ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

REDUCCIÓN Y CLASIFICACIÓN DE FALSAS ALARMAS EN LA UCI UTILIZANDO ÁRBOLES DE DECISIÓN

Autor: Fernando Rioja Checa

Tutor: Óscar Barquero Pérez

Curso académico: 2017/2018

Agradecimientos

En primer lugar, quisiera agradecer a mi tutor, bueno a Óscar. Gracias a él por introducirme y guiarme en el mundo de los datos, su análisis exploratorio, algoritmos de *machine learning*, análisis de resultados... Gracias por ayudarme en este trabajo, por dar las clases de la forma que lo haces y por dejarte ganar al mus algún año que otro...

Gracias al resto de profesores de la escuela, en su inmensa mayoría grandes profesionales y personas, siempre dispuestos a atenderte personalmente, a buscar huecos, creo que es algo que no valoramos lo suficiente. Recomendaría una y otra vez estudiar con vosotros.

Gracias a mis padres, por el esfuerzo que han hecho a todos los niveles. Gracias por apoyarme y exigirme cuando ha tocado. Este final es vuestro final también.

Gracias a los compañeros que conocí al principio, que a día de hoy se han transformado en amigos. Oter, Cris, Javi, Jony, Ángela, Samu, Merino, Sofía y más nombres que no terminaría de escribir. Por las clases particulares cuando han sido necesarias, por las escapadas, viajes, noches, tardes de laboratorio, de seminarios... Sois grandes.

Y gracias a esa persona especial, esa persona que siempre, pase lo que pase, está de tu lado. Que sabe que lo quieres hacer y te anima a ello. Ya que has estado en las malas, te mereces compartir también las buenas. Gracias por todo Marta.

Resumen

En la Unidad de Cuidados Intensivos saltan una cantidad desmedida de alarmas diarias. Estas alarmas son de una prioridad elevada, ya que el estado de los pacientes así lo requiere. Es por ello que las alarmas alcanzan los 80 dB, lo que provoca falta de sueño, estrés... Y otros trastornos añadidos al resto de pacientes hospitalizados. Muchas de estas alarmas surgen debido a que se procesan por separado las señales que están siendo monitorizadas y comparadas con un umbral predefinido, que, en ocasiones, no es el adecuado para la Unidad de Cuidados Intensivos.

Estas innecesarias interrupciones también se ha demostrado que tienen un efecto negativo en la recuperación y la duración de la estancia.

Tan solo entre el 2 % y el 9 % son realmente importantes para el manejo del paciente. Por lo que, en este campo, existe un gran margen de mejora.

En este contexto, este Trabajo Fin de Grado, se centra en utilizar técnicas de *machine learning*, concretamente árboles de decisión realizando un análisis exploratorio de los datos previo, para obtener una buena clasificación. La base de datos consta de 750 pacientes con 3/4 señales recogidas de cada uno de ellos en hospitales de EEUU y Europa, que fue utilizada en el Challenge de Physionet de 2015 para el Computing in Cardiology.

Finalmente los resultados demuestran que el algoritmo planteado mejora a otros dos modelos, *naive bayes* y regresión logística, dejando abiertas las lineas futuras de investigación a la multiclasificación o a probar otras técnicas en el análisis exploratorio de datos.

Abstract

In the Intensive Care Unit, an excessive amount of daily alarms. These alarms are a high priority events, since the condition of the patients thus it requires. That is why the alarms reach 80 dB, which causes lack of sleep, stress ... And other disorders added to the rest of hospitalized patients.

Many of these alarms arise because signals are processed separately, that are being monitored and compared with a predefined threshold, which it's usually not to be in the Intensive Care Unit.

These unnecessary interruptions have also been shown to have a negative effect on recovery and length of stay. Only between 2 % and 9 % are really important for patient management. So, in this field, there is a great room for improvement.

In this context, this Final Degree Project, focuses on using machine learning techniques, specifically decision trees performing an exploratory data analysis, to obtain a good classification. The database consists of 750 patients with 3/4 signals collected from each of them in hospitals in the USA and Europe, used in the Physionet Challenge 2015 at the Computing in Cardiology conference.

Finally, the results show that the proposed algorithm improves two other models, naive bayes and logistic regression, leaving future research lines open to the multiclassification or to try other techniques in the exploratory data analysis.

Índice general

Índice de figuras

1 Introducción

El objetivo de este Trabajo Fin de Grado (TFG) es la reducción de alarmas producidas por diferentes tipos de arritmias que se producen en la Unidad de Cuidados Intensivos (UCI), ya que muchas de ellas se demuestran falsas o que no necesitan atención inmediata. Para ello, se va a utilizar una técnica muy empleada en el *machine learning*, como es la clasificación mediante agregación de árboles de decisión. Buena parte del trabajo en machine learning reside en la adecuación de las señales, por ello, antes de la construcción del modelo será necesario preprocesar los datos y llevar a cabo un análisis exploratorio.

1.1. Motivación

Las estancias hospitalarias no son agradables. Cortas, largas, graves o leves, todos los pacientes quieren evitarlas o suavizarlas lo más posible. El interés crece cuando se trata de una estancia en la UCI, ya de por sí, complicada.

El excesivo número de alarmas de la UCI perjudican tanto al paciente que la eleva, como al resto de pacientes ingresados, e incluso al personal sanitario. Provoca interrupciones en la atención, ralentiza los tiempos de respuesta del personal, ya que si suena una alarma en la UCI, se atiende, dejando la tarea que se esté realizando. Un problema añadido es que el personal sanitario pierde sensibilidad ante las alarmas, porque en su mayoría no son situaciones críticas que haya que tratar en este instante. [1]

Una de los mótivos más usuales de alarma en UCI son las arritmias cardíacas, que

serán el objetivo del presente TFG. Para este fin se utilizará una base de datos con señales medidas en pacientes que ingresaron con algún tipo de arritmia y que hicieron saltar alguna alarma en la UCI.

En este TFG se va a explorar la utilización de modelos de machine learning basados en agregación de árboles de decisión, en congreto, un algoritmo novedoso llamado *Xgboost*. La elección está basada en que los árboles de decisión son una técnica muy eficaz para la clasificación binaria *(decisión)*. Adicionalmente, Xgboost ha demostrado un desempeño interesante en casos similares; no resulta difícil encontrar competiciones en Kaggle [2] donde los ganadores, incluso los puestos de podio, son implementaciones apoyándose en Xgboost.

Los árboles de clasificación son una herramienta muy utilizada en el aprendizaje máquina. Existen otros algoritmos como las redes neuronales, regresión logística o Naïve Bayes. En este TFG los resultados obtenidos con Xgboost se van a comparar con dos algoritmos clásicos de clasificación: Naïve Bayes y Regresión Logística. Uno de los problemas que se tienen que resolver en el TFG es que tratamos con *series temporales*, por lo que tenemos que tratar la extracción de características de estas series. Para ello, se va a utilizar la técnica de análisis en componentes principales (PCA, por sus siglas en inglés Principal Component Analysis), inspirandonos en **FRC**: [4] [3]

1.2. Objetivos

El objetivo principal del TFG es clasificar las alarmas de arritmias de la UCI en dos clases:*False Alarm* (FA) o *True Positive* (TP).

Vamos a utilizar técnicas de machine learning para este objetivo. Podemos dividir el objetivo principal del TFG en varios objetivos más específicos:

- Cargar los datos y analizarlos.
- Extracción de características de las señales utilizando técnicas de PCA.
- Procesar las señales que componen cada alarma.

- Clasificar las alarmas como FA o TP utilizando modelos de machine learning, en concreto siguiendo el modelo diseñado basado en árboles de clasificación, Naïve Bayes y regresión logística.

1.3. Estructura de la memoria

La estructura de este TFG es la siguiente:

- En el Capítulo 2 se presenta Physionet así como nociones básicas sobre cardiología y arritmias.
- En el Capítulo 3 se presentan los datos con los que vamos a tratar.
- En el Capítulo 4 se exponen los conceptos del análisis exploratorio y de *machine learning*. Se explica en detalle cómo funciona el módulo de Python mediante el cual implementaremos los árboles de decisión. Se explicarán también cómo se construyen los modelos de Naïve Bayes y de regresión logística.
- En el Capítulo 5 se exponen los resultados obtenidos.
- En el Capítulo 6 se presentan las conclusiones y las líneas de investigación futuras.

2 Estado del Arte

En este capítulo se va a introducir el interés por analizar y procesar señales médicas así como la evolución de las bases de datos hasta la actualidad.

A su vez, se hará una breve introducción teórica a los tipos de arritmias que componen la base de datos.

2.1. Contexto del estudio

El procesamiento digital de señales médicas ha ido muy ligado a la evolución de las mejoras en los sistemas de computo, y en los últimos años ha experimentado un gran auge la utilización de métodos de *machine learning* para analizar señales médicas de todo tipo.

- En 1952 Arthur Samuel escribe el primer programa de ordenador capaz de aprender. Consistia en un programa que jugaba a las damas y que mejoraba su juego después de cada partida. [6]

- Durante los años 1974 y 1980 tiene lugar el llamado AI Winter, ya que numerosas empresas que financiaban los proyectos de inteligencia artificial, dejan de invertir porque no se producen tantos avances para las expectativas creadas. [6]

- Los años 80 están marcados por la aparición de sistemas expertos basados en reglas y condiciones. Por ejemplo, en 1981 Gerald Dejong introduce el concepto “Explanation

Based Learning" (EBL), donde un computador analiza datos de entrenamiento y crea reglas generales que le permiten descartar los datos menos importantes. [6]

- A finales de los 80 y durante buena parte de los 90, tiene lugar el segundo AI Winter. En esta ocasión los efectos duraron prácticamente hasta los 2000. Mencionar que en 1997, el ordenador Deep Blue, de IBM ganó al campeón del mundo de ajedrez Gary Kaspárov. [6]

El aumento de la potencia y rapidez de cálculo junto con la gran abundancia de datos disponibles han vuelto a lanzar el campo de *machine learning*. Muchas empresas están transformando sus negocios hacia los datos y están incorporando técnicas de aprendizaje máquina en sus procesos, productos y servicios para obtener ventajas sobre la competencia. También en el campo de la medicina, donde además de los diagnósticos, se incorpora el análisis de datos puramente estadístico como ayuda para diagnosticar al paciente.

En este contexto surge la plataforma Physionet, de donde se han obtenido los datos para este TFG.

2.2. ¿Qué es Physionet?

PhysioNet ofrece acceso gratuito a través de la web a grandes colecciones de señales fisiológicas registradas y software relacionado de código abierto. El sitio web de PhysioNet es un servicio público de PhysioNet Research Resource for Complex Physiologic Signals, financiado por el Instituto Nacional de Imágenes Biomédicas y Bioingeniería (NIBIB) y el Instituto Nacional de Ciencias Médicas Generales (NIGMS) en los Institutos Nacionales de Salud. [9]

PhysioNet Resource, creado en 1999, tiene como objetivo estimular la investigación actual y las nuevas exploraciones en el estudio de señales fisiológicas y biomédicas complejas. Tiene tres componentes estrechamente independientes: [9]

- **Physiobank:** es un gran y creciente archivo de grabaciones digitales bien caracterizadas de señales fisiológicas, series de tiempo y datos relacionados para su uso por parte de la comunidad de investigación biomédica.

- **PhysioToolKit:** es una biblioteca de software para procesamiento y análisis de señal fisiológica, detección de eventos fisiológicamente significativos utilizando técnicas clásicas y métodos novedosos, como estadística avanzada o dinámica no lineal.

- **PhysioNetWorks:** es un laboratorio virtual donde se puede trabajar con colaboradores de cualquier parte del mundo para crear, evaluar, mejorar, documentar y preparar nuevos datos y trabajos de software para su publicación en PhysioNet.

El sitio web de PhysioNet fue establecido por el Resource como su mecanismo para el intercambio libre y abierto de señales biomédicas grabadas y software de código abierto para analizarlos, proporcionando instalaciones para el análisis cooperativo de datos y la evaluación de nuevos algoritmos propuestos.

Además de proporcionar acceso electrónico gratuito a datos PhysioBank y software PhysioToolkit, y espacios de trabajo seguros para el desarrollo colaborativo de nuevos datos y software dentro de PhysioNetWorks, el sitio web PhysioNet ofrece servicio y capacitación a través de tutoriales en línea para ayudar a los usuarios en su inicio y para niveles más avanzados.

Physionet lanza competiciones (challenges) todos los años, para celebrar como parte de la conferencia internacional Computing in Cardiology. Son competiciones abiertas en las que se plantea un problema de señales médicas cardíacas para que la comunidad aporte soluciones. El seleccionado para este TFG, corresponde al del año 2015 y tiene por título: *Reducing False Arrhythmia Alarms in the ICU: the PhysioNet/Computing in Cardiology Challenge 2015.* [10]

2.3. Señales electro-fisiológicas

El objetivo del Challenge fue la reducción de las falsas alarmas arrítmicas en UCI. Para este objetivo concreto se contaba con una serie de señales que monitorizaban la actividad eléctrica del corazón y la presión arterial. En esta sección se introducen brevemente los conceptos necesarios para entender la señales electro-fisiológicas que se van a utilizar a lo largo del TFG.

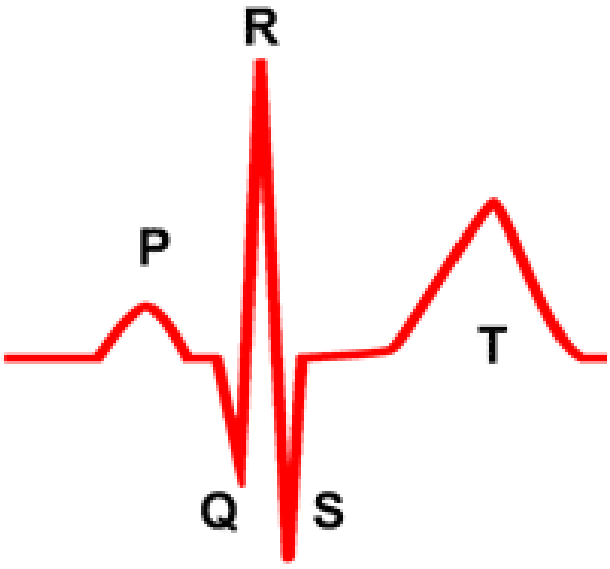

Figura 2.1: Señal de ECG y sus ondas más más significativas. [13]

2.3.1. Señal Electrocardiográfica

Electrocardiograma (ECG) El ECG es la representación gráfica de la actividad eléctrica del corazón en función del tiempo [5].

El ECG durante un latido tiene una serie de deflexiones que son caractarerísticas de cada una de las partes del latido. Se puede observar en la Figura 2.1 las partes significativas en la onda. A continuación se detallan brevemente:

- Onda P: Es la primera onda del ciclo cardíaco. Representa la despolarización de las aurículas. El principio corresponde a la despolarización de la aurícula derecha y el final al de la izquierda. La duración normal de la onda P es menor de 0,10s (2,5mm de ancho) y un voltaje máximo de 0,25 mV (2,5 mm de alto).

- Complejo QRS: El complejo QRS corresponde al impulso eléctrico que causa la contracción de los ventrículos derecho e izquierdo (despolarización ventricular). Tiene una mayor amplitud que la de las aurículas, debido a que los ventriculos tienen más masa cardiaca, es por esto, que se aprecia más en el ECG.

    - Onda Q: Primera onda del complejo QRS, es una onda negativa. Suele ser estrecha y poco profunda (menor de 0.04 s de ancho, 2 mm de profundidad)

(depende de la derivación concreta). Si hay una mínima onda positiva en el complejo QRS previa a una onda negativa, la onda negativa no es una Q, es una onda S, por muy pequeña que sea la onda positiva previa.

- Onda R: Es la primera onda positiva del complejo QRS, puede estar precedida de una onda negativa (onda Q) o no.
- Onda S: Es la onda negativa que aparece después de la onda R.
- Onda Q-S: Cuando un complejo es completamente negativo, sin presencia de onda positiva, se le denomina QS. Suele ser un signo de necrosis cardíaca.
- Ondas R' y S': Cuando hay más de una onda R o más de una onda S, se les denomina R' y S'.

- Onda T: Representa la repolarización de los ventrículos. Generalmente es de menor amplitud que el QRS que le precede. Su amplitud máxima es menor de 5 mm en derivaciones periféricas y menor de 15 mm en derivaciones precordiales.

Esta actividad se recoge mediante dos electrodos. En función de dónde se coloquen esos electrodos obtenemos distintas derivaciones. Siguiendo un símil fotográfico, las derivaciones son como diferentes ángulos o perspectivas de la misma fotografía. [11] [12]

- **I Derivación:** Mide la diferencia de potencial entre el electrodo del brazo derecho y el izquierdo.
- **II Derivación:** Se mide del brazo derecho a la pierna izquierda.
- **III Derivación:** Diferencia de potencial entre el brazo izquierdo y la pierna izquierda.
- **La derivación aVR (augmented vector right):** tiene el electrodo positivo (blanco) en el brazo derecho. El electrodo negativo es una combinación del electrodo del brazo izquierdo (negro) y el electrodo de la pierna izquierda (rojo), lo que .ªumenta"la fuerza de la señal del electrodo positivo del brazo derecho.
- **La derivación aVL (augmented vector left):** tiene el electrodo positivo (negro) en el brazo izquierdo. El electrodo negativo es una combinación del electrodo del brazo derecho (blanco) y la pierna izquierda (rojo), lo que äumenta"la fuerza de la señal del electrodo positivo del brazo izquierdo.

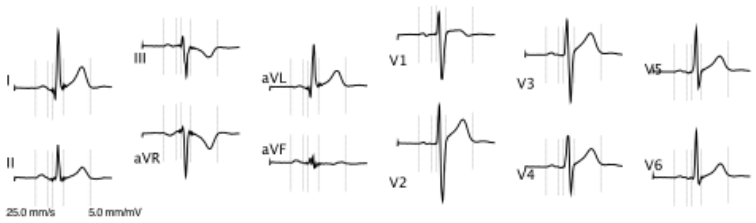

Figura 2.2: Forma de onda de las 12 derivaciones posibles del ECG.

- **La derivación aVF (augmented vector foot):** tiene el electrodo positivo (rojo) en la pierna izquierda. El electrodo negativo es una combinación del electrodo del brazo derecho (blanco) y el brazo izquierdo (negro) lo que .ªumenta"la señal del electrodo positivo en la pierna izquierda.

- Los electrodos para las derivaciones precordiales (V1, V2, V3, V4, V5, y V6) están colocados directamente sobre el pecho. Debido a su proximidad con el corazón, no es necesario aumentarlas. Las derivaciones precordiales ven la actividad eléctrica del corazón en el denominado plano horizontal. El eje eléctrico del corazón en el plano horizontal se denomina el eje Z.

Por lo tanto hay 12 derivaciones como se puede apreciar en la figura 2.2: Las derivaciones inferiores (III y aVF) detectan la actividad eléctrica desde el punto superior de la región inferior (pared) del corazón. Esta es la cúspide del ventrículo izquierdo.

Las derivaciones laterales (I, II, aVL, V5 y V6) detectan la actividad eléctrica desde el punto superior de la pared lateral del corazón, que es la del ventrículo izquierdo.

Las derivaciones anteriores, V1 a V6 representan la pared anterior del corazón o la pared frontal del ventrículo izquierdo.

aVR raramente se utiliza para la información diagnóstica, pero indica si los electrodos se han colocado correctamente en el paciente.

2.3.2. Arritmia Cardíaca

El objetivo del Challenge, y por tanto de este TFG, es identificar las FA arrítmicas en la UCI. El ECG es una de las herramientas fundamentales para la detección de arritmias. En esta Sección se expone la información necesaria sobre arritmias para poder entender el objetivo del TFG.

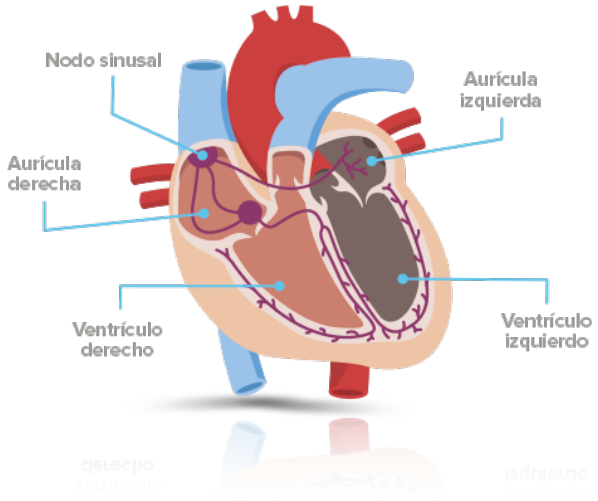

Figura 2.3: Partes del corazón.

Una arritmia es un trastorno de la frecuencia cardíaca (pulso), o del ritmo cardíaco [14]. Su estudio es importante ya que puede ser un indicio de graves problemas cardíacos o un peligro inmediato para la salud del paciente. Es por esta razón por la que las arritmias se consideran como alarmas en la UCI. [14]

En condiciones normales, la función del corazón es la bombear sangre a los pulmones y al resto del cuerpo. Para que el bombeo se produzca de forma eficiente, es decir que la contracción sea eficiente,el corazón posee un sistema eléctrico que garantiza que se contraiga (comprima) de manera ordenada. En la figura 2.3 se observa un esquema sencillo de las partes del corazón.

El impulso eléctrico que da la señal al corazón para contraerse comienza en un área del corazón llamada nodo sinoauricular (también llamado nodo sinusal o nódulo SA). Este es el marcapasos natural del corazón. Diferentes mensajes nerviosos le dan la señal al corazón para latir más lento o más rápido.

Las arritmias pueden presentar anomalías en las señales cardíacas (o pueden aparecer otras nuevas), las señales eléctricas pueden sufrir variaciones temporales, se pueden retrasar o adelantar o incluso pueden viajar por rutas diferentes por el corazón. Esta son las causas generales que desembocan en arritmias cardíacas [14]

Algunas causas usuales de latidos anormales del corazón son:

- Niveles anormales de potasio u otras sustancias en el cuerpo.
- Ataque al corazón o daños que existan en el miocardio debidos a un ataque previo.
- Cardiopatía congénita.
- Insuficiencia cardíaca o un agrandamiento desmedido del corazón.
- Hipertiroidismo.

A continuación detallamos las arritmias más relevantes para este TFG: [1]

1. **Asístola:** Sin complejo QRS durante al menos 4 segundos.
2. **Bradicardia extrema:** durante 5 latidos seguidos, el pulso cardiaco más bajo de 40bpm. Menos de 5 latidos en 6 segundos. Ritmo excesivamente lento.
3. **Taquicardia extrema:** durante 17 latidos, el ritmo cardíaco más alto de 140 bpm. Más de 17 latidos en 6.85 segundos. Funcionamiento muy rápido.
4. **Taquicardia ventricular:** 5 o más latidos ventriculares con el ritmo cardíaco mayor de 100bpm (en un periodo de 2.4 segundos). Latido que no debería partir del ventrículo.
5. **Fibrilación ventricular:** fibrilación o alteración de la forma de onda de al menos 4 segundos, de orgien ventricular.

2.4. Señales fisiológicas

Para la detección de arritmias en UCI se suelen registras otro tipo de señales fisiológicas que permiten completar la información de las señales de ECG. En concreto, para el Challenge que trata este TFG se tuvieron diferentes señales de presión arterial y señales de respiración además del ya mencionado electrocardiograma:

Ambulatory Blood Pressure (ABP) :método técnico no invasivo, que pretende obtener medidas sobre la presión arterial durante un cierto periodo de tiempo. [17]

Figura 2.4: Instrumentos necesarios para medir la señal ABP.

Photoplethysmogram PPG Consiste en medir las variaciones del nivel sanguíneo en alguna zona del cuerpo. Existen varios métodos, pero este en concreto mide la variación de la longitud de onda en la piel para determinar la concentración. [16]

Respiración Se mide la respiración a través del ECG. Se ha demostrado que es mucho menos estresante para el paciente y, por tanto, provoca menos errores en la medida. [18]

3 Descripción de los datos

En este capítulo se definen los datos del TFG, también se explica en qué porcentaje aparecen los diferentes tipos de arritmias, así como la forma original en la que el *challenge* estaba planteado.

3.1. Obtención de las señales

En el organismo humano se pueden medir numerosas señales, que en esencia, son magnitudes físicas: presión, potencial eléctrico, frecuencia...

En este TFG se disponen de los siguienetes datos:

- 750 pacientes, de los que se tienen 3-4 señales de cada uno.
- Los pacientes fueron monitorizados en diferentes hospitales de Europa y USA.

El objetivo concreto del challenge (y de este TFG), es la reducción de falsas alarmas producidas por arritmias. En la siguiente figura 3.1 se muestra como están repartidas las señales que aparecen en la base de datos:

	formación (*N* = 750)		Prueba (*N* = 500)	
	Falso	Cierto	Falso	Cierto
ASY	100	20	90	12
EBR	45	45	38	26
ETC	8	131	5	68
VTA	253	90	176	45
VFB	52	6	34	6
PPG	227	178	158	83
ABP	59	63	58	39
Ambos	172	51	127	35
Total	458	292	343	157

Nota. Cada una de las *norte* registros incluyen dos canales de ECG.

Figura 3.1: Número de señales registradas de cada paciente según su tipo de arritmia y si es falsa alarma o no.

3.2. Descripción de la competición

Como ya se ha mencionado anterioremente el objetivo del *challenge* es clasificar las arritmias detectadas como reales o falsas. A su vez también se propone que se identifique, en el caso de que exista arritmia a que clase corresponde de las 5 que tenemos en la base de datos.

La muestra fue elegida con el fin de que represente fielmente la distribución de la realidad, quedando de la siguiente forma:

- 47 % Taquicardia Ventricular.
- 17 % Asístola.
- 17 % Taquicardia extrema.
- 11 % Bradicardia extrema.
- 7 % Fibrilación ventricular.

El desafío se divide en dos partes [15]

- **1ª Parte:** El algoritmo se basa sólo en la información en tiempo real, es decir, que sólo disponemos de la(s) señal(es) antes de que se produjese la alarma.

- **2ª Parte:** El algoritmo era capaz de incorporar a su esquema 30 segundos una vez la alarma se había producido. Es ahí cuando utiliza esta información como realimentación al algoritmo y decide si la alarma era falsa o verdadera.

Se va a analizar como queda repartida la base de datos según las señales disponibles de cada paciente. Hay que tener en cuenta que sólo se dispone del conjunto de entrenamiento *(training)*, ya que, el conjunto de prueba *(test)* es el conjunto reservado para Physionet como organizador del *challenge* para comprobar los resultados del mismo.

Luego, como sólo está disponible para su estudio el primer conjunto de datos, se procederá a subdividirlo para poder evaluar el rendimiento del los algoritmos.

El *challenge* propone una puntuación para ponderar los aciertos y los fallos de los participantes. Siguen la siguiente fórmula:

$$Score = \frac{100 * (TP + TN)}{TP + TN + FP + 5 * FN} \tag{3.1}$$

Finalmente, por simplificar el problema a tratar, se escogieron los pacientes de los que se tiene registro de 4 señales y utilizando los 5 primeros minutos de cada señal, independientemente de si se tenían sólo esos 5 o si se disponía de 5:30.

4 Métodos

En este capítulo se van a exponer las diferentes técnicas de *machine learning* que se van a aplicar al problema, como son los árboles de clasificación, *naive Bayes* y regresión logística. También se hará hincapié en cómo implementar dichos modelos en Python. A su vez se introducirán conceptos más generales como *statistical learning* y *machine learning* y sus principales diferencias.

4.1. Statistical Learning

No resulta sencillo encontrar una definición para statistical learning, se puede definir como el área de las matemáticas que pretende obtener determinadas conclusiones a partir de datos conocidos [19]. El esquema general seria el siguiente:

$$Y = f(X) + e \tag{4.1}$$

Se puede ver cómo el *statistical learning* corresponde a encontrar una ecuación o función que explique la relación entre los datos. f es una función de *X1, X2,... Xn*, es decir, de los datos de entrada y el término e se corresponde con el error. De forma muy sencilla podríamos decir que el objetivo del statistical learning es estimar *f*.

Existen dos motivos principales por los que puede resultar de interés estimar f: predicción o inferencia. [19]

1. **Predicción**: en muchos casos el conjunto de datos es conocido y está bien definido,

pero obtener la salida que ofrecen esos datos no suele ser una tarea sencilla.

2. **Inferencia**: probablemente se conozca la salida Y, y las características de entrada X, pero no se encuentra una forma de relacionarlos. Su objetivo es obtener conclusiones útiles para hacer deducciones sobre una totalidad, basándose en la información numérica de la muestra, es decir, ver cómo están relacionadas cada una de las variables de entrada X con la salida Y. [20]. Normalmente, este tipo de problemas responde a una de las siguientes preguntas:

   a) ¿Qué variables están asociados con la respuesta?

   b) ¿Cuál es la relación entre la salida y cada variable?

   c) ¿Se puede resumir la relación entre Y con cada variable de entrada con una relación lineal o es más complicada?

4.2. Machine Learning

Asociado al concepto anterior, surgen diversas técnicas para llevarlo a cabo, este TFG se centra en el *machine learning* el cual podemos definir como: un campo de la inteligencia artificial y las ciencias de la computación, que puede aprender de los datos, en vez de reglas de programación clásicas. [21]

Para relacionar ambos términos, se puede concluir que el *machine learning* es capaz de encontrar relaciones entre los datos, aplicando técnicas de *statistical learning*.

4.2.1. Motivación del Machine Learning

Las aplicaciones que existían en la programación clásica, funcionan en base a estructuras de control, los ya conocidos *if, else, switch, case...* Pero de esto se haya principalmente un incoveniente, para resolver un problema de este tipo se tiene que estudiar a fondo la aplicación es decir, se deberían programar infinitas situaciones para, por ejemplo, reconocer una cara en una aplicación móvil o identificar que alimentos se tienen delante en el plato. Detrás de este problema surge otro, la necesidad de que un humano esté muy

encima del desarrollo del algoritmo, para por ejemplo, ver que decisiones se tomarían si fuese un humano el que decidiera. [22]

A la hora de abordar un problema de *machine learning* es necesario saber que se trata de un problema complejo, normalmente con muchas observaciones y que el mayor tiempo en un proyecto de estas características se lo lleva el análisis de los datos y elegir un modelo que optimice la solución. Existen varios tipos de problemas dentro del *machine learning* y no todos responden igual ante todos los datos. Suele ser de utilidad plantearse algunas preguntas básicas al principio del proyecto para ayudar a la hora de elegir qué técnica concreta utilizar. [22]

- ¿Qué objetivo se quiere conseguir? ¿Es posible con los datos disponibles?
- ¿De cuántos datos se pueden analizar/extraer conclusiones?
- ¿Faltan más ejemplos? ¿Qué hacer con los datos erróneos o incompletos?

Cómo se observa, son preguntas sencillas, pero de mucha utilidad a la hora de enfocar un problema de *machine learning*. La primera hace referencia al objetivo o *goal* del proyecto. La segunda a tener acotados los datos de los que se dispone el estudio y confirmar que todos son válidos para el propósito y la última, se corresponde con el necesario preprocesamiento de los datos, no siempre están en un formato adecuado, totalmente completos, en la forma que interesa para el algoritmo, etc.

Dentro del *machine learning* se pueden diferenciar dos tipos de algoritmos o problemas bastante diferenciados: *supervised learning* y *unsupervised learning*

4.2.2. Supervised Learning

El *machine learning* se divide en dos áreas principales: aprendizaje supervisado y aprendizaje no supervisado. Aunque pueda parecer que el primero se refiere a la intervención humana y la segunda no, estos dos conceptos tienen más que ver con qué se quiere hacer con los datos. [23]

Estos problemas son los más usuales y los que suelen tener mejor éxito en cuanto a su *accuracy*. Consiste en aportar al algoritmo datos de entrada de los que conocemos su salida, una vez disponemos de ambos conjuntos de datos (entrada y salida) se procede a entrenar el algoritmo. [19] Una vez hecho el entrenamiento (*training*), se procede a

evaluarlo en la fase de prueba (*test*) dónde se espera que el algoritmo genere determinadas salidas para ciertas entradas dadas. [22]

Realmente, no es el usuario el que elige qué problema quiere resolver, este viene definido por las características que tengan sus variables de entrada. Existen variables categóricas o cualitativas que diferencian entre distintas clases de una variable, por ejemplo si la variable de entrada es una fruta, podremos asignarla a la clase manzana, pera, plátano...

Por otra parte, existen variables cuantitativas registran valores numéricos, el precio de una acción, la temperatura... [19] En función del tipo que sea la variable de salida del problema se tienen dos grandes grupos de problemas en el aprendizaje supervisado: clasificación vs regresión.

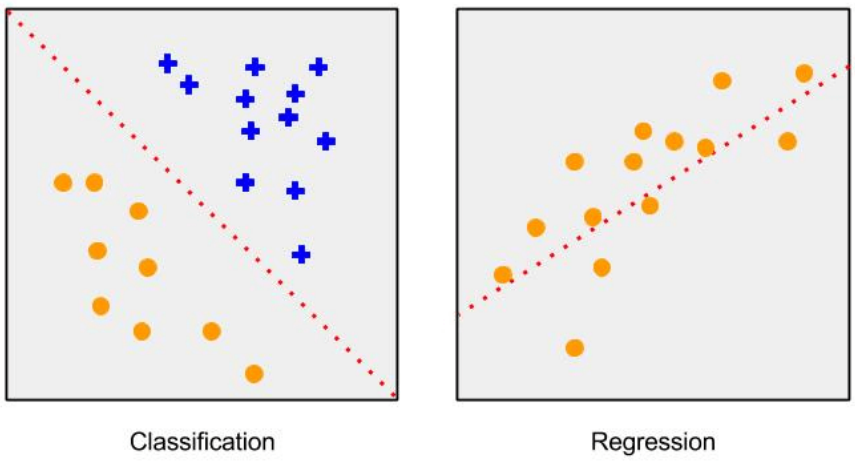

Figura 4.1: Ejemplos de clasificación y regresión.

- **Regresión**: problemas donde la naturaleza de la variable de salida es cuantitativa.
- **Clasificación**: algoritmos qué su respuesta es una variable cualitativa o clase.

4.2.3. Unsupervised Learning

Es el otro gran tipo de problemas en *machine learning*. En el *unsupervised learning* sólo se conocen los datos de entrada. [19] Suelen ser problemas más complejos, ya que no se pueden contrastar contra una salida conocida. El fin es explorar los datos para encontrar alguna estructura o forma de organizarlos. [23] No se sabe si existe algo que predecir, se trata de entender comportamientos, agrupaciones, tendencias...

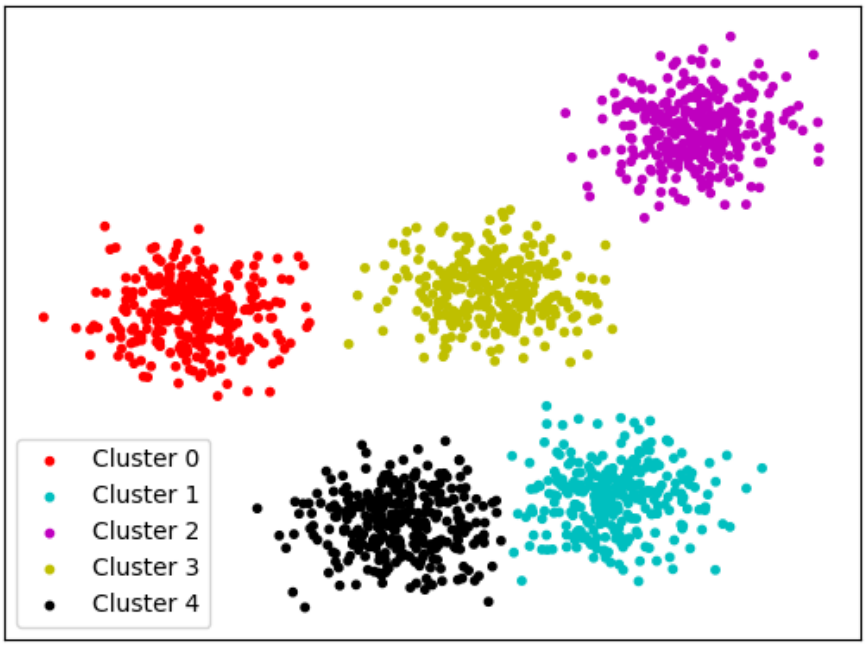

Figura 4.2: Algoritmo de *clustering* dónde se aprecian 5 *clusters*

Una de las técnicas más extendidas en el *unsupervised learning* es el conocido como *clustering* o agrupamiento. En pocas palabras, el objetivo es segregar grupos con rasgos similares, es decir, .ªgrupar"vectores con características similares y separar los que sean diferentes. [24]

Ahora se van a explicar tres conceptos claves en los algoritmos de *machine learning* como son: generalización, sobreajuste o *overfitting*, subajuste o *underfitting*.

4.2.4. Entrenamiento y generalización

Como ya se ha mencionado, en *machine learning* se entrena un determinado algoritmo para después ser evaluado con otro conjunto de datos nuevos al que llamamos *test*.

Para cualquier aplicación que se desee diseñar con estas técnicas, es importante el concepto de **generalización**. Es necesario que el algoritmo que se haya diseñado obtenga buenas respuestas (clasifique bien, falle pocas veces, estime correctamente ciertas muestras críticas, etc), con ejemplos nuevos.

Existen dos problemas que surgen asociados al hecho de tener que entrenar cada algoritmo: *overfitting* y *underfitting*.

- **Overfitting**: Es un problema muy habitual en los comienzos con el *machine learning*. Lo que ocurrirá es que la máquina sólo se ajustará a aprender los casos particulares que le enseñen y será incapaz de reconocer nuevos datos de entrada. En nuestro conjunto de datos de entrada muchas veces introducimos muestras atípicas (ó anómalas) o con "ruido/distorsión" en alguna de sus dimensiones, o muestras que pueden no ser del todo representativas. Cuando "sobre-entrenamos" nuestro modelo y caemos en el *overfitting*, el algoritmo estará considerando como válidos sólo los datos idénticos a los del conjunto de entrenamiento. [7]

- **Underfitting**: Se dice que un modelo estadístico o un algoritmo sufre *underfitting* cuando no es capaz de capturar la forma o las relaciones subyacentes de los datos. [8] Un indicador claro de *underfitting* es obtener malos resultados en entrenamiento, lo que parece indicar que en el *test* se obtendrán malos resultados también.

Para solucionar este problema se deben tener en cuenta ciertas consideraciones: [7] tener una cantidad mínima de muestras tanto para entrenar como para probar, disponer de clases variadas, tener uno o varios conjuntos de test, para ello podemos utilizar técnicas de validación cruzada si no se disponen de muchas muestras.

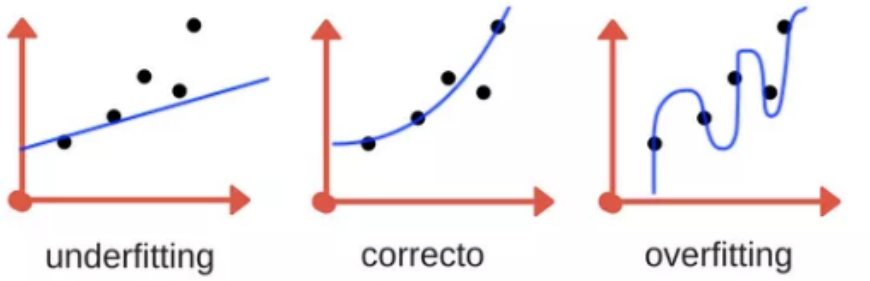

Figura 4.3: *Overfitting* VS *Underfitting*

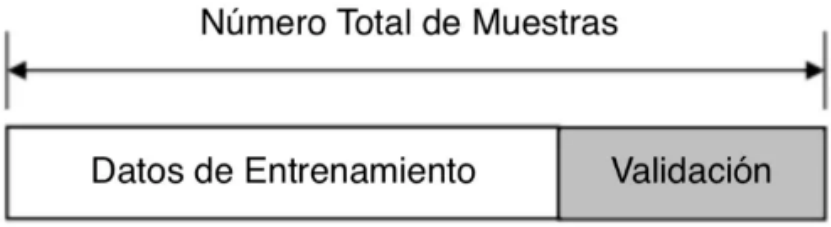

Figura 4.4: Proporciones de los conjuntos de entrenamiento y *test* en función del número total de muestras.

4.3. Enventanado de las señales

Procesar señales de 5 minutos de duración puede ser tedioso y computacionalmente muy exigente. Por esta razón, se procede a enventanar las señales.

Enventanar una señal no es más que divirla en tramos para poder procesarla más fácilmente. La expresión para enventanar una señal es la siguiente:

$$w(t) = x(t)\Delta h(t) \tag{4.2}$$

Existen diferentes tipos de ventanas atendiendo a su forma en tiempo y frecuencia.[1]. También dos principales métodos de aplicar estas ventanas, con solapamiento o sin solapamiento:

[1] `https://en.wikipedia.org/wiki/Window_function`

- **Con solapamiento:** Esta técnica se utiliza para evitar pérdida de información en los bordes de las ventanas. No obstante, las ventanas resultantes quedan más grandes.
- **Sin solapamiento:** Se pierde un poco de resolución en los bordes, con el beneficio de acortar el tamaño de las ventanas.

En este TFG se va a utilizar un enventanado rectangular y sin solapamiento, donde la función de la ventana quedaría de la siguiente forma:

$$h(t) = \begin{cases} 1 & si \quad t \in [0, T] \\ 0 & resto. \end{cases} \tag{4.3}$$

4.4. Extracción de características con PCA

Una vez se tienen unos conceptos básicos sobre *machine learning*, es conveniente recordar el esquema principal de un proyecto de esta índole.

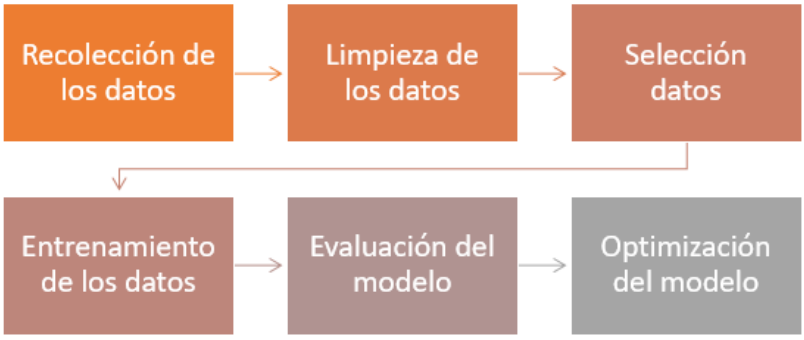

Figura 4.5: Fases necesarias en un proyecto de *machine learning*.

La técnica que se aplica en este TFG, se puede englobar en el apartado "Limpieza de los datos". Existen diferentes formas de plasmar estas etapas, pero en todas ellas en análisis exploratorio de los datos, es previo an entrenamiento del modelo.

No obstante cabe destacar que PCA es la técnica utilizada para la extracción de características de cada paciente, es decir, una vez aplicada, se dejan de tener 3-4 señales

por paciente para tener los autovalores generados al aplicar PCA.

En este punto, se detalla la técnica PCA (*principal component analysis*), que es la que se va a utilizar.

- **Problema**: ¿Se puede agrupar la información de un conjunto de datos mediante un número de variables menor que el de variables originales?

- **Idea**: Si una variable es función de otras, aporta información redundante.

Por tanto, si las m variables observadas están fuertemente correlacionadas, sería posible sustituirlas por menos variables sin gran pérdida de información. [25]

PCA combina linealmente las componentes del vector de entrada X para obtener otro conjunto de características f, denominadas componentes principales. Las nuevas componentes se generan de forma que están incorrelacionadas entre sí. [26] Las componentes f están ordenadas en función de la varianza (de las componentes originales) explicada, luego, descartar las últimas compnentes de f implica descartar las componentes con menos información.

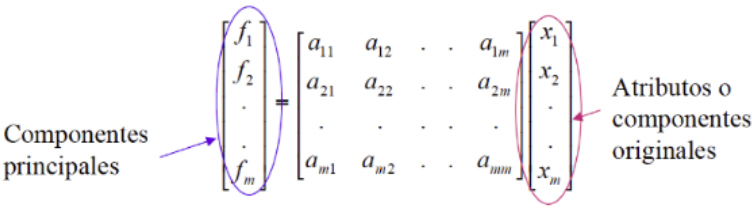

Figura 4.6: Componentes principales relacionadas con la matriz de datos.

El objetivo ahora debe ser como calcular los elementos de la matriz central. Cómo el objetivo es maximizar la varianza, se impone que el módulo del vector a sea 1: [26]

$$a_j = \sum_{m=1}^{p} a_{kj}^2 = 1 \tag{4.4}$$

Se elige a_1 con el objetivo de maximizar la varianza de f_1 sujeta a la restricción de que el módulo sea 1:

$$Var(f_1) = Var(a_1^T * x) = a_1^T * \sum a_1 \tag{4.5}$$

Ahora el problema es maximizar dicha función. Cabe destacar que la incógnita es precisamente a_1 (el vector desconocido que ofrece la combinación lineal óptima). De este modo de constriye la función L: [26]

$$L(a_1) = a_1^T * \sum a_1 - \lambda(a_1^T * a_1 - 1) \tag{4.6}$$

Para maximizar la función hay que derivar despecto de a_1 (el valor que se quiere maximizar) e igualar a 0:

$$\frac{\delta L}{\delta a_1} = 2\sum a_1 - 2\lambda l a_1 = 0 \tag{4.7}$$

El resultado, es en realidad un sistema lineal de ecuaciones. Por el teorema de Rouché-Frobenius[2], para una solución distinta de 0 de la matriz $(\sum -\lambda l)$ tiene que ser singular. Esto implica que el determinante tiene que ser igual a 0: [26]

$$|\sum -\lambda l| = 0 \tag{4.8}$$

La matriz de covarianzas obtenida es de orden p y además está definida como positiva tal que $\lambda_1 > \lambda_2 ... > \lambda_p$. Se puede llegar entonces a relacionar la varianza de f_1 de la siguiente forma:

$$Var(f_1) = Var(a_1^T x) = a_1^T \sum a_1 = a_1^T \lambda a_1 = \lambda a_1^T a_1 = \lambda 1 = \lambda \tag{4.9}$$

Luego para maximizar la varianza de f_1 se tiene que coger el mayor autovalor, es decir λ_1 y su correspondiente autovector a_1. [26] Se pueden resumir dos características fundamentales de PCA:

1. La varianza es maximizada. Se quiere ver qué variables o características del modelo explican más varianza que otras, es decir, cuáles aportan más información.
2. El error de reconstrucción final es mínimo (el error que medimos de volver de las características transformadas a las originales).

Desde un punto de vista geométrico, y posiblemente más intuitivo, el aplicar PCA, es equivalente a efectuar una rotación de los ejes.

[2] `https://bit.ly/2JvVo4a`

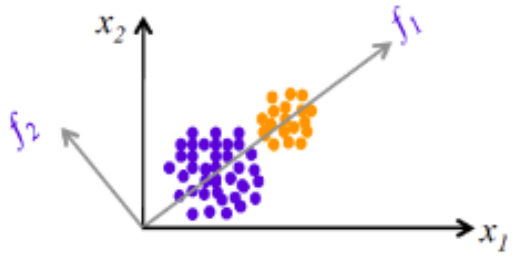

Figura 4.7: Rotación de los ejes originales X, en los transformados f

Resulta muy común encontrar esta técnica aplicada en diferentes escenarios, como por ejemplo, extracción de características en sistemas de regresión o de clasificación como es el caso de este TFG. Son situaciones donde existen muchas muestras de entrada que no aportan demasiada información, en este proyecto, por ejemplo, se disponen de 3-4 señales por paciente. Para acotar el problema, se han seleccionado sólo aquellos pacientes de los que se tienen registradas 4 señales, 498, en 247 casos sólo se dispone de la señal hasta que se produjo la alarma, es decir, 5 minutos de señal o 75000 muestras por cada señal, y en otros 251 casos se tiene la señal 30 segundos después de que saltáse la alarma, lo que haría un total de 5:30 minutos y 82500 muestras.

Para establecer una aproximación, sólo con el primer grupo de pacientes (de los que se tienen 5 minutos) ya se pueden obtener unos 74 millones de muestras de los cuales no todos son relevantes, es por esto que se decidió proceder a reducir los datos utilizando esta técnica.

Suele ser común quedarse con un determinado número de *eigenvalues* (autovalores) según diferentes criterios. Quedarse con los X primeros, que son los que más varianza explican, quedarse con los que expliquen un porcentaje importante, por ejemplo valores habituales son un 70-80 %.

El obejtivo de la utilización de PCA en este TFG es la reducción al espacio señal-ruido. Aplicando PCA, lo que se quiere (además de reducir los datos de entrada como se mencionó previamente) separar en cuatro señales nuevas la señal (que contendrá la información) y el ruido. Los autovalores muy altos indicarán que en ese tramo de señal existe mucha información útil para la clasificación, miesntras que por el contrario, autovalores bajos indicarán que tenemos poca información o ruido.

La matriz de datos X queda de la siguiente forma: Una matriz de 471x20, donde las 471 filas son los pacientes de los que disponemos 4 señales registradas y 20 columnas que corresponden a los autovalores asociados a cada paciente. Cuatro señales por paciente y cinco ventanas por señal.

4.5. Árboles de clasificación: XGBoost

4.5.1. Introducción y conceptos

En esta sección se van a explicar los conceptos generales de los árboles de clasificación así como del módulo de Python que se ha utilizado para implementarlo en este TFG.

Los árboles de clasisicación son una técnica muy extendida de *machine learning* dónde la salida puede tomar un conjunto finito de valores. Esta técnica se basa en utilizar árboles de decisión: dado un conjunto de datos, se fabrican diagramas de construcciones lógicas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. [28]

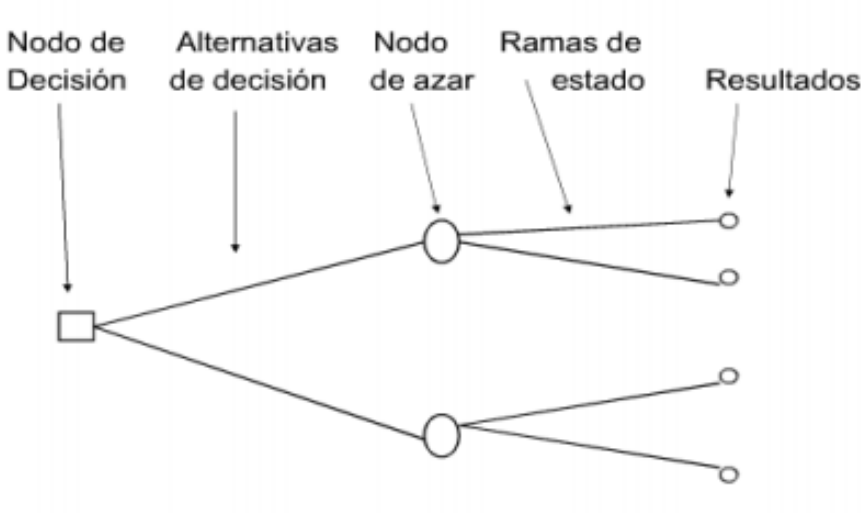

Figura 4.8: Esquema de un árbol de decisión.

Los árboles de clasificación son, en esencia, encadenar varias series de árboles de decisión para resolverun modelo más complejo, que tiene como salida un resultado finito.

Las principales ventajas de utilizar esta herramienta son: [29]

- Plantean el problema para que todas las opciones sean contempladas.
- Proveen un esquema para cuantificar el coste de un resultado y la probabilidad de que suceda.
- Llevan a adoptar la mejor alternativa con la información existente

Y estos inconvenientes:

- Las reglas de asignación son bastante sencillas a pequeñas perturbaciones en los datos.
- Al no tener claridad de objetivos, es difícil de organizar las ideas.
- Presenta inconvenientes cuando la cantidad de alternativas es grande y cuanto las decisiones no son racionales.

4.5.2. XGBoost: Introducción

XGboost es una librería de código abierto, que permite hacer algoritmos basados en *Gradient boosting* en diferentes sistemas operativos (Windows, Linux, macOS) y en direfentes lenguajes de programación (Python, Java, C++). [30]

Gradient boosting es una de las técnicas más potentes para construir modelos predictivos. La idea principal consiste en desarrollar un algoritmo que consiga convertir las hipótesis débiles en buenas. [31] El boosting es un algoritmo de aprendizaje automático supervisado que reduce el sesgo y la varianza. Responde a la pregunta que hicieron Kearns y Valiant: ¿Puede un conjunto de clasificadores débiles, formar un clasificador robusto? En resumen, utilizar a las hipótesis débiles o alumnos débiles *(weak learners)* en repetidas ocasiones generando una sucesión de hipótesis, cada una centrada en los ejemplos que anteriormente se hubiesen encontrado dificultades o no estén clasificados aún. ¿Cómo es esto posible?

La primera implementación con éxito de boosting fue Adaptative Boosting, AdaBoost de aquí en adelante. Los weak learners en AdaBoost son árboles de decisión de un solo escalón. ¿Cómo funciona?

El funcionamiento es el siguiente: [31] AdaBoost configura un peso a las observaciones, poniendo más énfasis a las que son difíciles de clasificar y menos con las que ya se maneja bien. Los nuevos alumnos son añadidos secuencialmente al modelo y su función es el entrenamiento de los patrones más complicados. Esto significa que las muestras más difíciles de clasificar tienen una cadena más larga de árboles, de pesos, que finalmente hace que clasifiquemos mejor estas muestras. De esta forma conseguimos invertir más carga computacional en observaciones más difícil de clasificar y menos en las que el algoritmo responde correctamente. Las votaciones se hacen por mayoría de los weak learners ponderados por su precisión individual. Los resultados más satisfactorios de AdaBoost fueron para problemas de clasificación binaria, que posteriormente se llamó AdaBoost M1.

Estos resultados son en los que se ha apoyado este TFG para la elección de esta implementación del *Gradient Boosting*.

Esta clase de algoritmos se describió como un modelo aditivo por etapas. Esto se debe a que se agrega un nuevo alumno débil a la vez y los estudiantes débiles existentes en el modelo se congelan y se dejan sin cambios.

La idea de gradiente queda patente, ya que el gradiente es un operador matemático que indica la variación máxima de una magnitud. En el algoritmo, los cambios siempre son introducidos al final, cuando añadimos un nuevo weak learner, es decir, un escalón más al árbol.

4.5.3. XGBoost: Conceptos claves

En esta técnica se pueden diferenciar 3 conceptos clave: [31]

1. **Loss function:** Una función de pérdida que debe ser optimizada.
2. **Weak Learners:** Alumnos o hipótesis malos/as para hacer predicciones.
3. **Aditive model:** para añadir alumnos débiles que, paso a paso, vayan minimizando la función de pérdida.

4.5.3.1. Loss Function

La función de pérdida depende del problema que se esté abordando. Tiene que ser derivable, pero en definitiva se admiten muchos modelos, por ejemplo, la regresión

puede utilizar el error cuadrado y la clasificación puede utilizar una función logarítmica. Un punto positivo de trabajar con Gradient Boosting, es que no es necesario desarrollar otro algoritmo para cada función de pérdida que se quiera utilizar, ofrece un marco lo suficientemente amplio como para que se pueda utilizar cualquier función de pérdida derivable.

4.5.3.2. Weak Learners

Como ya se ha mencionado, en cada etapa(es un modelo aditivo) del modelo se añade un nuevo alumno débil. Concretamente se utilizan árboles de regresión que generan salidas reales que se puedan sumar. Esto permite que se puedan añadir salidas de elementos previos y corregir los residuos dejados por predicciones anteriores. Los árboles se construyen aprovechando los mejores puntos de división, donde sea posible minimizar las pérdidas. Inicialmente, como en el caso de AdaBoost se utilizaron árboles muy cortos y de un sólo nivel. Después es habitual que se utilicen árboles con 4-8 niveles así como que se limite a los alumnos débiles, para garantizar que sigan siendo débiles, se puede conseguir limitando el número de nodos o de niveles. [31]

4.5.3.3. Aditive Model

Los árboles son añadidos uno cada vez y hay árboles (o partes del árbol) que no cambian en el modelo. La disminución del gradiente se utiliza para minimizar un conjunto de parámetros como, por ejemplo, variables en una regresión o pesos en una red neuronal. Después de calcular el error, los pesos son actualizados para minimizarlo. [31]

En este caso, en lugar de varibles de una regresión o pesos en una red neuronal, se tienen árboles de decisión. Después de calcular el error, debemos añadir otro árbol que siga minimizando el error, para seguir con el gradiente. Este enfoque se denomina descenso funcional del gradiente o descenso del gradiente con funciones.

4.5.4. XGboost y el problema del Overfitting

Los algoritmos de *machine learning* se enfrentan a diversos problemas en su construcción, pero uno de los más destacados es el *overfitting*.

Gradient Boosting es un algoritmo potente, que a su vez puede sufrir *overfitting*

fácilmente. En este apartado se resumen 4 mejoras para aumentar la efectividad y reducir el sobreentrenamiento del gradiente:

1. **Tree constraints:** Una de las máximas más intuitivas y la primera restricción que se puede añadir al modelo es utilizar el mínimo número de árboles necesario para llevarlo a cabo. También se pueden apuntar algunas restricciones más a la hora de añadir árboles de decisión:

   a) El número de árboles puede hacer el entrenamiento muy costoso, tanto temporal como computacionalmente, luego el consejo es añadir árboles siempre y cuando se vea una mejora notable del algoritmo.

   b) La frondosidad del bosque. Los árboles sencillos son más eficaces y eficientes. Con 4-8 niveles es cuando se ven mejores resultados.

   c) Número de nodos o de hojas: como la profundidad. Se trata de acotar la dimensión del árbol. No obstante, dependerá mucho del estudio que vayamos a realizar.

2. **Weighted Updates:** Las predicciones de cada árbol se suman juntas secuencialmente. La contribución de cada árbol a esta suma se puede ponderar para acelerar/ralentizar el aprendizaje mediante el algoritmo. Esta ponderación se denomina contracción (shrinkage) o tasa de aprendizaje. Simplemente se pondera cada actualización por la tasa de aprendizaje. El efecto que produce es una ralentización del aprendizaje, requiriendo más árboles para el modelo, lo que conlleva más tiempo de entrenamiento generando una relación entre el número de árboles y la tasa de aprendizaje. Disminuir el valor de v (tasa de aprendizaje) aumenta el valor de M (la cantidad de árboles). Es común tener valores pequeños en torno a 0.1-0.3 así como valores más pequeños que 0.1. La contracción ayuda a reducir la influencia de cada árbol individual y ayuda que árboles futuros mejoren el modelo.

3. **Stochastic Gradient Boosting:** Resulta beneficioso que los árboles se creen a partir de muestras del training set. Esta técnica se puede utilizar también para reducir la correlación entre los árboles.

   En cada iteración, una muestra de los datos de entrenamiento se extrae al azar (sin reemplazo) del conjunto de datos de entrenamiento completo. La muestra se-

leccionada al azar se usa luego, en lugar de la muestra completa, para el alumno débil.

4.5.5. XGBoost: Resumen y justificación

En resumen, se utiliza *XGBoost* ya que sus dos características principales son también los objetivos de un modelo de machine learning y, por supuesto, de este TFG: rápida ejecución y rendimiento del modelo. Resulta sencillo encontrar numerosas competiciones de Kaggle[3] donde los ganadores (e incluso los que hacen podio) utilizan soluciones con esta librería.

Se pueden obtener muy buenos resultados teniendo en cuenta los consejos para sobreponerse al *overfitting*. En la teoría es una buena herramienta para la clasificación binaria o decisión y por estas razones se quiso implementar con estos datos.

4.6. Naive Bayes

El siguiente método a estudiar va a ser el clasificador de Bayes ingenuo (*naive Bayes*). Este clasificador, dado un ejemplo de entrada x se basa en encontrar la hipótesis más probable a la que corresponda el ejemplo. [27] El ejemplo puede venir dado por un conjunto de valores como $< a_1, a_2 ... a_n >$, entonces la hipótesis más probable será aquella que cumpla:

$$\hat{S}_{MAP} = arg\ max_{\hat{S}}\ P(S_j | a_1, a_2 ... a_n) \tag{4.10}$$

Es decir, la probabilidad de ya conocidos los valores que describen ese ejemplo, éste pertenezca a la clase $\hat{S}$. Aplicando el Teorema de Bayes quedaría la siguiente expresón:

$$\hat{S}_{MAP} = arg\ max_{\hat{S}} \frac{P(a_1, a_2 ... a_n | \hat{S}) p(\hat{S})}{P(a_1, a_2 ... a_n)} \tag{4.11}$$

Para calcular $P(\hat{S})$ se pueden contar las veces que aparece en el conjunto de entrenamiento y despúes dividirlo entre el número de ejemplos totales. Sin embargo, para conocer el

[3] `https://www.kaggle.com/`

término $P(a_1, a_2...a_n)$, es decir, las veces que aparece el ejemplo x en cada categoría, se debería recorrer todo el conjunto de entrenamiento, posiblemente más de una ocasión. Esta práctica no resultaría fácilmente computable en el momento en el que la dimensión de los datos creciera notablemente. [27] Es ahora cuando se añade la suposición *naive*. Hacer esta suposición implica consideran independientes las características de los datos de entrada, la expresión de los datos quedaría de la siguiente forma:

$$P(a_1, a_2...a_n|\hat{S}) = \prod_{i=1}^{n} P(a_i|\hat{S}) \tag{4.12}$$

Para dejar la expresión en función del estimador $\hat{S}$, se puede concluir que:

$$\hat{S} = arg\ \ max_{\hat{S}}\ \ P(y) \prod_{i=1}^{n} P(a_i|\hat{S}) \tag{4.13}$$

4.7. Regresión Logística

En esta sección se van a introducir los fundamentos del último método que se propone en este TFG.

La regresión logística (*Logistic Regression*) es un tipo de análisis utilizado para predecir el resultado de una variable categórica o dicotómica. Resulta muy útil para modelar la probabilidad de un evento que ocurre en función de otras características o varibles. [32] [33]

Es un concepto similar a la regresión lineal [4] en cuánto a ver cuánta dependencia existe entre la variable dependiente y las independientes. Suele ser habitual utilizar regresión lineal para problemas dónde la varible de salida es continua o escalar y regresión logísistica cuando es categórica.

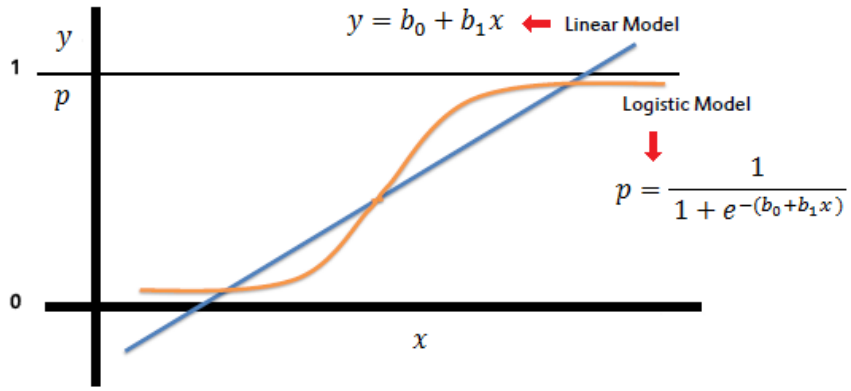

Figura 4.9: Regresión lineal vs regresión logística.

También se optó por este modelo, ya que es más apropiado para la clasificación binaria, que, por ejemplo, la regresión lineal.

[4] https://en.wikipedia.org/wiki/Linear_regression

Una vez conocida toda la formulación matemática que existe detrás del problema, a continuación se explica como hay que proceder para programarlo en Python.

4.8. *Sklearn*

Sklearn es una biblioteca abierta que contiene gran cantidad de módulos para poder resolver problemas de *machine learning*. Concretamente de esta biblioteca sale la función con la que vamos a utilizar *naive Bayes*. La función es: GaussianNB.

4.8.1. *GaussianNB*

GaussianNB es una función muy intuitiva y sencilla. Consta de un único parámetro *priors* [35], que establece dentro de un array las probabilidades *a priori* de cada clase de entrada. En este TFG se ha optado por no modificar este parámetro por la siguiente razón: aunque los datos no estén completamente balanceados, existen más casos de falsas alarmas, se quiere partir de una hipótesis equiprobable ya que cuando salta una alarma en la UCI se ha querido ponderar con más peso.

4.8.2. *LogisticRegression*

El segundo método importante, también procedente de *Sklearn*, es *LogisticRegression*, la función que se ha utilizado para el modelo basado en regresión logística. [36]

No obstante, en este TFG, por simplificar y porque no se trataba del objetivo principal, se ha optado por no modificar los parámetros por defecto de *LogisticRegression*, que son los siguientes:

- ***penalty***: La norma de la penalización. Por defecto L2.
- ***dual***: Formulación dual o primaria. *False* cuando hay más muestras que características. Por defecto *False*.
- ***tol***: Tolerancia para detener los criterios. Por defecto $1x10^{-4}$

- ***C***: Inversa de la fuerza de regularización. Valores pequeños especifican una regularización más fuerte. Por defecto 1.0.

- ***fit_intercept***: Especifica si se tiene que añadir alguna constante(sesgo, intersección...). Por defecto *True*.

- ***intercept_scaling***: Cuando se usa *liblinear* y self.fit_intercept está en True. En este caso, x se convierte en [x, self.intercept_scaling], es decir, se agrega una característica "sintéticaçon un valor constante igual a intercept_scaling. Por defecto 1,0

- ***class_weight***: Pesos asociados con las clases en la forma: class_label: weight. Si no se da, se supone que todas las clases tienen peso uno.

- ***random_state***: Se tiene la posibilidad de establecer una semilla para los números pseudoaleatorios. Por defecto esta a *None*.

- ***solver***: Algoritmo empleado para la optimización de la solución. Para conjuntos de datos pequeños *liblinear* es una buena opción y es la que viene por defecto. Existen otras configuraciones como: *newton-cg* o *sag* para problemas de multiclasificación.

- ***max_iter***: Número máximo de iteraciones. Útil sobre todo para los los modos de resolución *newton-cg* o *sag*, que son algoritmos más complejos. Por defecto viene fijado a 100.

- ***multi_class***: Puede ser *ovr* o *multinominal*. Si la opción es *ovr* se trata de un problema binario. El valor por defecto es *ovr*.

- ***verbose***: Posibilidad de incrementar el número de mensajes que ofrece el módulo. Por defecto viene con el valor 0.

- ***warm_start***: Por defecto en falso. Cuando se selecciona *True*, reutiliza la solución de la llamada anterior como ajuste de inicialización. No es útil para *liblinear*.

- ***n_jobs***: Número de CPUs que ejecutan en paralelo. Como el modelo planteado en este problema es sencillo, se queda en el valor por defecto, 1.

4.9. *XgBoost*

Este módulo de Python no pertenece ya a *Sklearn*. Ahora se explican los detalles (parámetros) que se pueden configurar en el clasificador que se utiliza.

Para poder llevar a cabo este TFG, se ha necesitado importar el clasificador *XGB-Classifier* de la libreria *xgboost*. Como se ha visto en los dos modelos anteriores, *XGB-Classifier* también dispone de varios parámetros configurables, y nos centraremos especialmente en tres:

- ***n_estimators***: Número de árboles en el modelo. En este TFG se ha probado con un total de 13 posibilidades, en saltos de 30 árboles: [20, 50, 80,..., 350, 380].
- ***max_depth***: Tamaño de los árboles de decisión. También se suele llamar número de capas o profundidad/frondosidad del bosque. En este trabajo se ha probado también varias opciones: [2, 4, 6, 8].
- ***learning_rate***: Tasa de aprendizaje. Cómo de rápido aprende el modelo. Es necesario fijarlo, ya que si se no se configura, este tipo de modelos aprende demasiado rápido y adolecería de *overfitting* [31] También se evalúan diferentes valores: [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3].

Como ya se ha mencionado, se entrena el modelo con varios de estos parámetros, escogiendo diferentes conjuntos para entrenamiento y *test*. Esto es necesario para evitar el problema del *overfitting* y esta técnica se llama validación cruzada o *cross-validation* (CV) y se procede a explicar el concepto y su implementación en Python.

4.10. Validación cruzada

Cross-Validation es una técnica muy extendida a la hora de intentar eliminar el *overfitting* en los problemas de *machine learning*. Consiste en dividir el conjunto de datos en dos, de entrenamiento y de prueba, y en cada experimento nuevo, variar esos dos conjuntos, es decir, escoger otro conjunto distinto de entrenamiento y de *test*.

Aunque es una buena solución que ayuda a tener más información, esto lleva el coste asociado de tener una alta carga computacional, lo que hace que el algoritmo que se está evaluando ralentice muy notablemente su velocidad de ejecución.

Existen diferentes formas de llevar a cabo esta técnica: [37]

- **Validación cruzada de K iteraciones** *(K-fold cross-validation)*: En cada iteración, se elige un subconjunto de datos de entrenamiento diferente k, y se valida sobre el subconjunto de test k-1.
- **Validación cruzada aleatoria**: la elección del subconjunto de datos de entrenamiento y de test se hace de forma aleatoria.
- **Validación cruzada dejando uno fuera** *(Leave-one-out cross- validation, LOOCV)*: se tiene un solo dato de prueba y el resto de entrenamiento. Se repite el modelo K veces, dejando cada vez un dato diferente fuera.

Es este trabajo se ha utilizado el primer método conocido como *K-fold cross-validation* . Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba [37]. Su contrapeso es que es de los más lentos a nivel computacional.

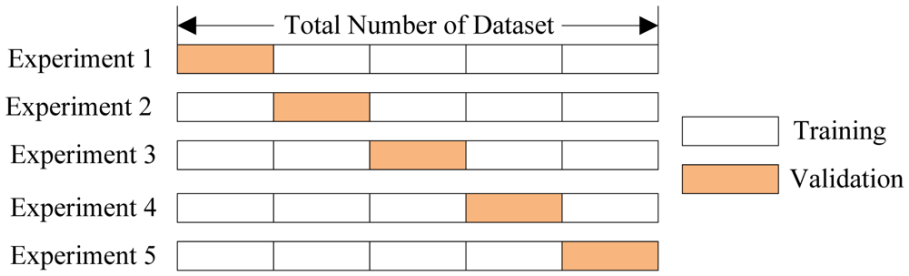

Figura 4.10: Ejemplo de un entrenamiento basado en *K-fold cross-validation.*

Para entrenar poner en práctica el entrenamiento con *cross-validation* se utilizará la función 'GridSearchCV' perteneciente a la librería sklearn.model_selection. Como parámetros se le han pasado los siguientes:

1. **model**: XGBClassifier.

2. **param_grid**: Conjunto de parámetros con: *n_estimators*, *max_depth*, *learning_rate*.

3. **scoring**: neg_loss_loss. Función de pérdida con forma logarítmica.

4. **n_jobs**: -1. Número de CPUs que ejecutan en paralelo. Cuando está en '-1' significa que ejecutan todos los procesadores de la máquina en paralelo.

5. **cv**: Validación cruzada que se desee definir. En este caso, k = 8.

4.11. Esquema de modelo de aprendizaje

En resumen, se dispone de una base de datos con un total de **750 pacientes**. De cada uno de esos pacientes, están medidas **3-4 señales** biológicas, en su mayoría derivaciones de ECG, ABP, PPG o la respiración del paciente.

Por seguir una coherencia, se han seleccionado los pacientes de los que se disponen 4 señales.

Estas señales son divididas en **ventanas** de un minuto para facilitar su manejo ya su vez poder caracterizar a los pacientes con más detalle del que se hubiese obtenido procesando las señales en su totalidad.

Después se aplica ***Principal Component Analysis*** **(PCA)** a cada ventana. Una vez hecho esto, disponemos de un vector de 20 *eigenvalues* por paciente que será el **vector de características** con el que, a partir de ahora, está definido cada paciente.

Una vez conseguido este vector, será el que se utilice de entrada para los algoritmos de *machine learning* propuestos. Una vez arrojen sus resultados *xgboost*, *naive Bayes* o regresión logística, ya se podrán tomar mejores decisiones sobre la veracidad o no de las alarmas, que era el objetivo principal de este TFG.

En la figura 4.11 se intenta concentrar la información aquí explicada en un sencillo esquema.

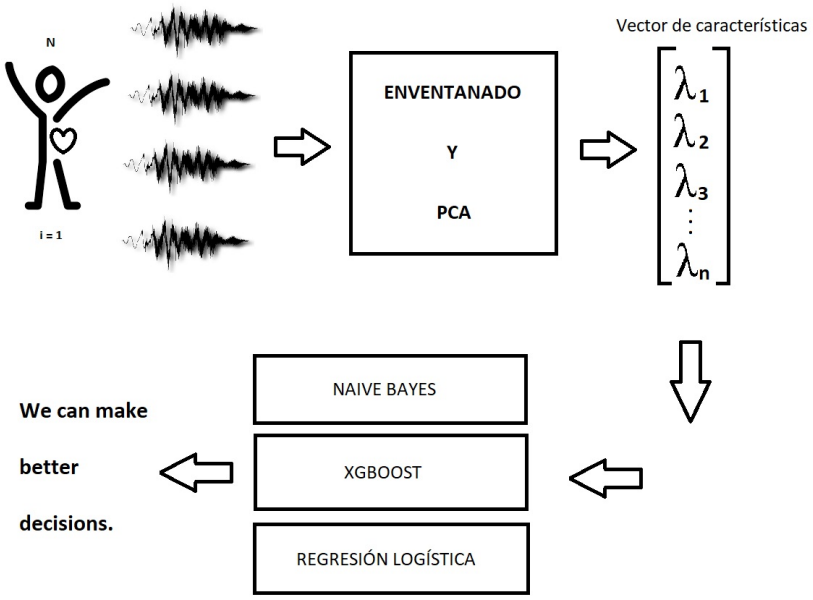

Figura 4.11: Esquema/resumen del modelo de aprendizaje seguido.

5 Análisis y resultados

5.1. Resultados

5.1.1. Matriz de confusión

La matriz de confusión es una buena técnica para resumir el rendimiento de un algoritmo de clasificación. [38] Más lo es aún para problemas de clasificación binaria, dónde es bastante sencillo observar como se comporta el modelo.

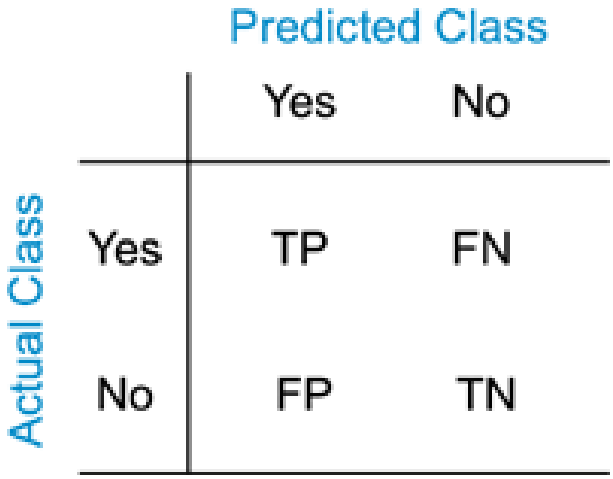

Figura 5.1: Matriz de confusión de dimensiones 2x2.

Asociados a esta matriz de confusión se pueden extraer distintos resultados. En este

TFG, la clase positiva corresponde con que la alarma era real y la negativa una falsa alarma. De esta forma la matriz sería:

- **TP** (*True positive*): Se estaría acertando sobre la existencia de una arritmia.
- **FN** (*False negative*): La predicción es que la arritmia es negativa, cuando si que era una alarma real.
- **FP** (*False positive*): Se estima que hay arritmia, cuando no la hay. Más comúnmente se la llama: *probabilidad de falsa alarma (FA).*
- **TN** (*True negative*): La predicción es que no existe arritmia, y efectivamente, no la hay.

Ahora que ya se sabe que significa cada apartado, se procede a comparar las matrices de confusión del ejemplo con *xgboost*, *naive bayes* y regresión logística.

El primer algoritmo a analizar va a ser regresión logística. Como podemos ver en la figura 5.2, la regresión logística no permite calcular, por ejemplo, *True negatives*. Tan solo ofrece los aciertos en la clase '1', que en este caso, es *True Alarm*.

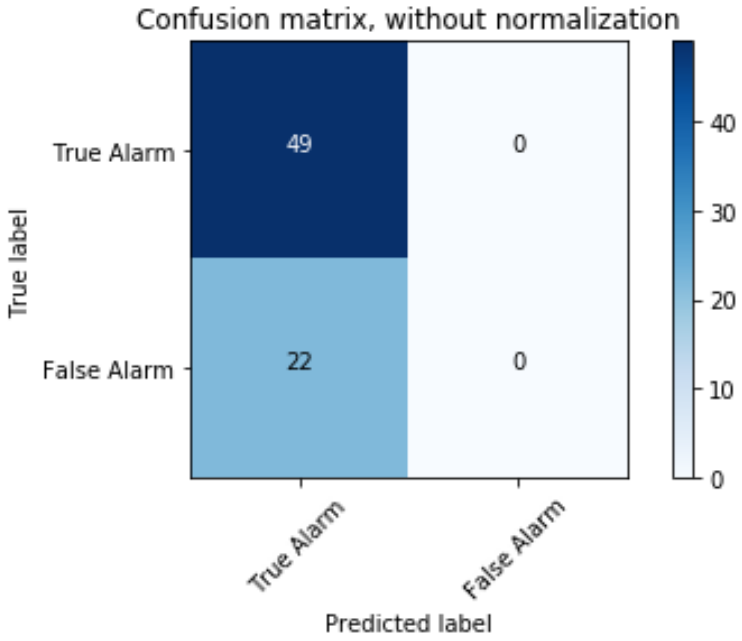

Figura 5.2: Matriz de confusión generada con el algoritmo de regresión logística.

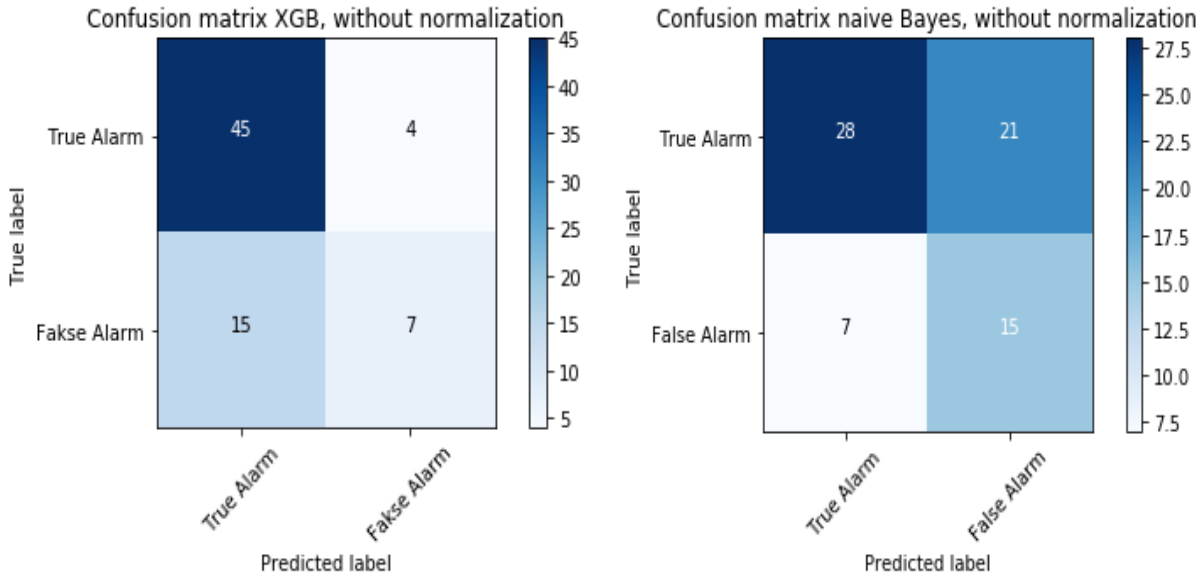

Figura 5.3: Matriz de confusión de *xgboost* vs *naive Bayes*.

En la siguiente figura se comparan los resultados del algoritmo propuesto basado en árboles de decisión (*xgboost*) y de *naive Bayes*.

Con estas dos matrices se pueden observar varias cosas:

1. En el primer caso se aciertan más los positivos en *xgboost*, es decir, cuando hay arritmia, el algoritmo propuesto la detecta bastante bien.

2. Por el contrario *naive Bayes* acierta más los *true negative*, casos en los que la alarma saltó, no era real.

3. En los fallos se aprecia como *xgboost* ofrece menos casos de falsos negativos, es decir, pocas veces ocurre que prediga ausencia de arritmia cuando si que existe. Mientras que *naive Bayes* ofrece mejores resultados de falsos positivos o falsa alarma.

También es de interés comparar valores numéricos. Para ello, existe un módulo en *sklearn* para calcular métricas en los algoritmos de *machine learning*, el módulo se llama: *sklearn.metrics*

5.1.2. Medidas de desempeño en clasificación

Antes de mostrar los resultados, es necesario explicar brevemente el significado de cada uno:

- ***Accuracy***: Precisión del modelo para detectar aciertos. Se calcula como:

$$\frac{TP + TN}{TOTAL} \tag{5.1}$$

- ***Precision***: Capacidad de n predecir como '1', una muestra que fuese un '0'. Se calcula de la siguiente forma:

$$\frac{TP}{TP + FP} \tag{5.2}$$

- ***Recall***: capacidad de clasificar las muestras como '1'. Se calcularía:

$$\frac{TP}{TP + FN} \tag{5.3}$$

- ***F1 score***: se promedia de forma ponderada entre las métricas *precision* y *recall*. Se calcula como:

$$2 * \frac{precision * recall}{precision + recall} \tag{5.4}$$

Para comparar de manera clara ambos algoritmos, se muestran sus métricas en la siguiente tabla:

	Accuracy	*Precision*	*Recall*	*F1*
Xgboost	0.7324	0.6364	0.3182	0.4242
naive Bayes	0.6056	0.4167	0.6818	0.5172

Cuadro 5.1: Tabla de métricas de ambos algoritmos.

Se observa como el algoritmo propuesto alcanza buenos resultado en *accuracy* y *precision*, mientras que, por otra parte, *naive Bayes* consigue mejor puntuación en *recall*, que son compensados en *F1* ya que es una medida que se pondera también con *precision*.

Por último, se propone un nuevo algoritmo para intentar conseguir mejores resultados. Se trata de un modelo sencillo basado en la comparación de *xgboost* y *naive Bayes*.

5.1.3. Combinación de algoritmos

A la vista de los resultados obtenidos con los modelos de Xgboost y Naïve Bayes, propusimos la combinación de ambos modelos en uno sólo. Esto es así, pues cada uno de los modelos parecía funcionar bien en clasificar pacientes en el que el otro modelo fallaba.

El algoritmo de combinación que propusimos es bastante sencillo, y se explica a continuación.

Este último algoritmo se basa en comparar las estimaciones de los evaluados anteriormente. Existen 3 casos posibles que se explican a continuación:

1. *Xgboost* y *naive Bayes* predicen que la salida es '1'. Luego el algoritmo que se propone elegirá '1'.

2. *Xgboost* y *naive Bayes* ofrecen como salida un '0'. Luego la salida de este modelo será un 0.

3. *Xgboost* y *naive Bayes* ofrecen salidas distintas. En este caso, se comparán las probabilidades con las que predicen ese valor y se asigna, como salida de este método la más alta de ambas. Por ejemplo, *xgboost* da como salida un '0' con una probabilidad de 0.687 y *naive Bayes* ofrece un '1' con probabilidad 0.574. Como es mayor la probabilidad de *xgboost*, este modelo pondrá un '0'.

En la siguiente tabla se pueden comparar los resultados de los tres modelos, *xgboost*, *naive Bayes* y el último basado en la combinación de ambos.

	Accuracy	*Precision*	*Recall*	*F1*
Xgboost	0.7324	0.6364	0.3182	0.4242
naive Bayes	0.6056	0.4167	0.6818	0.5172
Combine model	0.6479	0.4571	0.7273	0.5614

Cuadro 5.2: Tabla de métricas de ambos algoritmos.

Como se puede apreciar en la tabla 5.2, en términos de *accuracy* y *precision* el resultado es un paso intermedio entre *xgbost* que es el modelo que mejores resultados ofrece y *naive Bayes*. Por el contrario el algoritmo propuesto mejora notablemente las prestaciones en *recall* y en *F1 score*.

6 Conclusiones y líneas futuras

En el primer capítulo de este TFG, se establecieron una serie de objetivos:

1. Simplificar la base de datos extrayendo características más fácilmente computables.
2. Procesar las señales.
3. Clasificación de alarmas en falsas o positivas y evaluar el rendimiento de los algoritmos propuestos.

Los objetivos se han llevado a cabo con éxito:

1. Se ha aplicado una técnica de análisis exploratorio de los datos como es PCA, con el fin de tener datos más manejables. Cabe recordar que en la base de datos existían 750 pacientes cada uno de ellos con 3-4 señales de unas 80.000 muestras cada una. Ahora, habiéndose utilizado aproximadamente la mitad de la base de datos (los pacientes con 4 señales) se han reducido las características a 20 dígitos normalizados por paciente.
2. Para poder realizar esto, previamente se tuvo que realizar un enventanado de las señales, para subdividir el problema y a su vez poder tener mejor caracterizadas a los pacientes, que si, por ejemplo, se hubiese aplicado PCA directamente.
3. Se han llevado a cabo los algoritmos, poniendo especial interés en los árboles de clasificación implementados con la librería *xgboost* y comparándolo con modelos que teóricamente funcionan bien para la clasificación binaria. También se analizaron sus

métricas, así como se expuso un nuevo algoritmo combinando las salidas de los que mejores resultados obtuvieron.

Como también se ha mencionado en este TFG, el problema abarcada más facetas de las que se han abordado,(ver que tipo de arrimia provocaba las alarmas, analizar las señales antes y después de la alarma).

La extensión temporal de haber intentado resolver esos problemas, así como su complejidad son las razones de no haberlos planteado en este TFG. No obstante, ese sería el camino para futuras investigaciones sobre este tema, que requiere sin duda de bastante interés.

Las futuras acciones para profundizar en la solución completa de este problema podrían ser:

- Probar con otra técnica de análisis exploratorio de datos, como por ejemplo, una técnica no lineal *Kernel Principal Componet Analysis*.
- Evaluar si el tamaño de la ventana es el óptimo. Realmente no existe certeza de que parte de la señal es la que más información tiene una vez aplicado PCA, luego se podría probar con ventanas más extensas o más cortas.
- Se podría evaluar la multiclasificación original del problema. Bien con el algoritmo propuesto *xgboost* o con otros algoritmos de clasificacion como redes neuronales por ejemplo.
- El nuevo algoritmo propuesto en último lugar, podría mejorarse ponderando, por ejemplo con mas peso las decisiones de *xgboost* que es un modelo más completo y que no adolece de *overfitting*. Por el contrario, *naive Bayes* en este TFG se ha utilizado como una herramienta comparativa. Es por ello que no se ha puesto más interés en este modelo.

Estas extensiones requieren de técnicas más complejas y de *machine learnig* y con más carga computacional, ya que los algoritmos tendrían que ser más complejos.

Todo el código escrito para este TFG se encuentra en el siguiente repositorio de GitHub: `https://github.com/riojafernando/TFG`

Bibliografía

[1] PHYSIONET. Challenge 2015 Introduction, https://physionet.org/challenge/2015/#introduction,

[2] KAGGLE. About Kaggle, https://www.kaggle.com/kaggle,

[3] WIKIPEDIA. PCA, https://en.wikipedia.org/wiki/Principal_component_analysis,

[4] WIKIPEDIA. PCA, https://en.wikipedia.org/wiki/Principal_component_analysis,

[5] WIKIPEDIA. Electrocardiograma, https://es.wikipedia.org/wiki/Electrocardiograma,

[6] SYNERGIC PARTNERS. Una breve historia del machine learning, http://www.synergicpartners.com/una-breve-historia-del-machine-learning/,

[7] APRENDEMACHINELEARNING. Qué es el overfitting y cómo solucionarlo, http://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/,

[8] MACHINE LEARNING MASTERY. Overfitting and Underfitting With Machine Learning Algorithms, https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/,

[9] PHYSIONET. About PhysioNet, https://physionet.org/resource.shtml,

[10] PHYSIONET. Reducing False Arrhythmia Alarms in the ICU: the PhysioNet/Computing in Cardiology Challenge 2015, https://physionet.org/challenge/2015/,

[11] WIKIPEDIA. Electrocardiograma, http://www.my-ekg.com/generalidades-ekg/derivaciones-cardiacas.html,

[12] MY EKG. Derivaciones cardiacas, https://es.wikipedia.org/wiki/Electrocardiograma,

[13] MY EKG. Ondas del electrocargiograma, http://www.my-ekg.com/generalidades-ekg/ondas-electrocardiograma.html,

[14] MEDLINEPLUS. Enciclopedia Médica, https://medlineplus.gov/spanish/ency/article/001101.htm,

[15] PHYSIONET. Challengue 2015 Rules and Deadlines, https://www.physionet.org/challenge/2015/#rules-and-deadlines,

[16] TURMERO, P. Mediciones fotopletismográficas, http://www.monografias.com/trabajos104/mediciones-fotopletismograficas/mediciones-fotopletismograficas.shtml,

[17] MEDLINEPLUS. No invasivo, https://medlineplus.gov/spanish/ency/article/002269.htm,

[18] VALDERAS, MT. VALLVERDÚ, M. CAMINAL, P. «Extracción de la señal de respiración a partir del electrocardiograma», Actas de las XXXVI Jornadas de Automática, Bilbao, 2015,

[19] James, G., Witten, D., Hastie, T., and Tibshirani, R. An Introduction to Statistical Learning: with Applications in R. Springer Science and Business Media, 2013.

[20] WIKIPEDIA. Estadística Inferencial, https://es.wikipedia.org/wiki/Estad\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{\OT1\i\global\mathchardef\accent@spacefactor\spacefactor}\accent19\OT1\i\

egroup\spacefactor\accent@spacefactor\futurelet\@let@token\penalty\@M\
hskip\z@skipstica_inferencial,

[21] SRIVASTAVA, T. Difference between machine learning and statistical modeling. https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/

[22] Müller, A. C., and Guido, S. Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, Inc., 2016.

[23] GONZÁLEZ, A. Conceptos básicos de Machine Learning, http://cleverdata.io/conceptos-basicos-machine-learning/

[24] BUSINESS ANALYTICS. An Introduction to Clustering & different methods of clustering, https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

[25] GRANÉ, A. Análisis de Componentes Principales, http://halweb.uc3m.es/esp/Personal/personas/agrane/ficheros_docencia/MULTIVARIANT/slides_comp_reducido.pdf

[26] DE LA FUENTE, FERNÁNDEZ, S. Componentes Principales, http://www.fuenterrebollo.com/Economicas/ECONOMETRIA/MULTIVARIANTE/ACP/ACP.pdf

[27] MALAGÓN, LUQUE, C. Clasificadores bayesianos. El algoritmo Naive Bayes, https://www.nebrija.es/~cmalagon/inco/Apuntes/bayesian_learning.pdf

[28] WIKIPEDIA, Árboles de decisión, https://es.wikipedia.org/wiki/\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{A\global\mathchardef\accent@spacefactor\spacefactor}\accent19A\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\penalty\@M\hskip\z@skiprbol_de_decisi\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\accent19o\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\penalty\@M\hskip\z@skipn

[29] CUENCA, J. Ventajas y desventajas. Teoría de decisiones y árboles de decisión. http://teoriadedecisionesyarbolesdedecision.blogspot.com/

[30] WIKIPEDIA. XGBoost. https://en.wikipedia.org/wiki/Xgboost

[31] BROWNLEE, J, XGBoost With Python. Gradient Boosted Trees with XGBoost and scikit-learn. Machine Learning Mastery. 2016

[32] WIKIPEDIA. Logistic Regression. https://en.wikipedia.org/wiki/Logistic_regression

[33] MORAL, PELÁEZ, I. Modelos de regresión: lineal simple y regresión logística. http://www.revistaseden.org/files/14-cap%2014.pdf

[34] DE LA FUENTE, FERNÁNDEZ, S. Regresión Logística. 2011. http://www.estadistica.net/ECONOMETRIA/CUALITATIVAS/LOGISTICA/regresion-logistica.pdf

[35] SCIKIT-LEARN. sklearn.naive_bayes.GaussianNB. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[36] SCIKIT-LEARN. sklearn.linear_model.LogisticRegression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[37] WIKIPEDIA. Validación Cruzada. https://es.wikipedia.org/wiki/Validaci\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\accent19o\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\penalty\@M\hskip\z@skipn_cruzada

[38] BROWNLEE, J. What is a Confusion Matrix in Machine Learning. https://machinelearningmastery.com/confusion-matrix-machine-learning/