UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de soluciones software mediante Aprendizaje Automático en el ámbito de la Salud: situación tecnológica y perspectivas

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIONES

Autor: Carlos Sánchez Gómez

Director: Pedro Sánchez Palma

Codirector: Jesús Martín Jiménez

Cartagena, Noviembre 2019

Agradecimientos

Al Dr. Luciano Consuegra Sánchez, por facilitar el acceso a la base de datos del Estudio SIESTA sin la cual la última parte de este trabajo no habría sido posible.

A Pedro y Jesús, por su confianza y ayuda que han hecho posible este trabajo.

A Lourdes, por su cariño y apoyo.

A Lucas, por la promesa de futuro.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Introducción

Objetivos

Las técnicas de Aprendizaje Automático están ganando terreno en los últimos años como medio para dotar a los especialistas sanitarios de herramientas informáticas que les permita procesar los datos de información sanitaria de tal forma que puedan realizar una serie de acciones relacionadas con la atención y gestión de la salud favoreciendo el diagnóstico precoz, mejorar la medicina preventiva, permitir una atención médica más personalizada, además de un ahorro en costes y aumento de la productividad de los profesionales sanitarios. El objetivo de este trabajo fin de grado es:

1. Realizar una síntesis de las técnicas y herramientas asociadas a Aprendizaje Automático.
2. Realizar un análisis detallado de la situación actual de Aprendizaje Automático en el ámbito de la salud, con posible concreción tanto geográfica (España) como de especialidad (Cardiología).
3. Identificar los principales retos por superar para una adecuada expansión de dichas técnicas.
4. Desarrollar un modelo de Aprendizaje Automático sobre una base de datos reales de interés para la Cardiología.

Estructura

Este trabajo está organizado en cinco capítulos. El primero da una introducción al Aprendizaje Automático y al Aprendizaje Profundo, con conceptos básicos que servirán como base sobre la que continuar indagando en los conceptos de Aprendizaje Automático, su desarrollo y su aplicación. Se explican las técnicas básicas de Aprendizaje Automático y los principales algoritmos que se utilizan. Se establecerán las pautas para el uso de ellos a nivel teórico.

El segundo capítulo se centra en el soporte software para Aprendizaje Automático, tanto el usado para el desarrollo del trabajo y el aprendizaje previo como un repaso a las principales herramientas de distintas empresas.

El tercer capítulo desarrolla la aplicación del Aprendizaje Automático a la salud, el conocido como eSalud, y el papel que el Aprendizaje Automático juega en él. Muestra la situación actual de la aplicación de técnicas de Aprendizaje Automático en España orientadas a la salud y particularizando en la rama de la cardiología.

El cuarto capítulo muestra una propuesta de desarrollo de una aplicación de Aprendizaje Automático en la rama de la Cardiología, consistente en reanalizar los datos de un trabajo de investigación sobre valor predictivo de eventos al año de los biomarcadores de inflamación en el síndrome coronario agudo sin elevación del segmento ST.

Para finalizar, el quinto y último capítulo habla de las conclusiones tras la realización del trabajo fin de grado y de posibles líneas de trabajo futuras para seguir desarrollando a partir de lo conseguido en el capítulo cuarto.

Capítulo 1

Introducción a Aprendizaje Automático y técnicas básicas

Este capítulo es una introducción al Aprendizaje Automático con conceptos básicos que servirán de base sobre la que continuar indagando en su desarrollo y aplicación. Se expondrán sus características, los fundamentos para su funcionamiento y las distintas categorías de aprendizaje. Por último se mostrarán algunos de los algoritmos fundamentales.

1.1 Inteligencia Artificial

La Inteligencia Artificial (IA) es el campo de la ciencia informática dedicado a la resolución de problemas cognitivos asociados comúnmente con la inteligencia humana, como el aprendizaje, la resolución de problemas y el reconocimiento de patrones. La Inteligencia Artificial, que normalmente se abrevia IA, puede evocar escenas futuristas o robóticas, pero va mucho más allá de los robots de la ciencia ficción, llegando a la ciencia informática avanzada de hoy en día. El profesor Pedro Domingos, investigador destacado en este campo, describe "cinco tribus" del aprendizaje virtual, compuestas por: simbolistas, provenientes de la lógica y la filosofía; conexionistas, procedentes de la neurociencia; evolutivos; relacionados con la biología evolutiva; bayesianos, interesados en la estadística y la probabilidad; y analogistas, procedentes de la psicología [1]. Recientemente, los avances en la eficacia de la computación estadística han permitido a los bayesianos ampliar su campo en varias áreas, englobadas bajo el nombre "Aprendizaje Automático". Del mismo modo, los avances en la computación de red han llevado a los conexionistas a crear un subcampo denominado "Aprendizaje Profundo". Aprendizaje Automático y Aprendizaje Profundo son campos de la ciencia informática derivados de la disciplina de la Inteligencia Artificial.

1.2 Aprendizaje Automático

El Aprendizaje Automático es un campo de la informática, englobado en la Inteligencia Artificial, que se encarga de construir algoritmos cuya utilidad se basa en una serie de datos ejemplo de un fenómeno. Estos ejemplos pueden venir de la naturaleza, creados por una persona o generados por otro algoritmo.

Otras definiciones sobre el Aprendizaje Automático [2]:

- Arthur Samuel lo describió como *"El campo de estudio que da a los ordenadores la capacidad de aprender sin estar explícitamente programados."* Aunque esta definición es demasiado informal y está un poco anticuada.
- Tom Mitchell da una definición más moderna y acertada: *"El Aprendizaje Automático es un programa informático que aprende de una experiencia E relacionada con una actividad T y con una medida de desempeño P, de manera que su desempeño en T medido por P mejora con la experiencia E."*

Como ejemplo sería un programa de Aprendizaje Automático de jugar al ajedrez donde:

- E: la experiencia de muchas partidas de ajedrez
- T: La tarea de jugar al ajedrez
- P: La probabilidad de que el programa gane la siguiente partida de ajedrez.

El Aprendizaje Automático se puede separar en dos tipos, atendiendo al tipo de ejemplo que se le suministre: Aprendizaje Supervisado y Aprendizaje no Supervisado.

1.2.1 Terminología

Parámetros e hiperpárametros

Los **parámetros** son variables que define el modelo creado mediante un algoritmo de aprendizaje. El propio parámetro los modifica en función de los datos de entrenamiento. El objetivo del entrenamiento es encontrar los valores de los parámetros que hacen que nuestro modelo sea óptimo.

Por el contrario, los **hiperparámetros**, que suelen tomar un valor numérico, son una propiedad de los algoritmos de aprendizaje. Los hiperparámetros tienen que ser ajustados por el analista de datos antes de ejecutar el algoritmo y su valor influye en la manera en la que funciona el algoritmo.

Clasificación y regresión.

Se denomina **Clasificación** al problema de asignar automáticamente una etiqueta a un ejemplo que carece de ella, perteneciendo esta etiqueta o clase a un conjunto finito. En Aprendizaje Automático, los problemas de clasificación los resuelven algoritmos de

clasificación que toman los ejemplos etiquetados del conjunto de entrenamiento como entrada y producen un modelo que puede asignar etiquetas a datos carentes de ella.

En los problemas de clasificación se considera **clasificación binaria** en el caso de que solo haya dos posibles etiquetas de salida ("Si" o "No", "Enfermo" o "Sano") y **clasificación multiclase** en el caso de 3 o más etiquetas.

Se llama **Regresión** al problema de predecir un valor real. Es similar a la asignación de una etiqueta, pero en este caso la predicción no es una opción entre un conjunto finito sino un número real. En este caso los algoritmos de aprendizaje crean una función en muchos casos continua y es la que utilizan para generar la salida ante una entrada nueva.

Aprendizaje Profundo.

La mayoría de algoritmos de aprendizaje fijan los parámetros del modelo directamente de las variables del conjunto de datos de entrenamiento. La excepción más notable son las **redes neuronales**, específicamente las que tienen más de una capa entre la entrada y la salida. En esos casos los parámetros del modelo no se aprenden directamente del conjunto de datos de ejemplo sino de las salidas de las capas anteriores. Es el denominado Aprendizaje Profundo.

1.2.2 Aprendizaje Supervisado

Es el tipo de Aprendizaje Automático más utilizado en la práctica. En el aprendizaje supervisado los ejemplos están etiquetados, de manera que se conoce la respuesta correcta para una entrada concreta.

Los datos de entrenamiento están formados por pares (normalmente vectores): un componente del par será los parámetros de entrada ($\vec{x}$) y el otro componente serán los resultados deseados ($\vec{y}$).

Cada elemento del vector x contiene los parámetros que definen los ejemplos en los datos de entrenamiento. El vector y contiene los datos de salida de los ejemplos en los datos de entrenamiento.

El objetivo de los algoritmos de aprendizaje supervisado es usar los datos de entrenamiento para crear un modelo que tome un vector de parámetros de entrada y deduzca un valor para los datos de salida.

1.2.2.1 ¿Cómo funciona el aprendizaje supervisado?

El proceso empieza con la recopilación de datos de entrenamiento. En el aprendizaje supervisado los datos son colecciones de parejas (entrada, salida). Los datos de entrada pueden ser muy variados, desde correos de ejemplo en el caso de un programa que clasifique si un correo es correo basura o no, datos financieros para programas de predicción de precios, distintos datos médicos para programas que buscan enfermedades, datos de sensores, fotografías… Los datos de salida pueden ser desde números reales, etiquetas de categoría (“correo basura”, “perro”, “enfermo”, etc.). Todo esto dependerá de lo que se pretenda conseguir con nuestro algoritmo y de los datos de partida.

El analista de datos decide, basado en su experiencia, como convertir una entidad real como puede ser una imagen o un correo en un vector de parámetros para el algoritmo. Puede ser pasando una imagen a su estado en pixeles, transformando el correo en una serie de números u otras opciones.

Una vez dispuestos los datos de ejemplo hay que elegir un algoritmo de aprendizaje al que aplicar dichos datos. Existen multitud de algoritmos ya establecidos además de la posibilidad de crear un algoritmo nuevo. Estos algoritmos se desarrollan posteriormente con mayor detalle.

Con los datos de ejemplo y una vez decidido el algoritmo se puede aplicar este para obtener el modelo. Para ello los algoritmos de aprendizaje suelen encontrar los mejores valores que resuelvan el problema de optimización planteado al incluir los parámetros de ejemplo.

Aprendizaje, por tanto, es el proceso por el que se obtiene el **modelo** a partir de los **datos de ejemplo**.

1.2.2.2 Algoritmos de aprendizaje

Todos los algoritmos de aprendizaje, de los que luego se trataran los ejemplos más comunes, están formados por tres partes [3]:

1. Una función de pérdidas.

2. Un criterio de optimización basado en la función de pérdidas (la función de coste suele ser el más utilizado).
3. Una rutina de optimización, que encuentra una solución al criterio de optimización.

Estos son los bloques de diseño de cualquier algoritmo de aprendizaje. Existen algoritmos que se diseñan específicamente para un criterio de optimización concreto como los algoritmos de **regresión lineal** y **logística** y **Máquinas de Soporte Vectorial (SVM, Support Vector Machines)**, mientras que otros lo hacen implícitamente, como **k-Vecinos más cercanos (kNN, k-Nearest Neighbors)** o los **Árboles de decisión**. El caso de los Árboles de decisión es curioso puesto que se puede considerar el algoritmo de aprendizaje más antiguo que se creó experimentalmente basándose en la intuición y el criterio de optimización se desarrollo después para explicar cómo funcionaba el algoritmo.

Gradiente de descenso.

El **gradiente de descenso** es el algoritmo de optimización más utilizado en los casos en los que el criterio de optimización es diferenciable [3]. Es un algoritmo iterativo de optimización que sirve para encontrar el mínimo de una función. Se inicia en un punto aleatorio y va dando pasos en cada iteración hacia el negativo del gradiente hasta encontrar un mínimo. Es utilizado para la optimización de los algoritmos de regresión lineal y lógica, en SVM y en redes neuronales entre otros. En el caso de la regresión logística y SVM, al tratarse de modelos que producen funciones convexas, solo tienen un mínimo, por lo tanto el mínimo obtenido por el gradiente de descenso será global.

¿Cómo funciona el gradiente de descenso?

Para ilustrarlo, se propone un set de entrenamiento con variable única. Sin embargo el criterio de optimización tendrá dos parámetros: w y b.

Con esto podemos establecer el modelo de Regresión lineal [3]:

$$f(x) = \mathrm{w}x + b$$

que será el modelo sobre el que se trabaje en el ejemplo. Se propone el ejemplo de un modelo predictivo de las unidades vendidas de un producto según la cantidad de

dinero que se invierte en publicidad. Para ello se dispone del set de entrenamiento, formado por distintas compañías para el mismo producto y pares "Dinero invertido" y "Unidades vendidas".

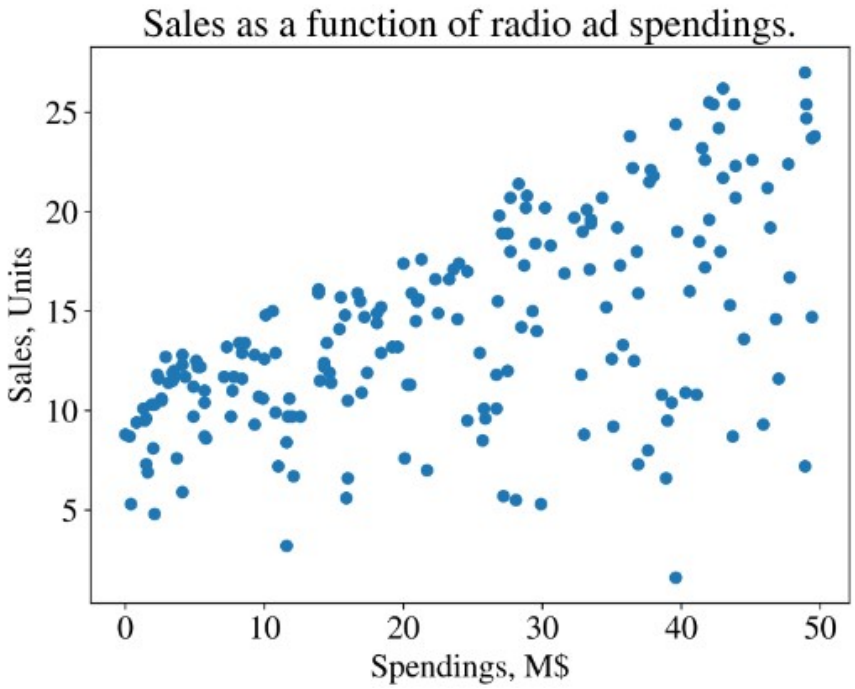

Figura 1 Datos de entrenamiento para un modelo en representación grafica. En el eje Y, unidades vendidas. Eje X, gastos en anuncios de radio. Extraída de [3]

Company	Spendings, M$	Sales, Units
1	37.8	22.1
2	39.3	10.4
3	45.9	9.3
4	41.3	18.5
..	..	..

Figura 2 Datos de entrenamiento representados como una tabla. Extraída de [3]

Con el modelo que se desea aplicar y los datos del set de entrenamiento se deben encontrar los valores óptimos para w y b, aprendidos de los datos de los que se dispone. Para ello se buscan los valores que minimicen el error cuadrático medio:

$$l \overset{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b)^2$$

Esta será la función de pérdidas, que arroja la diferencia entre el valor asignado por el modelo y el valor real. Se buscan los valores de los parámetros w y b que minimicen la función de perdidas. Ese será el criterio de optimización.

El gradiente de descenso comienza con el cálculo de las derivadas parciales para cada parámetro:

$$\frac{dl}{dw} = \frac{1}{N}\sum_{i=1}^{N} -2x_i\big(y_i - (wx_i + b)\big);$$

$$\frac{dl}{db} = \frac{1}{N}\sum_{i=1}^{N} -2\big(y_i - (wx_i + b)\big);$$

La estrategia de uso del Gradiente de descenso funciona por pasos. En cada paso se actualiza el valor de los parámetros w y b usando las derivadas parciales, así como el hiperparámetro α que controla el paso siguiente dentro del gradiente.

La actualización de los parámetros será:

$$w \leftarrow w - \alpha\frac{dl}{dw}$$

$$b \leftarrow b - \alpha\frac{dl}{db}$$

Repitiendo hasta la convergencia. Son precisas bastantes iteraciones hasta ver que b y w apenas cambian, en cuyo caso se habrá alcanzado un mínimo tal y como muestra la figura 3:

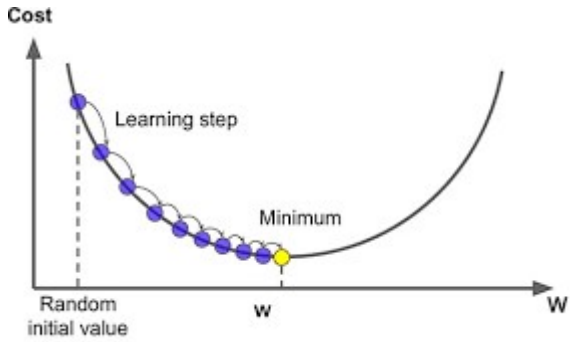

Figura 3 Gradiente de descenso y representación de las iteraciones [2]

La elección del hiperparámetro α es importante en el gradiente de descenso, pues la elección de un buen tamaño puede influir en la rapidez con la que este converja o incluso impedirlo, como se observar en la figura 4:

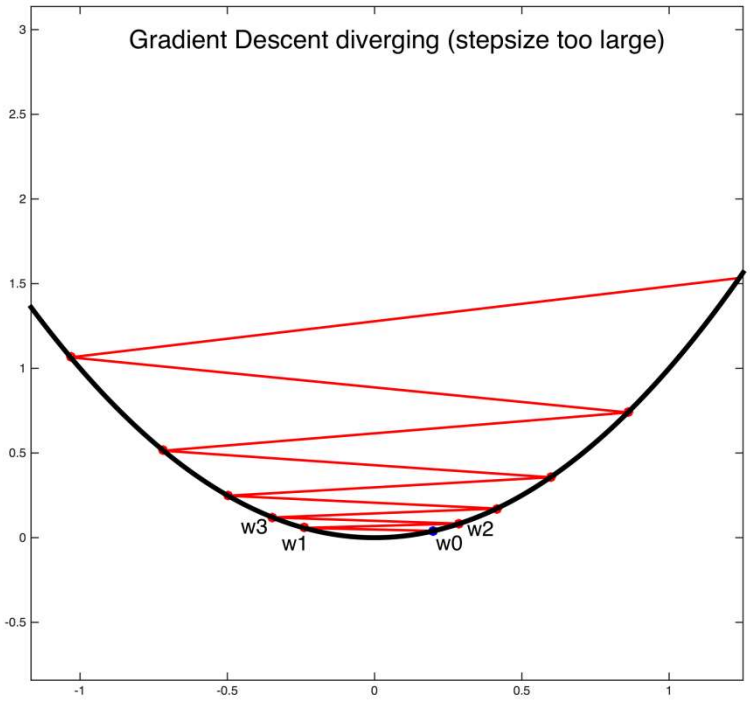

Figura 4 Influencia del valor de α en el gradiente de descenso [4]

1.2.2.3 Algoritmos de aprendizaje fundamentales

Regresión Lineal

Regresión lineal es, como su propio nombre indica, un algoritmo de regresión cuyo modelo es una combinación lineal de las variables de entrada.

Como ya se adelanto en la explicación del gradiente de descenso, considerando una colección de ejemplos $\{x_i, y_i\}_{i=1}^{N}$, siendo N el numero de ejemplos, x es un vector de variables de dimensión D que contiene todos las variables de entrada del set de entrenamiento e y_i es un numero real para cada ejemplo y cada x_i es también un numero real, el modelo será el siguiente [3]:

$$f_{w,b}(x) = \mathrm{w}x + b$$

Donde w es un vector de dimensión D y b es un número real.

Este modelo se usa para predecir una salida y ante cada nueva entrada x se cumple que $y \leftarrow f_{w,b}(x)$. Dos modelos parametrizados con w y b diferentes darán dos resultados distintos para la misma entrada, por se pretenden encontrar los valores óptimos de w y b que ofrezcan la predicción más precisa (figura 5).

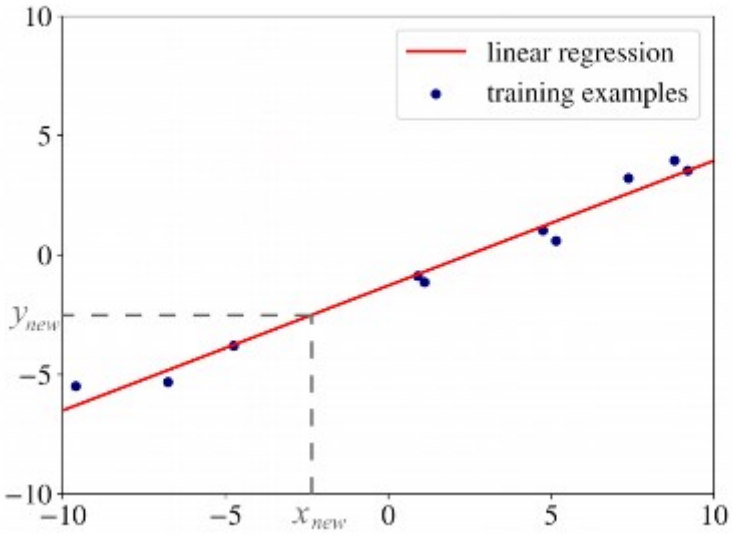

Figura 5 Regresión Lineal para ejemplos unidimensionales extraído [3]

Para encontrar los valores óptimos de los parámetros w y b hay que minimizar la siguiente expresión:

$$\frac{1}{N}\sum_{i=1\ldots N}(f_{w,b}(x_i)-y_i)^2$$

Esta expresión es la función objetivo, también en esta forma se denomina Función de Pérdidas, que da una medida del error cometido en una predicción. En este caso la función de pérdidas es la del error cuadrático medio.

Regresión Logística

A pesar de su nombre, el algoritmo de Regresión Logística es un algoritmo de clasificación y no de regresión. El nombre viene de su similitud con la regresión lineal.

En Regresión Logística el modelo sigue calculando y_i como una función lineal de x_i. Si se usa la misma ecuación que en regresión lineal se obtienen salidas que van de menos infinito a infinito, mientras que y_i solo tiene dos posibles valores.

La forma más fácil de actuar es definir uno de los valores de salida como 0 (salida negativa) y el otro como 1 (salida positiva) y utilizar un función con codominio (0,1). Si la función devuelve un valor próximo a 0 se asigna la etiqueta negativa y si devuelve uno próximo a 1, la etiqueta positiva.

La función con esta propiedad que se suele utilizar es la **Función Sigmoidea**, que definida para el modelo queda [2]:

$$f_{w,b}(x) \stackrel{\text{def}}{=} \frac{1}{1+e^{-(wx+b)}}$$

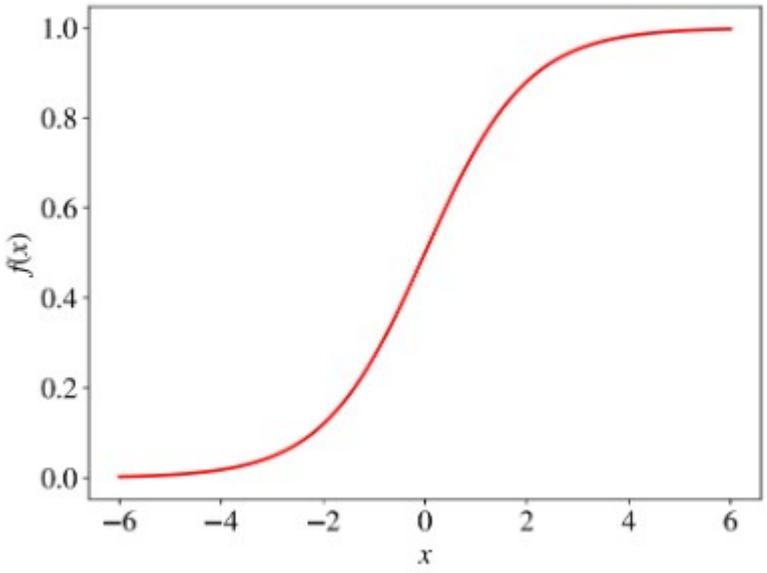

Figura 6 Función Sigmoidea

Para encontrar los valores óptimos de w y b el criterio de optimización al que se recurre será la **función de similaridad** que, dado el carácter de la función sigmoidea, se puede transformar con logaritmos y encontrar el máximo de similaridad según:

$$L_{w,b} \stackrel{\text{def}}{=} \prod_{i=1,\dots} f_{w,b}(x)^{y_i}(1-f_{w,b}(x_i))^{(1-y_i)}$$

$$LogL_{w,b} \stackrel{\text{def}}{=} \ln(L_{w,b}(x)) = \sum_{i=1}^{N}\left[y_i \ln f_{w,b}(x) + (1-y_i)\ln(1-f_{w,b}(x)\right]$$

Como método de optimización se puede recurrir nuevamente al gradiente de descenso.

Maquina de Soporte Vectorial

El algoritmo de Máquina de Soporte Vectorial es un algoritmo de clasificación que establece un hiperplano que separa los posibles resultados de la clasificación y_i (Ver figura 7).

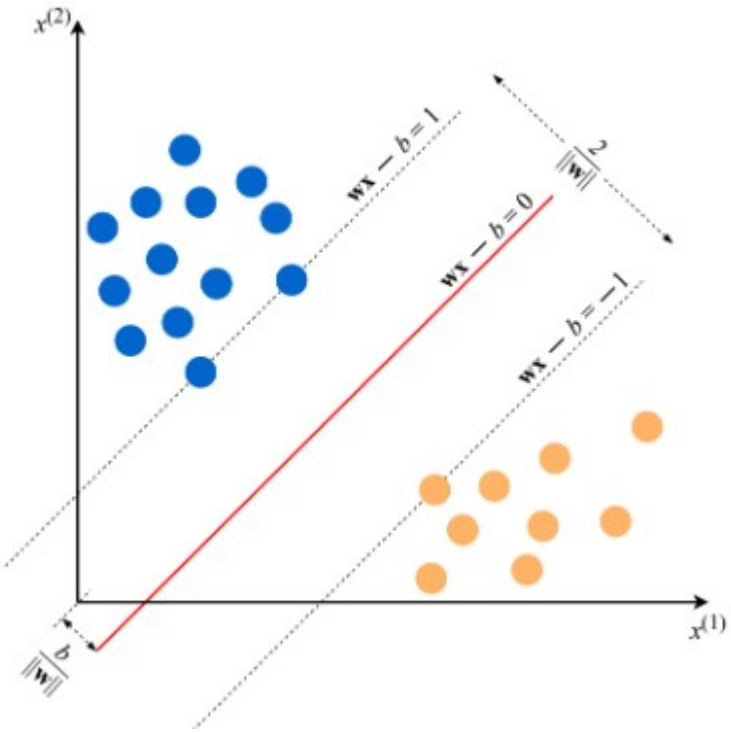

Figura 7 Representación de la frontera establecida por SVM en un modelo de dos dimensiones [3]

Para ello deben cumplirse las siguientes condiciones:

$$wx_i - b \geq +1 \; si \; y_i = +1$$

$$wx_i - b \leq +1 \; si \; y_i = -1$$

Es decir, establece un hiperplano que separa los ejemplos positivos de los negativos con el margen más grande, siendo el margen la distancia entre los dos ejemplos más cercanos de las dos clases. Cuanto mayor sea el margen, mejor será la clasificación que realice el modelo.

El objetivo de optimización será la norma euclidea $\|w\|$ Con esto el problema de optimización quedará asi:

$$min \frac{1}{2}\|w\| \text{ de manera que } y_i(x_i w - b) - 1 \geq 0, i = 1, \dots, N.$$

Es común también introducir el hiperparámetro C que penaliza la clasificación errónea.

En tal caso la función a minimizar pasaría a ser:

$$C\|w\|^2 + \frac{1}{N}\sum_{i=1}^{N} \max\,(0,1 - y_i(wx_i - b))$$

SVM permite también la adaptación para datos que no se pueden separar inicialmente, es decir, que tienen no linealidad inherente. Para ello transforma el espacio en uno de dimensión superior donde sí sea linealmente separable. Para ello hace uso de núcleos Gaussianos, que trabajan en espacios dimensionales superiores sin necesidad de realizar la transformación explícitamente (figura 8)

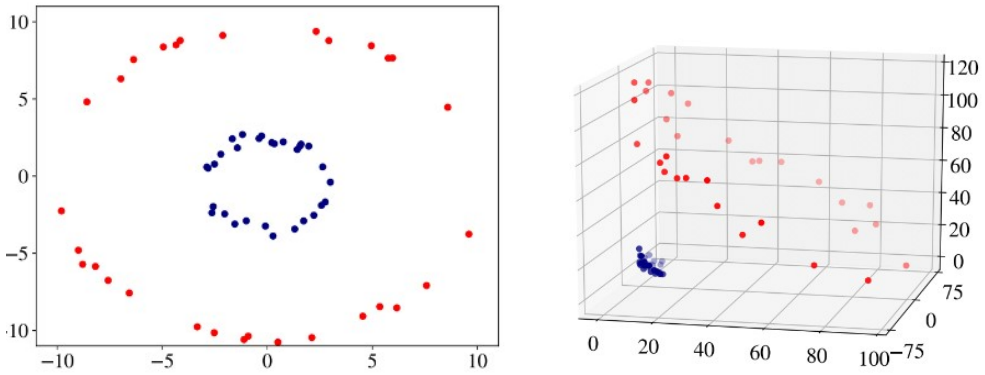

Figura 8 Los datos de la figura de la izquierda se convierten en linealmente separables al aumentar la dimensión en la figura de la derecha [3].

K-Vecinos más cercanos

K-Vecinos más cercanos es un algoritmo de aprendizaje no paramétrico, porque no hace suposiciones previas respecto a los datos. Al contrario que otros algoritmos que permiten deshacerse de los datos de entrenamiento una vez creado el modelo, kNN mantiene todos los ejemplos en memoria. Cada vez que recibe un ejemplo x_i que no ha visto antes, kNN encuentra los k ejemplos más cercanos a éste y le asigna la etiqueta mayoritaria en el caso de clasificación o la media en el caso de regresión.

La cercanía entre dos ejemplos nos da la función de distancia, que puede ser la distancia Euclidea o en otros casos se puede utilizar la similitud coseno:

Distancia Euclidea: $d(x_i, x_k) = \sqrt{\sum_{j=1}^{D}(x_i{}^{(j)} - x_k{}^{(j)})^2}$

Similitud coseno: $d(x_i, x_k) = \cos(\angle(x_i, x_k)) = \frac{\sum_{j=1}^{D} x_i{}^{(j)} x_k{}^{(j)}}{\sqrt{\sum_{j=1}^{D}(x_i{}^{(j)})^2}\sqrt{\sum_{j=1}^{D}(x_k{}^{(j)})^2}}$

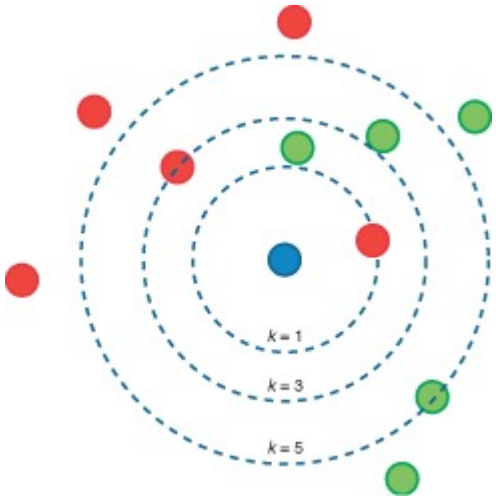

Figura 9 Ejemplo de kNN, para distintos valores de k extraído de [3]

La elección de la métrica de la distancia así como el valor para k son las elecciones a realizar antes de aplicar el algoritmo tal y como ilustra la figura 9, por tanto pueden considerarse hiperparámetros.

Redes Neuronales

Las redes neuronales son un algoritmo de clasificación. Es más, podemos considerar que el algoritmo de regresión logística es la unidad estándar de una red neuronal.

Una red neuronal es una función matemática:

$$y = f_{NN}(x)$$

con la particularidad de tratarse de una función que es composición de otras funciones. Las redes neuronales se construyen en capas, de manera que una red neuronal de 3 capas que devuelve un escalar tendrá la siguiente forma:

$$y = f_{NN}(x) = f_3(f_2(f_3(x)))$$

Las funciones que aplican cada neurona, entendiendo ésta como las unidades mínimas de la red neuronal, como ilustra la figura 10, reciben el nombre de **función de activación**. Suelen ser funciones no lineales elegidas por el analista de datos.

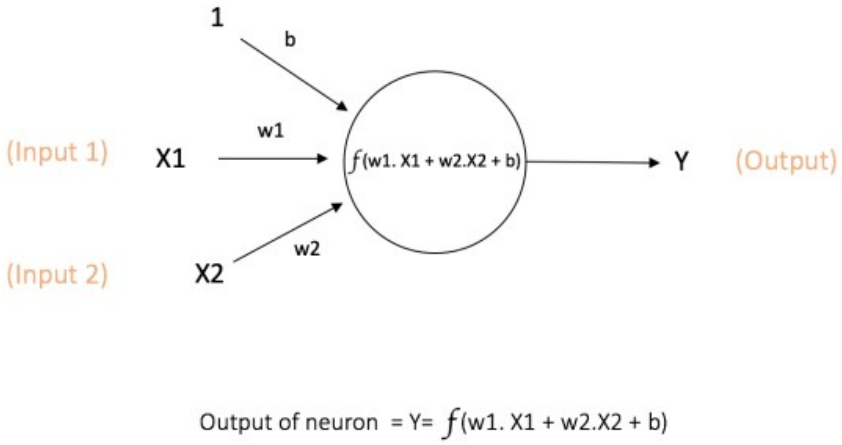

Figura 10 Una neurona, la unidad básica de las redes neuronales, extraído de [4]

En una red neuronal se parte de la primera capa, conocida comúnmente como capa de entrada, cuyos nodos de entrada pasan al siguiente nodo de la capa 2, y así sucesivamente según la arquitectura de la red hasta alcanzar la ultima capa que será la capa de salida. Un ejemplo de esta arquitectura se muestra en la figura 11.

Cada una de las capas intermedias entre la de entrada y la de salida se conocen como capas ocultas. Cuantas más capas ocultas, mayor será la complejidad de la red neuronal. En ese caso se entra en el campo del Aprendizaje Profundo, que suele referirse de manera general como entrenamiento de redes neuronales con más de dos capas ocultas.

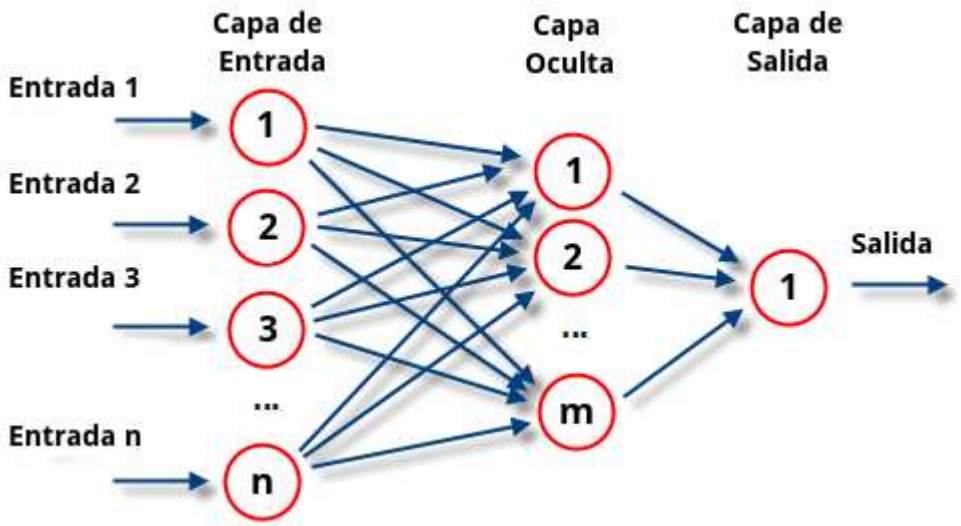

Figura 11 Representación de la arquitectura de una red neuronal [4]

Los hiperparámetros de cada una de las capas de una red neuronal suelen llamarse "pesos" de la capa.

La salida de cada unidad o neurona es el resultado de varias acciones. Primero, todas las entradas de la neurona se juntan formando un vector de entrada. La neurona aplica una transformación lineal al vector de entrada, igual que hace la regresión lineal con el vector de entrada. Por último, la neurona aplica la función de activación al resultado de la transformación lineal que dará como resultado la salida de la neurona, que a su vez será la entrada de la neurona siguiente.

Para encontrar los valores óptimos de los hiperparámetros de una red neuronal se utiliza el algoritmo de propagación hacia atrás de errores o **retropropagación**. Es un algoritmo eficiente para el cálculo de gradientes en redes neuronales. Recorre la red desde el final hasta el principio realizando actualizaciones en los pesos de las capas en función de su influencia en la capa posterior.

1.2.3 Aprendizaje no Supervisado.

En el Aprendizaje no Supervisado los datos de ejemplo no están formados por pares "entrada, salida", sino que están formados solo por el conjunto de parámetros de entrada de ejemplo.

El objetivo de un algoritmo de Aprendizaje no Supervisado es crear un modelo que coja los parámetros del vector X como entrada y los transforme en otro vector o valor de salida que pueda ser usado para resolver un problema. Por ejemplo, en los algoritmos de agrupamiento el algoritmo se encarga de establecer relaciones entre los datos de entrada y dar un valor de agrupamiento a cada uno de ellos. En los algoritmos de compresión de datos la salida es un vector de parámetros que contiene menos parámetros que la entrada. En los algoritmos de detección de anomalías, el algoritmo detecta que entradas se pueden considerar una anomalía y se escapan de lo usual.

1.2.3.1 Algoritmos de Aprendizaje no Supervisado.

K-Medias

El algoritmo k-Medias es un algoritmo no supervisado de agrupamiento, es decir, asigna etiquetas de clasificación a datos nuevos según la agrupación con respecto a los datos del set de entrenamiento, que tampoco tiene etiquetas. Funciona de la siguiente manera:

Primero, se elige el hiperparámetro k que determina el número de agrupaciones en las que dividiremos nuestros datos. Inicializamos k vectores denominados “centroides” de manera aleatoria en el espacio de los datos.

Tras esto, calcula la distancia de cada ejemplo x_i a cada centroide usando la medida considerada conveniente. Se asigna el centroide más cercano a cada ejemplo. Por último, se calcula el punto medio entre todos los datos asignados a un centroide y se asigna esta posición como la nueva localización del centroide. Se repite esta secuencia hasta que no se produzca variación entre las posiciones de los centroides entre una iteración y la siguiente como se puede ver en la figura 12.

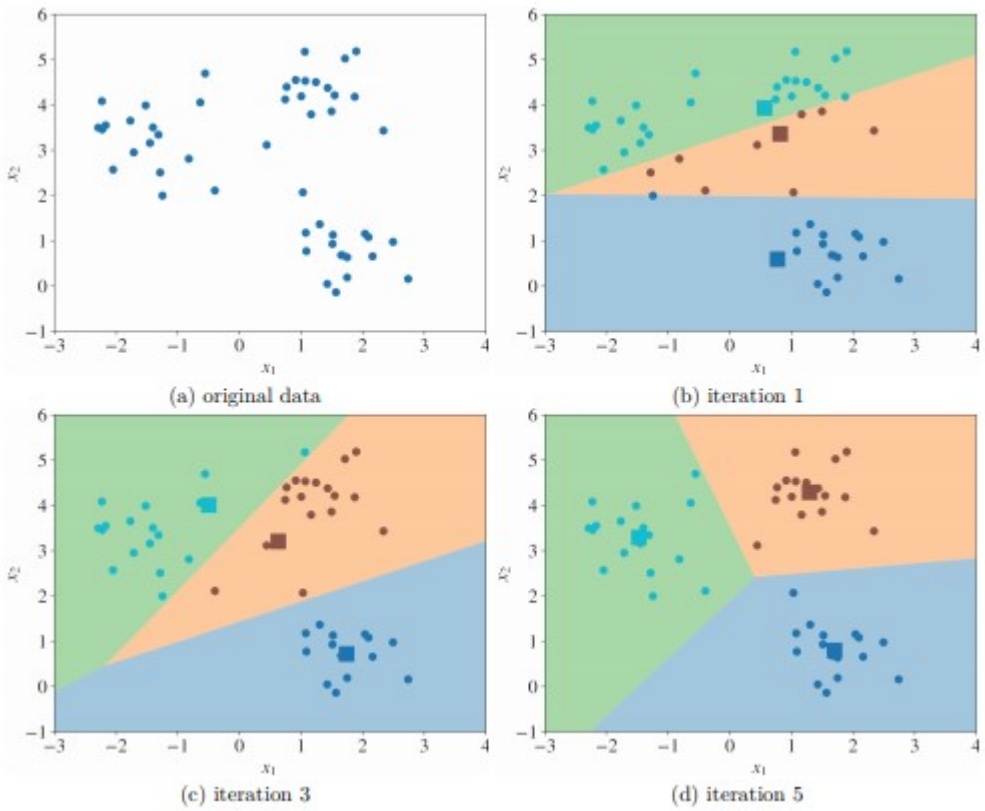

Figura 12 Progresión del Algoritmo K-Medias para k=3 con distintas iteraciones [3]

La posición inicial de los centroides influye en su posición final, por tanto el algoritmo puede dar posiciones finales diferentes para posiciones iniciales distintas. También influye el valor de k, el número de centroides, que es el hiperparámetro que se debe elegir. En ambos casos mejor es "observar" los datos iniciales para una mejor elección de los hiperparámetros.

Algoritmo de Detección de Anomalías

El Algoritmo de Detección de Anomalías es un algoritmo de Aprendizaje no Supervisado basado en el modelado de la función de densidad de probabilidad (pdf). Se puede utilizar para la detección de errores o de datos que se salgan de unos parámetros que se hayan determinado como "normales" para cierta actividad sobre la se aplicará el modelo.

El funcionamiento es bastante sencillo. La principal asunción es que los datos de entrenamiento siguen una distribución normal o gaussiana con media μ y desviación σ^2. Se usan estos datos sobre los parámetros que pueden llevar a detectar casos anómalos y con ello se crea la función de densidad de probabilidad. Se elige un valor de dicha

función de probabilidad de manera que cualquier valor que si $p(x) < \epsilon$ se considera un valor anómalo y marcaría una detección de error.

Este algoritmo se puede usar, por ejemplo, para la detección de anomalías en la fabricación industrializada de manera que entrenando un modelo con ciertas medidas sea capaz de marcar piezas mal fabricadas.

Análisis de Componentes Principales

El algoritmo de Análisis de Componentes Principales (PCA, Principal Component Analysis) es un método de reducción de dimensión.

Su funcionamiento se basa en el uso de componentes principales, que son vectores que definen un nuevo sistema de coordenadas en el que el primer eje va en la dirección de la mayor varianza de los datos. El segundo eje es ortogonal a este (en el caso de haberlo). El paso siguiente sería proyectar los datos sobre los nuevos ejes.

De esta manera ahora solo se precisa una coordenada para describir cada componente como ilustra la figura 13:

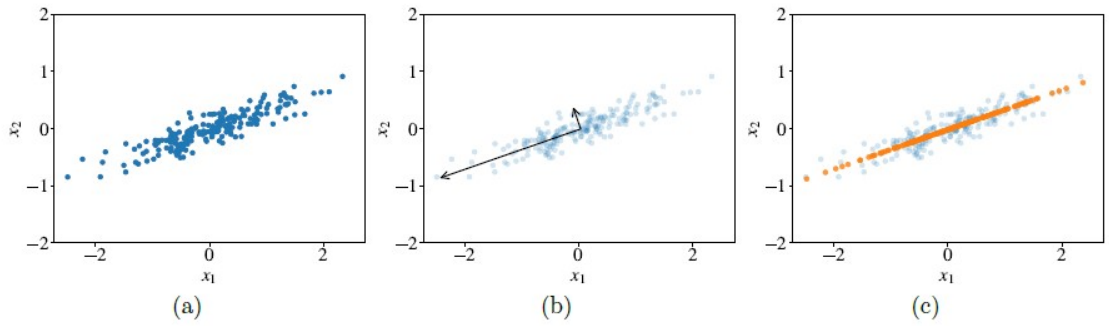

Figura 13(a) Los datos originales (b) los componentes principales como vectores (c) los datos proyectados sobre el componente principal [3]

1.2.5 Selección del algoritmo de aprendizaje

Elegir el algoritmo de aprendizaje que mejor puede aplicarse para la tarea a realizar es complejo, muchas veces limitado por el tiempo y los recursos.

Esta decisión puede estar influenciada por varios aspectos. Como ejemplo:

- Numero de características y ejemplos

El número de variables y de ejemplos de los que se disponga para el modelo influye mucho en la decisión. Las redes neuronales suelen tolerar un gran número de ejemplos y miles de características, otros como SVM tienen menor capacidad.

- No linealidad de los datos

Si los datos son linealmente separables o pueden ser modelados usando un modelo lineal se pueden utilizar algoritmos como SVM, Regresión lineal o Regresión Logística. En caso contrario se recurrirá a Aprendizaje Profundo con redes neuronales.

- Explicable

Si las relaciones que hace el modelo tienen que ser explicables en cuanto a las relaciones que establece, puede limitar los algoritmos a utilizar. El Aprendizaje Profundo puede ser difícil de entender y en muchos casos de explicar.

- Tiempo de entrenamiento y de predicción

El tiempo de entrenamiento y el tiempo necesario para realizar una predicción son factores a tener en cuenta a la hora de elegir un algoritmo de aprendizaje. SVM por ejemplo es un algoritmo que realiza predicciones muy rápidas. En el caso de kNN y algunas redes neuronales recurrentes, el tiempo de predicción es mucho más lento.

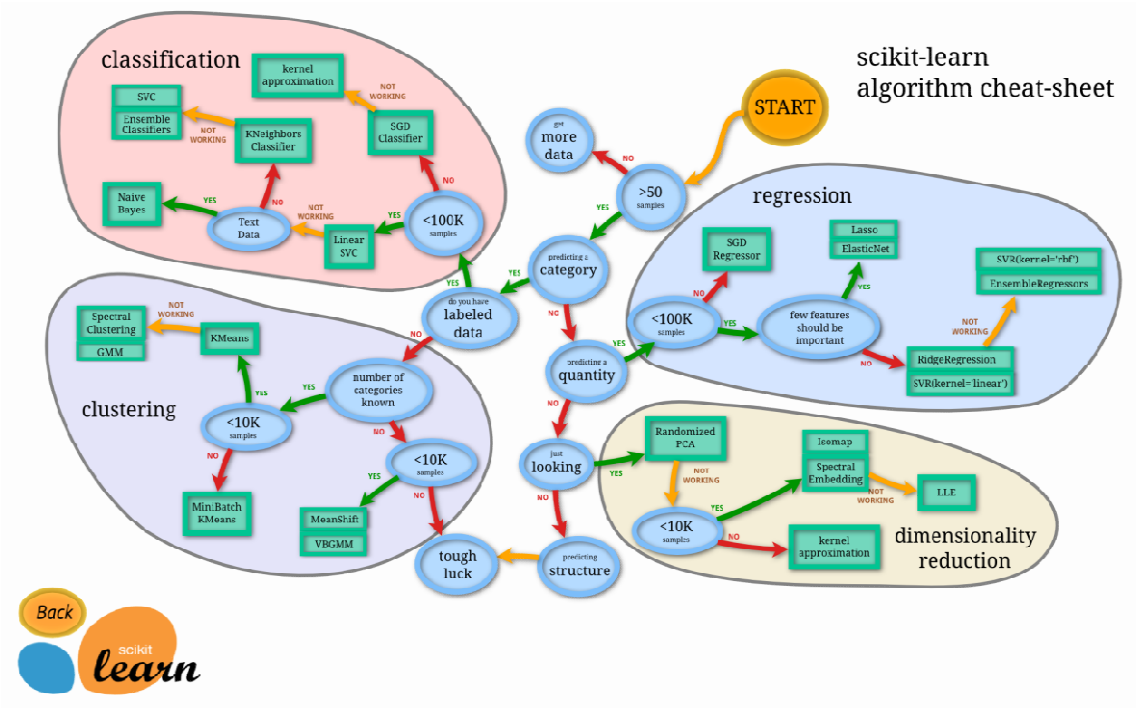

Figura 14 Diagrama de elección de algoritmo de aprendizaje de Scikit-learn [6]

1.2.6 Tratamiento de los datos

Hay una serie de rutinas que deben tenerse en cuenta previamente al entrenamiento y creación del modelo. Estas rutinas tratan los datos de entrenamiento para lograr que el modelo sea más eficiente y preciso, y en algunos casos son necesarias para poder utilizar los algoritmos de entrenamiento.

1.2.6.1 Normalización

La normalización es el proceso para cambiar el rango de valores de una variable de los datos a unos rangos estándar, normalmente en el intervalo [-1,1], [0,1].

Se realiza con la aplicación de la fórmula:

$$\bar{x}_j = \frac{x_j - \min(j)}{\max(j) - \min(j)}$$

La función principal de la normalización es la de incrementar la velocidad de aprendizaje, pues si igualamos el rango de valores de las variables de manera que ninguna esté en un rango mucho mayor que otra, de manera que no domine esta al aplicar los procesos como el error cuadrático medio.

1.2.6.2 Estandarización

La estandarización es un proceso mediante el cual se reescala una variable para que tenga las propiedades de una distribución normal con media $\mu = 0$ y desviación típica de $\sigma = 1$ siguiendo la fórmula [2]:

$$\widehat{x_j} = \frac{x_j - \mu_j}{\sigma_j}$$

Su aplicación es la misma que la normalización y el uso de uno u otro dependen del algoritmo de aprendizaje y de los datos. No es infrecuente probar con ambos y ver cual beneficia más al modelo.

1.3 Subajuste y Sobreajuste del modelo

La función principal de la normalización es la de incrementar la velocidad de aprendizaje, igualando el rango de valores de las variables para que ninguna esté en un

rango mucho mayor que otra, de manera que no domine está al aplicar los procesos como el error cuadrático medio.

Subajuste se refiere a un modelo que no puede modelar los datos de entrenamiento ni generalizar a nuevos datos, esto ocurre cuando el modelo de Aprendizaje Automático es muy simple; es un indicativo de baja precisión del modelo. Su aparición significa que el modelo o el algoritmo no se ajusta a los datos lo suficientemente bien. Suele suceder cuando se tienen menos datos de los necesarios para construir un modelo preciso y también cuando se intenta construir un modelo lineal con datos no lineales.
En tales casos, las reglas del modelo de Aprendizaje Automático son demasiado fáciles y flexibles para aplicarse a datos tan mínimos y, por lo tanto, es probable que el modelo haga muchas predicciones erróneas. La falta de adaptación se puede evitar utilizando más datos y también reduciendo las variables mediante técnicas de reducción de dimensión. Un ejemplo de esto lo tenemos en la parte izquierda de la figura 15.

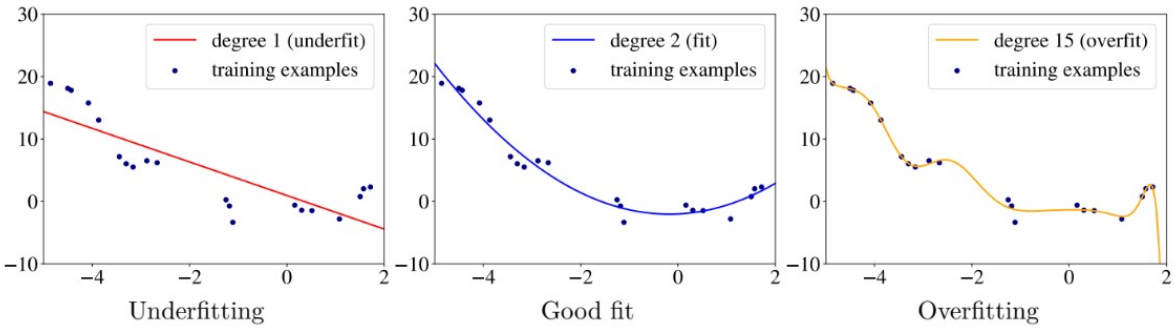

Figura 15 Modelo con subajuste (izquierda), un ajuste correcto (centro) y sobreajuste (derecha). Extraído de [3]

El sobreajuste ocurre cuando un modelo se ajusta a los datos de entrenamiento demasiado bien, como muestra la parte derecha de la figura 1.14. Esto ocurre cuando un modelo aprende el detalle, incluyendo el ruido en los datos de entrenamiento en la medida en que tiene un impacto negativo en el rendimiento del modelo en datos nuevos. Esto significa que el ruido o las fluctuaciones aleatorias en los datos de entrenamiento son recogidos y aprendidos por el modelo. El problema es que estos conceptos no se aplican a los datos nuevos y afectan negativamente a la capacidad de los modelos para generalizar.

El sobreajuste es más probable con modelos no paramétricos y no lineales porque estos tipos de algoritmos de Aprendizaje Automático tienen más libertad para construir el

modelo basado en el conjunto de datos, por lo tanto, pueden construir modelos poco realistas. Como soluciones al problema del sobreajuste se puede probar a entrenar el modelo con un algoritmo más sencillo, usar técnicas de reducción de la dimensión como PCA, regularizar o normalizar los datos o incluir más datos de entrenamiento.

Capítulo 2

Soporte Software para Aprendizaje Automático

En este capítulo incluye las principales utilidades software destinadas al desarrollo de programas con Aprendizaje Automático, así como las principales bibliotecas. También se abordará el Aprendizaje Automático como servicio y cómo las grandes empresas ofrecen sus propias aplicaciones para poder desarrollar Aprendizaje Automático para diferentes tareas, sin la necesidad de un amplio conocimiento previo tanto de programación como de Aprendizaje Automático en si mismo.

2.1 Uso del software para Aprendizaje Automático

Si bien en el capítulo anterior se ha dado una breve introducción a los fundamentos de varios algoritmos de aprendizaje, en la mayoría de los casos no es necesario desarrollarlos a la hora de crear un modelo de Aprendizaje Automático.

Todos estos algoritmos suelen tener implementaciones en bibliotecas de distintos lenguajes de programación, con los algoritmos ya depurados y listos para su aplicación en la creación de un modelo. Por poner un ejemplo, en la biblioteca Scikit Learn para el lenguaje Python se tienen ya implementados SVM, Regresión Lineal y Logística, K-Medias, entre otros

Por tanto, en muchos casos a la hora de crear un modelo de Aprendizaje Automático es importante conocer las herramientas de las que se dispone, que son muchas y muy diversas.

2.1.1 Lenguajes de Programación para Aprendizaje Automático

En este apartado citaré los principales lenguajes de programación utilizados para la implementación de Aprendizaje Automático así como algunas de las librerías más utilizadas.

2.1.1.1 Python

Python según la definición que podemos encontrar en su página web:

“Python is an interpreted, object-oriented, high-level programming language with dynamic semantics” [5]

Traducido quiere decir que Python es un lenguaje de programación interpretado, de alto nivel, orientado a objetos y con semántica dinámica. Es de propósito general y con fuente abierta, lo que hace que sea accesible como lenguaje en comparación con otros que veremos más adelante. Python combina una gran potencia con una sintaxis muy limpia. Tiene módulos, clases, excepciones y un manejo de los datos muy dinámico.

El hecho de ser de código abierto y la gran comunidad que ha se creado a su alrededor es uno de los factores que ha impulsado su avance como uno de los lenguajes más utilizados para el Aprendizaje Automático y el tratamiento de datos en general, aunque sus aplicaciones no se limitan a este campo. Tiene una filosofía de reutilización y baja complejidad que hace que sea más sencillo de aprender que otros lenguajes de programación.

Como puntos negativos, el gasto de memoria es mucho mayor que en otros lenguajes de programación. Se considera un lenguaje poco eficiente para el desarrollo de aplicaciones para móviles. Al ser un lenguaje interpretado, la velocidad es menor que en algunos lenguajes compilados.

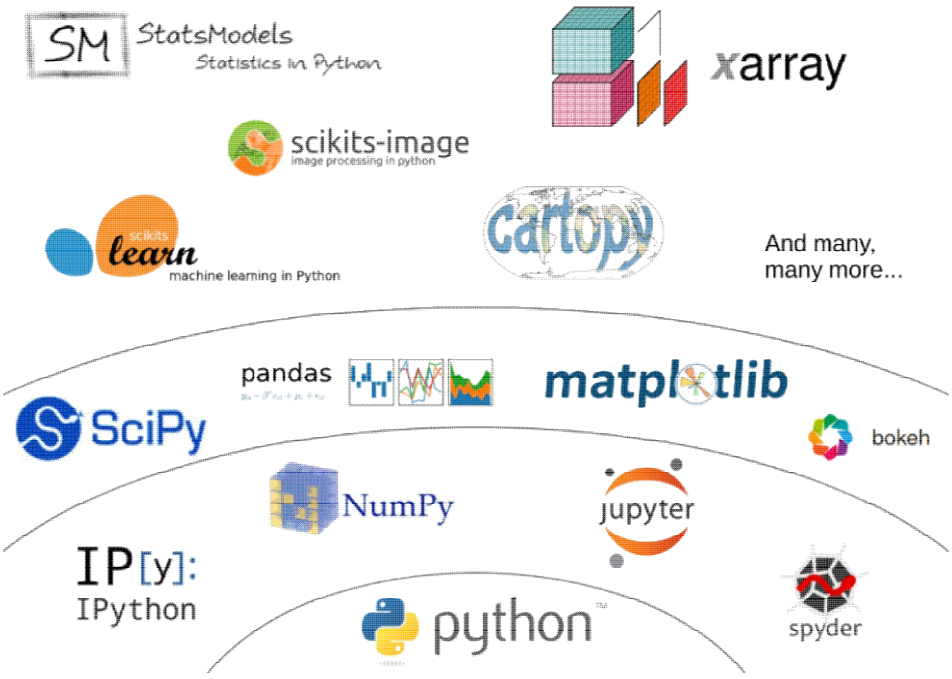

Figura 16 Distintas bibliotecas en Python con aplicaciones para Aprendizaje Automático [6]

2.1.1.2 Bibliotecas en Python

Una de las ventajas de Python es que, debido a que es de código abierto y a que existe una extensa comunidad, posee una gran cantidad de bibliotecas que pueden ser útiles [7], muchas de ellas específicamente creadas para Aprendizaje Automático y otras para otro tipo de tareas pero igualmente útiles, como se puede ver en la figura 16.

Scikit-learn

Scikit-learn (https://scikit-learn.org) es una librería de Python para Aprendizaje Automático y análisis de datos. Está basada en NumPy, SciPy y Matplotlib. Las ventajas principales de Scikit-learn son su facilidad de uso y la gran cantidad de técnicas de Aprendizaje Automático que implementa. Con Scikit-learn se puede realizar Aprendizaje Supervisado y no Supervisado. Se puede usar para resolver problemas tanto de clasificación como de regresión. Es muy fácil de usar porque tiene una interfaz simple y muy consistente. Otro punto a favor de Scikit-learn es que los hiperparámetros tienen unos valores preconfigurados por defecto adecuados para la mayoría de los casos. Estas son algunas de las técnicas de Aprendizaje Automático que podemos usar con Scikit-learn:

- Regresión lineal
- Regresión logística
- Máquinas de vectores de soporte
- Árboles de decisión
- Bosques aleatorios
- Agrupamiento
- Modelos basados en instancias
- Clasificadores bayesianos
- Reducción de dimensionalidad
- Detección de anomalías

Por todas estas características será la biblioteca elegida para desarrollar el modelo ejemplo (Capítulo 4).

NumPy

NumPy (https://numpy.org/) proporciona una estructura de datos universal que posibilita el análisis de datos y el intercambio de datos entre distintos algoritmos. Las estructuras de datos que implementa son vectores multidimensionales y matrices con

capacidad para gran cantidad de datos. Además, esta librería proporciona funciones matemáticas de alto nivel que operan en estas estructuras de datos.

SciPy

SciPy (https://www.scipy.org/) proporciona rutinas numéricas eficientes fáciles de usar y opera en las mismas estructuras de datos proporcionadas por NumPy. Por ejemplo, con SciPy puedes realizar: integración numérica, optimización, interpolación, transformadas de Fourier, álgebra lineal, estadística, etc.

Matplotlib

Matplotlib (https://matplotlib.org/) es la librería gráfica de Python estándar y la más conocida para generar gráficos de calidad:: series temporales, histogramas, espectros de potencia, diagramas de barras, diagramas de errores, etc.

Seaborn

Seaborn (https://seaborn.pydata.org/) es una librería gráfica basada en Matplotlib y especializada en la visualización de datos estadísticos. Se caracteriza por ofrecer un interfaz de alto nivel para crear gráficos estadísticos visualmente atractivos e informativos. Seaborn considera la visualización como un aspecto fundamental a la hora de explorar y entender los datos. Se integra muy bien con la librería de manipulación de datos Pandas.

Pandas

Pandas (https://pandas.pydata.org) es una de las librerías de Python más útiles para los científicos de datos. Las estructuras de datos principales en Pandas son *Series* (para datos en 1 dimensión) y *DataFrame* (para datos en 2 dimensiones). Estas son las estructuras de datos más usadas en muchos campos tales como finanzas, estadística, ciencias sociales y muchas áreas de ingeniería. Pandas destaca por lo fácil y flexible que hace la manipulación y análisis de datos.

Tensorflow

TensorFlow (https://www.tensorflow.org/) es una librería de Python, desarrollada por Google, para realizar cálculos numéricos mediante diagramas de flujo de datos. Esto

puede chocar un poco al principio porque en vez de codificar un programa codifica un grafo. Los nodos de este grafo serán operaciones matemáticas y las aristas representan los tensores (matrices de datos multidimensionales). Con esta computación basada en grafos, TensorFlow puede usarse para Aprendizaje Profundo y otras aplicaciones de cálculo científico.

¿Por qué se necesita diseñar un grafo en vez de un programa? La flexibilidad de ejecución de TensorFlow permite que el grafo que representa la red neuronal profunda y sus datos se pueda ejecutar en una o varias CPU o GPU en un PC, en un servidor o en un móvil.

Keras

Keras (https://keras.io/) es un interfaz de alto nivel para trabajar con redes neuronales. El interfaz de Keras es mucho más fácil de usar que el de TensorFlow. Esta facilidad de uso es su principal característica. Con Keras es muy fácil comprobar si las ideas propuestas tendrán buenos resultados rápidamente. Utiliza de forma transparente otras librerías de Aprendizaje Profundo.

2.1.1.3 Matlab/Octave

MatLab (abreviatura de "Matrix Laboratory") (https://es.mathworks.com/) es un programa de análisis numérico que ofrece un entorno de desarrollo integrado y con su propio lenguaje de programación interpretado. Fue creado inicialmente para trabajar con vectores y matrices desde una aproximación más matemática y es uno de los motivos por los que resulta muy intuitivo a la hora de utilizarlo para Aprendizaje Automático. Tiene multitud de herramientas de representación de datos, como ejemplo en la Figura 17, bastante intuitivas y un manejo de los datos que ayuda para la creación de algoritmos.

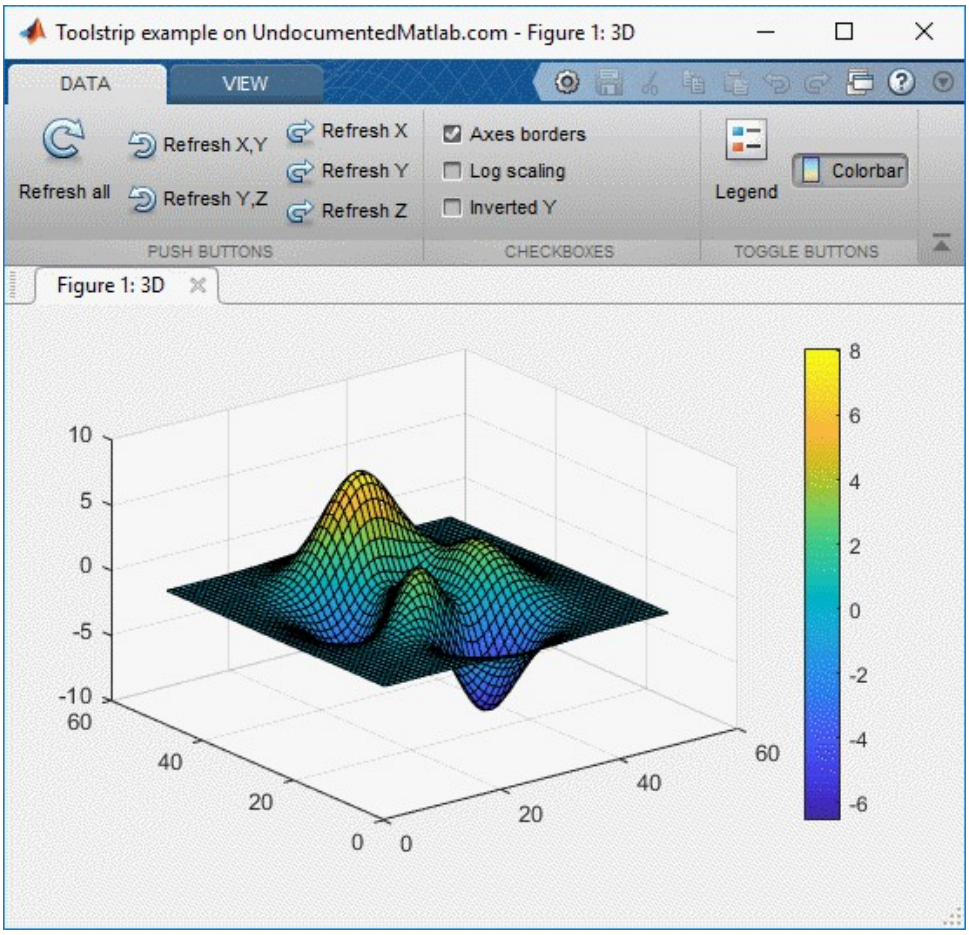

Figura 17 Representación gráfica en Matlab extraida de https://es.mathworks.com

Su uso está muy extendido en el ámbito universitario, lo que hace que tenga un gran número de usuarios. Además es un lenguaje muy sencillo para iniciarse [8].

Uno de los principales inconvenientes de Matlab, si no el mayor, es que requiere licencia, no sólo para el programa principal sino además para los distintos módulos necesarios.

No es un lenguaje recomendado para el desarrollo de productos finales, pero sí para hacer una primera aproximación a la hora de desarrollar algoritmos o como herramienta de aprendizaje [2].

Una alternativa de libre acceso a Matlab y bastante similar es GNU Octave (https://www.gnu.org/software/octave/). Se define directamente como un lenguaje de programación orientado a las matemáticas y compatible con la mayoría del código de Matlab y bastante semejante en muchos aspectos, tal y como muestra la figura 18.

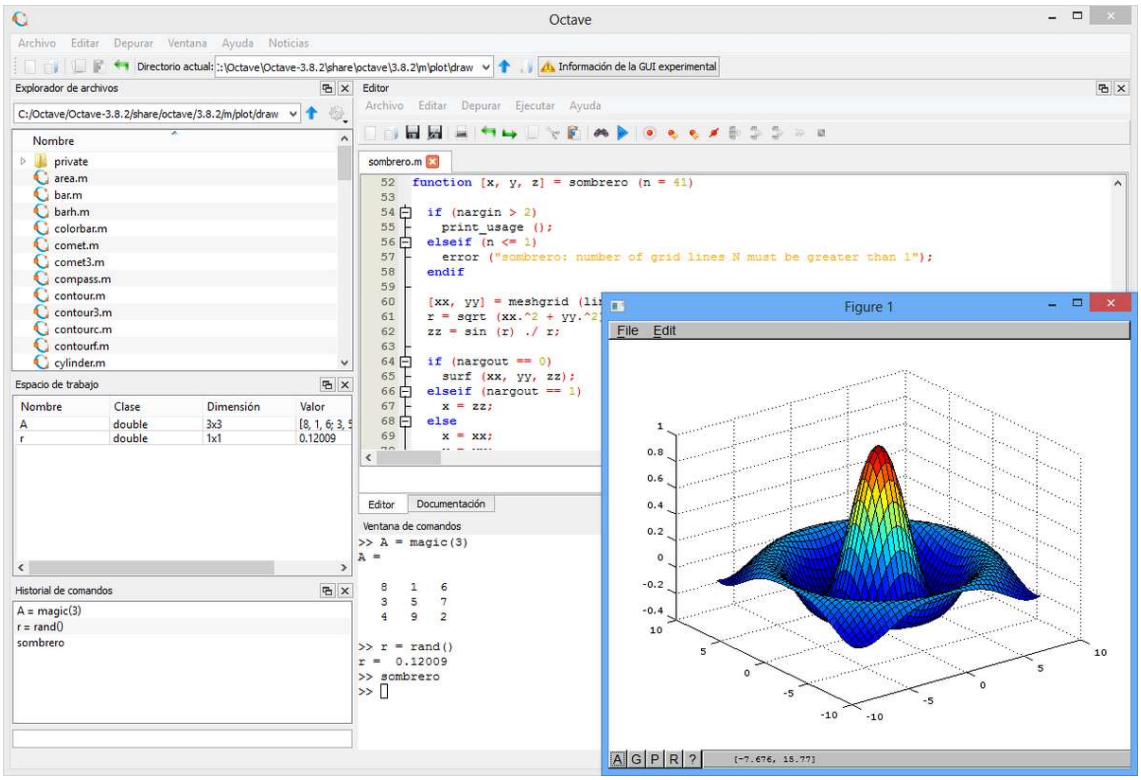

Figura 18 Interfaz de usuario de Octave extraida de https://www.gnu.org/software/octave/

2.1.1.4 Bibliotecas en Matlab/Octave

Las bibliotecas oficiales de Matlab y otros lenguajes reciben el nombre de *toolbox*. Una traducción aproximada seria *caja de herramientas* pero para una mejor comprensión se utilizara el termino original.

MatLab dispone de dos toolbox recomendadas para el Aprendizaje Automático:

Statistics and Machine Learning Toolbox

Esta *toolbox* (https://es.mathworks.com/products/statistics.html) proporciona funciones y apps para describir, analizar y modelar datos. Puede utilizar estadísticas descriptivas y gráficos para el análisis exploratorio de datos, ajustar distribuciones de probabilidad a datos, generar números aleatorios para simulaciones Monte Carlo y realizar pruebas de hipótesis. Los algoritmos de regresión y clasificación permiten extraer inferencias de los datos y crear modelos predictivos.

Para el análisis de datos multidimensionales, Statistics and Machine Learning Toolbox proporciona funciones para selección de características, regresión de pasos sucesivos,

análisis de componentes principales (PCA, por sus siglas en inglés), regularización y otros métodos de reducción de dimensionalidad que le permiten identificar variables o características que afectan a su modelo.

Esta *toolbox* proporciona algoritmos de Aprendizaje Automático supervisados y no supervisión, incluyendo máquinas de vector soporte (SVM), árboles de decisión, k-vecino más próximo, k-means, k-medoids, clustering jerárquico, modelos de mezclas de gaussianas y modelos ocultos de Markov. Muchos de los algoritmos estadísticos y de Aprendizaje Automático se pueden emplear para realizar cálculos con conjuntos de datos que son demasiado grandes como para almacenarlos en la memoria.

Deep Learning Toolbox

Deep Learning Toolbox (https://es.mathworks.com/products/deep-learning.html) (anteriormente conocida como Neural Network Toolbox™) proporciona un marco de pruebas para diseñar e implementar redes neuronales profundas con algoritmos, modelos previamente entrenados y apps. Utiliza redes neuronales convolucionales (ConvNets y CNNs) y redes Long-Sort Term Memory (LSTM) para realizar clasificación y regresión en imágenes, series temporales y datos de texto. Las apps y los gráficos ayudan a visualizar activaciones, editar arquitecturas de red y supervisar el entrenamiento.

En caso de conjuntos de entrenamiento pequeños se puede realizar transferencia del aprendizaje con modelos de red profunda previamente entrenados (incluidos SqueezeNet, Inception-v3, ResNet-101, GoogLeNet y VGG-19) y modelos importados desde TensorFlow™-Keras y Caffe.

LIBSVM y LIBLINEAR

LIBSVM (https://www.csie.ntu.edu.tw/~cjlin/libsvm/) y LIBLINEAR (https://www.csie.ntu.edu.tw/~cjlin/liblinear/) son dos librerías de Aprendizaje Automático desarrolladas por la universidad de Taiwán. Ambas implementan SVM, si bien LIBSVM lo hace con el algoritmo SMO (sequential minimal optimization) valido para clasificación y regresión, y LIBLINEAR implementa SVM lineal con un algoritmo de descenso coordinado.

Aunque están escritas originalmente en C++ pero ambas están disponibles para su uso en Matlab y Octave.

2.1.1.5 JAVA

Java es un lenguaje de programación de alto nivel, compilado e interpretado y orientado a objetos. Es uno de los lenguajes de programación de uso más extendido y es lógico pensar que también se usa para el desarrollo de Aprendizaje Automático.

Una de las ventajas de Java es que muchos de los programas de almacenamiento de datos están escritos con base de Java. Eso hace que el acceso a esa información sea a través de Java y sea más fácil trabajar si mantenemos el mismo entorno.

Por tanto, aunque a día de hoy no es uno de los lenguajes que se prioricen para Aprendizaje Automático, su extendido uso como lenguaje impulsa la creación de librerías y lo vuelve un lenguaje de programación para tener en cuenta [9].

2.1.1.6 Librerías de Java

Weka

Waikato Environment for Knowledge Analysis (Weka) es una plataforma de desarrollo para Aprendizaje Automático desarrollado por la Universidad de Waikato, Nueva Zelanda (https://www.cs.waikato.ac.nz/ml/weka/). Está escrito en Java y tiene su propia interfaz gráfica (figura 19), interfaz de comandos y Java API. Es quizás la librería de Aprendizaje Automático más popular en Java.

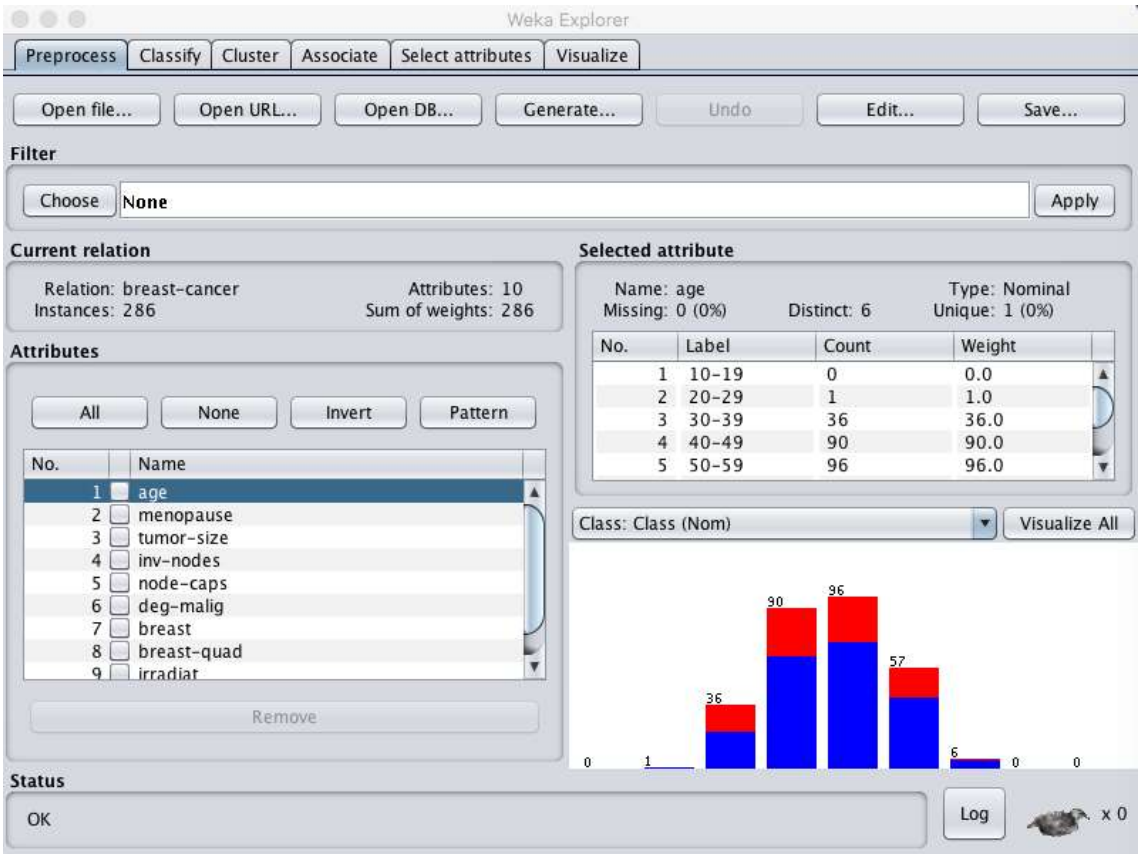

Figura 19 Interfaz gráfica de Weka extraida de www.cs.waikato.ac.nz/ml/weka/

Java-ML

La Java Machine Learning Library (http://java-ml.sourceforge.net/) contiene una gran cantidad de algoritmos de Aprendizaje Automático implementados en Java. Tiene una interfaz estándar para cada algoritmo y referencias a literatura científica relevante para un mayor entendimiento. Incluye métodos de manipulación de datos, clustering, selección de parámetros y clasificación. La gran desventaja es que la última publicación fue en junio de 2012.

SMILE

SMILE (http://haifengl.github.io/smile/) (Statistical Machine Intelligence and Learning Engine) es un sistema en Java y Scala, rápido y completo de Aprendizaje Automático, procesamiento de lenguaje, álgebra lineal, interpolación y visualización. Cubre varios aspectos del Aprendizaje Automático como los algoritmos de clasificación y regresión, agrupamiento, minería de datos, selección de característica, escalado multidimensional y valores perdidos, entre otros.

2.1.1.7 R

R (https://www.r-project.org) es un entorno y un lenguaje de programación diseñado para el análisis estadístico, distribuido bajo la licencia GNU GPL, que se creó como un dialecto libre del lenguaje S.

El hecho de ser un lenguaje estrechamente relacionado con la estadística hace que haya ganado su puesto destacado para el desarrollo del Aprendizaje Automático. Dado que es un lenguaje enfocado plenamente en la estadística y la relación de los modelos estadísticos con el Aprendizaje Automático, las nuevas ideas y tecnologías en muchos casos aparecen primero en R antes de llevarse a otros lenguajes.

Como desventaja podemos decir que la curva de aprendizaje de R es más pronunciada que en otros lenguajes de programación [10].

2.1.1.8 Librerías R

MLR: Machine Learning in R

MLR (https://mlr.mlr-org.com/) es una biblioteca que incluye un gran número de técnicas de regresión y clasificación, así como sus correspondientes métodos de optimización y evaluación. MLR posee enlace con el paquete OPENML y su plataforma en línea, que proporciona un entorno colaborativo donde se pueden compartir datos, algoritmos, modelos y distintas utilidades de Aprendizaje Automático.

2.2 Lenguaje de programación más utilizado

Una vez presentadas muchas opciones para programar algoritmos de Aprendizaje Automático, y obviando así como algunas de sus ventajas y desventajas, es lógico preguntarse cuál será el mejor lenguaje de programación a la hora de afrontar la implementación del Aprendizaje Automático.

Sin embargo, la respuesta a esta pregunta es ambigua y depende mucho del contexto. Puede depender de si se busca un lenguaje para comprender los conceptos del Aprendizaje Automático o bien para desarrollar estudios complejos. También depende mucho de los conocimientos previos en alguno de los lenguajes que se utilizan.

Atendiendo a la comparativa realizada por "Towards Data Science" en 2017 [11] se analizan los siguientes resultados:

Python se posiciona lider entre los 5 lenguajes más utilizados, con un 57% de uso y un 33% de prioridad para el desarrollo, lo cual es entendible si tenemos en cuenta el gran desarrollo de librerías que tiene y la aparición de paquetes como Tensorflow. Muchas veces se compara el uso de Python en el Aprendizaje Automático con el de R, sin embargo no son comparables en términos de popularidad, con R ocupando la cuarta posición de uso con un 31% y la quinta en términos de prioridad con un 5%. De hecho, de los cinco lenguajes más utilizados es el que menos prioridad tiene, con solo un 17% de prioridad entre los desarrolladores que lo utilizan. De aquí podemos extraer que en la mayoría de los casos R es un lenguaje complementario al desarrollo en otros lenguajes.

Por el contrario Python, con un 58% de prioridad entre los que lo usan, es con diferencia el que tiene el indicador mayor. No sólo es el más ampliamente usado sino que es la primera opción para la mayoría de usuarios. Los lenguajes C/C++ son los segundos por detrás Python, con un uso del 44% y una prioridad del 19%. El lenguaje Java sigue a C/C++ de cerca y Javascript se sitúa quinto con una prioridad sólo un poco mayor que R (7%). Sobre otros lenguajes como Julia, Scala, Ruby, Octave, MATLAB, etc, el uso era inferior al 26% y la prioridad inferior al 5% por eso en el estudio se descartaron para otros campos.

Además, se analizó en dicho estudio el uso de cada lenguaje según el tipo de proyecto para el que se empleaba. Así, se pudo extraer que para análisis de sentimiento la prioridad la tenían Python y R, en el caso de detección de fraude y seguridad en red la

prioridad era para Java. La inteligencia artificial para juegos y el movimiento robotizado son las áreas donde destaca el uso de C/C++. El lenguaje R tenía una alta prioridad en el campo de la bioingeniería y bioinformática, algo lógico si tenemos en cuenta lo extendido que está R en la estadística biomédica.

También es importante tener en cuenta el contexto profesional, según los datos analizados por la encuesta. Python es el lenguaje de programación más priorizado por aquellos para los que el Aprendizaje Automático es su primera profesión o campo de estudio. Esto confirma que Python se ha convertido en una parte integral de la ciencia de datos y el Aprendizaje Automático. R tiene la prioridad entre los analistas de datos y los estadísticos, dado que fue un lenguaje creado para esa rama como reemplazo a S.

Los desarrolladores de capa de acceso web se decantan más por JavaScript, mientras que los desarrolladores de aplicaciones priorizan más Java. Para los que provienen del campo de la ingeniería eléctrica y la electrónica, la mayoría continúan utilizando C/C++ ya que suelen ser los lenguajes más utilizados en esos campos.

Por tanto, no se puede decir que exista un único lenguaje de programación superior para el Aprendizaje Automático, sino que puede depender de qué tipo de aplicación se vaya a desarrollar, del campo de origen y de los conocimientos previos de algún otro lenguaje, dado que en la mayoria de los casos los desarrolladores continúan usando el lenguaje de trabajos previos.

Lo que no se puede negar es la popularidad de Python, quizás por su gran cantidad de librerías especializadas y su curva de aprendizaje reducida. Eso lo convierte en la opción más elegida si el Aprendizaje Automático es la primera aproximación también a la programación. [12]

2.3. Aprendizaje Automático como servicio

El Aprendizaje Automático como servicio (conocido por sus siglas en ingles MLaaS, “Machine Learning as a Service”) es un paraguas con el que se definen varias plataformas basadas en la nube que solucionan los principales problemas de infraestructura que se pueden tener a la hora de implementar Aprendizaje Automático,

como el procesamiento previo de los datos, el entrenamiento de los modelos y la evaluación del modelo.

Cuatro grandes plataformas que pertenecen a cuatro de las empresas tecnológicas más grandes lideran estos servicios: Amazon Machine learning services de Amazon (https://aws.amazon.com/es/machine-learning/) , Azure Machine Learning de Microsoft (https://azure.microsoft.com/), Google Cloud AI de Google (https://cloud.google.com/products/ai/) e IBM Watson de IBM (https://www.ibm.com/watson). Pretenden atender un mercado que quiere aplicar Aprendizaje Automático pero que quizás no dispone de un equipo de trabajo necesario para el desarrollo de modelos propios [13].

A continuación se describen con algo más de detalle, aunque podemos ver una comparativa inicial en la tabla 1.

Tabla 1 Comparativa entre los distintos MLaaS, indicando aplicaciones automatizadas y desarrollo de modelos personalizados

	Amazon	**Microsoft**	**Google**	**IBM**
Servicios Automáticos o Semiautomáticos de Aprendizaje Automático				
	Amazon ML	Microsoft Azure ML Studio	Cloud AutoML	IBM Watson ML Model Builder
Clasificación	SI	SI	SI	SI
Regresión	SI	SI	SI	SI
Agrupamiento	SI	SI	NO	NO
Detección de anomalías	NO	SI	NO	NO
Recomendación	NO	SI	SI	NO
Ranking	NO	SI	NO	NO
Servicios de desarrollo de modelos personalizados				
	Amazon SageMaker	**Azure ML Services**	**Google ML Engine**	**IBM Watson Studio**
Algoritmos Incorporados	SI	SI	NO	SI

2.3.1 Amazon Machine Learning

Al hablar de los servicios para Aprendizaje Automático de Amazon se puede distinguir entre dos opciones: análisis predictivo con Amazon Machine Learning y la herramienta SageMaker para científicos de datos.

Amazon Machine Learning para análisis predictivo se puede considerar como una de las opciones más automatizadas para solucionar las necesidades de Aprendizaje Automático. El servicio permite cargar los datos desde múltiples fuentes, como Amazon RDS, Amazon Redshift, archivos CSV, etc. Todas las operaciones de procesado previo de datos se realizan de manera automática: Amazon ML detecta qué campos son categóricos y cuáles numéricos, y además realiza de manera automática operaciones más avanzadas como la reducción de la dimensión.

Las capacidades de predicción de Amazon ML están limitados a tres opciones: clasificación binaria, clasificación multiclase y regresión. No se incluye ningún método de Aprendizaje no Supervisado y el usuario debe elegir una variable como variable objetivo. No es necesario ningún conocimiento previo de Aprendizaje Automático dado que el servicio elige el algoritmo que mejor se ajuste a los datos a analizar.

Con todo lo expuesto, se puede ver que la automatización de casi todas las elecciones puede ser tanto una ventaja como una desventaja, pues apenas deja elegir opciones para el análisis. Si lo que el usuario necesita es una solución más personalizable, lo mejor en el caso de Amazon es recurrir a SageMaker (https://aws.amazon.com/es/sagemaker/).

SageMaker es un entorno cuyo objetivo es simplificar el trabajo del analista de datos mediante herramientas que permiten crear e implementar modelos de manera rápida. Entre sus servicios incluye Jupyter para simplificar la exploración y el análisis de datos. SageMaker dispone de algoritmos incorporados que están optimizados para conjuntos de datos grandes con computación en sistemas distribuidos. Incluye múltiples opciones de algoritmos para clasificación, regresión, arboles de decisión, reconocimiento de imagen, predicción de secuencias y traducción, agrupamiento, reducción de dimensión y redes neuronales entre otros.

Si no quieres utilizar ninguno de los modelos desarrollados siempre se pueden crear métodos propios mediante las opciones que da SageMaker o se puede integrar SageMaker con TensorFlow, Keras y otras librerías de Aprendizaje Automático.

Con todo esto, Amazon ML proporciona un servicio que da cobertura tanto a profesionales del análisis de datos como usuarios sin un conocimiento previo, lo que lo que convierte en una gran opción para empresas que ya utilizan los servicios de Amazon en la nube.

2.3.2 Azure Machine Learning

Azure Machine Learning platform tiene como objetivo establecer un entorno potente tanto para iniciados como para científicos de datos experimentados. Ofrece prácticamente los mismos servicios que Amazon ML pero Azure a día de hoy es más flexible con la construcción de algoritmos fuera de los predefinidos. Se pueden dividir los servicios de Azure en dos grandes categorías: Azure Machine Learning Studio y Bot Service.

Azure Machine Learning Studio es el mayor paquete de MLaas. Casi todas las operaciones en Azure pueden completarse usando una interfaz grafica que permite el manejo de los datos de forma intuitiva, incluyendo exploración de los datos, procesado previo, elección de los algoritmos y validación de los resultados del modelo.

El empleo de Azure para Aprendizaje Automático tiene una curva de aprendizaje elevada, pero puede ayudar al entendimiento de la mayoría de las técnicas. La interfaz

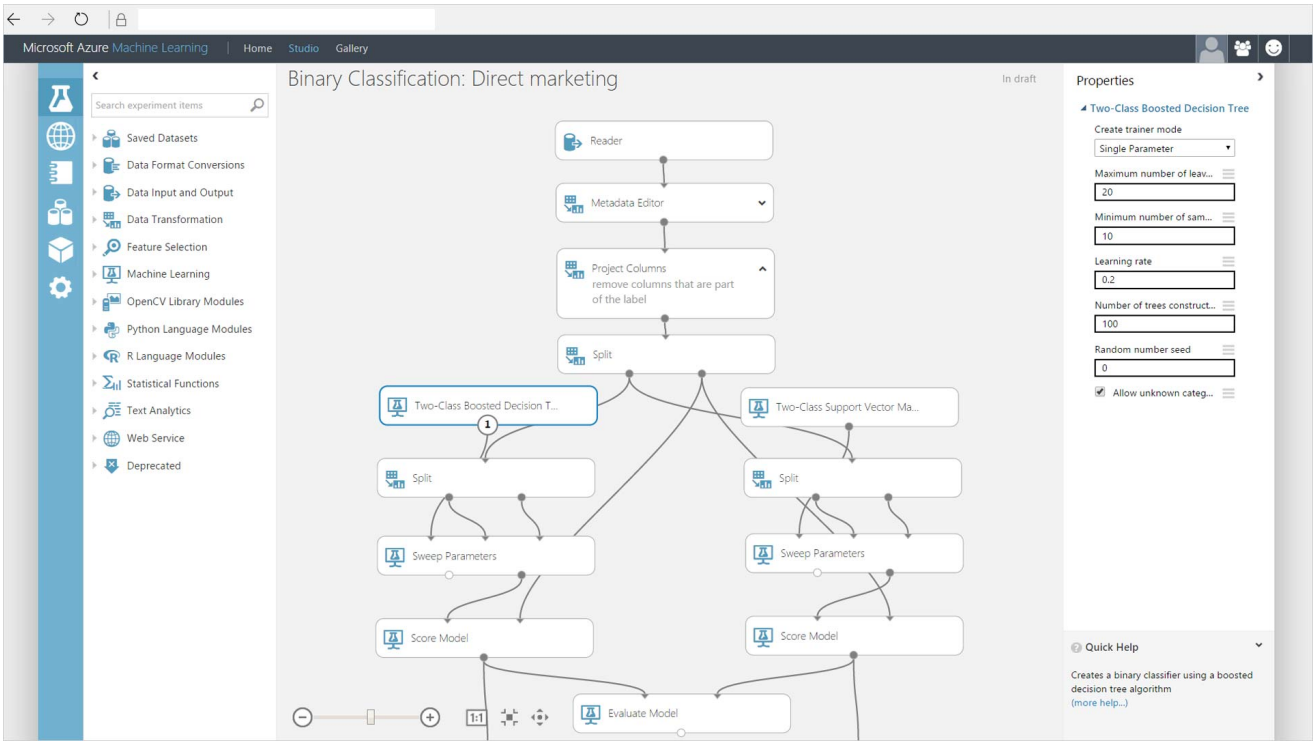

Figura 20 Ejemplo de interfaz gráfica de Azure studio extraido de https://azure.microsoft.com

grafica permite visualizar cada paso, así como el flujo de trabajo como muestra la figura 2.5. Azure dispone de un amplio abanico de algoritmos para implementar, unos 100 métodos de toda clase (aunque solo posee uno de agrupamiento, K-means).

Otro servicio para tener en cuenta de Azure es Cortana Intelligence Gallery, una colección de soluciones de Aprendizaje Automático creados por la comunidad que pueden ser revisados y reutilizados.

En 2017 se creó un nuevo conjunto de productos relacionados con el Aprendizaje Automático que Microsoft llamó Azure Machine Learning Services, pero no dispone de algoritmos previamente construidos sino que requiere un mayor trabajoy diseño. Está obviamente mas enfocada a usuarios experimentados. Integra entornos populares mencionados con anterioridad, como TensorFlow y Scikit-learn.

2.3.3 Google Cloud AI

Con una filosofía similar a la de Amazon, Google ofrece su producto MLaaS en dos niveles: Google Cloud Machine Learning Engine para usuarios experimentados y con amplios conocimientos y Cloud AutoML para usuarios menos experimentados.

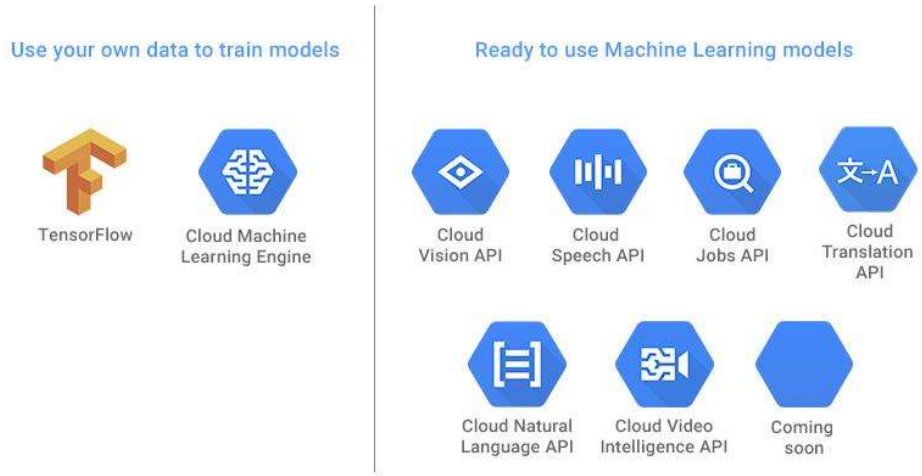

Figura 21 Distintos productos de Google Cloud AI extraidos de https://cloud.google.com/automl/

Google Cloud AutoML es una plataforma de Aprendizaje Automático en la nube diseñada para el usuario inexperto. Los desarrolladores pueden cargar los datos, entrenar los modelos y desarrollarlos en la web. Está completamente integrado con

todos los servicios de Google y almacena los datos en la nube. Algunos productos de AutoML tienen interfaz grafica.

Mientras que AutoML ofrece automatización al precio de perder flexibilidad en el diseño, Google ML Engine es todo lo contrario. Está enfocado a los usuarios experimentados y ofrece la posibilidad de usar librerías como TensorFlow, XGBoost, scikit-learn y Keras en la nube. Eso hace que sea muy similar a SageMaker.

TensorFlow es sí mismo es otro producto de Google, una librería de Aprendizaje Automático de código abierto. No tiene una interfaz grafica, la curva de aprendizaje es pronunciada y está más enfocada al Aprendizaje Profundo, pero es una herramienta muy potente.

2.3.4 IBM Watson Machine Learning Studio

La propuesta de MLaaS de IBM es una plataforma integrada tanto para el usuario experimentado como para el iniciado en la materia del Aprendizaje Automático. El sistema ofrece dos aproximaciones, la automática y la manual. De manera parecida a Amazon ML, Watson studio tiene AutoAI que provee procesado de datos Automático y una interfaz de construcción de modelos que apenas requiere ningún paso previo para empezar a procesar los datos y desarrollar los modelos.

La parte automática puede resolver tres tipos de tareas: clasificación binaria, clasificación multiclase y regresión. Se puede elegir que sea completamente automático o elegir manualmente el algoritmo, entre los 10 que tiene disponible para cumplir con las tareas.

Además de AutoAI, Watson dispone de otros dos servicios que se pueden usar para construir modelos: SPSS Modeler, un paquete de software que se usa para transformar los datos en datos estadísticos y Neural Network Modeler, similar al anterior pero enfocado a redes neuronales.

De manera separada, IBM ofrece una interfaz para Aprendizaje Profundo y un entorno similar a Jupyter para programar manualmente modelos con frameworks como TensorFlow, scikit-lear, PyTorch, entre otros.

2.3.5 APIs de MLaaS

Además de las distintas plataformas completas para aplicar Aprendizaje Automático, existen APIs de alto nivel que proporcionan servicios concretos de alguna rama del Aprendizaje Automático. Estas APIs están preparadas para introducir los datos y obtener resultados usando modelos ya entrenados y probados. Se pueden clasificar en 3 grandes grupos:

1. Reconocimiento de texto, traducción y análisis textual
2. Reconocimiento de imagen y video
3. Otros

Cada una de las compañías anteriores posee varias de estas APIs e intentan cubrir el abanico completo de necesidades que puedan tener los usuarios.

2.3.6 Utilidad del MLaaS

Existe una cierta controversia sobre si el MLaaS tiene realmente un segmento de mercado [14]. Las empresas intentan vender productos de MLaaS como una capa más de sus servicios en la nube, pero existe un problema de fondo: el MLaaS no tiene un segmento real de mercado, no es realmente útil para el usuario competente ni para el usuario iniciado. El segmento formado por los usuarios (y potenciales clientes) que ya poseen una competencia en Aprendizaje Automático tienden a utilizar las herramientas de código abierto que ofrecen en muchos casos los MLaaS como parte herramientas para crear sus modelos, por lo tanto, no necesitan estas plataformas para aplicar Aprendizaje Automático.

En el otro extremo está el usuario que no posee ni conocimientos ni competencia en Aprendizaje Automático, que además no va a conseguir aplicar el Aprendizaje Automático con estas plataformas. Lo que necesita son aplicaciones que resuelvan problemas más elevados, para los que el Aprendizaje Automático es sólo una parte del proceso.

Por tanto, en muchos casos el MLaaS no suele aportar demasiado a los usuarios que ya manejan y desarrollan modelos de Aprendizaje Automático y no suele ser útil para los

usuarios iniciados puesto que no les aporta soluciones directas que puedan capitalizar. La parte realmente sencilla del MLaaS está demasiado automatizada, acotada y no permite flexibilidad, lo que impide abordar tareas que se salgan de sus opciones preestablecidas.

Capítulo 3

El Aprendizaje Automático y la eSalud, con aproximación al la cardiología

En este capítulo se revisan conceptos de eSalud así como las aplicaciones que ofrece el Aprendizaje Automático para centrar posteriormente el interés en el campo de la Cardiología.

3.1 La eSalud.

La eSalud (*eHealth* en inglés) es el término con el que se define al conjunto de Tecnologías de la Información y la Comunicación (TICs) que, a modo de herramientas, se emplean en el entorno sanitario en materia de prevención, diagnóstico, tratamiento, seguimiento y gestión de la salud, ahorrando costes al sistema sanitario y mejorando su eficacia.

Engloba diferentes productos y servicios para la salud, como aplicaciones móviles, la telemedicina, los dispositivos *wearables* (para la monitorización que se integran en ropa y accesorios), el *Big Data* (grandes cantidades de datos), los sistemas de apoyo a la decisión clínica, el Internet de las Cosas o los videojuegos de salud, entre otros.

En opinión del doctor Sergio Vañó, presidente de la Asociación de Investigadores en eSalud (AIES), la eSalud supone una "transformación radical de la sanidad y, por ello, es necesario una evaluación de la eficacia y la seguridad de los sistemas de eSalud", con el objetivo de que los profesionales sanitarios "*estén preparados y, los datos proporcionados por los dispositivos de monitorización, puedan integrarse en la asistencia sanitaria*"[15].

3.2 La medicina, la Inteligencia Artificial y el Aprendizaje Automático

La Inteligencia Artificial y las tecnologías basadas en el Aprendizaje Automático tienen potencial para cambiar la medicina, transformando la inmensa cantidad de datos por ella generados en nuevas e importantes aplicaciones e investigaciones. Ejemplos de aplicación de esta tecnología podrían ser la detección precoz de enfermedades, el

aumento de la precisión en el diagnostico, la identificación de nuevos patrones y el desarrollo de diagnósticos y terapias personalizadas. Entre los grandes beneficios del Aprendizaje Automático está la capacidad para aprender de los ejemplos del mundo real y su capacidad de mejorar su funcionamiento. Inmerso en el Aprendizaje Automático está el entrenamiento a través de ejemplos dados, extraídos del mundo real y su adaptación y mejora, constituyendo un gran avance respecto a otro software convencional de uso médico.

La práctica clínica se basa en el manejo de grandes cantidades de datos, tendiendo actualmente a su recogida automatizada. Esto puede parecer una ventaja para la aplicación de técnicas de Aprendizaje Automático en la medicina, pero existen dificultades. Las consideraciones legales y éticas del uso de esos datos así como, en muchos casos, la dificultad para que distintos centros los compartan, son desventajas no despreciables que tiene esta tecnología en su aplicación médica.

3.3 El Aprendizaje Automático y la Cardiología.

El objetivo principal de la aplicación de técnicas de Aprendizaje Automático en cardiología es crear herramientas con las que el cardiólogo pueda mejorar su atención al paciente. Los cardiólogos toman decisiones para el cuidado del paciente en relación a los datos que analizan y tienden a tener acceso a una mayor cantidad de datos en comparación con otras especialidades médicas. El volumen y complejidad de los datos biomédicos generados durante el proceso asistencial al paciente crece ininterrumpidamente, lo que induce a pesar que las aplicación de Aprendizaje Automático puedan contribuir a desarrollar modelos que aporten una mayor eficacia en el ejercicio de la actividad. La Cardiología como especialidad debería plantearse la incorporación de estas técnicas. Como ejemplo baste imaginar la implementación de algoritmos de visión computarizada, como en Radiología, a las muy diversas técnicas de imagen que el cardiólogo usa habitualmente.

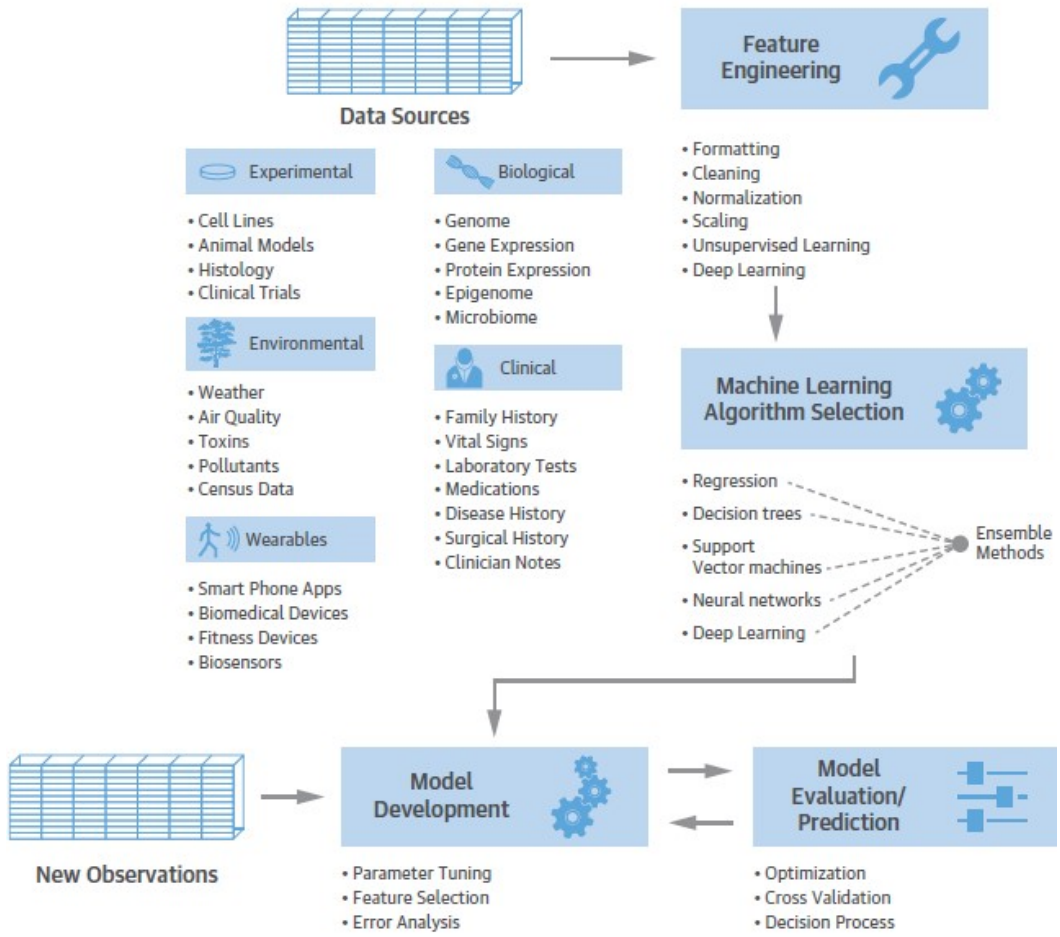

Figura 22 Flujo de trabajo del Aprendizaje Automático con la práctica clínica. Se pueden observar las distintas fuentes de datos médicos y el flujo de trabajo para crear un modelo de Aprendizaje Automático [16]

El cardiólogo habituado al ejercicio de su especialidad como ciencia y arte deberá modificar su actitud desde la aprensión hacia la curiosidad y desde el rechazo hacia el entusiasmo por estas técnicas. El Aprendizaje Automático y en general la Inteligencia Artificial puede mejorar el trato a los pacientes porque hará al cardiólogo capaz de analizar e interpretar más datos llegando a un nivel de profundidad que antes no era posible. El Aprendizaje Reforzado puede convertirse en una herramienta para adaptar los tratamientos del paciente en función de la mejora obtenida a través del análisis de los datos recogidos en tiempo real. El Aprendizaje no Supervisado puede mejorar la identificación de variables que afecten a enfermedades en los pacientes y con Aprendizaje Supervisado seleccionar un tratamiento mejor. Tampoco podemos obviar los avances que se pueden hacer en la interacción con los EHRs (del inglés Electronic Health Record o registro electrónico de los datos del paciente), con la automatización de la extracción de datos o del registro de variables mediante técnicas de reconocimiento de texto libre. Y aunque hoy en día estas técnicas deben ser desarrolladas por personal

técnico especializado, con el tiempo su aplicación puede llegar a ser más sencilla, universal y amigable.

3.3.1 Aumento del interés de la Cardiología en el Aprendizaje Automático

El interés que ha despertado el Aprendizaje Automático en la Cardiología queda patente por el aumento progresivo de artículos publicados en los últimos años. Realizando una búsqueda en la base de datos bibliográfica de National Center of Biotechnology Information, US National Library of Medicine (PubMed: https://www.ncbi.nlm.nih.gov/pubmed/) introduciendo los términos ["machine learning"] AND ["cardiology"] que abarque los últimos 10 años, se localizan 427 artículos, distribuidos temporalmente según la figura 23.

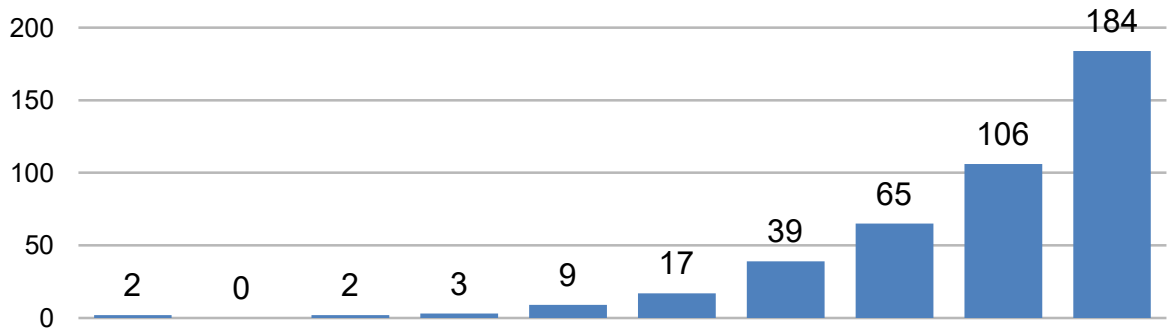

Figura 23 Evolución de las publicaciones sobre Aprendizaje Automatico en PubMed (2010-2019)

Depurando aquellos artículos no directamente relacionados con la Cardiología (59), la temática del resto se podría distribuir en la siguiente tabla:

Tabla 2 Publicaciones PubMed 2010-2019

Temática	Predicción	Clasificación	Diagnóstico	TOTAL
ECG/Arritmias	12		33	45
Insuficiencia cardíaca	12	15	11	38
Cardiopatía isquémica	23		10	33
Factores de riesgo	9	12		21
Congénitas			16	16
Cirugía cardíaca	10			10
Hospitalización	9			9
Biomarcadores			8	8
Miscelánea	12			12

Además de las publicaciones referentes a aspectos clínicos, destaca el elevado número de artículos referentes a aplicación del Aprendizaje Automático a las técnicas de imagen: ecocardiografía (15), coronariografía (15), tomografía computarizada (34) y resonancia magnética/ medicina nuclear (16).

Es necesario destacar el interés que despierta la aplicación de estas técnicas para el manejo, interpretación y análisis de los datos recogidos en las historias clínicas, tanto en formato electrónico como en texto libre: un total de 24 artículos.

Como novedad que ha sido la irrupción del Aprendizaje Automático en tantos campos de la actividad cardiológica, es lógico que se hayan publicado en este decenio 53 revisiones y editoriales, unos con intención divulgativo-formativa y otros como crítica excéptica al comparar sus propuestas con el método científico estándar y las prácticas habituales en el ejercicio asistencial.

A continuación en la tabla disponible en el ANEXO 1 se recogen algunos de los artículos relacionados con el Aprendizaje Automático en Cardiología. Como muestra de la candente actualidad de este tema, en el último Congreso de la Sociedad Europea de Cardiología -celebrado en Paris en septiembre de 2019- se presentaron 49 comunicaciones científicas basadas en el uso de Aprendizaje Automático en la Cardiología: imagen cardíaca (ecocardiografia, TAC coronario, resonancia magnetica) monitorización del paciente y telemedicina, modelos de predicción de riesgo cardiovascular y de estratificación del síndrome coronario agudo, predicción del pronóstico de insuficiencia cardíaca, entre otros.

3.4 Aplicación del Aprendizaje Automático a la Cardiología.

A continuación se detallan algunos de campos de la Cardiología donde se ha utilizado Aprendizaje Automático.

3.4.1 Electrocardiograma

La automatización de la interpretación de electrocardiogramas (ECG) es a día de hoy común y pionera entre las aplicaciones Inteligencia Artificial en la Cardiología. Los modelos de Aprendizaje Automático actuales son capaces de identificar la morfología de las ondas con gran precisión y, utilizando esta información, pueden calcular sus parámetros (desviaciones de ejes, longitud de intervalos y frecuencia cardíaca). Se ha probado la aplicación de modelos de alta precisión en la detección de desviaciones del segmento ST y alteraciones del ritmo como fibrilación auricular y retrasos en la conducción interventricular, aunque las arritmias más complejas todavía necesitan la validación de un médico. Para este tipo de modelos se han utilizado algoritmos como SVM y redes neuronales.

Los últimos avances en ese campo incluyen el procesamiento del ECG para crear un modelo que caracteriza la onda mediante una serie de vectores de variables y lo clasifica mediante un algoritmo SVM con kernel gaussianos entre una de las cinco arritmias más comunes. Con este método consiguieron una precisión en la clasificación entre ritmo sinusal, bloqueo de rama izquierda, bloqueo de rama derecha, extrasístoles ventriculares y extrasístoles auriculares del 100%, 98,66%, 100%, 99,66% y 100% respectivamente[17]. Un estudio reciente realizado por el Standford Machine Learning Group [18] usa una red neuronal convolucional de 34 capas para conseguir clasificar un ECG distinguiendo entre un amplio tipo de arritmias obteniendo sensibilidad y precisión comparables a la valoración visual por cardiólogos.

3.4.2 Ecocardiogramas

El ecocardiograma en 2 dimensiones es una técnica de imagen básica para el diagnóstico y seguimiento de numerosas cardiopatías. Sin embargo depende mucho de la interpretación visual de la imagen mostrada. Los modelos de Aprendizaje Automático pueden automatizar y mejorar muchos de estos procesos. Existen múltiples herramientas software comerciales que utilizan datos extraídos del ecocardiograma para realizar

detecciones de anomalías o medición del flujo sanguíneo, pero suelen depender de extraer los datos del ecocardiograma, evaluando previamente la imagen para decidir que parámetros medir. Aunque estos pasos pueden parecer triviales en el caso de un único paciente, constituyen una seria limitación si se quiere hacer un estudio con gran número de ecocardiogramas. Mediante algoritmos de Aprendizaje Automático, mayormente con redes neuronales por tratarse de análisis de imagen, se puede lograr un procesado de ecocardiogramas que se han utilizado para detectar y caracterizar patologías valvulares así como para mejorar la calidad de ecocardiogramas ya realizados, como muestra la figura 24.

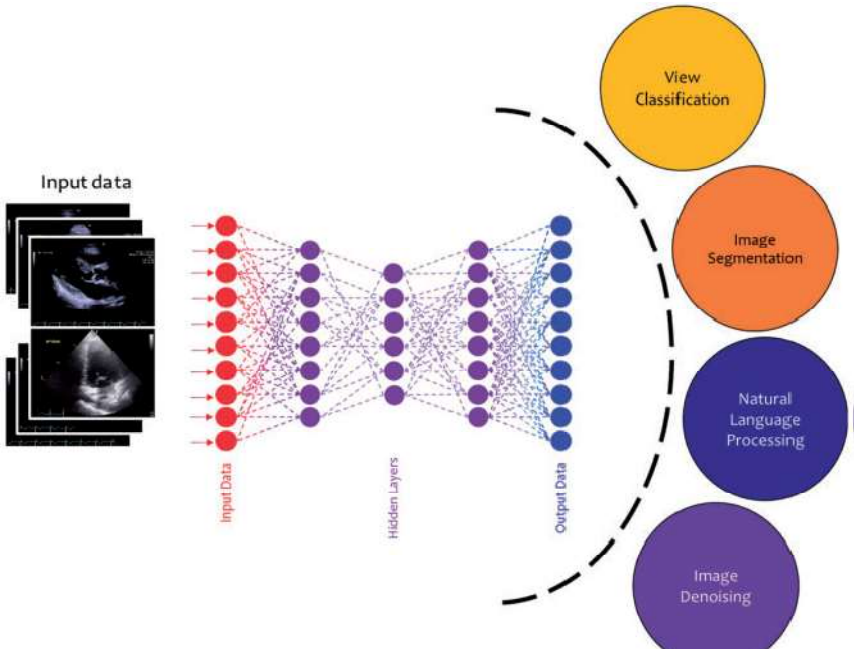

Figura 24 Esquema de la aplicación de diversas técnicas de Aprendizaje Automatizado a ecocardiogramas [19]

3.4.3 Arritmias cardíacas

La predicción de arritmias cardíacas es una de las aplicaciones que más se está extendiendo en el uso de Aprendizaje Automático para cardiología. Existen estudios que abordan modelos predictivos de aparición de fibrilación auricular utilizando Aprendizaje Supervisado con diversos procesos como preprocesado de la señal, extracción de variables significativas y algoritmos de clasificación. También se han publicado aplicaciones que mejoran la gestión de alarmas de telemonitorización [20], en la predicción de la respuesta de procedimientos de ablación invasivos e incluso en la predicción de mortalidad tras una parada cardíaca resucitada [21].

Con Aprendizaje Profundo se ha conseguido con existo la detección de arritmias mediante el análisis de señales electrocardiográficas o la búsqueda de fenotipos para clasificar miocardiopatias hipertróficas con diferente riegos arrítmico utilizando Aprendizaje no Supervisado [22].

3.4.4 Cardiopatía isquémica

Se ha probado a aplicar técnicas de Aprendizaje Automático a los datos habituales de la historia electrónica del paciente para predecir el riesgo de enfermedad cardiovascular en la población general y se ha mostrado superior a las escalas de riesgo utilizadas tradicionalmente [23]. También se ha conseguido aplicar con éxito técnicas de Aprendizaje Automático con Aprendizaje Supervisado para la predicción pronóstico de la cardiopatía isquémica estable, el síndrome coronario agudo o la predicción de mortalidad de enfermos con infarto de miocardio, tanto analizando resultados de hospitales individuales o grandes registros como el SWEDEHEART , un estudio aleatorio de 2.037 pacientes con angina estable o síndrome coronario agudo sometidos a tratamiento intervencionista guiado por análisis automático del flujo coronario. En los estudios con hallazgos dispares se ha logrado que el valor del Aprendizaje Automático mejora con el volumen de la muestra.

3.4.5 Insuficiencia Cardíaca

Con el uso de Aprendizaje Automático se puede optimizar los ingresos hospitalarios evitables por insuficiencia cardíaca identificando de manera más precisa qué pacientes son más susceptibles de descompensación tras el alta hospitalaria. Sin embargo, los resultados fueron contradictorios inicialmente demostrando la importancia del ajuste en la metodología. Hay estudios que abordan el uso de Aprendizaje Automático en los sistemas de gestión de la telemonitorizacion remota de pacientes con insuficiencia cardíaca, con la posibilidad de mejorar la evolución clínica de estos pacientes.

En el área del trasplante cardiaco se ha aplicado Aprendizaje Profundo para predecir la probabilidad de muerte o trasplante de pacientes incluidos en la lista de espera o la predicción de éxito del paciente trasplantado [24].

3.4.6 Calcificaciones Coronarias

El uso de imagen no invasiva se ha convertido en el instrumento utilizado para detectar la presencia de enfermedad coronaria y el diagnostico consecuente. Los marcadores de calcificación coronaria (CAC, Coronary artery calcium) y la tomografía computerizada para la cuantificación del calcio coronario (CCTA) permiten tanto calificar como cuantificar la ateroesclerosis. Se han utilizado algoritmos de Aprendizaje Automático para mejorar la información que se extrae de las imágenes obteniendose con SVM resultados con una precisión del 94% y un AUC de 0.94 para la automatización de la evaluación de estenosis coronaria en CCTA [25].

3.5 Limitación del Aprendizaje Automático en la Cardiología

Las mejoras y aplicaciones que se pueden alcanzar con Aprendizaje Automático en Cardiología no están exentas de limitaciones. El Aprendizaje Automático se basa en la identificación de patrones en conjuntos de datos, lo que se convierte en uno de sus puntos fuertes puesto que actualmente los ordenadores son eficientes en esta tarea pero a su vez es una de sus limitaciones. La cantidad de datos necesaria en algunos casos para hacer que el modelo sea fiable puede ser muy grande y en medicina los sistemas de recogida de datos actualmente no son automáticos o están en fase de implantación. Existen criterios legales y éticos a los que ceñirse en la recogida y uso de esos datos. En el caso concreto de las enfermedades poco prevalentes, la escasez tanto de casos clínicos como de estudios realizados sobre las mismas, constituyen una dificultad seria para la aplicación de Aprendizaje Automático.

Muchas bases de datos médicos están sesgadas, lo que obliga a que los modelos tengan que ajustarse o no serán precisos a la hora de identificar casos de categorías con pocos ejemplos.

La opacidad e interpretabilidad de los modelos de Aprendizaje Automático y sobretodo de Aprendizaje Profundo es también un problema. Los modelos con redes neuronales de muchas capas encuentran patrones complejos entre datos y arrojan una predicción estadística, pero sin explicación biológica. Es lo que se conoce como efecto de “caja negra” y hace que no se pueden obtener la explicación individualizada de cada predicción.

Por último podemos tener en cuenta la validación de aplicaciones basadas en algoritmos de aprendizaje reforzado que van mejorando respecto a datos que van recogiendo con el

tiempo. Establecer unos criterios para que la comunidad médica valide y acepte estas aplicaciones que pueden ir variando en funcionamiento con el tiempo y los casos que se introduzcan pueden causar un problema a la hora de aprobar el uso de estos algoritmos.

Por tanto, aunque el interés en el Aprendizaje Automático para su uso en Cardiología es evidente, no está exento de ciertas dificultades. El campo donde está teniendo una mayor aceptación es en el tratamiento de imagen, que normalmente requiere algoritmos de Aprendizaje Profundo, porque estas aplicaciones suelen generar unos resultados que los cardiólogos pueden identificar y comprobar de manera sencilla.

3.6 Cardiología y Aprendizaje Automático en España

Aún son muy escasas las investigaciones médicas españolas en este campo.

Destaca, como pionero, el Dr. Ignacio Hernández Medrano, neurólogo, fundador en 2014 de SAVANA, empresa tecnológica al servicio de hospitales para procesamiento del Lenguaje Natural médico y su implementación en la práctica clínica. Mediante diversas técnicas de Big Data, entre ellas Aprendizaje Automático, SAVANA permite que los distintos profesionales sanitarios puedan evaluar resultados en salud, conectar todas las fuentes de datos generados en la práctica clínica, reclutar pacientes para ensayos clínicos, predecir sucesos clínicos y tomar decisiones en tiempo real basadas en mejores prácticas. Actualmente SAVANA está en más de 30 hospitales públicos y privados de España y ha sido galardonada con el Premio Fundación Princesa de Girona Empresa 2019 (http://www.savanamed.com).

El Servicio de Cardiología del Hospital Universitario de Salamanca y el Instituto de Investigación Biomédica de Salamanca, también ocupan lugar preferente en la investigación de la aplicación de Aprendizaje Automático en Cardiología:

- Aprendizaje Automático en la predicción de éxito de la Cardioversión eléctrica programada en la fibrilación auricular, cuyos resultados fueron similares a los obtenidos con estadística clásica [26]
- SALMANTICOR: estudio colaborativo con los Centros de Salud de la provincia de Salamanca para detectar la distribución geográfica de la prevalencia de las cardiopatías y, en consecuencia, proponer una distribución más eficaz de los recursos sanitarios [27].

- Revisión sobre Inteligencia Artificial [28].

Otras publicaciones es este área, con aplicación de técnicas de Aprendizaje Automático son:

- Servicio de Cardiología del Hospital de la Princesa de Madrid sobre nuevos marcadores electrocardiográficos para predecir el riesgo de complicaciones cardiovasculares tras cirugía no cardíaca, según una estrategia machine learning, con hallazgo de tres marcadores del ECG que son predicadores independientes de complicaciones cardiovasculares previamente desconocidos [29].
- Estudio multicéntrico en hospitales de Euskadi (Vizcaya y Gupuzcoa) sobre el impacto de los factores medioambientales en las descompensaciones de insuficiencia cardíaca: ingresos previos, inverso de la temperatura , concentración de dióxido de azufre y óxido de nitrógeno presentan una correlación positiva con las descompensaciones [30].
- Estudio multicéntrico de varios departamentos de hospitales de Cataluña, Madrid y Almería que analiza el perfil aterogénico de los lípidos séricos en pacientes con insuficiencia renal crónica [31].
- Estudio colaborativo multicéntrico europeo con participación del Instituto Catalán para la Investigación y la Universidad Pompeu Fabra de Barcelona sobre el análisis de parámetros ecocardiográficos útiles para el diagnóstico de insuficiencia cardíaca con fracción de eyección preservada [32].
- Clasificación de genotipos clínicos cardiovasculares sobre la base de los datos analizados de historias clínicas electrónicas, estudio colaborativo entre Madrid y Barcelona aplicando la metodología ofrecida por SAVANA y que ha sido presentado en el Congreso de la Sociedad Europea de Cardiología celebrado en Paris en septiembre de 2019 [33].

Se puede considerar que es por tanto un campo aún por explotar a nivel nacional y abren la expectativa de desarrollar nuevas líneas de investigación que impliquen a muchos profesionales. En palabras del Dr. Alejandro Recio en una entrevista realizada durante el Congreso Nacional de la Sociedad Española de Cardiología, celebrado en Barcelona en octubre de 2019 *"estamos en un momento clave en lo que a nuevas tecnologías se refiere, y los avances en el campo del Aprendizaje Automático actúan como un*

catalizador **(...)** *el cardiólogo del futuro deberá tener un nuevo perfil añadido al clínico, siendo capaz de interpretar bases de datos y manejar nuevas herramientas que harán posible la utilización de esta tecnología como una 'segunda opinión' en tiempo real"*. La Dra. Vilahur , también entrevistada durante el congreso, considera "*es oportuno y necesario que los programas de formación médica y los sistemas de residencias incorporen a la ciencia informática en sus diferentes ramas, para permitir instituir una nueva generación de profesionales de la salud capaces de manejar las modernas herramientas tecnológicas en materia de inteligencia artificial para el advenimiento de una medicina de precisión basada en una combinación de conocimiento humano y de big data con el fin de progresar en la calidad de atención y resultados médicos*"[34].

Capítulo 4

Desarrollo de un modelo predictivo con Aprendizaje Automático y el estudio SIESTA

En este capítulo se expone el diseño y desarrollo de un modelo predictivo sobre el pronóstico del síndrome coronario agudo a partir de los datos de los pacientes que fueron reclutados para el estudio SIESTA. Se mostrarán todos los algoritmos ensayados, sus métricas de trabajo y el código utilizado.

4.1. El estudio SIESTA

Propuesta

SIESTA es el acróstico de Systemic Inflammation Evaluation in patients with non-ST elevation Acute coronary syndrome, estudio multicéntrico español, publicado en 2010 [35]. Investigaciones precedentes habían relacionado la elevación de la concentración plasmática de proteína C reactiva (PCR) -que es un marcador de inflamación activa- con muerte, infarto de miocardio y necesidad de revascularización miocárdica urgente [36][37][38][39]. Por ello, el objetivo del estudio SIESTA fue comparar el valor predictivo a un año de varios biomarcadores inflamatorios y no inflamatorios en pacientes con síndrome coronario agudo: analizar si la adición de biomarcadores de inflamación a las variables clínicas, electrocardiográficas, angiografías y marcadores de lesión miocárdica considerada en las escalas predictivas TIMI (Thrombolysis In Myocardial Infarction) y PEPA (Proyecto de Estudio Pronóstico de la Angina) mejoraba su valor predictivo.

Diseño

Se reclutaron pacientes atendidos en 25 hospitales españoles entre abril de 2002 y junio de 2004. En el diseño del estudio, y teniendo en cuenta los antecedentes predictivos de PCR, se había estimado un tamaño muestral de 1400 pacientes[40], pero se consiguió incluir finalmente a 610 pacientes.

Los criterios de inclusión fueron pacientes de ambos sexos, sin límite de edad, con dolor de pecho en las últimas 48 horas sugestivo de SCA y una de las siguientes condiciones:

1. Signos electrocardiográficos de isquemia miocárdica (descenso del segmento ST y/o inversión de onda T).
2. Enfermedad coronaria, cerebrovascular o vascular periférica documentada.
3. Angioplastia coronaria transluminal percutánea (ACTP) y/o cirugía de revascularización miocárdica realizada no menos de 12 semanas antes del episodio actual.
4. Elevación de la troponina cardíaca.

Los marcadores de inflamación que se incluyeron en el estudio fueron la proteína C reactiva (PCR), fibrinógeno, neopterina, interleucinas 6, 8, 10 y 18, factor de necrosis tumoral, e-selectina, endotelina 1, factor tisular, molécula de adhesión celular vascular-1 (VCAM-1) e intercelular-1 (ICAM-1), proteína plasmática-A asociada al embarazo (PAPP-A), péptido natriurético tipo B (NTproBNP), troponina I o T, leucocitos e isoforma MB de la creatinfosfocinasa (CK-MB).

Los pacientes dados de alta fueron visitados en la consulta por los médicos investigadores al mes, a los 6 meses y al año.

La variable principal de estudio es una variable compuesta: muerte por cualquier causa, muerte de origen cardíaco, infarto de miocardio no letal y angina que requiera hospitalización, ACTP o cirugía de revascularización urgente. Aunque en el diseño inicial se consideraban variables secundarias las componentes de la variable principal, en el análisis final se consideró variable secundaria la compuesta por muerte por cualquier causa o infarto de miocardio no letal. Los efectos de los niveles de los biomarcadores sobre las variables primaria y secundaria se examinaron mediante modelos de riesgo proporcional de Cox y su valor discriminante con estadístico C.

Resultados

El 90% del total (549 pacientes) completaron el seguimiento anual y durante este periodo un 37,5% sufrieron algún evento de los considerados en la variable principal del estudio.

Ninguno de los marcadores de inflamación incluidos resultó significativo para la predicción de eventos de la variable principal. Sí se encontró correlación entre la combinación de los valores de fibrinógeno y de péptido natriurético NTproBNP con la variable secundaria de estudio, con suficiente significación como para considerarlo un valor predictivo independiente, aunque de escasa potencia.

Los autores del estudio concluían que estos resultados, basados en pacientes de origen mediterráneo y riesgo bajo a moderado, con síndrome coronario agudo, sugieren que los marcadores de inflamación añaden información modesta a la ya aportada por los factores de riesgo convencionales y las variables clínicas incluidas habitualmente en las escalas de riesgo clínico.

4.2 Diseño de un Modelo Predictivo con el estudio SIESTA.

Usando la base de datos del estudio SIESTA se diseña un modelo predictivo con técnicas de Aprendizaje Automático, con el uso de distintos algoritmos con la intención de conseguir una predicción de evento de primer tipo según el estudio [40].

4.2.1 Diseño del modelo.

El acceso a la base de datos del estudio SIESTA ha sido amablemente facilitado por el Dr.D. Luciano Consuegra Sánchez en representación de los autores del mismo, para ser sometidos a un nuevo análisis mediante técnicas de Aprendizaje Automático. El acceso a los datos cuenta con la aprobación del Comité de Ética en la Investigación de la Universidad Politécnica de Cartagena. En el Anexo 3 se incluye la certificación de Informe Favorable.

Se diseñó un modelo cuyo **objetivo principal** es, con el uso de la base de datos recogida por el estudio SIESTA, utilizar los marcadores de inflamación y las variables clínicas recogidas para establecer una predicción de si el paciente sufrirá un evento de los establecidos para la variable principal del estudio: muerte por cualquier causa, muerte de origen cardíaco, infarto de miocardio no letal y angina que requiera hospitalización, ACTP o cirugía de revascularización urgente.

Con este objetivo se diseñó el modelo con las siguientes variaciones respeto al estudio SIESTA inicial:

- El modelo será predictivo en función de las variables de los pacientes al ingreso y al alta, por tanto se eliminarían las variables de seguimiento a lo largo del año. Esto se estableció por dos motivos: si el objetivo es lograr un modelo predictivo en relación a a los datos recogidos durante un ingreso hospitalario y ser capaz en el momento del alta médica de dar una predicción, no se podían tener en cuenta datos recogidos durante un seguimiento de un año. Además el número de datos nulos en el seguimiento anual era lo bastante elevado como para suponer un problema más que una ventaja a la hora de introducirlas en el modelo. Sobre el conflicto que generan los valores nulos y su presencia en la base de datos SIESTA se darán más detalles en los siguientes apartados.
- Solo se evaluaría la capacidad predictiva para la variable objetivo principal, puesto que es la que tiene inicialmente un número suficiente de ejemplos como para poder aplicar algoritmos de Aprendizaje Automático.
- Se eliminarían aquellas variables que proporcionasen información redundante ya recogida por otras variables y aquellas variables creadas por la combinación de otras. También las variables creadas para evaluar el estudio mediante estadística tradicional.
- El modelo sería un **modelo de clasificación binaria** pues la variable objetivo tiene dos categorías ("Evento No" y "Evento Si"). Se utilizarían varios algoritmos de Aprendizaje Automático con el fin de encontrar los mejores resultados de predicción.

Los pasos seguidos para la creación del modelo se pueden ver en la figura 25 y se introducirán a continuación.

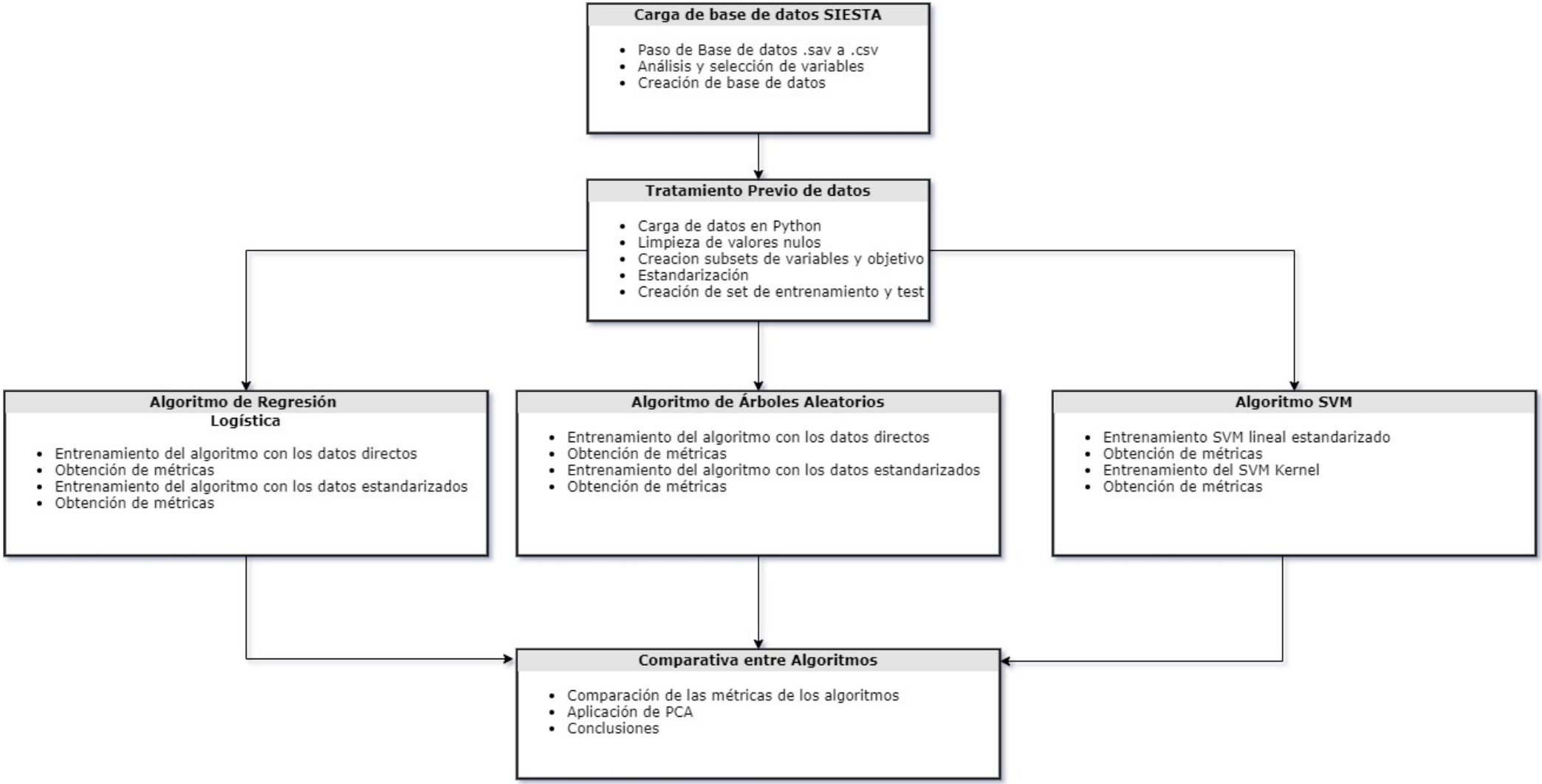

Figura 25 Diagrama de los pasos seguidos para el desarrollo del modelo predictivo

Carga de la base de datos SIESTA

La base de datos del estudio SIESTA, proporcionada para poder realizar el modelo de Aprendizaje Automático, está formada por 556 variables y 610 ejemplos. Entre las variables se encuentra la variable objetivo del modelo así como una variable para cada uno de los eventos que conforman dicha variable objetivo.

La base de datos está en formato .sav, que es el formato para utilizar SPSS (el software que se utilizó para el estudio SIESTA). Como primer paso para la creación del modelo se transformó esta base de datos a formato csv. El término CSV significa "valores separados por comas" (comma-separated values) y es el formato más utilizado para guardar las bases de datos que se utilizan en Aprendizaje Automático y Ciencia de Datos. Esta transformación se realizó utilizando R Studio, un entorno del lenguaje R.

Bajo el análisis y criterio de un cardiólogo se seleccionaron las variables a utilizar para el modelo. Para ello se eliminaron inicialmente todas las variables creadas para poder trabajar con estadística tradicional, como transformaciones logarítmicas de otras variables y variables que proporcionaban información redundante (como ejemplo, de las variables “Fumador”, “Tipo de Fumador” y “Fumador activo” se eliminó la variable “Fumador” que ya se encuentra recogida en las otras dos). Este primer análisis y selección dejó 272 variables.

Fue necesario un segundo análisis para eliminar las variables de datos de seguimiento anual y las variables objetivo que no se iban a analizar. Además se estableció la estrategia a la hora de sustituir los valores nulos existentes en la base de datos. Se eliminaron también aquellos ejemplos que tenían un valor nulo para la variable objetivo (23 ejemplos).

Como resultado de este análisis y selección se obtuvo la base de datos para el modelo: 119 variables clasificadas en distintas categorías como indica la tabla 3 y 587 ejemplos. Estas variables se pueden también clasificar como categóricas (80) y numéricas (40), siendo la variable objetivo categórica: Evento a los 365 días Sí o No.

Tabla 3 Categorías asignadas a las variables

Tipo de variable	Numero de variables
Demográfica	3
Antecedentes	8
Biomarcadores Convencionales	7
Biomarcadores Incluidos	12
Datos Clínicos	9
Datos Bioquímicos	14
Datos de evolución Clínica	9
Datos de tratamiento	50
Factor de riesgo	9
Marcador clínico	2

En el Anexo 2 se incluye una tabla con las variables seleccionadas, sus categoría y tipo, y el número de nulos encontrados para cada una.

Tratamiento previo de los datos

Antes de poder entrenar el modelo es necesario realizar una serie de tareas para poder utilizar el algoritmo seleccionado. La primera de ellas es la limpieza de los valores nulos. No se puede entrenar un algoritmo de Aprendizaje Automático con bases de datos que contengan valores nulos. Estos valores nulos representan generalmente o bien pérdida durante la recolección de los datos o por perdida durante alguna fase de manipulación de los datos. La base de datos para el modelo tenía 3999 valores nulos, recogidos en el Anexo 2. Para sustituir dichos valores nulos se recurrió a la estrategia definida según cada variable en el paso anterior.

También es necesario dividir los datos en dos subsets: el de variables y el de la variable objetivo. El subset de variables está formado por todas las variables excepto la variable objetivo. Este paso ese necesario para trabajar en la mayoría de lenguajes de programación utilizados en Aprendizaje Automático y lo es en el caso concreto del lenguaje utilizado para el desarrollo del modelo.

Como parte del procedimiento para evaluar un modelo también se dividen los datos en otros dos subsets: el subset de entrenamiento y el subset de test [4.8]. De esta manera se evalúa el modelo con unos datos distintos de los utilizados para su entrenamiento, evitando así que el modelo sufra de sobreajuste como se pudo ver en el capítulo 1.

Como último paso previo se realizó una estandarización de los datos siguiendo la formula vista en el capítulo 1:

$$\widehat{x_j} = \frac{x_j - \mu_j}{\sigma_j}$$

La función principal de la normalización es la de incrementar la velocidad de aprendizaje, pues si igualamos el rango de valores de las variables de manera que ninguna esté en un rango mucho mayor que otra, de manera que no domine esta al aplicar los procesos como el error cuadrático medio [3].

Entrenamiento del modelo y obtención métricas de evaluación

Una vez realizados los tratamientos previos de los datos se puede proceder a entrenar el modelo con el algoritmo seleccionado. Se seleccionaron tres algoritmos: Regresión Logística y SVM, vistos en el capítulo 1, y Bosques Aleatorios que es un algoritmo de clasificación basado en Arboles de Decisión. Se dará una explicación más extensa de cada uno de ellos en el apartado correspondiente a su implementación.

Con el modelo ya entrenado con el subset de datos de entrenamiento, se pueden obtener distintas métricas para la evaluación del desempeño del mismo. Las seleccionadas, por ser las más utilizadas como referencia a la hora de evaluar los modelos de clasificación binaria, son la precisión, la Exhaustividad, el Valor-F, la curva ROC y AUC.

La **Precisión** se define según [3] y [41] como el ratio de predicciones positivas correctas del total de predicciones positivas:

$$Precisión = \frac{TP}{TP + FP}$$

La **Exhaustividad** es el ratio de predicciones positivas correctas del total de ejemplos positivos del set:

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

Para evaluar completamente la efectividad de un modelo, debemos examinar la Precisión y la Exhaustividad. Lamentablemente, con frecuencia hay tensión entre Precisión y Exhaustividad. Esto quiere decir que, al mejorar la precisión, generalmente se reduce la Exhaustividad, y viceversa.

El **Valor-F** nos da un valor ponderado entre la Precisión y la Exhaustividad:

$$Valor - F = 2x \frac{Precisión * Exhaustividad}{Precisión + Exhaustividad}$$

Por último la **curva ROC** se realiza mediante una combinación del ratio de predicciones positivas correctas (la Exhaustividad) y el ratio de predicciones positivas incorrectas para realizar una curva que muestre el desempeño del modelo. Solo se puede emplear con algoritmos que devuelvan la probabilidad de predicción.

Cuanto mayor sea el **Área Bajo la Curva** (**AUC**), mejor desempeño tiene el clasificador. Si el AUC es mayor de 0.5 podemos decir que el clasificador es mayor que un clasificador aleatorio. Se pueden observar ejemplos de curvas ROC y AUC en la figura 26.

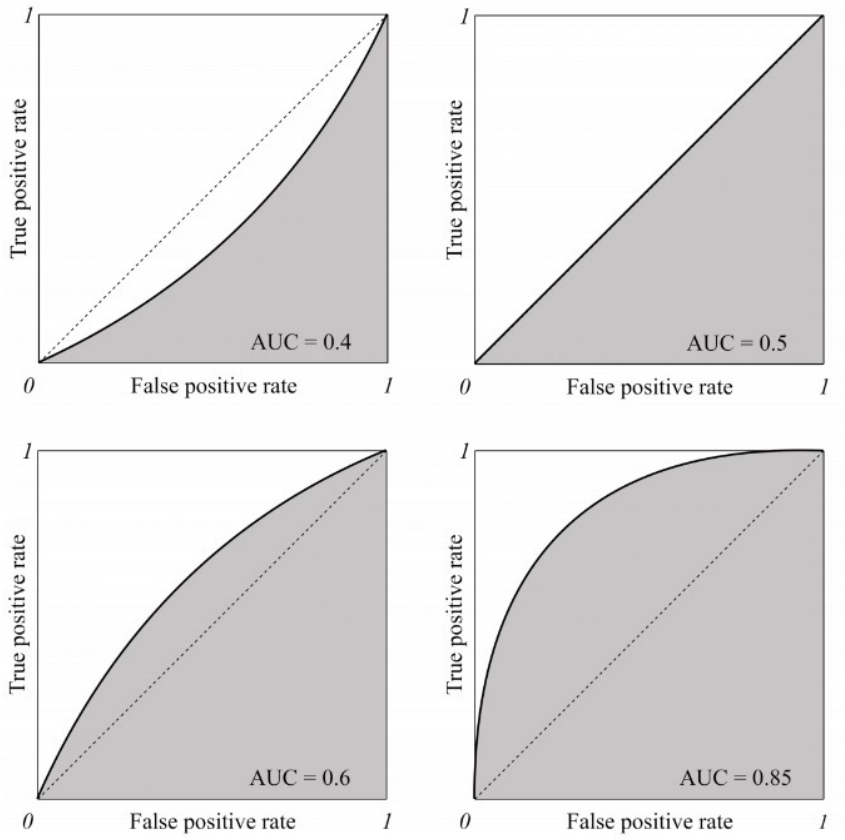

Figura 26 Con la curva ROC se puede obtener de manera gráfica una medida del desempeño del modelo [3]

4.2.2 Lenguaje de Programación utilizado.

La aproximación inicial durante el diseño del modelo contemplaba el uso de Octave como lenguaje y herramienta software a utilizar debido a que la formación previa a la realización de este Trabajo Fin de Grado se realizó con este lenguaje [2]. Sin embargo durante las primeras fases de desarrollo del modelo se decidió el cambio a Python. Los motivos principales son que a pesar del conocimiento previo de Matlab y su lenguaje, y su extrapolación a Octave, el trabajo con Python en el campo del Aprendizaje Automático es mucho más sencillo. Existen muchas más referencias bibliográficas ([3], [41], [42] entre otras muchas) y en definitiva mas información accesible de todos los ámbitos [4].

Se usó la distribución Anaconda de Python que incluye distintas opciones de desarrollo para Python y R así como una serie de paquetes y librerías ya instalados como muestra las figura 27 y 28, y que se utiliza especialmente para Ciencia de Datos y Aprendizaje

Automático, de distribución libre y abierta, y que se puede descargar directamente de su página web (https://www.anaconda.com/).

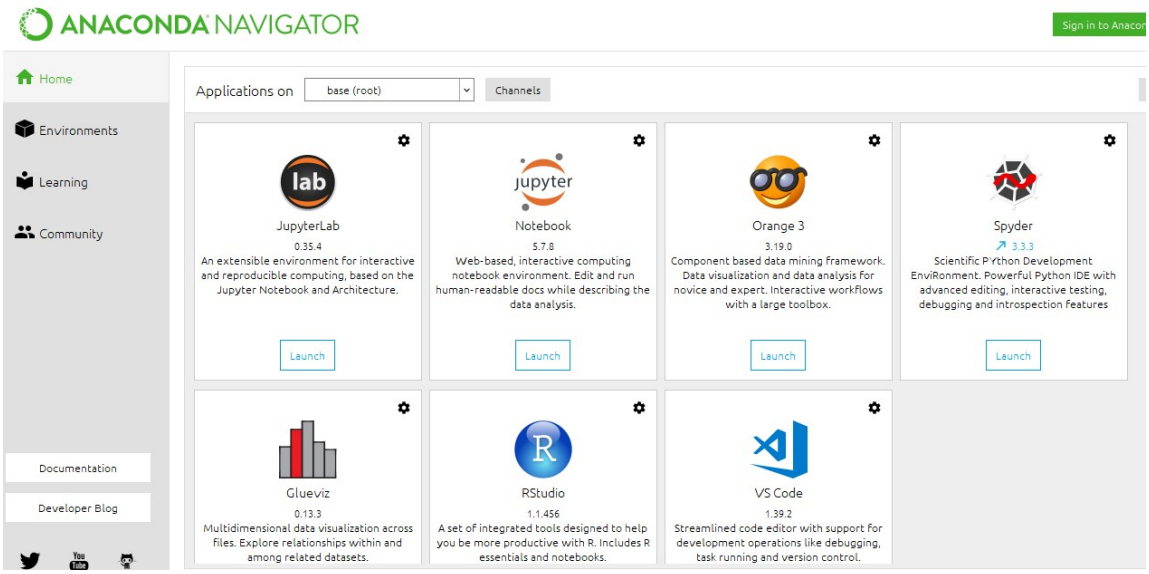

Figura 27 Interfaz de inicio de Anaconda con distintos entornos para Python y R

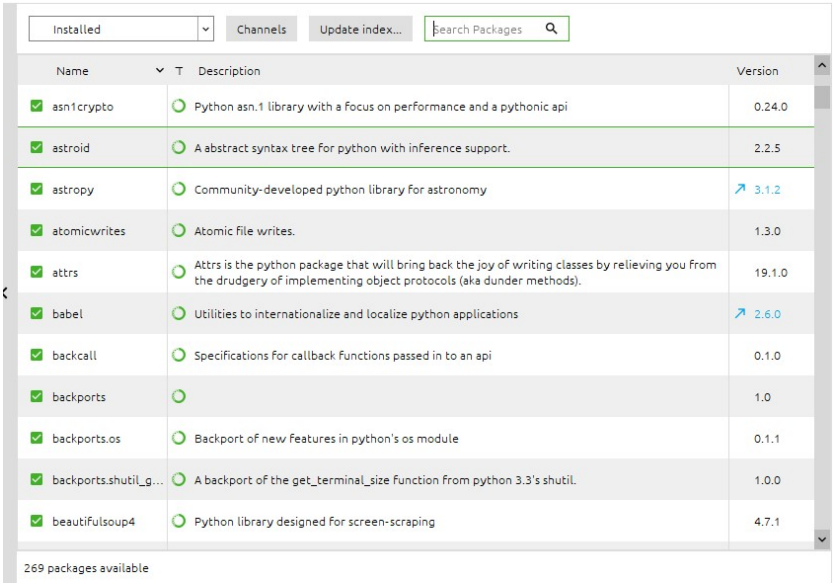

Figura 28 Interfaz de instalación de paquetes de Anaconda

Se utilizaron los algoritmos de Aprendizaje Automático implementados en Scikit-Learn, además de utilizar con frecuencia las librerías Pandas, Numpy, matplotlib y seaborn que ya se explicaron en el bloque 2. Se usó principalmente el entorno Jupyter notebook como entorno para el desarrollo del código del modelo, aunque es igualmente aplicable en cualquier entorno Python.

4.3 Implementación del modelo predictivo con Python

Para la implantación del modelo en Python se siguieron los siguientes pasos:

1. Carga de la base de datos y procesado previo.
2. Selección del algoritmo de Aprendizaje Automático con el que entrenar el modelo.
3. Entrenamiento del modelo con distintas técnicas para el tratamiento de los datos.
4. Métricas del modelo

Todos los modelos se caracterizaron con variaciones en sus hiperparámetros para intentar conseguir las mejores métricas de evaluación del modelo.

4.3.1 Carga de la base de datos SIESTA y procesado previo de los datos.

El trabajo previo al entrenamiento de cualquier modelo, como se ha visto anteriormente, incluye la carga de los datos que se van a utilizar como ejemplos para el entrenamiento del modelo además del procesado previo de estos datos para eliminar datos nulos o normalizar los valores.

En el caso del modelo predictivo que se iba a desarrollar se realizó previamente un trabajo de definir cómo sustituir los valores nulos de cada una de las variables. Para ello se recurrió a la clase **SimpleImputer**[1] de Scikit-Learn que permite sustituir datos seleccionando la estrategia a seguir entre media, moda, mediana o una constante. En el caso del modelo que se quiere desarrollar, fue necesario aplicar los 4 tipos de estrategia.

```
import pandas as pd
from sklearn.impute import SimpleImputer
si = SimpleImputer()
si_median = SimpleImputer(strategy='median')
si_moda = SimpleImputer(strategy='most_frequent')
## Lectura Base de datos
df = pd.read_csv('SIESTA_MODELO1.csv',sep=';')
df.head()
## Limpiar la base datos
df2=df
### Eliminamos TIMI nulos
```

1 https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

```
df2=df.dropna(subset=['TIMI'])
### Valores a sustituir por la media
si.fit(df2['PAD'].values.reshape(-1, 1))
df2["PAD"] = si.fit_transform(df2["PAD"].values.reshape(-1, 1))
si.fit(df2['PAS'].values.reshape(-1, 1))
df2["PAS"] = si.fit_transform(df2["PAS"].values.reshape(-1, 1))
si.fit(df2["Latidos"].values.reshape(-1, 1))
df2["Latidos"] = si.fit_transform(df2["Latidos"].values.reshape(-1, 1))
si.fit(df2["IMC"].values.reshape(-1, 1))
df2["IMC"] = si.fit_transform(df2["IMC"].values.reshape(-1, 1))
si.fit(df2["Temperatura"].values.reshape(-1, 1))
df2["Temperatura"] = si.fit_transform(df2["Temperatura"].values.reshape(-1,
1))
si.fit(df2["CPKingreso"].values.reshape(-1, 1))
df2["CPKingreso"] = si.fit_transform(df2["CPKingreso"].values.reshape(-1, 1))
si.fit(df2["CPK_MBingreso"].values.reshape(-1, 1))
df2["CPK_MBingreso"] = si.fit_transform(df2["CPK_MBingreso"].values.reshape(-
1, 1))
si.fit(df2["creatininaingreso"].values.reshape(-1, 1))
df2["creatininaingreso"] =
si.fit_transform(df2["creatininaingreso"].values.reshape(-1, 1))
si.fit(df2["hgbingreso"].values.reshape(-1, 1))
df2["hgbingreso"] = si.fit_transform(df2["hgbingreso"].values.reshape(-1, 1))
### Valores a sustituir por mediana
si_median.fit(df2['COLTOTALalta'].values.reshape(-1, 1))
df2['COLTOTALalta'] =
si_median.fit_transform(df2['COLTOTALalta'].values.reshape(-1, 1))
si_median.fit(df2['HDLCOLalta'].values.reshape(-1, 1))
df2['HDLCOLalta'] = si_median.fit_transform(df2['HDLCOLalta'].values.reshape(-
1, 1))
si_median.fit(df2['LDLCOLalta'].values.reshape(-1, 1))
df2['LDLCOLalta'] = si_median.fit_transform(df2['LDLCOLalta'].values.reshape(-
1, 1))
si_median.fit(df2['TRIGLICERalta'].values.reshape(-1, 1))
df2['TRIGLICERalta'] =
si_median.fit_transform(df2['TRIGLICERalta'].values.reshape(-1, 1))
si_median.fit(df2['CPKalta'].values.reshape(-1, 1))
df2['CPKalta'] = si_median.fit_transform(df2['CPKalta'].values.reshape(-1, 1))
si_median.fit(df2['CPKMBalta'].values.reshape(-1, 1))
df2['CPKMBalta'] = si_median.fit_transform(df2['CPKMBalta'].values.reshape(-1,
1))
si_median.fit(df2['TROPONalta'].values.reshape(-1, 1))
df2['TROPONalta'] = si_median.fit_transform(df2['TROPONalta'].values.reshape(-
1, 1))
si_median.fit(df2['CREATININAalta'].values.reshape(-1, 1))
```

```
df2['CREATININAalta'] =
si_median.fit_transform(df2['CREATININAalta'].values.reshape(-1, 1))
si_median.fit(df2['LEUCOalta'].values.reshape(-1, 1))
df2['LEUCOalta'] = si_median.fit_transform(df2['LEUCOalta'].values.reshape(-1,
1))
si_median.fit(df2['HGBalta'].values.reshape(-1, 1))
df2['HGBalta'] = si_median.fit_transform(df2['HGBalta'].values.reshape(-1, 1))
si_median.fit(df2['NEOPT_ING'].values.reshape(-1, 1))
df2['NEOPT_ING'] = si_median.fit_transform(df2['NEOPT_ING'].values.reshape(-1,
1))
si_median.fit(df2['IL6_ING'].values.reshape(-1, 1))
df2['IL6_ING'] = si_median.fit_transform(df2['IL6_ING'].values.reshape(-1, 1))
si_median.fit(df2['IL10_ING'].values.reshape(-1, 1))
df2['IL10_ING'] = si_median.fit_transform(df2['IL10_ING'].values.reshape(-1,
1))
si_median.fit(df2['P_SEL_ING'].values.reshape(-1, 1))
df2['P_SEL_ING'] = si_median.fit_transform(df2['P_SEL_ING'].values.reshape(-1,
1))
si_median.fit(df2['Cistatinaing'].values.reshape(-1, 1))
df2['Cistatinaing'] =
si_median.fit_transform(df2['Cistatinaing'].values.reshape(-1, 1))
si_median.fit(df2['Fibrinoging'].values.reshape(-1, 1))
df2['Fibrinoging'] =
si_median.fit_transform(df2['Fibrinoging'].values.reshape(-1, 1))
### Valores a sustituir por Moda
si_moda.fit(df2['HEPARINAingreso'].values.reshape(-1, 1))
df2['HEPARINAingreso'] =
si_moda.fit_transform(df2['HEPARINAingreso'].values.reshape(-1, 1))
si_moda.fit(df2['NITRATOS_INGRESO'].values.reshape(-1, 1))
df2['NITRATOS_INGRESO'] =
si_moda.fit_transform(df2['NITRATOS_INGRESO'].values.reshape(-1, 1))
si_moda.fit(df2['LDLIngreso'].values.reshape(-1, 1))
df2['LDLIngreso'] = si_moda.fit_transform(df2['LDLIngreso'].values.reshape(-1,
1))
### Valores a sustituir por 1/ Resto sustituir por 0
df2['Tipofumad']=df2['Tipofumad'].fillna(1)
df2=df2.fillna(0)
### Guardamos la base de datos sin valores nulos
df2.to_csv('DatosLimpios.csv',index=False)
```

Con esto la base de datos ya se encuentra en disposición de poder trabajar con ella para la creación del modelo. Para ello se crean los subconjuntos de variables y variable

objetivo, formado el primero por todas la variables del set de datos excepto la variable objetivo y el segundo formado únicamente por los valores de la variable objetivo.

```
print(df2.groupby('endpoinTOTALa360dias').size())
OUT: endpoinTOTALa360dias
0    388
1    199
dtype: int64
## Creo los subconjuntos X e Y
y = df2['endpoinTOTALa360dias']
X = df2.drop(['endpoinTOTALa360dias'],axis=1)
X.shape
```

Después de la limpieza se tienen por tanto 119 variables con 587 ejemplos, y la variable objetivo tiene 388 ejemplos con "Evento No" y 199 ejemplos con "Evento Si".

El siguiente paso es la creación de los datos en subconjuntos de entrenamiento y validación, siguiendo lo indicado en el libro [41]. Estos conjuntos se utilizarán para entrenar con un conjunto distinto del que luego usaremos para el cálculo de las métricas de validación del modelo, asegurando así que la validación de nuestro modelo no está afectada por sobreajuste. También se realiza una estandarización de los datos conforme a lo visto en el capítulo 1. Esta estandarización será la principal estrategia a seguir para impedir el sobreajuste del modelo, además de ser una estrategia recomendada para reducir los tiempos de procesado durante el entrenamiento del modelo [3].

Para ambos procesos también se utilizaron clases de Scikit-Learn [41]: **model_selection.train_test_split** [2] para el caso de la creación de sets de entrenamiento/evaluación y **StandardScaler** [3] para la regularización con media y desviación de los sets de datos.

```
## Separo en entrenamiento/validación
from sklearn import model_selection
validation_size = 0.30
seed = 42
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X,
y, test_size=validation_size, random_state=seed)
## Regularizacion de los datos
```

2 https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

3 https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(X_train)
X_train_std=sc.transform(X_train)
X_test_std=sc.transform(X_test)
```

Con esto ya se ha efectuado el tratamiento previo de los datos necesario para poder entrenar los distintos algoritmos de Aprendizaje Automático con su implementación en Scikit-Learn siguiendo las recomendaciones disponibles en [4.8].

4.3.2 Entrenamiento y métricas de algoritmo de Regresión Logística

El primer algoritmo que se implementó fue el algoritmo de Regresión Logística. Este algoritmo, del que ya hablamos en el capítulo 1, es uno de los más utilizados para clasificación. Está basado en la función sigmoidea [41]:

$$f_{w,b}(x) \stackrel{\text{def}}{=} \frac{1}{1+e^{-(wx+b)}}$$

Una de las ventajas de este algoritmo frente a otros algoritmos de clasificación como los perceptrones es que con Regresión Logística se obtiene la probabilidad asignada a cada predicción, por tanto proporciona no solo la etiqueta de la predicción sino el porcentaje que ofrece sobre esta.

Realizaremos el entrenamiento del modelo, primero sin estandarizar, mediante su implementación en Scikit-Learn **LogisticRegression**. El hiperparámetro C es un término inverso del parámetro λ que establece una penalización para el sobreajuste [41]. Se denomina C en Scikit-Learn por una convención con el algoritmo SVM.

```
## Entreno modelo Regresión Logística sin Regularizar
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(C=100.0, random_state=1)
model.fit(X_train,Y_train)
prediccion_test = model.predict(X_test)
### Metricas:
* Precisión: Ratio entre predicciones positivas correctas y el total
de positivos predecidos (TP/(TP+FP)
* Recall: Ratio entre predicciones positivas y el total de casos
positivos del dataset(TP/(TP+FN))
```

```
* f1 : Media armónica de precisión y recall
metrics.precision_score(Y_test, prediccion_test)
OUT: 0.48
metrics.recall_score(Y_test, prediccion_test)
OUT: 0.1875
metrics.f1_score(Y_test, prediccion_test)
OUT:0.2696
```

La obtención de las métricas de evaluación del modelo se realiza mediante la clase **metrics** que proporciona la Precisión, Exhaustividad y Valor-F. Se obtienen valores muy pobres de las 3 métricas para este algoritmo destacando el valor extremadamente bajo de Exhaustividad (0.1875).

El siguiente paso es probar el entrenamiento del modelo con los datos estandarizados, para así comprobar si este proceso previo mejora las métricas obtenidas.

```
## Entreno el modelo regularizado
modelR = LogisticRegression(C=100, random_state=1)
modelR.fit(X_train_std,Y_train)
prediccion_test_R = modelR.predict(X_test_std)
probabilidades_test_R=modelR.predict_proba(X_test_std)
metrics.precision_score(Y_test, prediccion_test_R)
OUT:0.41
metrics.recall_score(Y_test, prediccion_test_R)
OUT:0.3125
metrics.f1_score(Y_test, prediccion_test_R)
OUT:0.3539
```

Aunque la Exhaustividad de la clasificación mejora con respecto al caso sin estandarizar (pasa de 0.19 a 0.31), no podemos considerar que el valor medio de las métricas sean lo suficientemente buenas como para afirmar que su capacidad predictiva sea suficiente. La mejora no es los bastante alta, aunque se puede observar que la regularización ha contribuido a mejorar los resultados.

Como ultima métrica se recurre a la ROC y la AUC que mostramos en la figura 29.

```
### ROC y AUC
from sklearn.metrics import roc_auc_score
preds = probabilidades_test_R[:,1]
fpr, tpr, threshold = metrics.roc_curve(Y_test, preds)
```

```
roc_auc = metrics.auc(fpr, tpr)
# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

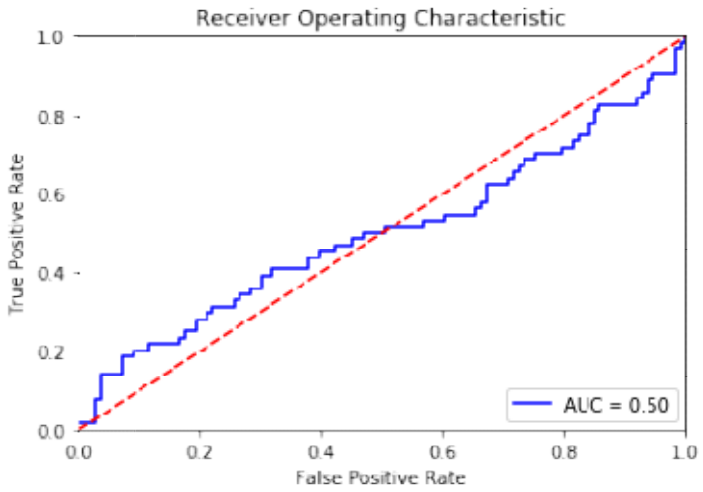

Figura 29 Curva ROC y AUC del modelo de Regresión Logística, con una AUC de 0.5 con lo no podemos considerar el modelo como un buen modelo a la hora de realizar predicciones.

4.3.3 Entrenamiento y métricas de algoritmo de Máquinas de Soporte Virtual (SVM)

El siguiente algoritmo que se entrenó para la caracterización del modelo es SVM. En este caso se parte de los datos previamente procesados y se comenzó con un SVM lineal. Este tipo de algoritmo SVM encuentra una separación lineal entre las variables para establecer la frontera con el margen más grande de entre los posibles.

Se utilizó la implementación de Scikit-Learn, **SVC**[4]. La selección del tipo de SVC a emplear se realiza mediante el atributo "kernel".

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, Y_train)
y_pred = svclassifier.predict(X_test)
print(confusion_matrix(Y_test,y_pred))
print(classification_report(Y_test,y_pred))
```

Este algoritmo no fue capaz de completar su entrenamiento, tras un tiempo. de ejecución de 2 horas por lo que se entiende que no hay convergencia del algoritmo SVM lineal.

Descartando el SVM lineal se pasó a implementar SVM con kernels. Esta es la versión del algoritmo de SVM que implementa un aumento de dimensión en los datos antes de encontrar una frontera de decisión [3]. Un conjunto de datos que no es linealmente separable en una dimensión puede volverse separable en una dimensión superior.

En este caso se cambiará el atributo "kernel" a "sigmoid" para establecer este tipo de algoritmo. El hiperparámetro "gamma" controla el comportamiento del kernel(entendida como la hipotética esfera que establece la frontera de decisión), cuando es muy pequeño, el modelo final es equivalente al obtenido con un SVM lineal, a medida que aumenta su valor, también lo hace la flexibilidad del modelo [41]. El atributo "probability" permite la obtención de estimación de probabilidad para las predicciones, necesario para obtener la curva ROC.

```
svclassifier = SVC(kernel='sigmoid', gamma=0.1,
C=10,probability=True)
svclassifier.fit(X_train_std, Y_train)
y_pred = svclassifier.predict(X_test_std)
y_prob = svclassifier.predict_proba(X_test_std)
metrics.precision_score(Y_test, y_pred)
OUT: 0.34
metrics.recall_score(Y_test, y_pred)
```

4 https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

```
OUT:0.31
metrics.f1_score(Y_test, y_pred)
OUT:0.3272
```

Tampoco se pueden considerar lo suficientemente buenas como para considerar que el modelo pueda utilizarse para predecir la variable objetivo. En el caso de la curva ROC es aun peor que en Regresión Logística, con una AUC de 0.46 frente a la AUC de 0.5 obtenida anteriormente como muestra la figura 30.

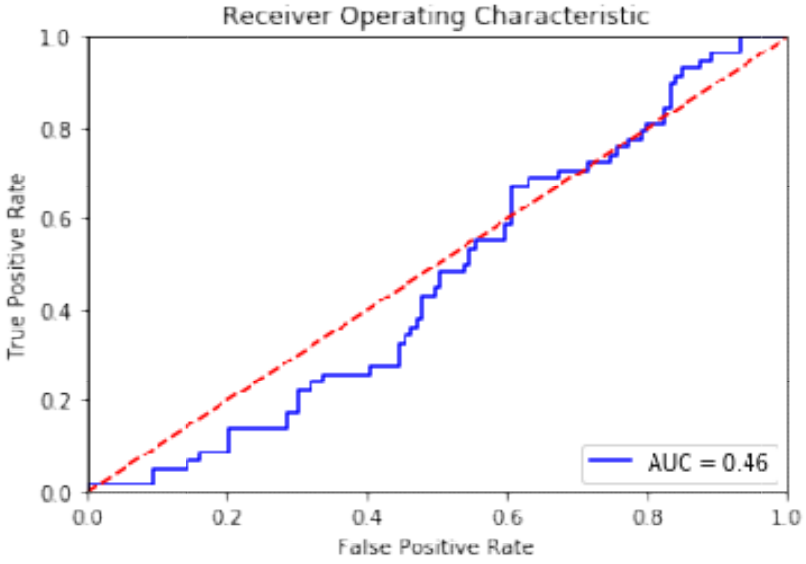

Figura 30 Curva ROC y AUC del modelo de SVM, con un valor de AUC de 0.46, no óptimo.

4.3.3 Entrenamiento y métricas de algoritmo de Bosques Aleatorios.

El ultimo algoritmo probado fue el de Bosques Aleatorios (Random forest) que no se ha explicado con anterioridad. Es un algoritmo de clasificación supervisado que, como su nombre sugiere, crea "bosques" con varios árboles de decisión.

En general, cuantos más árboles haya en el bosque, más robusto será el bosque. Del mismo modo, en el clasificador aleatorio de bosques, cuanto mayor sea el número de árboles en el bosque, mayor será la precisión. Una vez calculado cada árbol de decisión, el resultado de cada uno de ellos se promedia y con este se obtiene la predicción del problema. Como ventajas, suele ser un algoritmo de clasificación con resultados bastante buenos, es capaz de manejar gran cantidad de datos, identifica las variables más significativas y es capaz de mostrarlas. Como desventajas, es uno de los algoritmos

que produce el efecto de caja negra, pues se tiene poco control sobre lo que hace el modelo y sobreajusta los conjuntos de datos que tienen demasiado ruido.

Se entrena el algoritmo de Bosques Aleatorios con el modelo entandarizado, mediante su implementación en Scikit-Learn, **RandomForestClassifier**[5]. Nos permite mediante el parámetro "n_estimators" seleccionar el número de árboles de decisión que creará. Obtenemos sus métricas y su curva ROC en la figura 31.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
forest = RandomForestClassifier(n_jobs=2, oob_score=True,
n_estimators=1000)
forest.fit(X_train_std,Y_train)
prediccion_test_ST = forest.predict(X_test_std)
probabilidades_test_ST=forest.predict_proba(X_test_std)
metrics.precision_score(Y_test, prediccion_test_ST)
OUT:0.5747
metrics.recall_score(Y_test, prediccion_test_ST)
OUT:0.0689
metrics.f1_score(Y_test, prediccion_test_ST)
OUT:0.1231
```

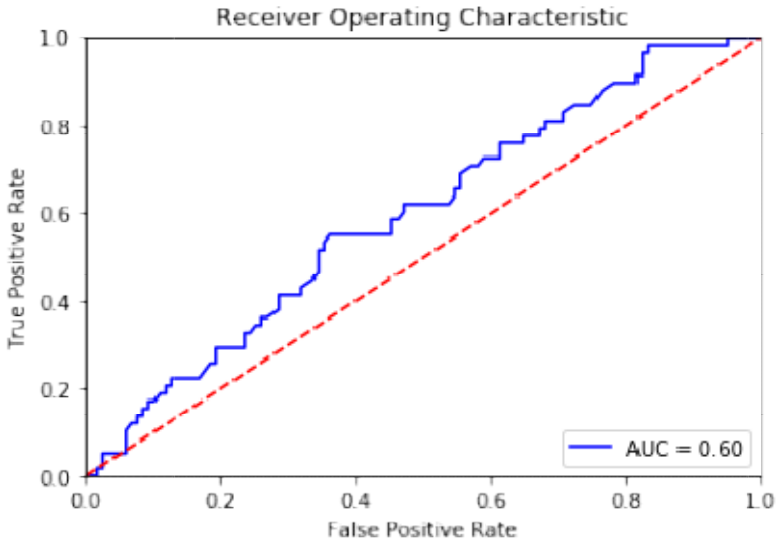

Figura 31 Curva ROC y AUC del modelo de Bosques Aleatorios, ligeramente superior a las vistas hasta el momento con una AUC de 0.6

Aunque el aumento en la Precisión es considerable con respecto a el obtenido hasta el momento (0.57), el valor de Exhaustividad es muy bajo (0.0689). E modelo falla mucho

5 https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

prediciendo los casos de “Evento Sí”, aunque los casos de “Evento No” predichos correctamente mejoran la precisión del modelo. Esto es así porque nuestra variable objetivo está sesgada, es decir, tiene más ejemplos de “Evento No” que de “Evento Si”

Una de las ventajas de los bosques aleatorios es que nos permite obtener la importancia que se otorga a cada una de las variables, según los datos que extrae el propio algoritmo durante su entrenamiento.

```
feature_list = list(X.columns)
# Get numerical feature importances
importances = list(forest.feature_importances_)
# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for feature,
importance in zip(feature_list, importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x:
x[1], reverse = True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
feature_importances];
```

La salida que proporciona este código no se va a reproducir completa por la extensión, al dar el valor de importancia de cada una de las 120 variables del modelo. Pero son destacables dos aspectos: más del cincuenta por ciento de las variables tenían valores de importancia por debajo de 0.001 y no existía ninguna variable que el algoritmo identificase como con un valor de importancia significativamente mayor que el resto.

4.3.4 Aplicación de Análisis de Componentes Principales a los algoritmos.

Como último paso antes e intento de lograr una caracterización del modelo que tuviese unos valores de métrica suficientemente buenos como para poder considerarlo un modelo de predicción efectivo, se aplico la técnica de Análisis de Componentes Principales (PCA) a los algoritmos de Regresión Logística y Bosques aleatorios.

PCA es un algoritmo de Aprendizaje no Supervisado de reducción de dimensionalidad. Identifica patrones en la correlación entre las variables de entrada del algoritmo.

Encuentra la dirección de máxima varianza en el espacio formado por los datos de las variables de entrada y proyecta estas en un espacio de dimensión inferior [41].

Para ello se siguió la siguiente implementación del algoritmo en Scikit-Learn, **PCA**[6]. Con esta clase primero se ajusta el algoritmo con el conjunto de datos de entrenamiento para después transformar el set de entrenamiento y de test.

```
from sklearn.decomposition import PCA
pca = PCA(0.55)
pca.fit(X_train_std)
X_train_PCA= pca.transform(X_train_std)
X_test_PCA = pca.transform(X_test_std)
```

Con ello se consiguieron las siguientes ROC y AUC en el modelo de Regresión Logística con los datos estandarizados (Figura 32) y Bosques Aleatorios (Figura 33).

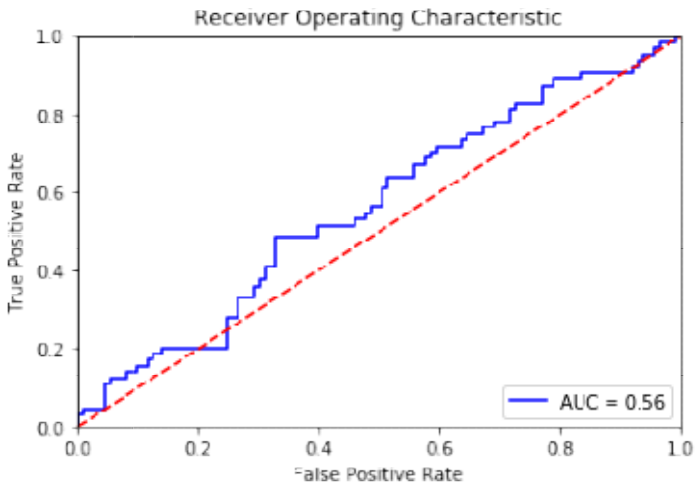

Figura 32 Curva ROC y AUC del modelo de Regresión Logística tras aplicar PCA, con un aumento de AUC de 0.5 a 0.56

6 https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

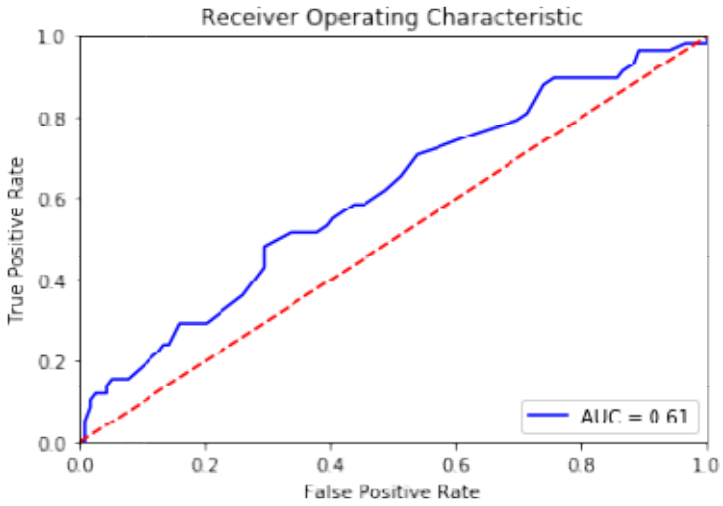

Figura 33 Curva ROC y AUC del modelo de Bosques Aleatorios tras aplicar PCA, con una mejora de 0.60 a 0.61

En ambos casos se consiguieron mejoras en las métricas, pero el modelo seguía siendo incapaz de predecir con unos valores aceptables de Precisión, Exhaustividad y Valor-F los casos de "Evento Si", recogidos en la tabla 4.

4.4 Comparativa de las métricas de los algoritmos

En la tabla 4 se recogen todas las métricas obtenidas al entrenar el modelo con los distintos algoritmos, utilizando los datos con o sin estandarizar y las métricas obtenidas tras la utilización de PCA.

Tabla 4 Métricas del modelo con distintos algoritmos

		Precisión	Exhaustividad	Valor-F
Regresión Logística	Datos originales	0.48	0.1875	0.2696
	Datos estandarizados	0.41	0.3125	0.3539
	PCA	0.45	0.14	0.2143
SVM	SVM Lineal	N/A	N/A	N/A
	SVM Kernel	0.3461	0.31	0.3271
Bosques Aleatorios	Datos originales	0.375	0.0517	0.0909
	Datos estandarizados	0.57	0.0689	0.123
	PCA	0.4782	0.1896	0.2716

En el caso del modelo diseñado y teniendo en cuenta el objetivo principal del diseño, se debe elegir el algoritmo con mayor **Exhaustividad**. Esta métrica nos da el ratio de predicciones positivas correctas del total de ejemplos positivos del set. Se pretende que

sea más eficiente detectando casos de “Evento Sí”, porque sería un marcador de riesgo para establecer por ejemplo un mayor seguimiento al paciente. Si hubiese que elegir el algoritmo que mejor desempeño tiene, a la vista de los resultados, debería de ser el algoritmo de Regresión Logística con los datos estandarizados. Su Exhaustividad y Valor-F son los más altos.

4.5 Conclusiones y limitaciones

La aplicación de tres algoritmos sobre las variables de entrada (biomarcadores de inflamación y datos clínicos) no son capaces de predecir la probabilidad de sufrir eventos de la variable objetivo principal.

Este resultado negativo es concordante con el hallado en el estudio SIESTA. La aplicación de algoritmos de Aprendizaje Automático no modifica la falta de predicción obtenida mediante pruebas estadísticas convencionales. Esto puede ser debido, entre otros factores, a:

- Un número insuficiente de ejemplos.
- Unas variables que no aportan suficiente información pues el algoritmo de Bosques Aleatorios indicaba que muchas de ellas no tenían ningún peso en la predicción.
- Una variable objetivo demasiado general. Al introducir “Muerte por cualquier causa” entre los eventos que daban lugar a un valor positivo para la variable objetivo del modelo se podían estar introduciendo ejemplos cuyo valor de variable objetivo no estuviese relacionado con las variables de entrada. El set de datos puede tener mucho ruido, valores dispares que empeoran enormemente las métricas.

Hay que tener en cuenta que la aplicación de Aprendizaje Automático a un diseño de estudio que no estaba pensado para ello también influye en su efectividad. Con un modelo plenamente creado para ser desarrollado con Aprendizaje Automático se podrían haber conseguido unos mejores resultados.

A pesar del resultado negativo tanto del Estudio SIESTA original como del modelo predictivo que se ha intentado desarrollar con técnicas de Aprendizaje Automático, la búsqueda de biomarcadores que mejoren el diagnóstico y la predicción pronóstica del

síndrome coronario agudo no ha perdido su interés. Por tanto una posible revisión del diseño del modelo junto a un incremento significativo en el número de ejemplos podría mejorar su capacidad de predicción.

Capítulo 5

Conclusiones

El Aprendizaje Automático, englobado en la Inteligencia Artificial, es un conjunto de técnicas que se encargan de construir algoritmos cuya utilidad se basa en una serie de datos ejemplo de un fenómeno. Estos ejemplos pueden venir de la naturaleza, creados por una persona o generados por otro algoritmos.

Cualquier actividad humana genera actualmente una ingente cantidad de datos. Esto, junto con las mejora en el procesamiento por ordenador, ha contribuido al auge del Aprendizaje Automático y su aplicación. Existen multitud de utilidades software con las que afrontar el diseño e implementación de modelos.

La medicina ha encontrado en el Aprendizaje una nueva vía para el análisis de la exhaustiva información que generan los procesos asistenciales y de investigación. En los próximos años podría llegar a incluirse en los programas de formación del médico y especialmente del investigador.

En el caso concreto de la Cardiología, que depende del análisis de datos de diversas fuentes, se está mostrando un interés progresivo en las técnicas de Aprendizaje Automático como herramienta de mejora diagnóstica, predictiva y de investigación. Existen ya numerosas publicaciones en todas sus subespecialidades, destacando su desarrollo en el campo del diagnóstico por imagen.

Algunos grupos de investigación cardiológica en España inician tímidamente su andadura en estas técnicas en los últimos 5 años, habiéndose despertado interés en su aportación a la práctica clínica.

De la experiencia obtenida al intentar aplicar técnicas de Aprendizaje Automático a una base de datos utilizada en investigación previa realizada con técnicas convencionales, se extrae como consecuencia principal que se requiere un diseño especifico para su aplicación.

Como trabajos futuros, el abanico es muy amplio. Estas técnicas, al ser empleadas para el análisis de gran cantidad de datos, posiblemente creen nuevas hipótesis de

investigación. La asociación estadísticamente significativa entre variables debe apoyarse en una explicación biológica para que tenga validez en medicina.

En cuanto a su implantación, en palabras de Alexander Zlotnik, Jefe de servicio de la Subdireccion General de Tecnologías de la Información del Ministerio de Sanidad, Consumo y Bienestar Social [43]: "*... en el sector sanitario público de España existen numerosos proyectos de investigación que emplean técnicas de inteligencia artificial - muchos de ellos de altísima calidad, reconocidos con publicaciones en revistas científicas internacionales de máximo nivel-, pero existe una falta de traslación de dichos resultados de investigación a la práctica clínica diaria. El Sistema Nacional de Salud de España no puede quedarse atrás en la evolución tecnológica, incluyendo la inteligencia artificial, pero creo que debe adoptar las nuevas tecnologías en general de manera metódica y coordinada.*"

Referencias

[1]Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* New York, USA: Basic Books.

[2] Ng, A. (2019). *Machine Learning Coursera*.://www.coursera.org/learn/machine-learning

[3] Burkov, A. (2019). *The Hundred-page Machine Learning Book*. Wroclaw, Polonia: Andriy Burkov.

[4] Gomila Salas, J. G. *Curso completo de Machine Learning: Data Science en Python* .https://www.udemy.com/course/machinelearningpython/

[5] Web Python. https://docs.python.org/

[6]Web Anaconda.https://www.anaconda.com/

[7] Martínez Heras, J. (2019, 5 febrero). *Librerías de Python para Machine Learning*https://iartificial.net/librerias-de-python-para-machine-learning/

[8] Disha, M. (2019, 12 mayo). *Why You Should Learn Matlab For Data Science.* https://analyticsindiamag.com/why-you-should-learn-matlab-for-data-science/

[9] Ahsan Khan, A. (2019, 18 marzo). *Can Java be used for Machine Learning /Artificial Intelligence?*https://medium.com/nestedif/can-java-be-used-for-machine-learning-artificial-intelligence-42b122d1747e

[10]Innat, M. (2018) *Preferable tools for machine learning - Python - MatLab – R.* https://www.codementor.io/innat_2k14/preferable-tools-for-machine-learning-python-matlab-r-jfozzpphz

[11] Voskoglou, C. (2017) *What is the best programming language for Machine Learning.* https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7

[12] Puget, J. F. (2018, 5 abril). *The Most Popular Language For Machine Learning Is.* https://medium.com/inside-machine-learning/the-most-popular-language-for-machine-learning-is-46e2084e851b

[13] *Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson.*https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/

[14] Broudford (2017)*Five AI Startup Predictions for 2017*http://www.bradfordcross.com/blog/2017/3/3/five-ai-startup-predictions-for-2017

[15] Página Web de la eSalud. https://laesalud.com/que-es-esalud/

[16] Johnson KW, Torres Soto J,Glicksberg BS, Miotto R, Ali M, Ashley E and Dudley JT. *Artificial Intelligence in Cardiology*.J Am Coll Cardiol 2018;71(23):2668-79

[17] Zhao Q, Zhang L. *ECG feature extraction and classification using wavelet transform and support vector machines.* Int Conf Neural Networks Brain 2005;2:1089–1092.

[18]Rajpurkar P, Hannun AY, Haghpanahi M, Bourn C, Ng AY. *Cardiologist-level arrhythmia detection with convolutional neural networks.* Comput Vis Pattern Recognit 2017.

[19]Al'ArefS, AnchoucheK, Singh G, et al. *Clinical applications of machine learning in cardiovascular disease and its relevance to cardiac imaging.* Eur Heart J. 2019 Jun 21;40(24):1975-1986

[20]Eerikainen LM, Vanschoren J, Rooijakkers MJ, Vullings R, Aarts RM. *Reduction of false arrhythmia alarms using signal selection and machine learning.* Physiol Meas.2016;37:1204–1216.

[21] Budzianowski J, Hiczkiewicz J, Burchardt P, et al. *Predictors of atrial fibrillation early recurrence following cryoballoon ablation of pulmonary veins using statistical assessment and machine learning algorithms.* Heart Vessels. 2019;34:352–359.

[22] Lyon A, Ariga R, Minchole A, et al. *Distinct ECG phenotypes identified in hypertrophic cardiomyopathy using machine learning associate with arrhythmic risk markers.* Front Physiol. 2018;9:213.

[23]Weng SF, Reps J, Kai J, Garibaldi JM, Qureshi N. *Can machine-learning improve cardiovascular risk prediction using routine clinical data?* PLoS One2017;12:e0174944.

[24] Medved D, Ohlsson M, Hoglund P, Andersson B, Nugues P, Nilsson J. *Improving prediction of heart transplantation outcome using deep learning techniques.* Sci Rep. 2018;8:3613.

[25] Kang D, Dey D, Slomka PJ, Arsanjani R, Nakazato R, Ko H, Berman DS, Li D,Kuo CC. *Structured learning algorithm for detection of nonobstructive and obstructive coronary plaque lesions from computed tomography angiography.* J Med Imaging (Bellingham) 2015;2:014003.

[26]Nuñez García JC, Vicente Palacio V, Dorado-Díaz PI y Sánchez Fernández PL. *Machine Learning en la predicción de éxito de la cardioversión eléctrica programada en la fibrilación articular*. Rev Esp Cardiol 2018;71(Supl 1):117

[27] Melero-Alegría J, Cascón M, RomeroA, et al. *SALMANTICOR study. Rationale and design of a population-based study to identify structural heart disease abnormalities: a spatial and machine learning analysis.* BMJ Open2019;9:e024605.

[28] Dorado-Díaz I, Sampedro-Gómez J, Vicente-Palacios V, y Sánchez PL. *Aplicaciones de la inteligencia artificial en cardiología: el futuro ya está aquí.*Rev Esp Cardiol 2019. DOI:10.1016/j.recesp.2019.05.016

[29] Cecconi Duca A, Sanz García, A, Alday E et al. *Nuevo marcadores electrocardiograficos para predecir el riesgo de complicaciones cardiovasculares tras cirugía no cardiaca: Estudio basado en una estrategia de Machine Learning.* Rev Esp Cardiol 2019;72 (Supl 1):75

[30] Escolar Pérez V, Lozano Bahamonde A, Larburu Rubio N et al. *Impacto de los factores mediambientales en la descompensación de insuficiencia cardiaca*. Rev Esp Cardiol 2019;72 (Supl 1):908

[31] Bermúdez López M, Forné C, Amigo N et al. *An in-depth analysis shows a hidden atherogenic lipoprotein profile in non-diabetic chronic kidney disease patients.* Expert Opin Ther Targets, 2019;23(7):619-630

[32] Sánchez Martínez S, Duchateau N, Erdei T et al. *Machine learning analysis of left ventricular function to characterize heart failure with preserved ejection fraction*. Circ Cardiovasc Imaging, 2018;11(4):e007138

[33] Cruz N, Serrano M, López A et al. *Electronic health records (EHRs) data validation in atherosclerotic/cardiovascular clinical phenotypes.* Congreso de la Sociedad Europea de Cardiología, Paris 2019

[34] Iglesias S. (2019). *Inteligencia artificial y machine learning al servicio de una mejor práctica cardiológica.* 24 October 2019, from https://secardiologia.es/comunicacion/noticias-sec/10982-inteligencia-artificial-y-machine-learning-al-servicio-de-una-mejor-practica-cardiologica

[35] Kaski JC, Fernández-Bergés DJ, Consuegra Sánchez L, Cruz Fernández JM, García Moll X, Mostaza JM, Toro Cebada R, González Juanetey JR, Guzmán Martínez G y Marrugat J. *A comparative study of biomarkers for risk prediction in acute coronary syndrome - Results of the SIESTA (Systemic Inflammation Evaluation in non-ST-elevation Acute coronary syndrome) study.* Atherosclerosis 2010;212:66-643

[36] Morrow DA, Rifai N, Antman EM, Weiner DL, McCabe CH, Cannon CP and Braunwald E. *C-reactive protein is a potent predictor of mortality independently of and*

in combination with troponin T in acute coronary syndromes: A TIMI 11A substudy. J Am Coll Cardiol 1998;13(7):1460-1465

[37] Ferreiros ER, Boissonet CP, Pizarro R, García Merletti PF, Corrado G, Cagide A and Bazzino OO. *Independent prognostic value of elevated C-reactive protein in unstable angina.* Circulation 1999;100:1958-1963

[38] Lindahl B, Toss H, Siegbahn A, Venge P, Wallentin L for the FIRSC Study Group. *Markers of myocardial image and inflammation in relation to long-term mortality in unstable coronary artery disease.* N Eng J Med 2000;343:1139-1147

[39] Mueller C, Buettner HJ, Hodgson JM, Marsch S, Perruchoud AP, Roskamm H and Neumann F-J. *Inflammation and long-term mortality after nosn-ST elevation acute coronary syndrome treated with a very early invasive strategy in 1042 consecutive patients.* Circulation 2002;105:1412-1415

[40] Kaski JC, Cruz-Fernández JM, Fernández-Bergés D, García-Moll X, Martín L, Mostaza J, et al. *Marcadores de inflamación y estratificación de riesgo en pacientes con síndrome coronario agudo: estudio SIESTA (Systemic Inflammation Evaluation in patients with non-ST segment elevation Acute coronary syndromes).* Rev Esp Cardiol 2003;56:389-95

[41] Raschka S, and Mirjalili V. (2017). *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow.* Birmingham, UK: Packt Publishing.

[42] Müller, A. C., and Guido, S.(2016). I*ntroduction to Machine Learning with Python: A Guide for Data Scientists.* Milton Keynes, UK: O'Reilly Media, Incorporated.

[43]García A, *La inteligencia artificial es útil en Sanidad, pero la tecnología no está aún madura,* en La Opinión (29/10/2019)

ANEXO 1

Tabla 5 Aplicación de técnicas de machine learning en Cardiología. Selección bibliográfica (modificada de PJ Dorado-Diaz, et al [3.4])

Área	Tipo	Aplicación	Autor	Referencia	Técnica
Arritmias	Predicción	Predicción de FA paroxística a partir de la variabilidad de la FC	Ebrahimzadeh, et al.	*Comput Methods Programs Biomed. 2018;165:5-67*	Aprendizaje Supervisado
Arritmias	Predicción	Predicción de recurrencia de FA tras crioablación de venas pulmonares	Budzianowski, et al.	*Heart Vessels. 2019;34:352-59*	Aprendizaje Supervisado
Arritmias	Predicción	Predicción de mortalidad hospitalaria en pacientes con parada cardíaca resucitada a partir de un registro	Nanayakkara, et al	*PLoS Med 2018;15:e1002709*	Aprendizaje Supervisado
Arritmias	Clasificación	Clasificación de alarmas por arritmias cardíacas en telemetría	Eerikainen, et al	*Phisiol Meas 2016;37:1203-1216*	Aprendizaje Supervisado
Arritmias	Clasificación	Detección de hasta 17 tipos de arritmias a partir de ECG	Yildirim, et al	*Comput Biol Med 2018;102:411-420*	Aprendizaje Supervisado
Riesgo CV	Predicción	Predicción de eventos CVs a 10 años a partir de HE	Weng, et al	*PloS One 2017;12:e0174944*	Aprendizaje Supervisado
C-Isq	Predicción	Predicción de eventos cardiacos mayores en pacientes con SCA a partir de HE	Huang, et al	*J Biomed Inform 2017;66:161-170*	Aprendizaje Supervisado
C-Isq	Predicción	Predicción de mortalidad a 30 días tras IM a partir de un registro	Shouval, et al	*Int J Cardiol 2017;246:7-13*	Aprendizaje Supervisado
C-Isq	Predicción	Predicción de supervivencia a 2 años tras IM a partir de datos poblacionales, clínicos y de ecocardiograma	Walert, et al	*BMC Med Inform Decis Mak 2017;17:99*	Aprendizaje Supervisado
C-Isq	Predicción	Predicción de mortalidad tras ingreso por IM a partir de datos de laboratorio y comorbilidad	Goldstein, et al	*Eur Heart J 2017;38:1805-1814*	Aprendizaje Supervisado

Área	Tipo	Aplicación	Autor	Referencia	Técnica
C-Isq	Diagnóstico	Mejoría del diagnóstico de IM en Urgencias a partir de datos demográficos y bioquímicos.	Than MP, et al	*Circulation 2019; D.O.I.: 11.1161/CIRCULATIONAHA/119.041980*	Aprendizaje Supervisado
I.C.	Predicción	Predicción de reingresos hospitalarios por IC a partir de HE	Shameer, et al	*Pc Symp Biocomput 2017;22:276-287*	Aprendizaje Supervisado
I.C.	Clasificación	Seguimiento remoto de pacientes con IC para ajuste de tratamiento y evitar hospitalizaciones a partir de señales ECG de dispositivos electrónicos inteligentes	Inan, et al	*Cir Heart Fail 2018;11:e007138*	Aprendizaje no supervisado
I.C.	Predicción	Predicción de diagnóstico en pacientes en lista de espera de trasplante cardiaco	Medved, et al	*Sci Rep 2018;8:3613*	Aprendizaje Supervisado
I.C.	Clasificación	Caracterización de la IC con FEp a partir de variables ecocardiográicas	Sánchez-Martínez et al	*Circ Cardiovas Imaging 2018;11:e007138*	Aprendizaje no supervisado
I.C.	Predicción / Clasificación	Predicción de evolución y fenotipado de pacientes de IC	Ahmad, et al	*J Am Heart Assoc 2018 D.O.I.:10.1161/JAHA.117.008081.*	Aprendizaje Supervisado y no supervisado
I.C.	Clasificación	Fenogrupos de respondedores a TRC a partir de parámetros clínicos y ecocardiográficos	Cikes, et al	*Eur J Heart Fail 2018; D.O.I.:10.1002/ejhf.1333*	Aprendizaje no supervisado
Imagen	Predicción	Predicción de mortalidad a partir de variables ecocardiográficas e HE	Samad, et al	*JACC Cardiovasc Imaging 2018;12:68-689*	Aprendizaje Supervisado
Imagen	Diagnóstico	Identificación de HV fisiológica vs. patológica a partir de variables ecocardiográficas	Narula, et al	*J Am Coll Cardiol 2016;68:2287-2298*	Aprendizaje Supervisado
Imagen	Diagnóstico	Estudio comparativo entre la cuantificación automática y manual de la función ventricular izquierda ecocardiográfica	Knackstedt, et al	*J Am Coll Cardiol 2015;66:1456-1466*	Aprendizaje Supervisado
Imagen	Predicción	Predicción de ECO en TCC a partir de variables clínicas y score de calcio coronario	Al'Aref et al	*Eur Heart J 2019; D.O.I.: 10.1093/eurheartj/ehz565*	Aprendizaje Supervisado
Imagen	Diagnóstico	Cálculo de RFF a partir de imágenes de tomografía coronaria	Tesche, et al	*Radiology 2018;288:64-72*	Aprendizaje Supervisado

Área	Tipo	Aplicación	Autor	Referencia	Técnica
Miscelánea	Diagnóstico	Diferenciación entre pericarditis constrictiva y miocardiopatía restrictiva a partir de variables clínicas y ecocardiográficas	Sengupta, et al	*Circ Cardiovasc Imaging 2016; D.O.I.:10.1161/CIRCIMAGING.115.004330*	Aprendizaje Supervisado
Miscelánea	Clasificación	Distribución geografica de cardiopatías a partir de datos demográficos y clínicos para previsión de recursos sanitarios	Melero-Alegría et al	*BMJ Open 2019;9:e024605*	Aprendizaje Supervisado

(C-Isq: cardiopatía isquémica; CV: cardiovascular; ECG: electrocardiograma; ECO: enfermedad coronaria obstructiva; FA: fibrilación auricular; FC: frecuencia cardíaca; FEp: fracción de eyección preservada; HE: historia electrónica; HV: hipertrofia ventricular; IC: insuficiencia cardíaca; IM: infarto de miocardio; SCA: síndrome coronario agudo; TCC: Tomografia computarizada coronaria.

ANEXO 2

Tabla 6 Variables utilizadas para el modelo

Nombre Variable	Descripcion de la variable	Tipo	Tipo variable	Categoría (si aplica)	Inclusión Modelo	Nulos	Estrategia de Sustitucion de nulos
edad	Edad (en años)	Demográfico	Numérica		SI	0	
SEXO	SEXO	Demográfico	Categórica	(0=HOMBRE , 1=MUJER)	SI	0	
ecg_scr	Inclusion: ECG		Categórica	(0=NO , 1=SI)	SI	0	
antenfvasc_scr	Inclusion Antec enf vascular		Categórica	(0=NO , 1=SI)	SI	0	
troponinas_scr	Inclusion troponinas		Categórica	(0=NO , 1=SI)	SI	0	
PAD	PAD	Dato Clinico	Numérica		SI	26	Media
PAS	PAS	Dato Clinico	Numérica		SI	26	Media
Latidos	LATIDOS	Dato Clinico	Numérica		SI	26	Media
IMC	IMC	Dato Clinico	Numérica		SI	35	Media
Temperatura	Temperatura	Dato Clinico	Numérica		SI	30	Media
HTA11	HTA	Factor de riesgo	Categórica	(0=NO , 1=SI)	SI	7	0
Dislipemia	DISLIPEMIA	Factor de riesgo	Categórica	(0=NO , 1=SI)	SI	8	0
Diabetes	DIABETES	Factor de riesgo	Categórica	(0=NO , 1=SI)	SI	7	0
Tipofumad	TIPOFUMADOR	Factor de riesgo	Categórica	(1=No Fumador , 2=Exfumador , 3=Fumador Activo)	SI	10	1
Fumador_activo	Fumador	Factor de riesgo	Categórica	(0=NO FUMADOR , 1=FUMADOR)	SI	8	0
Alcohol	ALCOHOL	Factor de riesgo	Categórica	(0=NO , 1=SI)	SI	7	0
Antfamil	ANTFAMILIAR	Factor de riesgo	Categórica	(0=NO , 1=SI)	SI	7	0
Clsquemica21	CISQUEMICA	Antecedentes	Categórica	(0=NO , 1=SI)	SI	7	0
IAM21	IAM	Antecedentes	Categórica	(0=NO , 1=SI)	SI	15	0
ANGINA21	ANGINA	Antecedentes	Categórica	(0=NO , 1=SI)	SI	13	0
ACTP21	ACTP	Antecedentes	Categórica	(0=NO , 1=SI)	SI	7	0
BYPASS21	BYPASS	Antecedentes	Categórica	(0=NO , 1=SI)	SI	7	0
VALVULOPATIAS 21	VALVULOPATIAS	Antecedentes	Categórica	(0=NO , 1=SI)	SI	7	0
AFECTCEREBROV ASC21	AFECTCEREBROV ASC	Antecedentes	Categórica	(0=NO , 1=SI)	SI	7	0
ARTPERIFERICA2 1	ARTPERIFERICA	Antecedentes	Categórica	(0=NO , 1=SI)	SI	8	0
TIMI	TIMI RISK SCORE	score	Categórica		SI	22	Eliminar datos nulos
ALT_ST	Cambios ST	Marcador clinico	Categórica	(0=NO , 1=SI)	SI	0	
Cambios_ECg41	Cambios_ECg	Marcador clinico	Categórica	(0=NO , 1=SI)	SI	0	
endpoinTOTALa 360dias	End point total a 360 dias		Categórica	(0=NO , 1=SI)	Variable de decision	0	
Angioplastiaing	Angioplastia ingreso	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	0	
Cirug_aing	Cirugia ingreso	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	0	
In_hosp_Revasc ul	Revascularizacio n	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	0	

	intrahospitalaria						
ECOCARDIO42	ECOCARDIO	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	39	0
FEimputed	FE ECO+ANGIO	Dato Clinico	Numérica		SI	0	
CORONARIOGRAFIA43	CORONARIOGRAFIA	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	13	0
Nvasosimputed	Numero vasos enfermos	Datos evolucion clinica	Categórica	(0 , 1 , 2 , 3)	SI	0	
IntervencionismoIng	Intervencionismo ingreso	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	0	
Stentingr	Stent ingreso	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	0	
CirRescateing	CirRescate	Datos evolucion clinica	Categórica	(0=NO , 1=SI)	SI	0	
Trat_Previo61	Trat_Previo	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	13	0
AAS61	AAS previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	22	0
Ticlopidina61	Ticlopidina previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
Clopidogrel61	Clopidogrel previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	22	0
Trifusal61	Trifusal previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
IECA61	IECA previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	22	0
ARA_II61	ARA_II previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
AntCalcio61	AntCalcio previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	22	0
BetaBloq61	BetaBloq previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
Alfa_Bloq61	Alfa_Bloq previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	21	0
Digoxina61	Digoxina previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
Diureticos61	Diureticos previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	22	0
Anticoagulantes61	Anticoagulantes previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	21	0
Estatinas61	Estatinas previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	22	0
Nitratos61	Nitratos previa	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
Aines_cortic61	Aines_cortic previos	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
ADO61	ADO previos	Datos tratamiento Previo	Categórica	(0=NO , 1=SI)	SI	23	0
IECAingreso	IECA ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	29	0
Diureticosingreso	Diureticos ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	31	0
Digoxinaingreso	Digoxina ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	32	0
ARA_Iingreso	ARA_II ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	32	0
Alfa_Bloqingreso	Alfa_Bloq ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	32	0
AASingreso	AAS ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	22	0
Ticlopidinaingres	Ticlopidina	Datos tratamiento	Categórica	(0=NO , 1=SI)	SI	32	0

o	ingreso	Ingreso					
Clopidogrelingreso	Clopidogrel ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	26	0
Trifusalingreso	Trifusal ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	32	0
Inh_IIbIIIaingreso	Inh_IIbIIIa ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	29	0
HEPARINAingreso	HEP HNF + HBPM	Datos tratamiento Ingreso	Numérica		SI	32	Moda
Beta_Bloqingreso	Beta_Bloq ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	24	0
NITRATOS_INGRESO	NITRATOS VO+IV	Datos tratamiento Ingreso	Numérica		SI	32	Moda
Ant_Calcicosingreso	Ant_Calcicos ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	28	0
Estatinasingreso	Estatinas ingreso	Datos tratamiento Ingreso	Categórica	(0=NO , 1=SI)	SI	27	0
AASalta	AAS alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	39	0
Ticlopidinaalta	Ticlopidina alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
Clopidogrelalta	Clopidogrel alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	41	0
Trifusalalta	Trifusal alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
IECAalta	IECA alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	42	0
ARA_IIalta	ARA_II alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	45	0
Ant_Calcicosalta	Ant_Calcicos alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	42	0
Beta_Bloqalta	Beta_Bloq alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	38	0
Alfa_Bloqalta	Alfa_Bloq alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
Digoxinaalta	Digoxina alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
Diureticosalta	Diureticos alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
Anticoagulantesalta	Anticoagulantes alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	45	0
ADOalta	ADO alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
Corticoidesalta	Corticoides alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
AINEalta	AINE alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	46	0
Nitratosalta	Nitratos alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	41	0
Estatinasalta	Estatinas alta	Datos tratamiento Alta	Categórica	(0=NO , 1=SI)	SI	37	0
CTIngreso	CT Ingreso	Datos Bioquimicos Ingreso	Numérica		SI	0	
HDLIngreso	HDL Ingreso	Datos Bioquimicos Ingreso	Numérica		SI	0	
TGIngreso	TG Ingreso	Datos Bioquimicos Ingreso	Numérica		SI	0	
LDLIngreso	LDL Ingreso	Datos Bioquimicos Ingreso	Numérica		SI	2	Moda
CPKingreso	CPK ingreso	Biomarcadores Convencional	Numérica		SI	59	Media

CPK_MBingreso	CPK_MB ingreso	Biomarcadores Convencional	Numérica		SI	91	Media
tropoINGimputed	troponinas ingreso	Biomarcadores Convencional	Numérica		SI	0	
creatininaingreso	creatinina ingreso	Datos Bioquimicos Ingreso	Numérica		SI	66	Media
MDRD_60	MDRD dicotomica por 60	Score	Categórica	(0=NO , 1=SI)	SI	0	
Leucosingimputed	leucocitos ingreso	Datos Bioquimicos Ingreso	Numérica		SI	0	
hgbingreso	hgb ingreso	Datos Bioquimicos Ingreso	Numérica		SI	49	Media
COLTOTALalta	COLTOTAL alta	Datos Bioquimicos Alta	Numérica		SI	175	Mediana
HDLCOLalta	HDLCOL alta	Datos Bioquimicos Alta	Numérica		SI	187	Mediana
LDLCOLalta	LDLCOL alta	Datos Bioquimicos Alta	Numérica		SI	195	Mediana
TRIGLICERalta	TRIGLICER alta	Datos Bioquimicos Alta	Numérica		SI	176	Mediana
CPKalta	CPK alta	Biomarcadores Convencional Alta	Numérica		SI	181	Mediana
CPKMBalta	CPKMB alta	Biomarcadores Convencional Alta	Numérica		SI	187	Mediana
TROPONalta	TROPON alta	Biomarcadores Convencional Alta	Numérica		SI	162	Mediana
CREATININAalta	CREATININA alta	Datos Bioquimicos Alta	Numérica		SI	140	Mediana
LEUCOalta	LEUCO alta	Datos Bioquimicos Alta	Numérica		SI	131	Mediana
HGBalta	HGB alta	Datos Bioquimicos Alta	Numérica		SI	130	Mediana
NTproimputed	NTpro BNP al ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	0	
CRP_ING	CRP ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	0	
CD40L_ING	CD40L ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	0	
NEOPT_ING	Neopterina ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	84	Mediana
IL6_ING	IL 6 ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	9	Mediana
IL10_ING	IL 10 ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	9	Mediana
IL18_ING	IL18 ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	0	
E_SEL_ING	E selectina ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	0	
P_SEL_ING	P selectina ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	2	Mediana
Cistatinaing	Cistatina al ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	36	Mediana
Fibrinoging	Fibrinogeno al ingreso	Biomarcadores a modelizar Ingreso	Numérica		SI	37	Mediana

Anexo 3

Certificación de Informe Favorable del Comité de Ética en la Investigación.

Carmen Alcaraz Tomás, Secretaria del Comité de Ética en la Investigación de la Universidad Politécnica de Cartagena

CERTIFICA

Que el Comité de Ética en la Investigación de la Universidad Politécnica de Cartagena, en su reunión del día 10 de octubre de 2019, ha evaluado la propuesta referida a la actividad:

Expediente: CEI19_005

Título: Contribución de Machine Learning al ámbito de la Cardiología

Investigador Responsable: Pedro Sánchez Palma

Y considera que se respetan los principios éticos básicos y es adecuado el procedimiento para obtener el consentimiento informado.

Por lo que este Comité **INFORMA FAVORABLEMENTE.**

En Cartagena