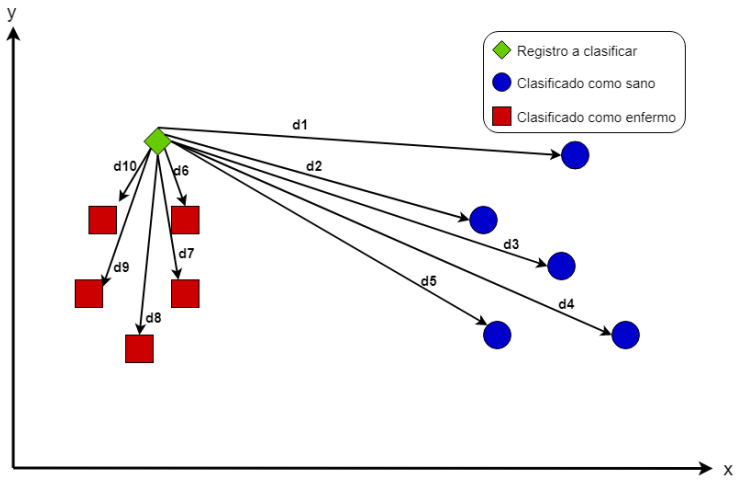

Ilustración 19. Ejemplo aplicación KNN

Como se ha comentado anteriormente, el objetivo será tratar de obtener la clase final para el registro a clasificar, es decir, el rombo verde. Previo a la aplicación del algoritmo, habría que definir cuántos vecinos queremos tener en cuenta a la hora de clasificar. Por ahora diremos que *k* es igual a tres. A partir de este momento se calculan todas las distancias entre el registro a clasificar y todos los del conjunto de entrenamiento, es decir, $d1, d2, d3, \dots, d10$. Tras esto, se realiza una ordenación de menor a mayor distancia, que quedaría algo parecido a lo siguiente:

Orden distancia de menor a mayor
d6
d10
d7
d9
d8
d2
d5
d1
d3
d4

Tabla 2. Ordenación de distancias

Tras la ordenación, como se ha declarado *k* a tres, únicamente se fija en la clase de las tres primeras distancias. En este caso, los registros que conforman las distancias *d6*, *d10* y *d7* se

han clasificado como pacientes enfermos, por lo que el registro a clasificar también será de la clase enfermo.

4.5 Árboles de decisión

Google Developers, s.f.-b, define la técnica de árboles de decisión como un modelo representado mediante una secuencia de declaraciones de ramificación [16]. El algoritmo es capaz de tomar la decisión más acertada, basándose en probabilidades.

Para entender esta técnica como un algoritmo de aprendizaje automático, veamos los distintos elementos que componen dicho árbol.

Fernando Sancho Caparrini (s.f.) define que los árboles de decisión están formados por:

- Nodos de decisión: nodo asociado a uno de los registros y tiene dos o más ramas que salen de él, cada una de ellas representando los posibles valores que puede tomar el registro asociado.
- Nodos-respuesta: nodo asociado a la clasificación que se quiere proporcionar, y devuelve la decisión del árbol con respecto al registro de entrada.
- Ramas: muestra los distintos caminos que se pueden emprender cuando tomamos una decisión o bien cuando ocurre algún evento aleatorio ("Introducción a los Árboles de Decisión", s.f.") [27].

Los árboles de decisión pueden ser utilizados tanto para tareas de regresión, como para tareas de clasificación, aunque su utilidad suele ser más común para tareas de clasificación.

Regresión	Clasificación
Variable dependiente es continua	Variable dependiente es categórica
Valores de los nodos se reducen a la media de las observaciones en esa región	El valor del nodo se reduce a la moda de las observaciones del conjunto de entrenamiento que han "caído" en esa región

Tabla 3. Árboles de regresión vs. Árboles de clasificación ("Árboles de Decisión y Random Forest" s.f.) [28].

Veamos un ejemplo sencillo de árbol de clasificación para entender su funcionamiento. Como se ha comentado anteriormente, un árbol de clasificación es una estructura jerárquica para la clasificación de objetos. Por lo tanto, imaginemos el siguiente objeto:

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \quad (20)$$

donde x_1, x_2 representan los valores de las distintas características del objeto a analizar.

Como ejemplo tomaremos que queremos analizar si el paciente x sufre algún tipo de enfermedad o no según su edad (x_1) y el colesterol en mg/dl (x_2).

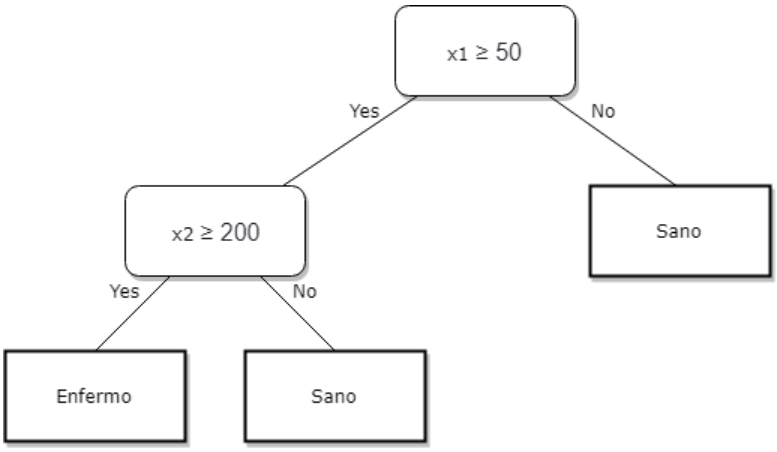

Ilustración 20. Ejemplo sencillo árbol de decisión

Existen dos flujos de decisiones posibles:

- Si $x_1 \geq 50$, es decir, el paciente tiene cincuenta años o más, se iría por la rama de la izquierda. Además, si $x_2 \geq 200$, es decir, el paciente tiene 200 mg/dl o más, el paciente sería clasificado como enfermo. Sin embargo, si $x_2 < 200$, es decir, el paciente tiene menos de 200 mg/dl de colesterol, el paciente sería clasificado como sano.
- Si $x_1 < 50$, es decir, el paciente tiene menos de cincuenta años, se iría por la rama de la derecha. Además, si $x_2 \leq 300$, es decir, el paciente tiene 300 mg/dl o menos de colesterol en sangre, el paciente sería clasificado como sano. Sin embargo, si $x_2 > 300$, es decir, el paciente tiene más de 300 mg/dl de colesterol, el paciente sería clasificado como enfermo.

Gráficamente se puede observar cómo mediante esta técnica particiona el espacio de representación, realizando cada una de estas particiones a partir de los nodos de decisión generados.

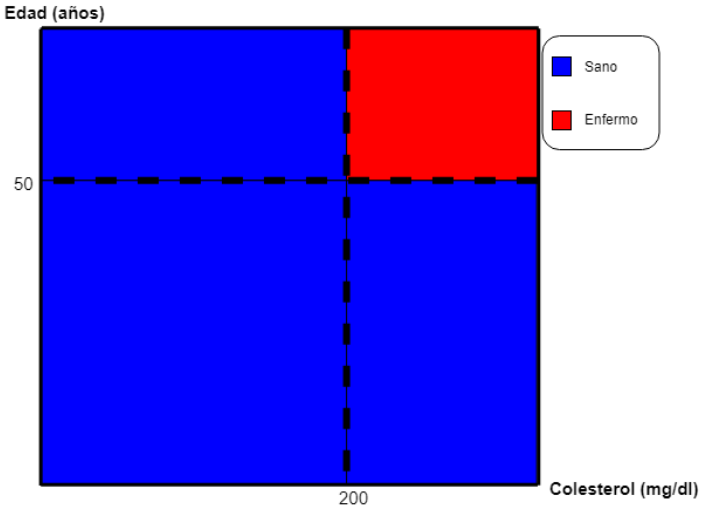

Ilustración 21. Representación gráfica árbol de decisión

Por lo tanto, el algoritmo de aprendizaje automático trata de dicotomizar, es decir, dividir el conjunto de datos en dos subconjuntos, aplicando un criterio de partición donde este criterio de partición consistirá en elegir un par variable-umbral, para luego poder dividir los datos. Pero en esta clasificación existen una serie de impurezas, denominadas impureza de un nodo. Esta es la incertidumbre que existe sobre el conjunto de datos. Esta impureza se calcula como la entropía de la distribución empírica, de la probabilidad posteriori de clases en el conjunto, donde, mientras la entropía sea menor, mejor (Universitat Politècnica de València, s.f.) [29].

Según "Árboles de Decisión y Random Forest" s.f., la entropía se calcula mediante la siguiente ecuación:

$$\boldsymbol{Entropía} = -\boldsymbol{p} \times \boldsymbol{\log_2(p)} - \boldsymbol{q} \times \boldsymbol{\log_2(q)} \quad \textbf{(21)}$$

donde:

- p es igual a la probabilidad de que el registro sea de clase enfermo.
- q es igual a la probabilidad de que el registro sea de clase sano.

En conclusión, cabe destacar la facilidad de interpretación desde un punto de vista visual a la hora de obtener las predicciones finales, ya que resulta sencillo la generación del árbol, para poder evaluarla de una forma visual. Sin embargo, existirá el peligro de que exista cierta entropía en la predicción, por lo que puede que el resultado final no sea el correcto.

4.6 Random Forest

En 2019, "Random Forest" establece que la técnica de Random Forest o Bosque Aleatorio, se define como una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos (2019d) [30]. Cabe destacar que este algoritmo puede utilizarse tanto para

tareas de regresión como para tareas de clasificación. Como se expresa en la definición, el algoritmo consiste en la creación de múltiples árboles de decisión donde posteriormente los "fusiona" para obtener una predicción más estable y precisa.

En Random Forest, cultivamos varios árboles en un único modelo. Para clasificar un nuevo conjunto de datos basado en sus características, cada árbol da una clasificación donde cada árbol "vota" por esa clase. Entonces, el bosque elige las clasificaciones que tienen la mayor cantidad de votos de todos los otros árboles en el bosque y toma la diferencia de los distintos árboles.

Al crear árboles aleatorios, éstos se dividen en diferentes nodos o subconjuntos. A partir de ahí, busca el mejor resultado de los subconjuntos aleatorios. Esto da como resultado el mejor modelo del algoritmo. Por lo tanto, en un bosque aleatorio, solo se tiene en cuenta el subconjunto aleatorio.

Para dar una idea más clara sobre el funcionamiento de Random Forest, veamos un ejemplo basado en *"Random Forest Analysis in ML and when to use it", s.f.* [31].

Supongamos que formamos mil árboles aleatorios para formar el bosque aleatorio para la detección de una mano humana. Cada bosque aleatorio predecirá los distintos resultados o la clase para las mismas características. Un pequeño subconjunto del bosque observará el conjunto aleatorio de características, por ejemplo, la mano o los dedos. Supongamos que unos cientos de árboles de decisión aleatorios predicen la clase dando lugar al pulgar, los dedos o los humanos. Luego, los votos del dedo se calculan a partir de cien decisiones aleatorias y también los votos del pulgar y del humano. Si los votos del dedo son más altos, entonces el bosque aleatorio final devolverá el dedo como un objetivo predicho. Este tipo de votación se llama votación por mayoría. Lo mismo se aplica al resto de los dedos de la mano, si el algoritmo predice que el resto de los dedos son dedos de una mano, entonces el árbol de decisión de alto nivel puede votar que una imagen es una mano.

4.7 SVM

En un estudio reciente (Tavish Srivastava, 2019) establece que SVM (*Support Vector Machine*) o máquinas de vectores de soporte, se definen como un modelo de aprendizaje supervisado utilizado tanto para tareas de clasificación como de regresión, donde los vectores de soporte son simplemente coordenadas de una observación individual [32]. SVM consiste en la representación de cada elemento como un punto en el espacio n-dimensional (donde *n* es el número de entidades totales) con el valor de cada elemento como una coordenada particular. Tras el posicionamiento de todos los elementos, la clasificación se realiza encontrando el hiperplano que diferencie ambas clases.

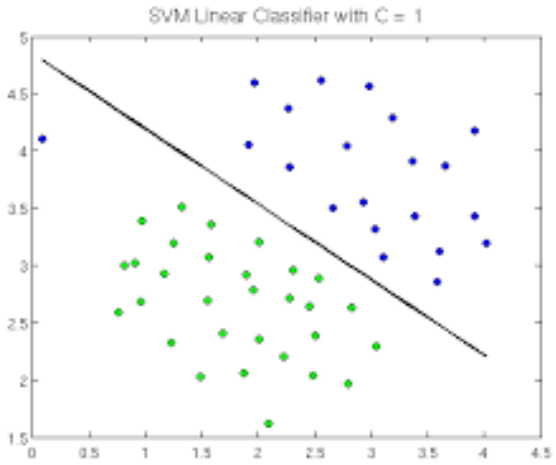

Ilustración 22. Ejemplo SVM [33]

En la ilustración se observa un caso prácticamente ideal, ya que las dos clases se pueden diferenciar claramente, por lo que trazar el hiperplano de separación resulta muy sencillo.

En 2019, "Máquina de Soporte Vectorial SVM" establece los siguientes conceptos necesarios para entender esta técnica [43]:

1. Hiperplano: plano que separa y clasifica linealmente un conjunto de datos.
2. Vectores de apoyo: puntos críticos más cercanos a la línea del hiperplano y que al eliminarse alterarían la posición del hiperplano.
3. Margen: distancia entre el hiperplano y el dato más cercano de cualquiera de los conjuntos.
4. Dimensión VC (Vapnik-Chervonekis): conjunto de funciones disponibles capaces de encontrar la cantidad máxima de puntos a separar. A mayor dimensión, menor será el error del conjunto.
5. Núcleo (Kernel): conjunto de funciones matemáticas utilizadas para transformar los datos de entrada en la forma deseada. Su utilidad es clara si los datos forman una curva y no existe un hiperplano que separe completamente ambas clases.

En problemas de clasificación su funcionalidad es clara (división de ambas clases mediante un hiperplano). Pero, ¿en problemas de regresión? En este caso se buscará encontrar cuál es la tendencia de los datos, para así poder predecir en función de dicha curva de tendencia.

Una vez analizados los conceptos básicos de SVM, veamos sus ventajas y desventajas.

Ventajas, donde resulta ser eficiente:

- cuando el margen de separación es claro;
- en problemas con una gran cantidad de espacios dimensionales;
- en casos donde el número de dimensiones es mayor que el número de muestras.

Desventajas, donde resulta ser ineficiente:

- cuando el conjunto de datos es muy grande, por lo que el proceso de entrenamiento es elevado;
- cuando existen datos con mucho ruido;
- al no definir directamente probabilidades a las estimaciones, sino que se calculan mediante un subproceso de *5-fold-cross-validation*.

Capítulo 5. Experimentos y resultados

A continuación, tras un estudio teórico de los distintos aspectos del aprendizaje automático, procederemos a aplicar dichas técnicas a la práctica.

Tal y como hemos comentado en el resumen, abarcaremos todas las fases del proceso, exceptuando la adquisición de los datos, ya que los datos que procedemos a analizar son obtenidos de una fuente de datos pública.

Dicha fuente de datos la obtendremos a través de la página web pública de Kaggle (Kaggle Inc. (s.f.)) [34], propiedad de Google LLC. Kaggle, consiste en una comunidad vía on-line de científicos de datos y principiantes del aprendizaje automático. Esta página web permite a los usuarios registrados acceder a distintos conjuntos de datos públicos, así como compartirlos a los demás usuarios. Destaca por lo denominado *"Kaggle Competitions"*, donde las compañías publican distintos problemas relacionados con el aprendizaje automático, y son los usuarios los que compiten por intentar diseñar el mejor algoritmo posible.

En nuestro caso, como únicamente queremos introducirnos en el aprendizaje de las distintas técnicas que se pueden aplicar, a partir del conjunto de datos público *"Heart Disease UCI"* (Kaggle Inc. (s.f.)) [35], trataremos de realizar un estudio de este, desarrollando así las distintas técnicas comentadas de forma teórica.

Para el estudio de las distintas técnicas empleadas en el aprendizaje automático mediante Python, debemos conocer las librerías utilizadas para tal fin. Las que vamos a utilizar serán: *matplotlib*, *pandas* y *sci-kit learn*. Esta última contiene todo tipo de funciones que abordaremos posteriormente.

Todas estas librerías se pueden obtener mediante la descarga del paquete Anaconda (Anaconda, s.f.) [36], ya que dispone de todas las librerías que utilizaremos.

5.1 Análisis teórico

Una vez tenemos todas las herramientas disponibles, procedemos al estudio del conjunto de datos que vamos a analizar. La propia página de Kaggle nos proporciona una breve descripción del *dataset*, donde introduce las características de la siguiente manera:

1. Age = Edad (en años)

2. Sex = Género (1 = Masculino, 0 = Femenino)

3. cp = Tipo de dolor de pecho. (1 = Angina típica, 2 = Angina atípica, 3 = Dolor no anginal 4 = Asintomática)

4. trestbps = Presión arterial al ingreso en el hospital (mmHg)

5. chol = Colesterol (mg/dl)

6. fbs = Azúcar en sangre en ayunas > 120 mg/dl (1 = Verdadero, 0 = Falso)

7. restecg = Resultados electrocardiográficos en reposo (0 = Normal, 1 = Anomalía en la onda ST, 2 = Hipertrofia ventricular izquierda probable o definitiva según el criterio de Estes)

8. thalach = Máximo ritmo cardíaco alcanzado

9. exang = Angina obtenida por el ejercicio (1 = Sí, 0 = No)

10. oldpeak = Depresión ST inducida por el ejercicio en relación con el descanso

11. slope = Pendiente del segmento pico del ejercicio ST (0 = Pendiente ascendente, 1 = Plano, 2 = Pendiente descendente)

12. ca = Número de vasos principales coloreados por fluoroscopia (0-3)

13. thal = Trastorno sanguíneo denominado talasemia = (1 = Normal, 2 = Defecto fijo, 3 = Defecto reversible)

14. target = Tiene o no enfermedad de corazón (1 = Sí, 0 = No)

Ya que no estoy especializado en el mundo de la medicina, existe una serie de características que no me aportan ninguna información significativa. Por ello, lo mejor será realizar un pequeño estudio de lo que estamos analizando, para así poder obtener mejores conclusiones de cara al futuro de nuestro análisis.

Empecemos por el principio: ¿qué es la angina de pecho? Según un artículo publicado por la Secretaría de la Salud ("¿Qué es la angina de pecho?" (s.f.)), la angina de pecho es un dolor o molestia que se presenta cuando el músculo cardiaco no recibe suficiente irrigación sanguínea y por lo tanto un deficiente aporte de oxígeno; este término viene del latín "angor pectoris" que significa estrangulamiento en el pecho [37].

¿Hay distintos tipos de angina de pecho? En un estudio reciente ("Angina de pecho: MedlinePlus" (s.f.)) [38], se definen tres tipos de angina:

- Angina estable: es la más común. Ocurre cuando el corazón trabaja más fuerte que lo usual. La angina estable tiene un patrón regular. Se trata con descanso y medicinas .

- Angina inestable: es la más peligrosa. No sigue un patrón y puede ocurrir sin hacer algún esfuerzo físico. No desaparece espontáneamente con el reposo o las medicinas. Es una señal de que podría ocurrir un infarto posteriormente.

- Angina variable: es la menos común. Ocurre cuando está descansando. Se trata con medicinas.

Con esto, somos capaces de distinguir entre los distintos tipos de dolor de pecho que nos muestra el *dataset*, donde la angina típica corresponde a lo denominado angina estable, la angina atípica a la angina inestable, y la angina asintomática a la angina variable.

Sabiendo esto, algunas de las características del *dataset* cobran sentido, como "trestbps" (presión arterial al ingreso en el hospital (mmHg)) o "exang" (angina obtenida por el ejercicio), ya que el sufrir una angina de pecho supone que la presión arterial deba ser mayor de lo normal, debido a que el corazón necesita más irrigación sanguínea. Lo mismo sucede cuando estamos realizando algún ejercicio físico: el corazón requiere una mayor cantidad de sangre para aportar el oxígeno necesario y, si esto no está sucediendo, puede provocar dicha enfermedad. En dicho artículo también se cita que la angina de pecho puede estar provocada por una placa de colesterol, es por ello por lo que valores altos de la característica "chol" pueden suponer un mayor porcentaje de probabilidad de que el paciente tenga dicha enfermedad de corazón.

Por otro lado, otras características como "fbs" (azúcar en sangre en ayunas) o "thalach" (máximo ritmo cardíaco alcanzado) nos quedan más claros, ya que sabiendo qué es la angina de pecho (que el azúcar en sangre en ayunas o que el ritmo cardíaco alcanzado sea mayor de lo esperado) supone una mayor cantidad de sangre requerida por el corazón y, por lo tanto, se requiera una mayor cantidad de oxígeno.

Vayamos ahora con el significado que pueda aportar los resultados electrocardiográficos. Un electrocardiograma (ECG), es una prueba que evalúa el ritmo y la función cardiaca del paciente mediante la actividad eléctrica del corazón. Dichas pruebas se aplican para evaluar el segmento ST. Y, ¿qué es el segmento ST? Según un artículo publicado por el Dr. Parrales (“La valoración y análisis del Segmento ST” (s.f.)), el segmento ST es la distancia que se encuentra entre el final del Complejo QRS y el inicio de la Onda T. De forma normal el segmento ST es una línea isoeléctrica. Representa el inicio de la repolarización ventricular. Es un momento de silencio eléctrico, el cual llega a su plenitud con la sístole ventricular [39].

En dicho artículo se menciona que un segmento ST normal corresponde a una línea isoeléctrica plana, aunque una ligera elevación o depresión de entre 0.5 y 1 mm también se considera como normal [39]. Como en la característica "slope" no existe tal nivel de detalle, consideraremos como normal cuando este nos indique que el segmento es plano.

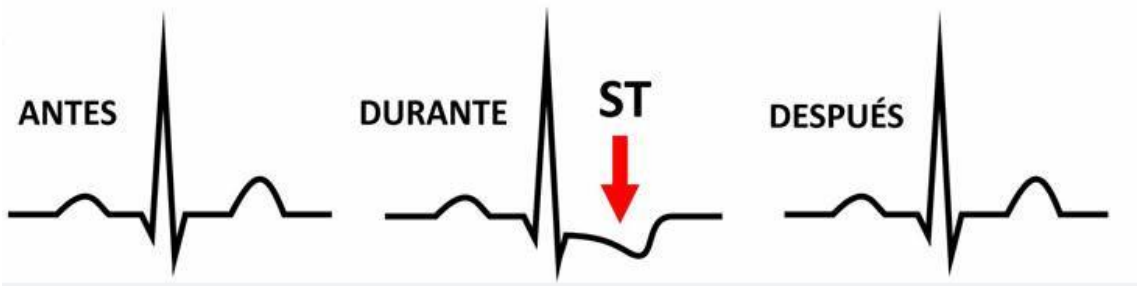

Ilustración 23. Electrocardiograma en la angina de pecho [40].

Se puede apreciar que, cuando el paciente sufre una angina de pecho, el segmento ST sufre un ligero descenso, indicando así que la irrigación sanguínea no es suficiente.

Además, dichos resultados pueden indicarnos la presencia de hipertrofia ventricular izquierda probable o definitiva. ¿Qué es la hipertrofia ventricular izquierda probable o definitiva? Según un artículo publicado en Mayo Clinic, la hipertrofia del ventrículo izquierdo es la dilatación y el engrosamiento (hipertrofia) de las paredes de la cámara de bombeo principal del corazón (ventrículo izquierdo).

Un estudio reciente ("Hipertrofia ventricular izquierda – Síntomas y causas – Mayo Clinic", 2018) cita que la hipertrofia ventricular izquierda puede manifestarse como una reacción a algún factor, como la presión arterial alta o una enfermedad cardíaca, que hace que el ventrículo izquierdo se esfuerce más. A medida que el esfuerzo aumenta, se engrosa el tejido muscular en la pared de la cavidad y, a veces, el tamaño de la propia cavidad también aumenta. El músculo del corazón dilatado pierde elasticidad y, finalmente, puede dejar de bombear con la fuerza necesaria [41].

Finalmente, encontramos un atributo relacionado con la talasemia (thal). En 2018, "Talasemia – Síntomas y causas Mayo Clinic", define la talasemia como un trastorno hereditario de la sangre que se caracteriza por tener menos hemoglobina y menos glóbulos rojos en su cuerpo de lo normal. La hemoglobina es la sustancia en sus glóbulos rojos que les permite transportar oxígeno. La baja hemoglobina y menos glóbulos rojos de la talasemia pueden causar anemia y dejarlo fatigado [42].

Si tiene una talasemia leve, es posible que no necesite tratamiento. Pero si tiene una forma más grave del trastorno, es posible que necesite transfusiones de sangre con regularidad. También puede tomar medidas por su cuenta para hacer frente a la fatiga, como elegir una dieta saludable y hacer ejercicio con regularidad [42]. Por lo tanto, interpretaremos el valor 2 (defecto fijo) como talasemia grave, mientras que el valor 3 (defecto reversible) como talasemia leve, donde no requiere tratamiento.

Una vez introducidos de manera superficial en el concepto de angina de pecho y los distintos conceptos relacionados con el mismo, procedemos al análisis práctico del *dataset* a estudiar.

5.2 Análisis práctico

Lo primero será importar las librerías necesarias:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sb
import statistics
from sklearn import linear_model
from sklearn import model_selection
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler as StandSc
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, roc_curve, auc, roc_auc_score
from sklearn.metrics import accuracy_score
```

Como primer paso, mediante la librería de *pandas* y la función *read_csv()*, se genera un dataframe a través la lectura del archivo CSV *(Comma Separated Value)*.

```
# delimiter --> carácter utilizado para separar las columnas en el archivo CSV.
df = pd.read_csv("C:/Users/pabnoc1/Desktop/TFG/heart-disease-uci/heart.csv", delimi
ter = ',')
```

Previo a realizar un algoritmo de aprendizaje automático óptimo, siempre es importante realizar una buena preparación de los datos, eliminando las posibles imperfecciones que pueda tener. Por ello, se analiza si nuestro conjunto de datos contiene valores nulos.

```
# Obtenemos los nombres de las columnas en una lista
col_names = df.columns.tolist()
# Iteramos la lista para encontrar valores nulos
for column in col_names:
    print("Valores nulos en {0}: {1}".format(column, df[column].isnull().sum()))
```

```
Valores nulos en age: 0, Valores nulos en sex: 0
Valores nulos en cp: 0, Valores nulos en trestbps: 0
Valores nulos en chol: 0, Valores nulos en fbs: 0
Valores nulos en restecg: 0, Valores nulos en thalach: 0
Valores nulos en exang: 0, Valores nulos en oldpeak: 0
Valores nulos en slope: 0, Valores nulos en ca: 0
Valores nulos en thal: 0, Valores nulos en target: 0
```

En este caso, el conjunto de datos está limpio en cuanto a nulos, debido a que normalmente en Kaggle los conjuntos de datos que se proporcionan para tareas de aprendizaje ya suelen estar corregidos.

Por otro lado, existe una función muy útil para conocer una serie de estadísticas del conjunto de datos, denominada *describe()*. Esta función permite conocer valores típicos de manera inmediata, como los distintos percentiles de cada una de las características, la media, los valores mínimos y máximos, etc.

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303	303	303	303	303	303	303	303
mean	54.366	0.68	0.96	131.62	246.26	0.14	0.53	149.65
std	9.082	0.46	1.03	17.53	51.83	0.35	0.52	22.91
min	29	0	0	94	126	0	0	71
25%	47.5	0	0	120	211	0	0	133.5
50%	55	1	1	130	240	0	1	153
75%	61	1	2	140	274.5	0	1	166
max	77	1	3	200	564	1	2	202

exang	oldpeak	slope	ca	thal	target
303	303	303	303	303	303
0.32	1.04	1.4	0.73	2.31	0.54
0.47	1.16	0.61	1.02	0.61	0.5
0	0	0	0	0	0
0	0	1	0	2	0
0	0.8	1	0	2	1
1	1.6	2	1	3	1
1	6.2	2	4	3	1

Tabla 4. Resultados tras función *describe()*.

Cuando se tiene una gran cantidad de datos, muchas veces no se quieren mostrar todos, sino únicamente una pequeña selección a modo de ejemplo para ver el aspecto de estos. Es por ello por lo que, como ejemplo, únicamente mostraremos las 5 primeras filas del dataframe.

```
# Muestra las 5 primeras filas del dataframe por defecto.
# df.head(x) , donde x = número de filas que se muestran.
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Tabla 5. Primeros 5 registros del *dataset*.

Entendamos la salida que acabamos de mostrar. Se tomará como ejemplo la primera fila, donde se identifican los siguientes datos del paciente en cuestión:

- age = 63 años

- sex = (1) Masculino
- cp = (3) Dolor no anginal
- trestbps = 145 mmHg
- chol = 233 mg/dl
- fbs = (1) Más de 120 mg/dl de azúcar en sangre en ayunas
- restecg = (0) Resultados electrocardiográficos en reposo normales
- thalach = (150) pulsaciones/minuto máximos alcanzados
- exang = (0) Angina de pecho no obtenida por el ejercicio
- oldpeak = (2.3) de depresión ST inducida por el ejercicio en relación con el descanso
- slope = (0) Pendiente ascendente
- ca = (0) Ningún vaso coloreado por fluoroscopia
- thal = (1) Trastorno sanguíneo denominado talasemia normal
- target = (1) Sí tiene enfermedad de corazón

Si analizamos este primer registro de datos, observamos la situación de un paciente de 63 años que sí sufre una enfermedad en el corazón, pero no debido a la angina de pecho, ya que la característica **cp** indica que el paciente no sufre dolor anginal. Sin embargo, existen otras características que sí pueden determinar la causa de su enfermedad. Por lo tanto, se destacan las siguientes:

- **fbs**, con un valor de más de 120 mg/dl de azúcar en sangre en ayunas.

  Esto es evidente que pueda ser un síntoma de peso para dicha enfermedad, ya que una cantidad elevada de azúcar en sangre puede estar produciendo una mayor necesidad de oxígeno.

- **chol**, con un valor de 233 mg/dl.

  Al no estar especializados en este tema, no se sabe con exactitud el significado del valor, es por ello por lo que haremos un estudio de los distintos niveles que existen para el colesterol.

Clasificación colesterol	Colesterol (mg/dl)
Deseable	< 200
Lmítrofe alto	200 - 239
Alto	> 249

Tabla 6. Clasificación de los niveles de colesterol.

Tal y como se observa en la tabla generada, cualquier valor de colesterol por debajo de los 200 mg/dl se encuentran dentro de los valores deseables, mientras que valores mayores o

iguales que 200 ya son cifras que no son óptimas. Es por ello por lo que el paciente, con un colesterol de 239 mg/dl no es un valor para nada óptimo, sino todo lo contrario.

- **trestbps**, con un valor de 145 mmHg.

  Lo mismo sucede con esta característica (no se entienden tampoco los valores con exactitud), por lo que también realizaremos un pequeño estudio de los distintos niveles.

Clasificación PA	Sistólica (mmHg)	Diastólica (mmHg)
Normal	< 120	< 80
Prehipertensión	120 - 139	80 - 89
Hipertensión Estadio 1	140 - 159	90 - 99
Hipertensión Estadio 2	≥ 160	≥ 100

Tabla 7. Clasificación de la presión arterial.

En esta tabla se muestra una clasificación de los distintos niveles que existen en función de la medición obtenida de la presión arterial, medida en mmHg. Por lo tanto, se observa como el paciente que estamos analizando se encuentra ante una situación de hipertensión, siendo este otro factor más de la causa de la enfermedad de corazón.

Pero ¿qué valor está proporcionando el *dataset*? ¿La presión arterial sistólica? ¿O la diastólica? Bien, como la información que se da no indica tal detalle, mediante técnicas de preparación de los datos como los histogramas o las cajas de bigotes, se puede deducir qué información se está proporcionando realmente.

Por otro lado, una vez analizado uno de los registros que se encuentran en el *dataset*, se llega a la conclusión que el ir uno por uno analizando cada caso supondría una gran cantidad de tiempo, e incluso sería imposible de analizar para grandes volúmenes de datos. Aquí es donde entra uno de los motivos por los que estas técnicas de aprendizaje automático son tan útiles, incluso en el sector de la medicina.

Para realizar un análisis correcto del *dataset* mediante técnicas de aprendizaje automático, primero se debe hacer una preparación de los datos. Para ello se debe realizar una limpieza, mediante la detección de *outlayers*, gracias a técnicas como histogramas y *box-plots* (cajas de bigotes).

Empezaremos mostrando un histograma de cada una de las características, y así podremos analizar a grandes rasgos que se encuentra en cada una de ellas para así poder detectar alguna anomalía/*outlayer*.

```
df.hist(figsize = (20,20));
```

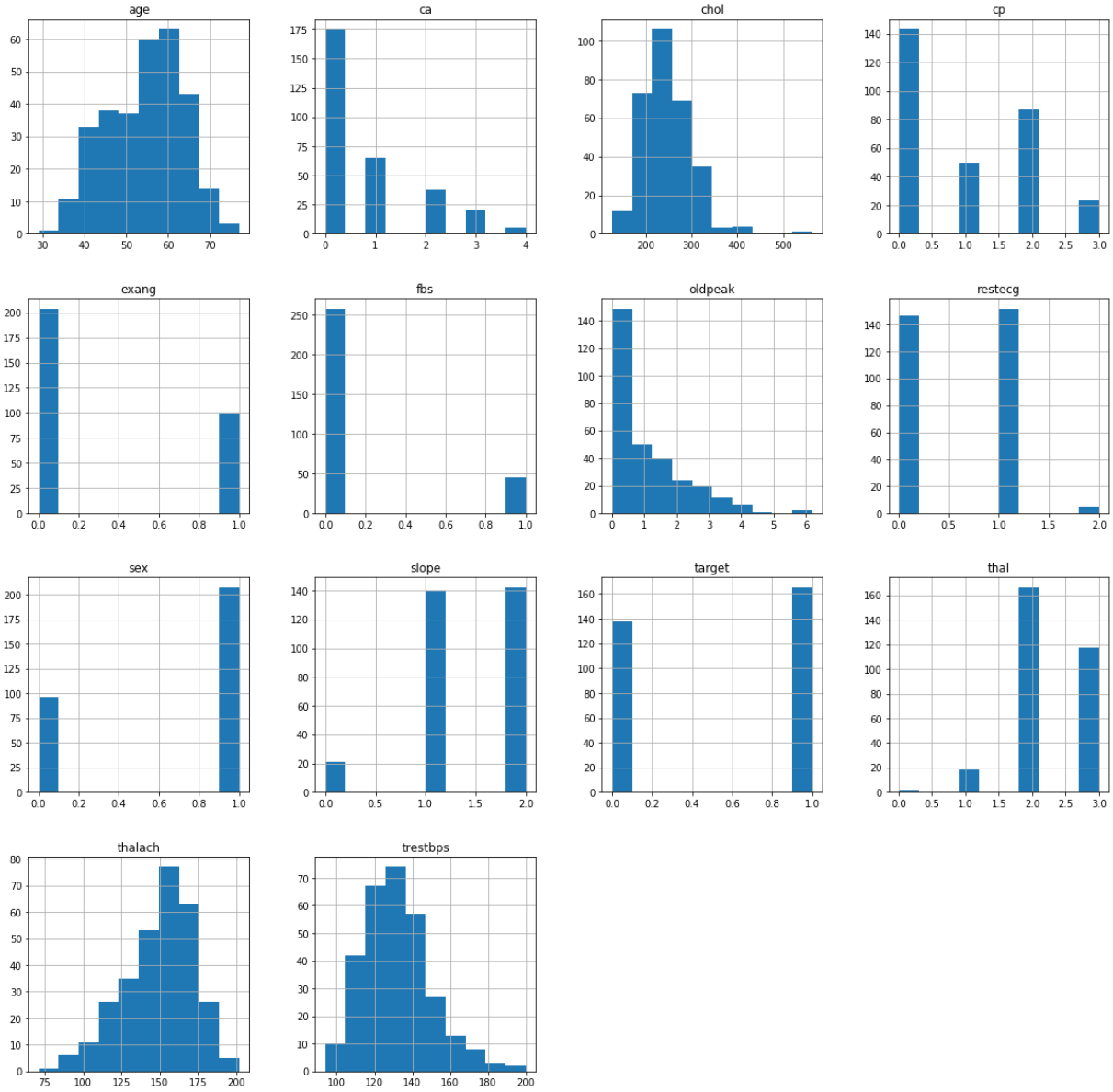

Ilustración 24. Histogramas de las distintas características.

Como se observa, el nivel de computación que ofrece es enorme. Con una única línea de código, mediante la función *hist()*, se obtiene un histograma de cada una de las características del *dataframe*, para poder así realizar un análisis global de manera muy sencilla.

Dichos histogramas también sirven para detectar anomalías/fallos en algunas de las características, en función del contexto en el que se trabaja. Con esto, al detectar posibles valores erróneos, dichos registros podrían ser detectados y eliminados con facilidad.

En este caso, detectar anomalías en características relacionados con el mundo de la medicina pueden ser incluso determinantes, es por ello por lo que no se realiza ningún tipo de limpieza en el *dataset*.

Observemos cada una de las características:

- age (edad): la franja de edades que se encuentran en el *dataset* son desde los 30 hasta casi 80 aproximadamente.
- ca (número de vasos principales coloreados por fluoroscopia): la información introductoria que se proporcionó indicaba que dichos valores estaban mapeados de 0 a 3, sin embargo, se observa como en realidad el mapeo de valores es de 0 a 4.
- chol (colesterol): con este histograma se observa como existen pacientes con valores muy por encima de lo deseado, incluso un valor anómalo aislado por encima de los 500.
- cp (tipo de dolor de pecho): lo mismo sucede aquí que con la característica *cp*, donde el mapeado real es de 0 a 3, y no de 0 a 4.
- exang (angina obtenida por el ejercicio): se observa como existen menos casos de angina obtenida por el ejercicio.
- fbs (azúcar en sangre en ayunas): donde casi alrededor del doble de los pacientes diagnosticados tienen más de 120 mg/dl de azúcar en sangre en ayunas.
- restecg (resutados electrocardiográficos en reposo): aquí el mapeo sí es correcto. Se detecta que es poco común encontrarse ante la situación de hipertrofia ventricular izquierda probable o definitiva según el criterio de Estes.
- sex (género): al parecer los hombres son más propensos a tener este tipo de enfermedades.
- slope (pendiente del segmento pico del ejercicio ST): la mayoría de los pacientes oscilan entre pendientes del tipo plano y del tipo descendente, indicando así que pocos tienen una pendiente del tipo normal.
- target (tiene o no enfermedad de corazón): se observa como existen muchos pacientes en el *dataset* que no parecen tener ninguna enfermedad de corazón.
- thal (trastorno sanguíneo denominado talasemia): se observa como existen algunos registros que presentan el valor 0, que se interpretan como pacientes que en un principio no tenían detectado dicho trastorno, por lo que, para que el *dataset* estuviera limpio de valores nulos, se le aplicó un valor 0.
- thalach (máximo ritmo cardíaco alcanzado): de primeras no se puede decir nada relevante viendo esta característica de forma aislada, ya que el máximo ritmo cardíaco que puede alcanzar una persona, que sea considerado como óptimo, depende de otros factores, entre ellos la edad y la cantidad de ejercicio físico a la que esté acostumbrado a hacer el paciente.

- trestbps (presión arterial al ingreso en el hospital): con esta información se deduce que es información respecto a la presión arterial sistólica, ya que los valores oscilan desde los 100 hasta los 200 aproximadamente.

Se han detectado una serie de fallos en cuanto al mapeo de valores en algunas características, por lo tanto, repasemos lo que finalmente significa cada una de las características, además de qué aportan los valores de las mismas.

1. Age = Edad (en años).
2. Sex = Género (1 = Masculino, 0 = Femenino).
3. cp = Tipo de dolor de pecho. (**0** = Angina típica, **1** = Angina atípica, **2** = Dolor no anginal, **3** = Asintomática).
4. trestbps = Presión arterial **sistólica** al ingreso en el hospital (mmHg).
5. chol = Colesterol (mg/dl).
6. fbs = Azúcar en sangre en ayunas > 120 mg/dl (1 = Verdadero, 0 = Falso).
7. restecg = Resutados electrocardiográficos en reposo (0 = Normal, 1 = Anomalía en la onda ST, 2 = Hipertrofia ventricular izquierda probable o definitiva según el criterio de Estes).
8. thalach = Máximo ritmo cardíaco alcanzado.
9. exang = Angina obtenida por el ejercicio (1 = Sí, 0 = No).
10. oldpeak = Depresión ST inducida por el ejercicio en relación con el descanso.
11. slope = Pendiente del segmento pico del ejercicio ST (0 = Pendiente ascendente, 1 = Plano, 2 = Pendiente descendente).
12. ca = Número de vasos principales coloreados por fluoroscopia (0-**4**).
13. thal = Trastorno sanguíneo denominado talasemia = (**0 =Valor nulo**, 1 = Normal, 2 = Defecto fijo, 3 = Defecto reversible).
14. target = Tiene o no enfermedad de corazón (1 = Sí, 0 = No).

Una vez observados a grandes rasgos las características del *dataset*, se analizan algunas de ellas en más detalle, mediante las cajas de bigotes. Analizando los histogramas se seleccionan las siguientes características:

- age (edad)
- chol (colesterol)
- thalach (trastorno sanguíneo denominado talasemia)
- trestbps (presión arterial sistólica en mmHg)

```
plt.boxplot(df.age);
```

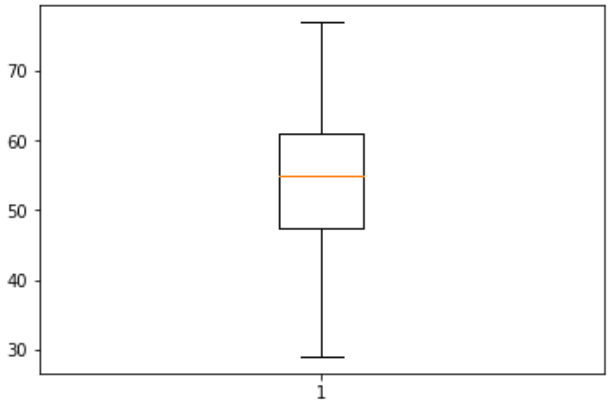

Ilustración 25. Caja de bigotes de la característica *age*.

A simple vista, destaca la media de edad en la que se encuentran los pacientes, alrededor de los 55 años, mientras que la mínima edad es de unos 30, y la máxima de más de 70 años. Esto, unido a saber el género del paciente, podría ser un factor determinante a la hora de diagnosticar en función de los distintos síntomas.

```
plt.boxplot(df.chol);
```

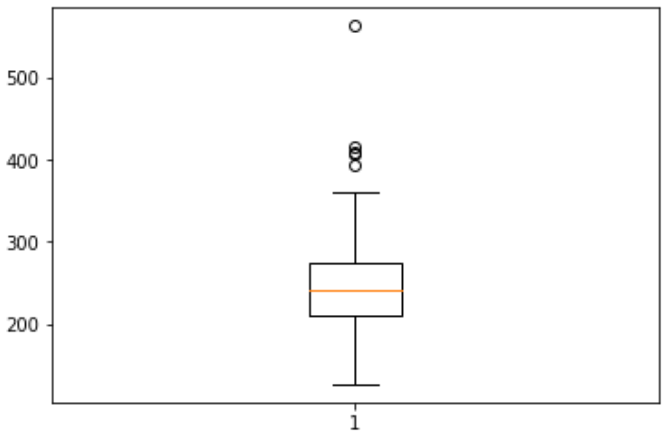

Ilustración 26. Caja de bigotes de la característica *chol*.

Aquí se observan datos atípicos, alejándose estos de los valores normales. En este caso, los valores podrían ser muy determinantes ya que, en el ámbito de la medicina, valores diferentes a lo normal suelen ser muy importantes. Además, la media de colesterol se encuentra en unos 250 mg/dl, siendo esto un valor muy alto para el colesterol de una persona normal.

```
plt.boxplot(df.thalach);
```

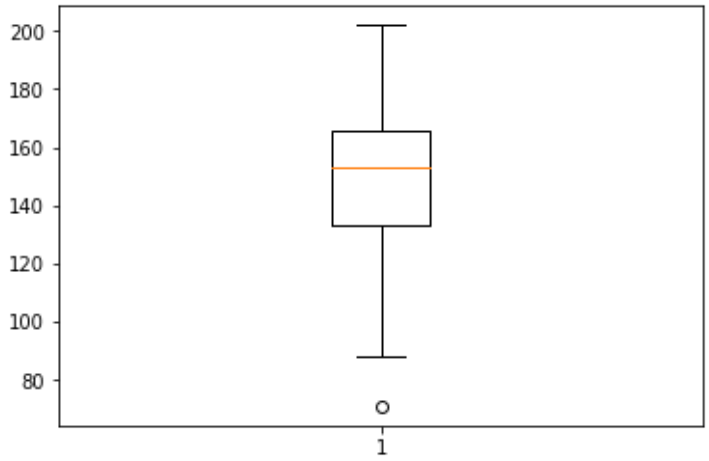

Ilustración 27. Caja de bigotes de la característica *thalach*.

En esta característica no se saca nada en claro ya que, en función de la edad y género del paciente, estos valores pueden ser normales o no, aunque sí es cierto que valores por debajo de 100 no son ideales.

```
plt.boxplot(df.trestbps);
```

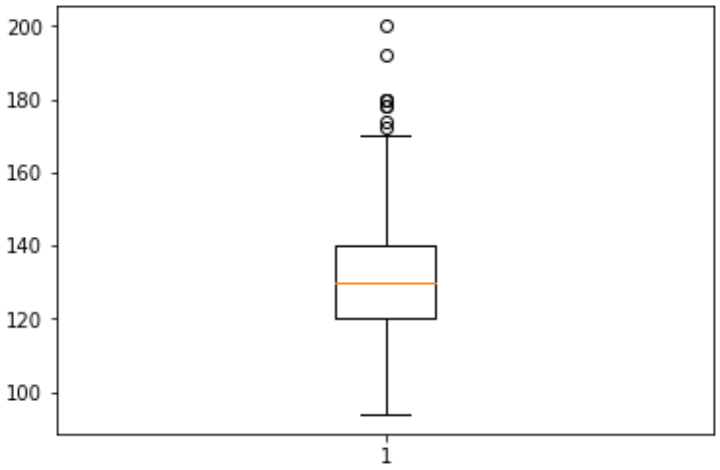

Ilustración 28. Caja de bigotes de la característica *trestbps*.

Aquí, la media de los pacientes se encuentra en un nivel entre prehipertensión e hipertensión de estadio 1 (sobre unos 130 mmHg aproximadamente). Además, se observan valores anómalos donde los niveles de presión arterial sistólica son muy elevados.

Como conclusión, a pesar de encontrar valores anómalos en algunas características, no las vamos a tomar como errores ya que pueden ser un punto definitivo a la hora de realizar un diagnóstico médico.

Respecto a la preparación de los datos, existen varias formas más de detectar datos faltantes o realizar una limpieza del conjunto de datos a analizar, mediante la media, la moda o analizando el valor previsto. También existen casos en el que se pueden ignorar características,

debido a que no aportan ningún tipo de información relevante al análisis. Pero, como nuestro conjunto de datos no es muy amplio, y que no queremos eliminar ningún registro y/o característica, debido a valores anómalos, el haber analizado el *dataset* con histogramas y cajas de bigotes resulta más que suficiente.

5.2.1 Aplicando técnicas predictivas

Una vez analizado el conjunto de datos, estamos preparados para generar una serie de modelos utilizando distintas técnicas estudiadas teóricamente. Cabe recordar que, para realizar un correcto análisis, se debe dividir el conjunto de datos en dos subconjuntos, denominados *training* y *test*, respectivamente.

La finalidad de dividir los datos es no realizar un sobreajuste en el algoritmo que se va a producir, ya que, si se emplearan todas las técnicas al conjunto de datos completo, como se conocen los valores de las etiquetas para cada registro, ajustaríamos el algoritmo a dichas etiquetas. Sin embargo, si se excluyen algunos registros al subconjunto de *test*, con este se podrá determinar si nuestro algoritmo está sobreajustado o no, y si el algoritmo está bien realizado para futuros registros que se vayan incluyendo en el conjunto de datos.

Como hemos visto en el apartado 2.4, Conjuntos de datos: *training*/*test*, validación cruzada, LOO, *bootstrapping*, existen varias formas de realizar dicha división. Además, resulta ser una técnica esencial para el correcto procesamiento del modelo. En esta ocasión utilizaremos la técnica de validación cruzada de K iteraciones (*K-Fold Cross Validation*), ya que como se describe en el apartado 2.4.1 Validación cruzada (*Cross Validation*), los datos, al dividirse en *k* iteraciones, siendo cada k distinta, evitaremos en gran medida el sobreajuste del modelo. Empecemos con las técnicas.

Se puede realizar la clasificación mediante estados binarios, es decir, sí o no, o mediante un número finito de estados, es decir, múltiples. Observando nuestro conjunto de datos, podremos realizar varios análisis, utilizando tanto estados binarios como múltiples, como, por ejemplo:

Estados binarios

- Género del paciente, ya que será hombre o mujer.
- Azúcar en sangre en ayunas, ya que está clasificado en 0 si es menor de 120 mg/dl, o 1 si es mayor de 120 mg/dl.
- Si sufre enfermedad de corazón o no.

Estados múltiples

- Tipo de angina de pecho que sufre, en el caso de que la sufra, según el género del paciente.
- Tipo de resultados electrocardiográficos en reposo que muestra el paciente.
- Tipo de trastorno sanguíneo denominado talasemia, en el caso de que lo sufra.

Existen muchos ejemplos que se podrían dedicar a nuestro conjunto de datos, solo muestro algunos ejemplos para entender mejor de que hablamos cuando mencionamos estados binarios o múltiples.

Analicemos los pacientes que sufren este tipo de enfermedad en función del género y para cada una de las edades:

```
# Edad vs target
sb.set_context("notebook", font_scale = 0.5, rc = {"font.size": 15,"axes.titlesize"
: 15,"axes.labelsize": 15})
sb.catplot(kind = 'count', data = df, x = 'age', hue = 'target', order = df['age'].
sort_values().unique())
plt.title('Variation of Age for each target class')
plt.show()
```

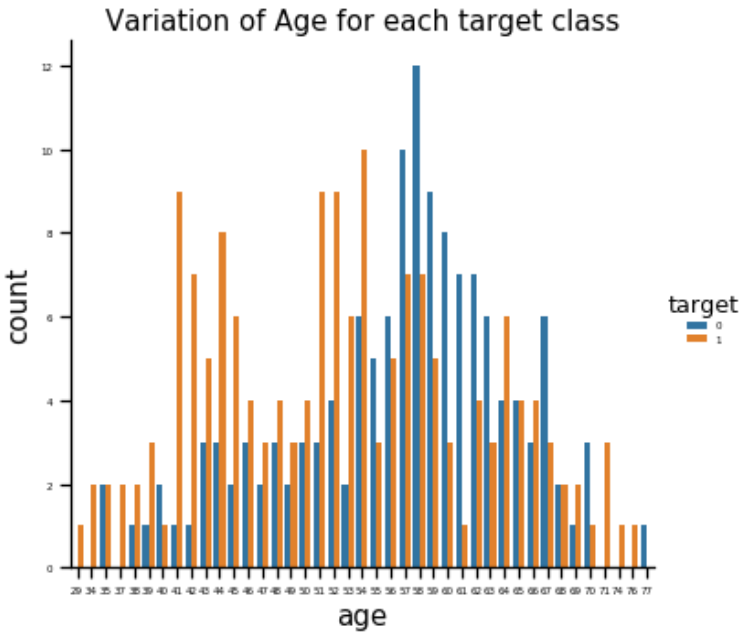

Ilustración 29. Gráfica del número de pacientes enfermos o no en función de la edad.

Esta gráfica muestra el número de pacientes que sufren o no este tipo de enfermedad en función de la edad. Como podemos observar, en la franja entre los 40 y 55 años se encuentran los pacientes que más sufren el riesgo de tener enfermedad de corazón. Mostremos ahora un gráfico diferenciando por género:

```
# Agrupamos por genero para observar número de pacientes por genero
print(df.groupby("sex").size())

# Diagrama de barras edad vs género en funcion de si sufre enfermedad o no
sb.catplot(kind = 'bar', data = df, y = 'age', x = 'sex', hue = 'target')
```

```
plt.title('Edad vs género en función de target')
plt.show()
```

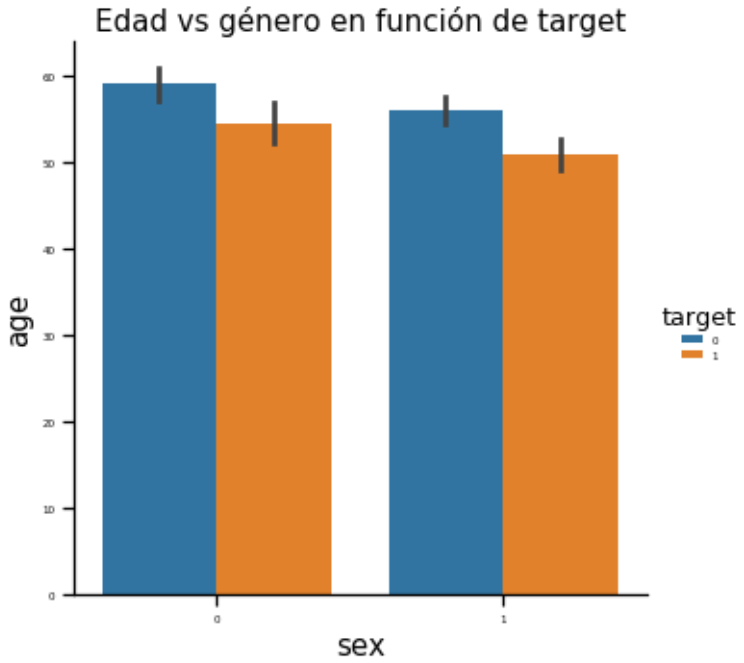

Ilustración 30. Gráfica edad vs género en función de *target*.

Como se puede observar, tenemos 96 pacientes de género masculino y 207 de género femenino. Ya solo con esto identificamos que la ratio de mujeres es el doble respecto a la de hombres, observando un desbalanceo importante en cuanto al número de hombres y mujeres, pudiendo producir que dicho desbalanceo afecte notablemente a nuestro algoritmo si quisiéramos realizar un modelo según el género del paciente. También podemos interrelacionar las características en función del sexo, para ver cómo se concentran las distintas salidas. Sin embargo, observamos que existen más hombres que mujeres que sufren la enfermedad, a pesar de que haya más de un 50% de mujeres respecto a los hombres.

A continuación, se aplicarán las siguientes técnicas predictivas:

- Regresión Logística
- Naive Bayes
- Random Forest
- KNN

Posterior a su aplicación, se analizarán y compararán los resultados según una serie de métricas que evalúan la calidad del modelo predictivo.

5.2.2 Regresión Logística

Utilizaremos la función "*sklearn.linear_model.LogisticRegression*" como técnica de clasificación [44].

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
# Para normalizar los datos
sc = StandSc();


# Omitimos columna target en el conjunto de entrenamiento
dataX = df.drop('target', axis=1)


# Incluimos únicamente target al conjunto de prueba
dataY = df['target']


# Variable para almacenamiento de los distintos accuracy's para cada modelo
results = {}


# Variable para almacenar cada uno de los accuracy's para cada split de datos
lr_resultList = []


# Inicialización de matriz de confusión vacía
lr_confmat_test_result = np.zeros((2,2))


# Variable que almacena las probabilidades a posteriori
lr_y_test = []
lr_pred_prob = []


# Modelo regresión logística
lr = LogisticRegression(solver="lbfgs")
kf= KFold(n_splits = 5, shuffle=True)
for train_index,test_index in kf.split(dataX):
    # Seteamos índices generados por KFold en subconjuntos de entrenamiento y test
    X_train, X_test = dataX.iloc[train_index], dataX.iloc[test_index];
    y_train, y_test = dataY[train_index], dataY[test_index];


    # Guardamos valores verdaderos para posterior análisis de curva ROC
    lr_y_test.extend(y_test)


    # Normalizamos nuestros datos previo a aplicar la técnica
    # ya que la escala de las columnas es distinta
    X_train = sc.fit_transform(X_train);
    X_test = sc.transform(X_test);


    # Aplicamos la técnica
    fit_lr = lr.fit(X_train, y_train);


    # Realizamos la predicción
    y_pred = lr.predict(X_test);


    # Almacenamos probabilidades a posteriori
    lr_pred_prob.extend(lr.predict_proba(X_test)[:,1])


    # Matriz de confusión del conjunto de datos de prueba
```

```
    lr_confmat_test = confusion_matrix(y_pred, y_test);
    lr_confmat_test_result += lr_confmat_test;

    # Almacenamos el resultado, en % acertado
    result = lr.score(X_test, y_test)*100
    lr_resultList.append(result)

    # Almacenamos resultado para posterior análisis y comparación
    results['Logistic Regression'] = statistics.mean(lr_resultList)

    # Mostramos resultados por cada split de datos
    print("Precision Regresión Logistica = {:.3f}%".format(result))
```

Precisión Regresión Logística = 88.525%

Precisión Regresión Logística = 78.689%

Precisión Regresión Logística = 83.607%

Precisión Regresión Logística = 86.667%

Precisión Regresión Logística = 78.333%

5.2.3 Naive Bayes

Aplicaremos el método de Naive Bayes gaussiano, adecuado para datos con resultados binarios, siendo este nuestro caso. En el caso de que se quisiera evaluar en función de más de una clase, como por ejemplo la clasificación en función del tipo de "cp", tendríamos que utilizar la función de Naive Bayes Multinomial. Para ello utilizaremos la función "*sklearn.naive_bayes.GaussianNB"* como técnica de clasificación [45].

```
# Naive Bayes gausiana
from sklearn.naive_bayes import GaussianNB

# Variable para almacenar cada uno de los accuracy's para cada split de datos
nb_resultList = []

# Inicialización de matriz de confusión vacía
nb_confmat_test_result = np.zeros((2,2))

# Variable que almacena las probabilidades a posteriori
nb_y_test = []
nb_pred_prob = []

# Modelo Naive Bayes
nb = GaussianNB()
kf= KFold(n_splits = 5, shuffle=True)
for train_index,test_index in kf.split(dataX):
    # Seteamos índices generados por KFold en subconjuntos de entrenamiento y test
    X_train, X_test = dataX.iloc[train_index], dataX.iloc[test_index];
    y_train, y_test = dataY[train_index], dataY[test_index];

    # Guardamos valores verdaderos para posterior análisis de curva ROC
    nb_y_test.extend(y_test)
```

```
    # Normalizamos nuestros datos previo a aplicar la técnica
    # ya que la escala de las columnas es distinta
    X_train = sc.fit_transform(X_train);
    X_test = sc.transform(X_test);

    # Aplicamos la técnica
    fit_nb = nb.fit(X_train, y_train);

    # Realizamos la predicción
    y_pred = nb.predict(X_test);

    # Almacenamos probabilidades a posteriori
    nb_pred_prob.extend(nb.predict_proba(X_test)[:,1])

    # Matriz de confusión del conjunto de datos de prueba
    nb_confmat_test = confusion_matrix(y_pred, y_test);
    nb_confmat_test_result += nb_confmat_test

    # Almacenamos el resultado, en % acertado
    result = nb.score(X_test, y_test)*100
    nb_resultList.append(result)

    # Almacenamos resultado para posterior análisis y comparación
    results['Naive Bayes'] = statistics.mean(nb_resultList)

    # Mostramos resultados por cada split de datos
    print("Precision Naive Bayes = {:.3f}%".format(result))
```

Precisión Naive Bayes = 83.607%

Precisión Naive Bayes = 81.967%

Precisión Naive Bayes = 77.049%

Precisión Naive Bayes = 80.000%

Precisión Naive Bayes = 85.000%

5.2.4 Random Forest

Utilizaremos la función "*sklearn.ensemble.RandomForestClassifier*" como técnica de clasificación [46].

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

# Variable para almacenar cada uno de los accuracy's para cada split de datos
rf_resultList = []

# Inicialización de matriz de confusión vacía
rf_confmat_test_result = np.zeros((2,2))

# Variable que almacena las probabilidades a posteriori
rf_y_test = []
rf_pred_prob = []

rf = RandomForestClassifier(n_estimators = 400)
kf= KFold(n_splits = 5, shuffle=True)
for train_index,test_index in kf.split(dataX):
    # Seteamos índices generados por KFold en subconjuntos de entrenamiento y test
    X_train, X_test = dataX.iloc[train_index], dataX.iloc[test_index];
    y_train, y_test = dataY[train_index], dataY[test_index];
```

```
    # Guardamos valores verdaderos para posterior análisis de curva ROC
    rf_y_test.extend(y_test)

    # Normalizamos nuestros datos previo a aplicar la técnica
    # ya que la escala de las columnas es distinta
    X_train = sc.fit_transform(X_train);
    X_test = sc.transform(X_test);

    # Aplicamos la técnica
    fit_rf = rf.fit(X_train, y_train);

    # Realizamos la predicción
    y_pred = rf.predict(X_test);

    # Almacenamos probabilidades a posteriori
    rf_pred_prob.extend(rf.predict_proba(X_test)[:,1])

    # Matriz de confusión del conjunto de datos de prueba
    rf_confmat_test = confusion_matrix(y_pred, y_test);
    rf_confmat_test_result += rf_confmat_test

    # Almacenamos el resultado, en % acertado
    result = rf.score(X_test, y_test)*100
    rf_resultList.append(result)

    # Almacenamos resultado para posterior análisis y comparación
    results['Random Forest'] = statistics.mean(rf_resultList)

    # Mostramos resultados por cada split de datos
    print("Precision Random Forest = {:.3f}%".format(result))
```

Precisión Random Forest = 81.967%

Precisión Random Forest = 83.607%

Precisión Random Forest = 81.967%

Precisión Random Forest = 88.333%

Precisión Random Forest = 81.667%

5.2.5 KNN

Uno de los parámetros determinantes a la hora de utilizar la técnica de KNN es determinar el número de vecinos que se deben fijar, "*n_neighbors*". Por ello, resulta imprescindible tratar de averiguar cuál es la *k* óptima previo a la generación del modelo de predicción. Utilizaremos la función "*sklearn.neighbors.KNeighborsClassifier"* como técnica de clasificación [47].

```
from sklearn.neighbors import KNeighborsClassifier

k_resultList = []
k_counter = 1

# Tratamos de averiguar cuál es la mejor K para el conjunto de datos
for k in range(1,21):
    knn = KNeighborsClassifier(n_neighbors = k)

    # Aplicamos la técnica
    fit_knn = knn.fit(X_train, y_train)

    # Realizamos la predicción
```

```
    y_pred = knn.predict(X_test);

    # Almacenamos el resultado, en % acertado
    result = knn.score(X_test, y_test)*100
    k_resultList.append(result)

    if k_counter == 20:
        # Pintamos gráfica de precisión para cada k
        plt.title("Gráfica elección mejor k")
        plt.plot(range(1,21), k_resultList)
        plt.xticks(np.arange(1,21,1))
        plt.yticks(np.arange(min(k_resultList),max(k_resultList)))
        plt.xlabel("K")
        plt.ylabel("Precision")
        plt.show()

        #Volvemos a setear valores
        k_counter = 1
        k_resultList = []

    else:
        k_counter += 1
```

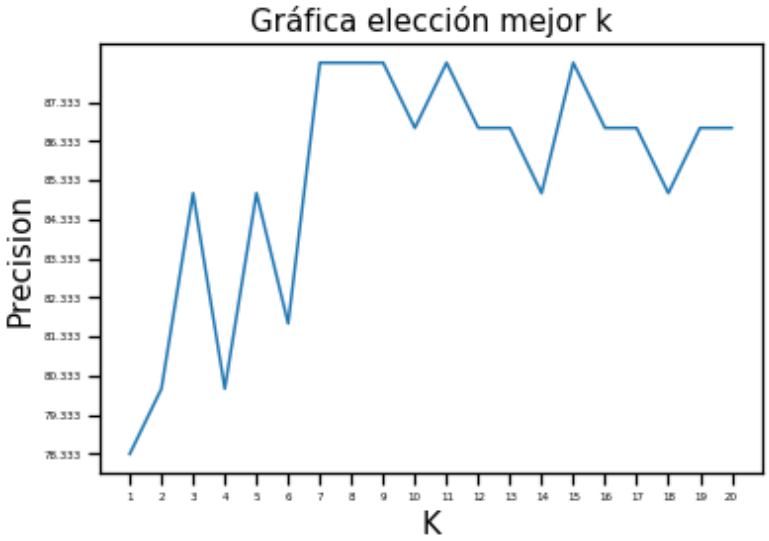

Ilustración 31. Gráfica de precisión en función de *k*.

Como se observa en la gráfica, para $K = 9$ la precisión es la más elevada. Por lo tanto, utilizaremos esta K para la generación del modelo.

```
from sklearn.neighbors import KNeighborsClassifier
# Variable para almacenar cada uno de los accuracy's para cada split de datos
knn_resultList = []

# Inicialización de matriz de confusión vacía
knn_confmat_test_result = np.zeros((2,2))

# Variable que almacena las probabilidades a posteriori
knn_y_test = []
knn_pred_prob = []

knn = KNeighborsClassifier(n_neighbors = 9)
kf= KFold(n_splits = 5, shuffle=True)
for train_index,test_index in kf.split(dataX):
```

```
    # Seteamos índices generados por KFold en subconjuntos de entrenamiento y test
    X_train, X_test = dataX.iloc[train_index], dataX.iloc[test_index];
    y_train, y_test = dataY[train_index], dataY[test_index];

    # Guardamos valores verdaderos para posterior análisis de curva ROC
    knn_y_test.extend(y_test)

    # Normalizamos nuestros datos previo a aplicar la técnica
    # ya que la escala de las columnas es distinta
    X_train = sc.fit_transform(X_train);
    X_test = sc.transform(X_test);

    # Aplicamos la técnica
    fit_knn = knn.fit(X_train, y_train);

    # Realizamos la predicción
    y_pred = knn.predict(X_test);

    # Almacenamos probabilidades a posteriori
    knn_pred_prob.extend(knn.predict_proba(X_test)[:,1])

    # Matriz de confusión del conjunto de datos de prueba
    knn_confmat_test = confusion_matrix(y_pred, y_test);
    knn_confmat_test_result += knn_confmat_test

    # Almacenamos el resultado, en % acertado
    result = knn.score(X_test, y_test)*100
    knn_resultList.append(result)

    # Almacenamos resultado para posterior análisis y comparación
    results['KNN'] = statistics.mean(knn_resultList)

    # Mostramos resultados por cada split de datos
    print("Precision KNN = {:.3f}%".format(result))
```

Precisión KNN = 78.689%

Precisión KNN = 85.246%

Precisión KNN = 73.770%

Precisión KNN = 85.000%

Precisión KNN = 85.000%

Una vez realizados los distintos modelos para cada una de las técnicas, entendamos y analicemos cada una de sus salidas.

Como se puede observar en cada una de ellas, al estar utilizando la función “KFold” fijando el número de “splits” a cinco, se genera cinco resultados distintos de predicción. Para poder realizar una comparación entre ellas, se obtiene como resultado final la media de cada uno de los valores obtenidos para cada algoritmo.

5.2.6 Análisis de resultados

Como hemos comentado en el Capítulo 3. Métricas, existen muchas formas de estudiar si el modelo que hemos generado resulta ser eficiente o no. Por lo tanto, para analizar los modelos que hemos generado, vamos a basarnos en la comparación de las siguientes métricas:

- Precisión (Accuracy)
- Matriz de confusión (Confusion matrix) y sensibilidad y especificidad
- Ranking: curvas ROC y AUC

5.2.6.1 Precisión (*Accuracy*)

```
sb.set_style("darkgrid")
plt.figure(figsize=(20,7))
plt.yticks(np.arange(0,100,10))
plt.xlabel("Precisión")
plt.ylabel("Técnicas predictivas")
sb.barplot(y=list(results.keys()), x=list(results.values()), orient="h")
plt.show()

for key, value in results.items():
    print("Precisión para %s = " % key, end = '')
    print("{0:.3f}%".format(value))
```

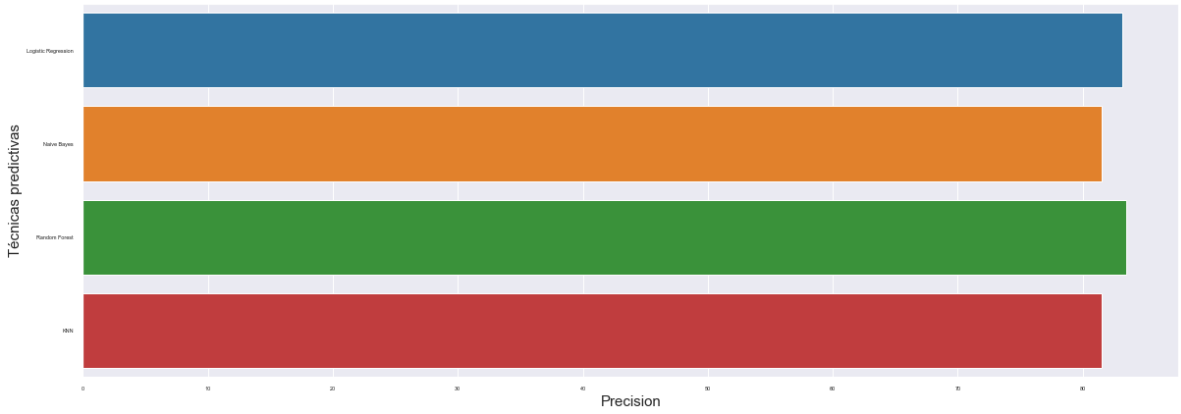

Ilustración 32. Gráfica de comparación de precisiones.

Precisión para Logistic Regression = 83.164%

Precisión para Naive Bayes = 81.525%

Precisión para Random Forest = 83.508%

Precisión para KNN = 81.541%

Como vimos en 3.1.3. Exactitud (*Accuracy*), esta métrica se define como el porcentaje total de elementos clasificados correctamente, es decir, el número de predicciones que se han realizado correctamente, dividido por el número total de predicciones. Esto es:

$$\boldsymbol{Exactitud} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} \quad (22)$$

Comparando los distintos modelos generados se puede observar como el algoritmo de Random Forest resulta ser el modelo más preciso, mientras que Naive Bayes y KNN se sitúan como los peores clasificadores de la comparación realizada. Sin embargo, parar nuestro análisis de resultados únicamente basándonos en la exactitud sería un error, ya que esta métrica por sí

sola no define la calidad del modelo generado. Por ello, veamos las matrices de confusión, la sensibilidad y la especificidad de estos.

5.2.6.1.1 Matriz de confusión (Confusion matrix) y sensibilidad y especificidad

Para ello haremos uso de la función *"sklearn.metric.confusion_matrix"* [48].

```
fig = plt.figure(figsize = (10,10))
lr_cm_plot = fig.add_subplot(2, 2, 1) # row, column, position
nb_cm_plot = fig.add_subplot(2, 2, 2)
rf_cm_plot = fig.add_subplot(2, 2, 3)
knn_cm_plot = fig.add_subplot(2, 2, 4)

# Matriz confusión Logistic Regression
lr_cm_plot.title.set_text('Logistic Regression')
sb.heatmap(data = lr_confmat_test_result, annot=True, fmt = 'g', ax=lr_cm_plot);

# Matriz confusión Naive Bayes
nb_cm_plot.title.set_text('Naive Bayes')
sb.heatmap(data = nb_confmat_test_result, annot=True,fmt = 'g', ax=nb_cm_plot);

# Matriz confusión Random Forest
rf_cm_plot.title.set_text('Random Forest')
sb.heatmap(data = rf_confmat_test_result, annot=True,fmt = 'g', ax=rf_cm_plot);

# Matriz confusión KNN
knn_cm_plot.title.set_text('KNN')
sb.heatmap(data = knn_confmat_test_result, annot=True,fmt = 'g', ax=knn_cm_plot);
```

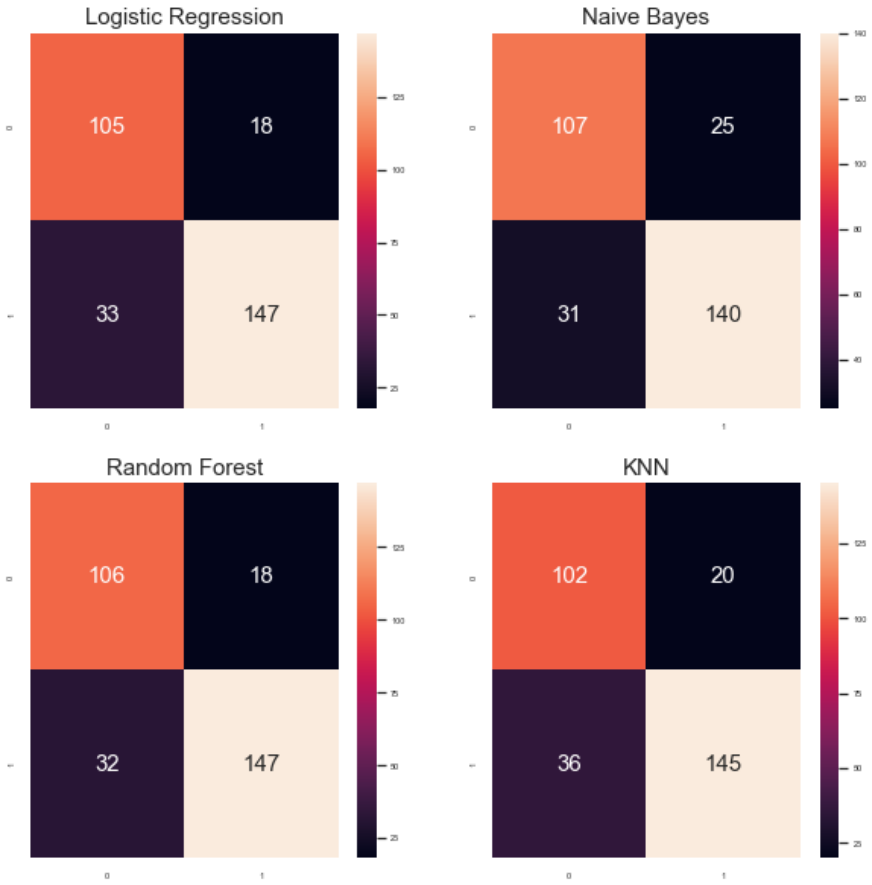

Ilustración 33. Matrices de confusión para las distintas técnicas.

Los valores para nuestra matriz de confusión para la técnica de **Regresión Logística** han sido:

- Verdaderos positivos: 105
- Verdaderos negativos: 147
- Falsos positivos: 18
- Falsos negativos: 33

Del modelo que hemos generado, ha sido capaz de acertar 252 de las 303 predicciones posibles, es decir, un 83.2% de las veces. A priori no parece un mal resultado, analicémoslo más en detalle. Es cierto que el modelo ha sido capaz de predecir 106 verdaderos positivos, pero ha fallado en un total de 33 ocasiones, indicando que estos pacientes no sufrían ninguna enfermedad, cuando en realidad sí. Esto quiere decir que, de 139 verdaderos positivos, 33 han sido mal identificados, provocando que exista un fallo de un 23.8% cuando hablamos de pacientes que sufren este tipo de enfermedad realmente.

Por otro lado, para pacientes que no sufren ninguna enfermedad, ha sido capaz de identificar 147 pacientes de los 165 posibles, queriendo decir esto que únicamente ha fallado en un 10.9% de los casos aproximadamente, siendo esto un porcentaje de error bastante bajo.

Resumiendo, la técnica de regresión logística resulta ser más precisa a la hora de identificar pacientes que no sufren ninguna enfermedad, pero no tanto cuando hablamos de tratar de identificar pacientes con algún tipo de enfermedad de corazón. Si estuviéramos en un contexto donde los falsos negativos fueran medianamente permitidos, esta técnica resultaría ser un buen clasificador. Pero como nos encontramos ante un contexto médico, el margen de error debe ser mínimo.

Los valores para nuestra matriz de confusión para la técnica de **Naive Bayes** han sido:

- Verdaderos positivos: 107
- Verdaderos negativos: 140
- Falsos positivos: 25
- Falsos negativos: 31

Del modelo que hemos generado, ha sido capaz de acertar 247 de las 303 predicciones posibles, es decir, un 81.5% de las veces. Es cierto que el modelo ha sido capaz de predecir 107 verdaderos positivos, donde solo ha fallado un total de 25 resultados. Esto quiere decir que, de 132 verdaderos positivos, 25 han sido mal identificados, provocando que exista un fallo de un 18.9% cuando hablamos de pacientes que sufren este tipo de enfermedad realmente.

Por otro lado, para pacientes que no sufren ninguna enfermedad, ha sido capaz de identificar 140 pacientes de los 171 posibles, queriendo decir esto que ha fallado en un 18.1% de los casos aproximadamente.

Resumiendo, la técnica de Naive Bayes resulta ser prácticamente igual de precisa a la hora de identificar tanto pacientes enfermos, como sanos ya que según los resultados, tiene un porcentaje de error bastante similar tanto en la predicción de pacientes enfermos, como de pacientes sanos. Comparando con los resultados obtenidos a partir de la clasificación mediante la técnica de Regresion Logística, resulta ser un mejor clasificador a la hora de predecir pacientes que sí sufren alguna enfermedad, pero menos preciso a la hora de identificar pacientes sanos.

Los valores para nuestra matriz de confusión para la técnica de **Random Forest** han sido:

- Verdaderos positivos: 106
- Verdaderos negativos: 147
- Falsos positivos: 18
- Falsos negativos: 32

Del modelo que hemos generado, ha sido capaz de acertar 253 de las 303 predicciones posibles, es decir, un 83.5% de las veces. Es cierto que el modelo ha sido capaz de predecir 106 verdaderos positivos, pero ha fallado un total de 18 resultados. Esto quiere decir que, de 124 verdaderos positivos, 18 han sido mal identificados, provocando que exista un fallo de un 14.5% cuando hablamos de pacientes que sufren este tipo de enfermedad realmente.

Por otro lado, para pacientes que no sufren ninguna enfermedad, ha sido capaz de identificar 147 pacientes de los 179 pacientes posibles, queriendo decir esto que ha fallado hasta en un 17.9% de los casos aproximadamente.

Resumiendo, la técnica de Random Forest resulta ser más preciso a la hora de identificar pacientes enfermos que pacientes sanos. Comparado con los resultados obtenidos a partir de la clasificación mediante la técnica de Naive Bayes, resulta ser un mejor clasificador a la hora de predecir pacientes que sí sufren alguna enfermedad, y similar a la hora de identificar pacientes sanos. Y comparando con el modelo de regresión logística las conclusiones son similares, ya que resulta ser un mejor clasificador para identificar pacientes con enfermedad, pero bastante peor para predecir pacientes sanos.

Los valores para nuestra matriz de confusión para la técnica de **KNN** han sido:

- Verdaderos positivos: 102
- Verdaderos negativos: 145

- Falsos positivos: 20
- Falsos negativos: 36

Del modelo que hemos generado, ha sido capaz de acertar 247 de las 303 predicciones posibles, es decir, un 81.5% de las veces. El modelo ha sido capaz de predecir 102 verdaderos positivos, pero ha fallado un total de 20 ocasiones. Esto quiere decir que, de 122 verdaderos positivos, 20 han sido mal identificados, provocando que exista un fallo de un 16.4% cuando hablamos de pacientes que sufren este tipo de enfermedad realmente.

Por otro lado, para pacientes que no sufren ninguna enfermedad, ha sido capaz de identificar 145 pacientes de los 181 pacientes posibles. Esto quiere decir que ha fallado hasta en un 19.9% de los casos aproximadamente.

Resumiendo, la técnica de KNN resulta ser más precisa a la hora de identificar pacientes enfermos, que pacientes sanos. Comparando con el resto de los clasificadores, resulta ser el modelo menos preciso a la hora de identificar pacientes sanos, mientras que para identificar pacientes sanos sólo el modelo de Random Forest ha sido capaz de realizar una mejor predicción.

Ahora mismo, habiendo realizado un análisis más exhaustivo de los resultados, se podría decir que el algoritmo de Random Forest como clasificador resulta ser el más preciso en comparación al resto, pero veamos algunas métricas más.

```
def specAndSens(cm, name):
    specificity = cm[1,1]/(cm[1,1]+cm[0,1])*100
    print('Especificidad para %s = ' % name, end = '')
    print("{0:.2f}%".format(specificity))
    sensitivity = cm[0,0]/(cm[0,0]+cm[1,0])*100
    print('Sensibilidad para %s = ' % name, end = '')
    print("{0:.2f}%\n".format(sensitivity))
```

```
specAndSens(lr_confmat_test_result, "Logistic Regression")
specAndSens(nb_confmat_test_result, "Naive Bayes")
specAndSens(rf_confmat_test_result, "Random Forest")
specAndSens(knn_confmat_test_result, "KNN")
```

Especificidad para Logistic Regression = 89.09%

Sensibilidad para Logistic Regression = 76.09%

Especificidad para Naive Bayes = 84.85%

Sensibilidad para Naive Bayes = 77.54%

Especificidad para Random Forest = 89.09%

Sensibilidad para Random Forest = 76.81%

Especificidad para KNN = 87.88%
Sensibilidad para KNN = 73.91%

Como comentamos teóricamente en el apartado 3.1.1 FP, FN, TP, Sensibilidad y especificidad, la sensibilidad es una métrica que evalúa la capacidad para acertar los verdaderos positivos, es decir, los pacientes que sufren la enfermedad, y la predicción se ha realizado correctamente; mientras que la especificidad evalúa la capacidad para acertar los verdaderos negativos, es decir, los pacientes que no sufren esta enfermedad, y la predicción también se ha realizado de forma correcta.

Por lo tanto, de los resultados obtenidos respecto a la sensibilidad y la especificidad se obtienen conclusiones similares, ya que los cuatro modelos coinciden en la gran capacidad para detectar a pacientes sanos (especificidad elevada), pero resultan ser menos precisos a la hora de predecir pacientes enfermos (sensibilidad).

5.2.6.1.2 Ranking: curvas ROC y AUC

Generaremos la curva ROC a partir de las probabilidades a posteriori obtenidas de los distintos modelos de predicción, y mediante la función *"sklearn.metric.roc_curve"* [49] y *"sklearn.metric.auc"* [50] compararemos dichas predicciones con sus clases reales, trazando así la curva que se ve en la gráfica.

```
# Función para generar gráfica curva ROC
def createCurveROC(real_values, pred_probs, name):
    fpr, tpr, thresholds = roc_curve(real_values, pred_probs)

    # Cuanto mayor sea el AUC, mejor será nuestro modelo
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, lw=3, alpha=1.0, label='AUC = %0.3f' % (roc_auc))
    plt.plot([0,1],[0,1],ls = '--',color = 'black')
    plt.title("Curva ROC para clasificador %s" % name)
    plt.xlabel("FPR (1 - Especificidad)")
    plt.ylabel("TPR (Sensibilidad)")
    plt.legend(loc="lower right", prop={'size': 20})
```

```
createCurveROC(lr_y_test, lr_pred_prob, "Logistic Regresion")
```

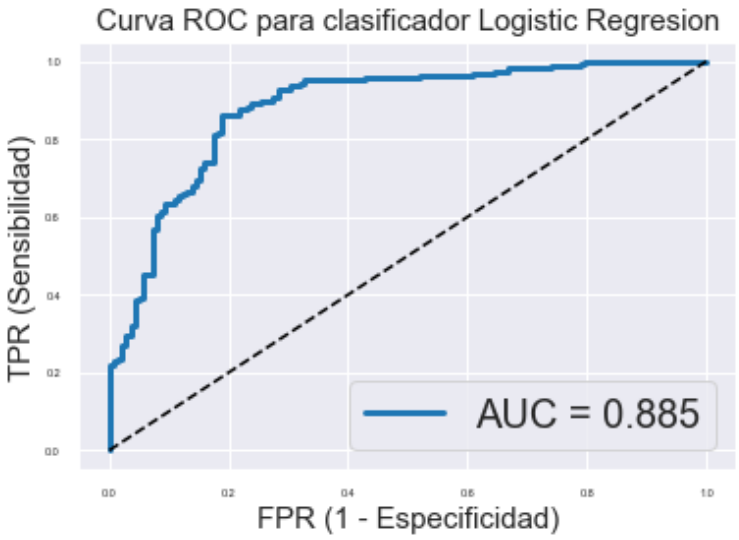

Ilustración 34. Gráfica curva ROC Regresión Logística.

```
createCurveROC(nb_y_test, nb_pred_prob, "Naive Bayes")
```

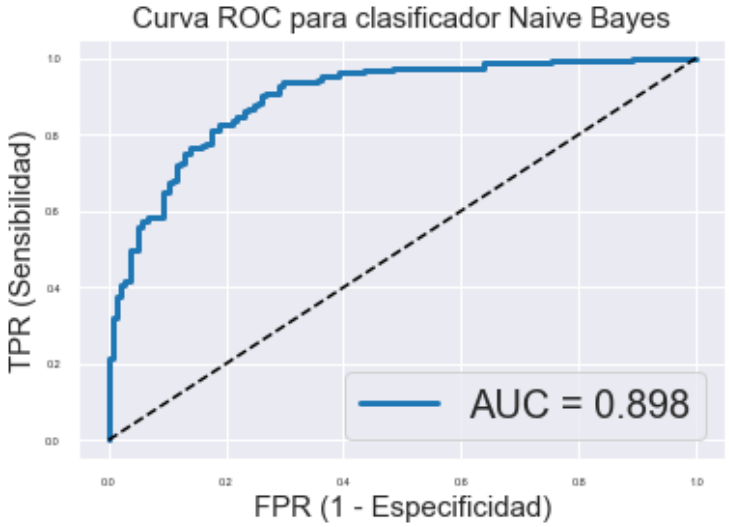

Ilustración 35. Gráfica curva ROC Naive Bayes.

```
createCurveROC(rf_y_test, rf_pred_prob, "Random Forest")
```

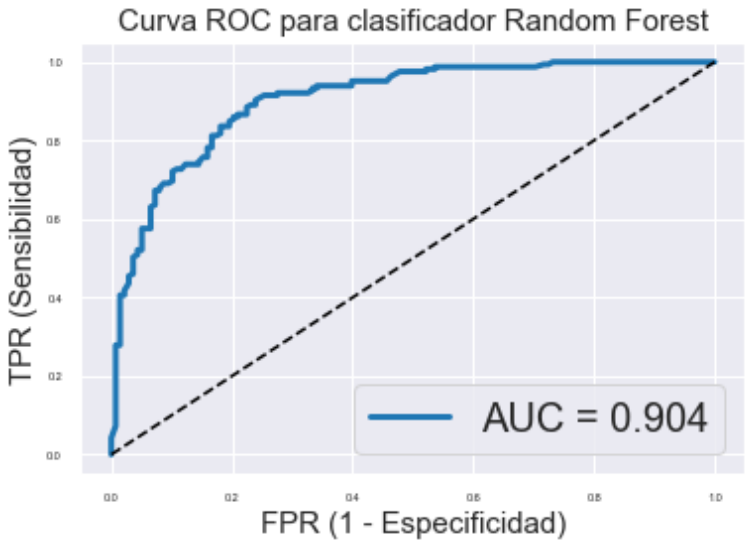

Ilustración 36. Gráfica curva ROC Random Forest.

```
createCurveROC(knn_y_test, knn_pred_prob, "KNN")
```

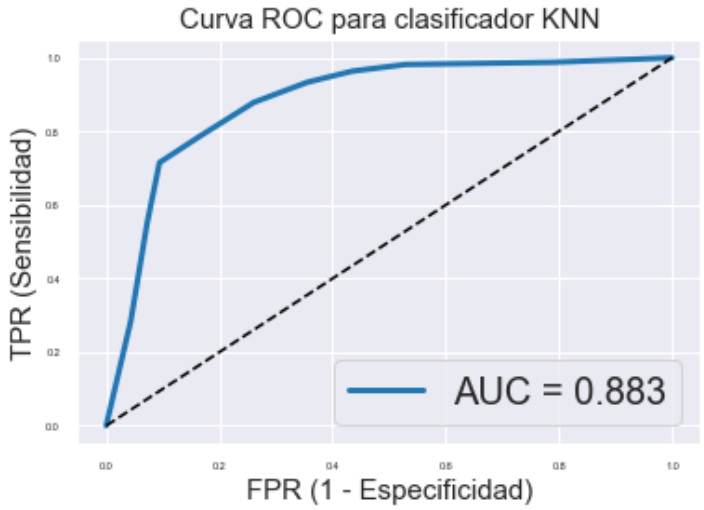

Ilustración 37. Gráfica curva ROC KNN.

Como se explica en el apartado 3.3 Ranking: Curvas ROC y AUC, consiste en una representación de la tasa de verdaderos positivos frente a la tasa de falsos positivos, para cada uno de los umbrales. Cabe recordar que el AUC (*Area Under the Curve*) resulta ser el porcentaje que el modelo ha sido capaz de predecir correctamente. Por lo tanto, cuanto más se aproxime a uno el valor de AUC, mejor será el modelo, mientras que si el AUC es 0.5 o menor, se entiende como un modelo inútil o no válido ya que se interpreta que el modelo está clasificando de una manera completamente aleatoria.

En nuestro caso podemos observar como el AUC resulta ser bastante óptimo, teniendo un 88%-90% de predicciones correctas para cada uno de los modelos. A continuación, se muestra

una guía establecida por el estudio “Curva ROC” donde interpreta el significado del valor del AUC según una serie de intervalos [51]:

- [0.5]: Es como lanzar una moneda (completamente aleatorio).
- [0.5, 0.6): Test malo.
- [0.6, 0.75): Test regular.
- [0.75, 0.9): Test bueno.
- [0.9, 0.97): Test muy bueno.
- [0.97, 1): Test excelente.

Según esta guía podemos clasificar nuestros modelos como buenos, incluso el de Random Forest como muy bueno, ya que su AUC es mayor que 0.9.

Capítulo 6. Conclusiones y futuros trabajos

El principal objetivo ha sido conocer los fundamentos del aprendizaje automático para posteriormente poder aplicar algunas técnicas predictivas. Para ello hemos hecho uso de un conjunto de datos público orientado al área de la salud, para así poder realizar predicciones respecto a si el paciente sufre algún tipo de enfermedad o no. Para a ello hemos hecho un estudio previo de los datos, tanto a nivel teórico como a nivel práctico. A nivel teórico para entender más en detalle cuál es el significado de los datos (como cuáles son los umbrales de la presión arterial y el colesterol, qué significa la talasemia, el segmento ST, etc.). A nivel práctico para conocer si existen anomalías o errores en el mapeo de los datos, donde hemos podido observar gracias a técnicas como los histogramas y las cajas de bigotes, que existían errores en el mapeo de características como *thal* (talasemia) y *cp* (tipo de dolor de pecho). Por otro lado, hemos encontrado valores anómalos como en el caso del colesterol, siendo en estos casos datos determinantes, ya que pueden ser importantes para el diagnóstico de la enfermedad de un paciente. Además, se han implementado diversos clasificadores como la Regresión Logística, Naive Bayes, Random Forest y KNN con el objetivo de analizar y comparar sus predicciones.

Se observa que los mejores resultados se obtienen con la técnica de Random Forest, alcanzando una precisión del 83.5% y un AUC de un 91% aproximadamente. Estos resultados sugieren que es posible predecir con bastante precisión si el paciente sufre algún tipo de enfermedad a partir del conjunto de datos utilizado. Mediante la matriz de confusión hemos podido observar cómo, de 124 pacientes enfermos, ha sido capaz de predecir 106 de forma correcta, indicando que existe un fallo de un 14.5% aproximadamente. Resulta ser bastante preciso ya que, a pesar de disponer de pocos datos (303 registros en este caso), su porcentaje de acierto resulta ser bastante elevado.

Mediante este TFG podemos concluir con que sí hemos cumplido el objetivo, ya que hemos realizado un estudio teórico respecto a los dos tipos de aprendizaje automático que existen (supervisado y no supervisado), los tipos de tareas más comunes (clasificación y regresión), cómo generar correctamente un modelo de predicción mediante la división del conjunto de datos en entrenamiento y prueba, conociendo los distintos modos en los que se pueden dividir. Además, hemos visto la importancia de la preparación de los datos previa a realizar la predicción, para así obtener una información general de lo que vamos a analizar y poder detectar posibles valores anómalos/ incorrectos, algunas técnicas predictivas esenciales, y lo importante que resulta aplicar distintas métricas para analizar correctamente el modelo de predicción.

Además, tras este estudio teórico hemos sido capaces de generar modelos de predicción

mediante algunas de las técnicas estudiadas, como Regresión Logística, Naive Bayes, KNN o Random Forest para, de este modo, poder analizar las salidas de estas con métricas como la matriz de confusión, la precisión o las curvas ROC para poder comparar sus resultados finalmente.

En definitiva, hemos aprendido cuáles son los pasos básicos para realizar un modelo de predicción óptimo, además de conocer los conceptos básicos de este campo de la inteligencia artificial.

A pesar de haber aprendido una gran cantidad de conceptos, queda claro que existen aún muchos por aprender. Tras habernos introducido en el mundo del aprendizaje automático, existe una serie de puntos los cuales me gustaría seguir investigando. Una de ellas es aprender que hay detrás del preprocesamiento y limpieza de los datos, ya que, en nuestro caso, Kaggle nos proporciona los datos limpios, pero en condiciones normales esto no sucede. Por otro lado, en la parte práctica se han aplicado técnicas para problemas de clasificación, donde el objetivo siempre ha sido buscar si el paciente está sano, o por el contrario sufre una enfermedad. Sin embargo, no se ha abordado la aplicación de técnicas para problemas de regresión, por lo que sería una futura línea de trabajo por la que continuar. Finalmente, se ha comentado que existen dos tipos de aprendizaje, supervisado y no supervisado, pero el trabajo práctico se ha centrado principalmente en el aprendizaje supervisado, por lo que aún queda mucho por aprender respecto al aprendizaje no supervisado. Por ello, una buena opción seria estudiar este campo con más profundidad, para poder así aplicar técnicas orientadas al mismo.

Capítulo 7. Bibliografía

[1] Victor Gonzalez Pacheco. (2019, 17 junio). "Una Breve Historia del Machine Learning - Think Big Empresas" https://empresas.blogthinkbig.com/una-breve-historia-del-machine-learning/

[2] "History of Machine Learning. (s.f.)." https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html#top

[3] "Aprendizaje Automático". (2019a, 18 noviembre). https://es.wikipedia.org/wiki/Aprendizaje_automático

[4] Julio Cesar Carpio Ticona. (2017, 10 abril). "Modelo de Predicción de la Morosidad en el otorgamiento de crédito financiero aplicando Metodología CRISP-DM" [PDF] http://repositorio.uancv.edu.pe/handle/UANCV/743

[5] Álex Rayón. (2017, 25 abril). "Guía para comenzar con algoritmos de Machine Learning – Deusto Data" https://blogs.deusto.es/bigdata/guia-para-comenzar-con-algoritmos-de-machine-learning/

[6] Amazon Web Services. (s.f.). "Validación cruzada – Amazon Machine Learning" https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/cross-validation.html

[7] Prashant Gupta. (2017, 5 junio). "Cross-Validation in Machine Learning" https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

[8] Alice Zheng. (2015, 22 octubre). "Evaluating Machine Learning Models" https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/4/offline-evaluation-mechanisms-hold-out-validation-cross-validation-and-bootstrapping

[9] Ricardo Moya. (2017, 2 junio). "Qué es el Machine Learning – Overfitting y Underfitiing" https://jarroba.com/que-es-el-machine-learning/

[10] Microsoft. (2017, 9 noviembre). "Limpieza y preparación de los datos para Azure Machine Learning: Team Data Science Process" https://docs.microsoft.com/es-es/azure/machine-learning/team-data-science-process/prepare-data

[11] RAE-ASALE (2019, 20 noviembre). "Histograma | Diccionario de la lengua española" https://dle.rae.es/histograma?m=30_2

[12] "Diagramas de Caja y Bigotes (s.f.)" https://es.scribd.com/document/305753007/Diagramas-de-Caja-y-Bigotes

[13] "Machine Learning: Selección Métricas de clasificación (2019, 27 octubre)." http://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/#

[14] Colaboradores de Wikipedia. (2019b, 18 noviembre). "Sensibilidad y especificidad (estadística)" https://es.wikipedia.org/wiki/Sensibilidad_y_especificidad_(estadística)

[15] Google. (s.f.). "Clasificación: ROC y AUC | Curso intensivo de aprendizaje automático" https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419

[16] Google Developers (s.f.). "Machine Learning Glossary | Google Developers" https://developers.google.com/machine-learning/glossary

[17] Colaboradores de Wikipedia. (2019c, 18 noviembre). "Regresión lineal" https://es.wikipedia.org/wiki/Regresión_lineal

[18] Ángel Franco García. (s.f.). "Ajuste de datos. Regresión" http://www.sc.ehu.es/sbweb/fisica3/datos/regresion/regresion.html

[19] Google Developers (s.f.). "Machine Learning Glossary | Google Developers" https://developers.google.com/machine-learning/glossary#logistic_regression

[20] Centro de Servicios Informáticos de la Universidad Nacional de Educación a Distancia. (s.f.). "Regresión Logística: Fundamentos y aplicación a la investigación sociológica" [PDF] https://www2.uned.es/socioestadistica/Multivariante/Odd_Ratio_LogitV2.pdf

[21] Colaboradores de Wikipedia. (2019d, 18 noviembre). "Función sigmoide" https://es.wikipedia.org/wiki/Función_sigmoide

[22] "Naive Bayes classifier". (2019a, 16 noviembre). https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[23] José Francisco López. (2018, 21 febrero). "Teorema de Bayes – Definición, qué es y concepto | Economipedia" https://economipedia.com/definiciones/teorema-de-bayes.html

[24] "k-nearest neighbors algorithm". (2019b, 16 noviembre). https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[25] EcuRed. (s.f.). "Distancia euclídea" https://www.ecured.cu/Distancia_eucl%C3%ADdea

[26] Fernando Sancho Caparrini. (s.f.). "Aprendizaje Inductivo: Árboles de Decisión" http://www.cs.us.es/~fsancho/?e=104

[27] "Introducción a los Árboles de Decisión (s.f.)." [PDF] http://www.dmae.upct.es/~mcruiz/Telem06/Teoria/arbol_decision.pdf

[28] Johanna Orellana Alvear. (s.f.) "Árboles de decisión y Random Forest" https://bookdown.org/content/2031/arboles-de-decision-parte-i.html

[29] Universitat Politècnica de València. (s.f.). "Árboles de clasificación | UPV" [Archivo de vídeo]. https://www.youtube.com/watch?v=q63hE-74Gq8

[30] "Random Forest". (2019d, 19 noviembre). https://es.wikipedia.org/wiki/Random_forest

[31] Anurag. (s.f.). "Random Forest Analysis in ML and when to use it" https://www.newgenapps.com/blog/random-forest-analysis-in-ml-and-when-to-use-it

[32] Tavish Srivastava. (2019, 6 septiembre). "Classification Algorithm Support Vector Machine" https://www.analyticsvidhya.com/blog/2014/10/support-vector-machine-simplified/?utm_source=blog&utm_medium=understandingsupportvectormachinearticle

[33] foto svm ideal http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex7/ex7.html

[34] Kaggle Inc. (s.f.). "Kaggle: Your Home for Data Science" https://www.kaggle.com

[35] Kaggle Inc. (2018, 25 junio). "Heart Disease UCI" https://www.kaggle.com/ronitf/heart-disease-uci/

[36] Anaconda. (s.f.). "Anaconda Python/ R Distribution – Free Download" https://www.anaconda.com/distribution/

[37] Secretaría de la Salud. (s.f.). "¿Qué es la angina de pecho?" https://www.gob.mx/salud/articulos/que-es-la-angina-de-pecho

[38] Biblioteca Nacional de EE. UU. (s.f.) "Angina de pecho: MedlinePlus" https://medlineplus.gov/spanish/angina.html

[39] Dr. Hugo Parrales. (2018, 16 diciembre). "La valoración y análisis del Segmento ST" https://cerebromedico.com/electrocardiograma/segmento-st/

[40] SEIC. (2018, 21 febrero). "¿Cómo es el ECG en la angina de pecho?" https://ecocardio.com/documentos/biblioteca-preguntas-basicas/preguntas-al-cardiologo/1062-ecg-en-angina-pecho.html

[41] Mayo Clinic. (2018, 20 noviembre). "Hipertrofia ventricular izquierda – Síntomas y causas" https://www.mayoclinic.org/es-es/diseases-conditions/left-ventricular-hypertrophy/symptoms-causes/syc-20374314

[42] Mayo Clinic. (2018, 10 noviembre). "Talasemia – Síntomas y causas" https://www.mayoclinic.org/es-es/diseases-conditions/thalassemia/symptoms-causes/syc-20354995

[43] Máquina de Soporte Vectorial SVM. (2019, 20 octubre). http://numerentur.org/svm/

[44] sklearn.linear_model.LogisticRegression – scikit-learn 0.21.3 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[45] sklearn.naive_bayes. GaussianNB – scikit-learn 0.21.3 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[46] sklearn.ensemble.RandomForestClassifier – scikit-learn 3.2.4.3.1 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[47] sklearn.neighbors.KNeighborsClassifier – scikit-learn 0.21.3 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[48] sklearn.metrics.confusion_matrix – scikit-learn 0.21.3 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[49] sklearn.metrics.roc_curve – scikit-learn 0.21.3 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

[50] sklearn.metrics.auc – scikit-learn 0.21.3 documentation. (s.f.). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

[51] "Curva ROC – Wikipedia, la enciclopedia libre" (2019e) https://es.wikipedia.org/wiki/Curva_ROC