



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE LA SALUD

INTELIGENCIA ARTIFICIAL APLICADA A UN SISTEMA DE MONITORIZACIÓN DEL NIVEL DEL DOLOR

Realizado por

Julia García Flores

Dirigido por

Juan Pedro Domínguez Morales

Lourdes Durán López

Departamento

Arquitectura y tecnología de computadores

Sevilla, junio del 2023

I- AGRADECIMIENTOS

*A mi compañero de proyecto, a mis profesores y
a mis amigos y familiares por su apoyo
emocional y comprensivo, y el ánimo
depositado en mi durante la
elaboración del trabajo.*

II-RESUMEN

Este proyecto está dirigido para todas aquellas personas que sufren dolor. Hoy en día, todavía no hay ningún método que nos diga con certeza el nivel de dolor que se puede llegar a padecer. Este nuevo sistema, ayudará a la comunicación médico-paciente, lo que permitirá conocer mejor los niveles de dolor que podemos llegar a padecer. De esta forma, los médicos podrán amoldar sus métodos a sus pacientes con mayor facilidad en diferentes ámbitos de la medicina, desde, una rehabilitación, para aquellos que sufran algún tipo de rotura o daño que necesite de esta práctica, hasta, para una operación en la que el paciente este sedado. También, vemos gran aplicabilidad de este sistema en personas con dificultades a la hora de comunicarse como pueden ser aquellas con enfermedades como la parálisis cerebral, ELA o cualquier otra que provoque la pérdida del habla.

El objetivo de este proyecto es elaborar un sistema que permita la monitorización de un paciente a partir de la obtención de datos proporcionados por señales fisiológicas, tales como la frecuencia cardíaca, la frecuencia respiratoria, la respuesta galvánica de la piel y la temperatura corporal. Estos parámetros fisiológicos serán recogidos en un conjunto de entrenamiento que nos permitirá llevar a cabo un estudio de investigación, utilizando como base principal la *Inteligencia Artificial*. Es decir, analizaremos la información que proporciona dicho conjunto de entrenamiento utilizando técnicas de preprocesamiento, algoritmos de *deep learning* y *machine learning* que nos permitan entrenar una serie de modelos que generen variables indicadoras del mejor camino para clasificar estas señales fisiológicas en función del nivel de dolor que padece el paciente, siguiendo los estándares médicos establecidos en las escalas métricas, como mecanismo para conseguir un sistema de manejo, diagnóstico y tratamiento del nivel de dolor.

III- ABSTRACT

This project is aimed at all those who suffer from pain. Nowadays, there is still no method that can accurately tell us the level of pain we may experience. This new system will help improve the doctor-patient communication, allowing for a better understanding of the levels of pain that patients may endure. In this way, doctors will be able to adapt their methods to their patients more easily in different areas of medicine, from rehabilitation for those who have suffered some kind of injury or damage that requires this practice, to sedated operations. Additionally, we see great applicability of this system in people with communication difficulties, such as those with diseases like cerebral palsy, ALS, or any other condition that causes speech loss.

The objective of this project is to develop a system that allows monitoring a patient by obtaining data from physiological signals, such as heart rate, respiratory rate, galvanic skin response, and body temperature. These physiological parameters will be collected in a training set that will enable us to conduct a research study, using Artificial Intelligence as the main basis. In other words, we will analyze the information provided by this training set using preprocessing techniques and machine learning to train a series of models that generate indicative variables for classifying these physiological signals according to the patient's level of pain, following established medical standards on metric scales. Neural networks will be used as a mechanism to achieve the expected results.

IV- ÍNDICE

I- Agradecimientos	2
II-Resumen	3
III- Abstract.....	4
IV- Índice.....	5
V- Índice de figuras.....	7
VI- Índice de cuadros.....	8
BLOQUE 1: DESCRIPCIÓN DEL PROYECTO.....	9
1- Introducción y motivación	9
1.2- Objetivos del proyecto.....	13
1.2.1- Objetivos educativos.....	13
1.2.2- Objetivos técnicos.....	14
1.3- Estado del arte	15
1.3.1- PMD_200.....	15
1.3.2- Painometer V2	17
1.3.3- Algómetro analógico FPK 60	18
1.3.4- Patente: US5533514A	20
1.4. Elicitación de requisitos	23
1.4.1- Requisitos funcionales	23
1.4.2- Requisitos no funcionales	23
1.4.3- Requisitos de información	24
BLOQUE 2: EJECUCIÓN DEL PROYECTO.....	26
2.1. Diseño del sistema	26
2.2. Implementación	29
2.2.1- Metodología	29
2.2.2- Tecnologías empleadas	31
2.2.2.1- Bloque Instrumental	31
2.2.2.2- Bloque Inteligente.....	33
2.2.2. DESARROLLO	40
2.2.2.2. Implementación	52
2.2.2.3- Resultados.....	54
2.2.2.3.2- Red Neuronal	61
2.3. Pruebas del sistema	78
BLOQUE 3: PLANIFICACIÓN DEL PROYECTO.....	80

3.1- Planificación temporal	80
3.1.1- Planificación temporal inicial	80
3.1. 2. Planificación temporal final.....	82
3.2- Planificación financiera	82
3.2.1- Planificación financiera inicial.....	82
3.2.1.1- Recursos materiales	83
3.2.1.2- Recursos humanos	84
3.2.2.3- Gasto total inicial	91
3.2.2- Planificación financiera final	91
3.2.2.1-Recursos materiales	91
3.2.2.2-Recursos humanos	94
3.2.2.3-Gasto total final.....	97
3.3- Estudio de mercado	100
3.3.1- Clientes potenciales	100
3.3.2. PLAN DE COMERCIALIZACIONES	101
VII. Conclusiones	105
VIII. Trabajo futuro	107
IX. Bibliografía	108
X. Anexos:.....	111
X.I. Programación en entorno Google Colab.....	111
X.II. Programación en entorno Weka	135

V- ÍNDICE DE FIGURAS

Figura 1: Dispositivo PMD_200.....	16
Figura 2: Aplicación Painometer V2	18
Figura 3: Algómetro analógico FPK60	19
Figura 4: Diagrama de bloques del sistema	26
Figura 5: Leyenda del diagrama de bloques del sistema.	27
Figura 6: Sensor de pulso.....	31
Figura 7: Micrófono.....	31
Figura 8: Sensor de respuesta galvánica.....	32
Figura 9: Sensor de temperatura corporal.....	32
Figura 10:Arduino.	32
Figura 11: Cheat-sheet de scikit-learn.	37
Figura 13: Esquema de los sprints.....	40
Figura 14: Estructura de red neuronal.....	43
Figura 15: Funciones de activación de la neurona.	44
Figura 16: Ejemplo de red neuronal.....	45
Figura 17: Codificación ordinal del atributo nivel de dolor.	48
Figura 18: Codificación OneHot del atributo nivel de dolor.....	49
Figura 19: Rango de datos por cada atributo.	50
Figura 20: Red neuronal del proyecto.	53
Figura 21: Valores nulos del conjunto de entrenamiento.....	54
Figura 22: Gráficas del escalado de la frecuencia cardiaca.	57
Figura 23: Gráficas del escalado de la frecuencia respiratoria.	58
Figura 24: Gráfica del escalado de la respuesta galvánica.	58
Figura 25: Gráfica del escalado de la temperatura corporal.	59
Figura 26: Normalización de los datos con escalado min-max.....	60
Figura 27: Análisis de correlación de las variables.....	60
Figura 28: Evolución del error durante el entrenamiento.....	62
Figura 29: Matriz de confusión de la red neuronal.	62
Figura 30: Esquema de tipos de aprendizaje supervisado en machine learning.	63
Figura 31:Esquema del aprendizaje supervisado.....	64
Figura 34: Ejemplo de una iteración de validación cruzada.....	65
Figura 35: Algoritmo de iteración de validación cruzada.	66
Figura 36: Ejemplo de resultado del algoritmo de regresión lineal.	67
Figura 37:Conjunto de datos en Weka.	70
Figura 38: Opciones de clasificación en Weka.....	71
Figura 39: Comparación de los atributos en función del nivel de dolor.	72
Figura 40: Resultados del modelo de regresión lineal.....	74
Figura 12: Screen1 y Screen2 de la aplicación Lepasy en funcionamiento.	77
Figura 41: Diagrama de Gantt. Planificación temporal inicial.	81
Figura 42: Diagrama de Gantt. Planificación temporal final.....	82
Figura 43: Comparación del gasto total que supone cada tipo de recurso.	98
Figura 44:Meta de comparación de los gastos empleados en cada recurso a futuro. ..	99
Figura 45: Análisis DAFO del proyecto.	102

VI- ÍNDICE DE CUADROS

Cuadro 1: Comparación de los productos incluidos en el estado del arte	21
Cuadro 2: Requisitos funcionales.....	23
Cuadro 3: Requisitos no funcionales.....	24
Cuadro 4: Requisitos de información.	25
Cuadro 5: Sensores del bloque instrumental.	32
Cuadro 7: Funciones de scikit-learn.....	37
Cuadro 9: Funciones para las etapas de preprocesamiento.	52
Cuadro 10: Funciones para el escalado de los datos.....	52
Cuadro 11: Características de la red neuronal.	53
Cuadro 12:Estudio estadístico de los datos faltantes.....	55
Cuadro 13: Categorías de la codificación OneHot.	56
Cuadro 14: Nivel de correlación entre las variables y el nivel de dolor.....	60
Cuadro 15: Métricas del modelo de la red neuronal.	62
Cuadro 16: Características más importantes de los algoritmos.....	65
Cuadro 17:Equivalencia de los modelos reales en Weka.....	72
Cuadro 18: Información sobre cada uno de los atributos.	73
Cuadro 19: Entornos de programación para cada modelo.	74
Cuadro 20: Estudio de los valores obtenidos por los modelos.	75
Cuadro 8: Botones y etiquetas de lepasy.....	76
Cuadro 21: Carga y procesamiento de los datos.	79
Cuadro 22: Estudio de los modelos.	79
Cuadro 23: Horario del calendario del proyecto.	80
Cuadro 24: Precios de los recursos materiales empleados en la planificación financiera inicial.	83
Cuadro 25: Precio de licencia software.....	84
Cuadro 26:Salario de los recursos humanos del proyecto.	85
Cuadro 27: Salario de recursos humanos con la cotización de la seguridad social....	86
Cuadro 28: Cálculo del gasto de los recursos humanos en función de las horas estimadas.	90
Cuadro 29: Gasto total estimado del proyecto.	91
Cuadro 30: Precios de los recursos materiales empleados en la planificación financiera final.....	93
Cuadro 31: Precio de las licencias reales.	93
Cuadro 32: Precio de los recursos humanos reales con la cotización en la seguridad social.	94
Cuadro 33: Calculo del gasto real de los recursos humanos.....	97
Cuadro 34: Gasto total del proyecto.....	98

BLOQUE 1: DESCRIPCIÓN DEL PROYECTO.

1- INTRODUCCIÓN Y MOTIVACIÓN

En la actualidad, los sistemas biomédicos inteligentes van adquiriendo un peso cada vez mayor dentro del complejo hospitalario. El sistema sanitario se está adaptando a un cambio tecnológico que avanza a pasos de gigantes. Cada vez surgen nuevas ideas sobre cómo poder recoger el mayor número de información sobre el paciente en el menor tiempo posible con el fin de desarrollar, definir o detectar diagnósticos o tratamientos de la forma más eficiente y mucho más segura.

Donde antes, si el doctor no encontraba explicación a los síntomas de un paciente, podía tardar meses en analizar y contrastar información hasta dar con el diagnóstico y su posterior tratamiento, ahora, hay muchas bases de datos en las que simplemente introduciendo una serie de palabras clave el doctor obtendrá una serie de información que le facilitará poder obtener un diagnóstico más temprano que permita que se pueda empezar con el tratamiento lo antes posible. Ya sabemos que muchas veces en la medicina el tiempo es muy importante, cuanto más avanzada sea la enfermedad, más complicada será curarla.

Estas personas sufren grandes dolores ya sea por, un accidente, la rehabilitación, o el postoperatorio. Estar pendientes en todo momento, administrarles sus analgésicos para que sufran el mínimo dolor posible, ser amable y atento es una muy buena forma de dignificarlos, poniéndonos en su piel, e intentando comprender su dolor. Todos sabemos que mañana podemos ser nosotros mismos los que estemos en ese lugar, ya que por desgracia estas cosas ocurren todos los días y a todas horas. De hecho, según el Portal Estadístico del Sistema Nacional de Salud, España dispone de 468 hospitales en los cuales se producen aproximadamente 4 millones de ingresos hospitalarios cuyo coste medio de cada hospitalización es de 6398 de euros, así como la realización de 82 millones de consultas cuyo tiempo de espera medio para primera consulta es de 95 días [1].

Estos datos son alarmantes puesto que 95 días es mucho tiempo para que una persona pueda ser entrevistada por un médico para informar de su situación sanitaria incluso sabiendo que no todas las personas son tratadas en hospitales, hay muchas que debido a su situación son tratadas en sus propias casas, donde también es importante que estén atendidas lo mejor posible. Por ello, cada vez es mayor el número de sistemas de

monitorización a distancia que permiten al médico el estado del paciente sin necesidad de su asistencia al hospital, lo que reduciría el tiempo de espera interconsulta.

En este proyecto, nos centraremos en crear un sistema de monitorización para personas que sufren dolor, este aparato nos permitirá conocer el nivel de dolor que sufre un paciente para poder comprender sus necesidades, más allá de saber las horas a las que hay que administrarle sus medicamentos.

El dolor, como todos sabemos, no solo es un sentimiento, es una respuesta del sistema nervioso que se produce cuando algo no va bien. Normalmente se sienten pinchazos, hormigueo, ardor, picor o molestia.

El dolor puede tener distintas intensidades, por eso hay veces que es casi inexistente, y en muchas otras ocasiones es insopportable. Conocer el nivel de dolor de las personas es muy complicado, ya que, hasta que no lo sientes por ti mismo, no sabes bien de qué tipo de dolor hablamos. Por ello, podemos decir que el nivel de dolor es una variable subjetiva, que depende de muchos factores.

Todos hemos sentido dolor alguna vez en nuestra vida. De hecho, según indican algunos estudios el 29% de la población española no hospitalizada sufre algún tipo de dolor, y el 17% padece dolor crónico. [2]

Si solamente en España hay este número tan elevado de personas afectadas en su día a día por el dolor, esta cifra a nivel mundial será inimaginable.

Hoy en día, los doctores emplean distintos métodos para intentar hacerse una idea lo más cercana posible, para poder tratar al paciente y sobre todo a la hora de administrar medicamentos. El método comúnmente utilizado, es la escala visual analógica (EVA), en la que mediante una serie de dibujos entre los que se distinguen 6 niveles de dolor, situándose el nivel inferior pegado a la izquierda y el máximo nivel en la derecha, el paciente señala de manera subjetiva cual piensa que se corresponde al nivel de dolor que padece en ese instante.

Otro método también utilizado es el empleo de lo que comúnmente se conoce como dolorímetro o algómetro. Este instrumento es capaz de medir el umbral del dolor, utilizando la fuerza aplicada sobre la zona afectada, de tal manera que el medico va aplicando presión y cuando el paciente no la aguanta más se lo comunica. Pero una vez más, al igual que en la escala EVA, hasta qué punto esto llega a ser fiable del todo.

Como podemos observar, estos métodos no son precisamente exactos, muchas veces el dolor se puede cubrir o intensificar dependiendo del paciente, y no le vamos a poner

un polígrafo para saber hasta qué punto dice la verdad. Aunque, la mayoría de los pacientes no mentirían sobre este tema, sigue siendo algo subjetivo. Por ello, en este estudio analizaremos las variables fisiológicas más relacionadas con el nivel de dolor, según indiquen los profesionales para poder obtener resultados más concretos. De esta forma, también ayudaríamos a que el personal sanitario o los familiares que se hagan cargo, detecten con más exactitud el dolor cuando el paciente no se pueda comunicar, ya sea por alguna enfermedad, o porque esté en algún proceso asistencial que impliquen estados de cero comunicaciones, como por ejemplo durante la sedación en una cirugía.

El dolor y la mente van ligados, ya que esta limitación puede llegar a afectar al vínculo con familiares y amigos, esto se debe a que en numerosas ocasiones ellos tendrán que adaptarse a las nuevas necesidades de la persona que padece el dolor, hasta tal punto que pueden llegar a sentirse una carga para sus amigos, familiares, compañeros de trabajo etc. Por todo esto tienden a desarrollar frustración, estrés e incluso depresión, lo cual no solo empeora normalmente los dolores, sino que también puede producir un aumento de la presión arterial, incremento de la frecuencia respiratoria y cardíaca, así como causar tensión muscular dando lugar a sentimientos de fatiga, problemas para dormir y cambios en el apetito, entre otros.

Nuestro sistema aparte de ser un dolorímetro, dispondrá de sensores que midan las señales vitales, para que los médicos puedan llevar un control exhaustivo, previniendo así al máximo todos los problemas de salud citados anteriormente para que el dolor sea lo más llevadero posible y no se agrave aún más.

La finalidad de este proyecto es conseguir que nuestra propuesta de dolorímetro será completamente autónomo, es decir, que no se necesite de la intervención del paciente para conocer su dolor.

Para desarrollar este instrumento utilizaremos una serie de sensores, de bajo coste, que midan señales fisiológicas para ser recogidas, procesadas y transformadas a través de un microcontrolador que proporcione datos de la frecuencia cardíaca, la respiración, respuesta galvánica de la piel y temperatura corporal, para definir un conjunto de entrenamiento que, posteriormente, serán analizados para determinar un patrón que demuestre que, dependiendo del rango en el que se encuentren los parámetros, el paciente tendrá dolor bajo, medio o alto.

Hoy en día, solo hay un aparato capaz de medir el nivel de dolor de manera independiente a la subjetividad de cada paciente, y debido a su elevado precio no es

accesible para la inmensa mayoría. Con este nuevo sistema también pretendemos que, en el futuro, cualquiera que lo necesite pueda tener un sistema así, ya sea en un hospital, como en su propia casa.

Mejorar el precio del sistema de monitorización del dolor, e intentar conseguir que sea lo más exacto posible, es uno de nuestros principales objetivos, y nuestra motivación, por supuesto, es mejorar al máximo la calidad de vida de las personas, que al final es la clave principal que nos impulsa a realizar este proyecto.

1.2- OBJETIVOS DEL PROYECTO

Los objetivos del proyecto son las metas que se pretenden conseguir en base a los conocimientos adquiridos a lo largo del desarrollo de este. A continuación, se presentan los dos tipos de objetivos presentes en este proyecto.

1.2.1- OBJETIVOS EDUCATIVOS.

Por un lado, aparece este tipo de objetivos que entran dentro de lo que se espera conseguir, de manera crítica, resolutiva y comunicativa, a lo largo de la realización de este proyecto. A continuación, se muestra una lista con estas metas:

- Conocer la estructura del microcontrolador Arduino UNO y sus componentes, así como el funcionamiento informático de este.
- Definición de los requisitos fundamentales que componen el sistema.
- Capacidad de selección y decisión de los elementos que van a formar parte del sistema, refiriéndose así al microcontrolador, sensores y componentes que en su totalidad dan lugar a un sistema de adquisición y procesamiento de señales biomédicas.
- Elaborar un circuito utilizando los elementos seleccionados, así como diagramas que muestren el funcionamiento de este sistema.
- Capacidad de programación del sistema, en lenguaje C ++, para que sea capaz de recoger señales y transformarlas a las unidades correspondientes en función del sensor que las obtenga y, a partir de ahí, realizar unas acciones concretas, todo ello desde el entorno de desarrollo de Arduino.
- Obtención de un conjunto de entrenamiento a partir los datos obtenidos del sistema en un formato concreto (.csv).
- Aplicar técnicas de análisis y preprocesamiento de los datos obtenidos en el conjunto de prueba.
- Capacidad de estudio y contraste de los datos obtenidos a partir de la aplicación de diferentes modelos sobre el conjunto de datos, comparando las ventajas y desventajas que presentan, a través de gráficos, tablas, matrices y datos estadísticos.
- Capacidad de decisión y justificación de los modelos más adecuados para la clasificación del nivel de dolor a partir de las variables biomédicas obtenidas a

través del sistema, en función de los datos de precisión y error, entre otros datos proporcionados por dichos algoritmos.

- Capacidad de planificación temporal y financiera aplicada al proyecto.
- Aprender a elaborar una memoria con fines de investigación siguiendo la normativa vigente y cumpliendo los entandares de estilos definidos.

1.2.2- OBJETIVOS TÉCNICOS

Por otro lado, los objetivos técnicos definen la finalidad de este proyecto a nivel de tecnológico y de herramientas que permitan conseguir el objetivo principal del proyecto.

A continuación, se muestra una lista con estas metas:

- Habilidad para el diseño y construcción de un sistema de bioinstrumentación utilizando elementos básicos y asequibles, tales como sensores, microcontrolador Arduino UNO, componentes analógicos y digitales, etc.
- Programación de variables biomédicas, tales como frecuencia cardíaca, frecuencia respiratoria, respuesta galvánica de la piel y temperatura corporal, a partir de los datos obtenidos por el sistema biomédico dentro del lenguaje de programación de Arduino.
- Capacidad de aprendizaje y empleo de técnicas que se ajusten al rango del problema y al entorno de desarrollo de Python, tales como: pandas, numpy, TensorFlow, Keras o matplotlib, entre otras librerías que permitan desarrollar la aplicación de algoritmos de IA al proyecto, prevaleciendo los resultados con mayor precisión posible.
- Análisis de datos, obtenidos al completar el estudio, representados a través de gráficos, tablas y matrices, entre otros, que permitan la definición de los resultados y conclusiones del proyecto.
- Implantación del sistema en una interfaz física que permita obtener los datos biomédicos, de personas seleccionadas para un estudio, de manera sencilla, fiable y ergonómica.

En resumen, el objetivo principal de este proyecto es la consecución de un sistema de monitorización del nivel de dolor en función de los datos proporcionados por las señales fisiológicas adquiridas y definidas dentro de modelos de Inteligencia Artificial, tal y como recalcamos al principio, con todos los aprendizajes y metas que el desarrollo de este motivo conlleva, tanto a nivel de aprendizaje como a nivel técnico.

1.3- ESTADO DEL ARTE

Tras numerosas búsquedas en el mercado, hemos identificado una serie de productos e ideas que comparten similitudes con el sistema que pretendemos diseñar en este proyecto. A continuación, se presenta una descripción de estos, así como las ventajas y desventajas que presentan en comparación con el cumplimiento de algunos de los requisitos que componen nuestro sistema.

1.3.1- PMD_200

En primer lugar, aparece el único mecanismo automatizado que permite monitorizar el estado de dolor de un paciente disponible en el mercado. Estamos hablando del dispositivo “PMD-200” [3]. Este sistema es un monitor de nocicepción no invasivo que permite representar la respuesta fisiológica de los pacientes en función de los estímulos dolorosos a través de un monitor. Este aparato ha sido diseñado por la empresa MedaSense y presenta la única solución inteligente para la evaluación del dolor en el mercado hoy en día.

El objetivo principal de este instrumento consiste en la evaluación del nivel de dolor que padecen los pacientes que están sometidos a la anestesia general, en una intervención quirúrgica o en una unidad de cuidados intensivos, y no pueden transmitirlo a los profesionales sanitarios. Esto permite que la monitorización del paciente anestesiado sea individual y eficaz, consiguiendo una evaluación nociceptiva importante que produce la disminución tanto de opioides (fentanilo, remifentanilo, morfina, propofol) hasta un 30% [4], como el riesgo de sufrir complicaciones durante la intervención para que los pacientes se puedan despertar de la anestesia con el nivel de dolor postoperatorio, relacionado con un exceso o una insuficiencia de administración de opioides (2), más bajo posible. Además, consigue reducir considerablemente los eventos adversos durante el tratamiento, tanto quirúrgico como farmacéutico, permitiendo una reducción considerable de la estancia hospitalaria del paciente y un ahorro hospitalario de, aproximadamente \$325 980 (304243,65 EUR) [5], económicamente hablando.

Antiguamente, se llevaba a cabo la monitorización nociceptiva del paciente a través de la frecuencia cardíaca y la presión arterial, entre otros. Sin embargo, algunos estudios han manifestado que estas constantes vitales no son suficientes para llevar un control del nivel del dolor que sufre un paciente anestesiado. Por ello, el sistema está diseñado en base al índice de nivel de nocicepción (NOL) que permite clasificar el dolor del

paciente en un umbral variable de 0 a 100 mediante una serie de algoritmos fundamentados en Inteligencia Artificial. Este umbral se calcula a través de una correlación entre los datos obtenidos a través de unos sensores que actúan como biopotenciales y la puntuación clínica basada en la estimación de la concentración de opioides y la capacidad del estímulo nociceptivo [6].

Para obtener los datos de los sensores, el sistema se compone de un aparato con forma de pinza, similar al pulsioxímetro, formado por cuatro sensores que permiten obtener variables fisiológicas de manera continua y no invasiva: temperatura, fotopletismografía, respuesta galvánica de la piel y acelerómetro. Estos datos serán los que principalmente permitan cuantificar dichas señales dentro del umbral del NOL. Esta escala aparece, de forma gráfica y numérica, en un monitor que permite al profesional sanitario obtener información sobre el estado de nocicepción del paciente de la siguiente manera:

Índice NOL = 0, ausencia de nocicepción (el paciente no padece dolor).

Índice NOL = 28, nivel de nocicepción calibrado al individuo.

Índice NOL = 100, respuesta nociceptiva extrema (el paciente padece dolor extremo).

En un pequeño resumen, podemos afirmar que los estudios realizados para la realización de este proyecto recalcan que, además de la frecuencia cardíaca y la presión arterial, es importante obtener más información sobre distintas señales fisiológicas para poder clasificar el nivel de dolor de forma más precisa. Por ello, podemos aprovechar este aporte de conocimiento para poder seleccionar una mayor cantidad de sensores que permitan obtener los datos que hagan que la herramienta sea lo más completa y eficiente posible, teniendo en cuenta lo económico, sin dudar.



Figura 1: Dispositivo PMD_200

A nivel funcional, aparece la capacidad de monitorizar el dolor a través de aplicaciones disponibles en dispositivos portátiles (smartphones y tablets) en cualquier lugar en el que se encuentre el paciente, siempre y cuando esté conectado a la red wifi. Lo más normal es que sea el paciente el que administre datos a estos programas que irán realizando un estudio en función de los parámetros que se hayan definido en su diseño en lugar de que la aplicación sea la que capte dicha información.

Según un estudio, hay más de doscientas ochenta aplicaciones disponibles en las tiendas oficiales de los principales sistemas operativos (Android/iPhone) que permiten monitorizar y evaluar el estado nociceptivo del paciente [7]. Sin embargo, son muy escasos los programas que están desarrollados con el contraste y el análisis científico y mucho menos los que hayan sido sometidos a ciclos de pruebas que cumplimentan las exigencias planteadas en los certificados de calidad que rigen este tipo de software. Es interesante conocer que dos de las aplicaciones cuyos desarrollos están contrastados empíricamente están programadas en España. A continuación, se muestran, de manera resumida, las características de estos programas que permiten llevar un manejo de la enfermedad:

1.3.2- PAINOMETER V2

Como parte de la competencia dentro del ámbito software aparece una aplicación desarrollada por el grupo de investigación del dolor de la Universidad Rovira y Virgili (ALGOS) cuyas características y diseño se asemejan mucho a la comentada en el apartado anterior. La principal diferencia es que este programa permite que los pacientes registren la información de manera interactiva, mediante una serie de escalas que permiten referenciar el nivel de dolor que estos padecen:

- La escala de caras en las que aparecen seis rostros faciales con signos de dolor incrementados de izquierda (ausencia de dolor) a derecha (dolor extremo).
- La escala numérica, similar a la anterior, en la que aparece un rango cerrado de números [0 -10] que representa el nivel de dolor que el usuario está padeciendo en ese instante.
- La escala visual analógica presenta una línea horizontal y otra vertical que el usuario desplazará, de izquierda (ausencia de dolor) a derecha (dolor extremo) hasta ubicarla en la zona que más represente la nocicepción.

- La escala analógica visual es muy parecida a la anterior con la diferencia de que, en esta evaluación, el paciente debe mover el marcador a través de un triángulo creciente de izquierda (ausencia de dolor) a derecha (dolor extremo).

Una vez que se han registrado los datos en las escalas mencionadas anteriormente, la aplicación los procesa en un fichero que se envía al profesional médico que ha pautado el tratamiento para que pueda realizar un estudio preciso del nivel de dolor que padece el paciente a partir de esta información, incluso de forma telemática.



Figura 2: Aplicación Painometer V2.

También es interesante comentar que, el programa genera automáticamente un gráfico que representa la cuantificación de los datos obtenidos en el estudio. Sin embargo, la clasificación de estos datos es muy poco fiable pues el software del programa no ha sido sometido a ningún tipo de entrenamiento de aprendizaje ni validación, sino que la representación se realiza directamente desde los datos recopilados.

En oposición con la mayoría del resto de aplicaciones disponibles en el mercado, cuyo software no ha sido desarrollado cumpliendo los requisitos impuestos por sellos y certificados de calidad como los que propone la Agencia de Calidad de la Junta de Andalucía y supongan “*un problema que plantea un riesgo potencial para la salud física y mental de los usuarios*” [8], este programa permite que los estudios que se realicen a partir de la información registrada estén contrastados empíricamente.

1.3.3- ALGÓMETRO ANALÓGICO FPK 60

Actualmente, a nivel comercial hay una gran cantidad de sistemas que cumplen la función de cuantificar el nivel nociceptivo que padece un paciente. Estamos hablando de los algómetros, o también comúnmente conocidos como dolorímetros.

Estos dispositivos son estructuras hardware compuestas por algún tipo de sensor que recoge bioseñales, las procesa y les da un determinado valor en función del dolor. Sin embargo, el principal problema es que su arquitectura informática es inmodificable (cerrada), es decir, es solamente accesible por los desarrolladores, y, además, la gran mayoría están basados en células de carga, lo que produce que su valor económico sea muy alto.

En resumen, podemos decir que son sistemas que permiten localizar la ubicación del dolor y, de alguna manera, darle un valor, pero no analizan estos datos para su posterior estudio o clasificación.

En este caso, se presenta un mecanismo de medición que permite calcular la dolencia en función de la fuerza de presión que se aplique a la superficie en la que se encuentre permitiendo dar un valor al dolor dentro del umbral de presión llamado algómetro analógico FPK 60. En otras palabras, se coloca el gancho de presión que compone el instrumento en el lugar donde se encuentra el dolor y se presiona, y, de forma gradual, se irán incrementando la presión aplicada sobre el tejido correspondiente, de forma rectilínea y uniforme, hasta que dicha fuerza supere el umbral de presión, o, mejor dicho, que “la presión aplicada se transforme de indolora a dolorosa” [9]. El resultado será un valor que cuantifique el nivel nociceptivo que ha sentido el paciente durante el tratamiento.



Figura 3: Algómetro analógico FPK60

El objetivo de este producto es que se pueda realizar estudios que permitan conocer el test de umbral y tolerancia al dolor, entre otros. Sin embargo, el principal inconveniente es que es necesario someter al paciente a la dolencia para poder cuantificar su valor.

A nivel técnico, el sistema está compuesto por una serie de materiales que permiten que sea ergonómico, debido a sus 500 gramos de peso, y un marcador circular que representa el valor cuantificado con un error relativo de +-2 grados. Estas características permiten que este sea uno de los productos líderes en su sector, de hecho, la empresa desarrolladora tiene cerca de 18000 valoraciones verificadas [10], dentro del marco europeo, cuya nota media es de 4,8/5 (fuente: Google), pero su elevado precio, 644,95 EUR, hacen que no sea una herramienta muy accesible al alcance de todos los profesionales sanitarios que lo necesiten.

Como hemos visto, hay una gran variedad de dispositivos que cumplen la función de monitorizar el nivel de dolor que sufre un paciente de diferentes maneras, ya sea de forma analógica, ya sea de forma digital o bien, a través de una aplicación. Aun así, se siguen realizando estudios que permitan mejorar y complementar los instrumentos disponibles a día de hoy.

Por ello, es importante señalar algunos estudios relacionados con el diseño de dispositivos similares a los descritos anteriormente, realizando una pequeña investigación sobre algunas patentes.

1.3.4- PATENTE: US5533514A

La herramienta desarrollada en esta invención permite medir el umbral de dolor aplicando un sensor de presión sobre un punto de la piel de forma incremental mientras un contador va registrando el valor de la fuerza aplicada, en gramos, de manera directa, a lo largo de cada segundo y que será representada en una pantalla contenida dentro del mismo instrumento utilizando un desarroll software que realice mediciones del nivel del dolor que padece el paciente para poder determinar una secuencia en cada punto[11]. Esta información será almacenada en un archivo en el que se establece, automáticamente, un umbral de dolor específico y concreto para cada punto que será evaluado con los datos que se vayan adquiriendo en posteriores usos para que así, los profesionales sanitarios, puedan observar cómo varía el tratamiento en función de la secuencia inicial y dicho umbral. Por ejemplo, supongamos que el profesional aplica la solución al paciente y este le indica que le molesta, en este caso, suponemos que la fuerza aplicada era de 150 gramos cuando sintió la dolencia, por lo tanto, el umbral de dolor estará entre [0g-150g]. Si los valores medidos en posteriores tomas de datos, el dolor habrá disminuido, incluso desaparecido en caso de obtener 0 gramos. Sin embargo, si el valor supera lo establecido en el umbral, se supone que el paciente se encuentra en un estado extremo dolorosamente hablando.

A parte, el paciente debe señalar la sensación y la incomodidad que ha sentido cuando se le estaba realizando la medición en la zona afectada, dando lugar a que esta herramienta sea una mezcla entre dispositivo digital y aplicación.

Esto proporcionará una gráfica visual siguiendo lo establecido en la escala analógica visual que será mostrada en una pantalla propia del sistema y permitirá al profesional sanitario realizar un estudio sobre el estado del paciente.

A continuación, presentamos una breve tabla comparativa de los instrumentos comentados anteriormente para poder realizar un mejor análisis de los instrumentos disponibles en el mercado.

Dispositivo	Precio	Ventajas
PMD-200	N/S	Obtención, monitorización y clasificación de datos efectivas debido a la tecnología NOL.
Painometer v2	Gratis	4 escalas de medida del nivel de dolor.
Algómetro analógico FPK 60	644,95	Cálculo de la fuerza aplicada en la zona nociceptiva de forma fiable y analógica.
Patente US5533514A	N/S	Cálculo de secuencias en función de la presión aplicada en un punto para obtener el umbral de dolor.

Cuadro 1: Comparación de los productos incluidos en el estado del arte

Tras este análisis mercantil, podemos concluir que actualmente hay varias herramientas que permiten cubrir algunos de los objetivos principales de este proyecto. Sin embargo, solamente hay un dispositivo capaz de cumplir con las expectativas que se proponen, ese dispositivo es el pmd-200 pues es el único instrumento que permite monitorizar el nivel nociceptivo y clasificarlo en un parámetro concreto en función de los datos obtenidos, en este caso utilizando la tecnología NOL.

Esto no quiere decir que el resto de los dispositivos sean de menor importancia pues, cada uno de ellos cumple con una característica individual concreta.

Por ejemplo, el cálculo de secuencias y el nivel del dolor en función aplicada, utilizando los dispositivos definidos en las patentes, permiten la localización de la dolencia, además, la precisión de los algómetros, permite que la obtención de bioparámetros sea lo más fiable y precisa posible, y, la evaluación del nivel del dolor de un paciente a través del feedback que este proporciona, a través de cuestionarios o escalas de medición del nivel nociceptivo, hacen que estos sistemas puedan formar parte de la construcción del tratamiento más correcto posible para un paciente que padezca esta enfermedad.

Con todo ello, podemos afirmar que una combinación de estas funcionalidades sumando un coste que permita que la herramienta sea accesible económicamente a todas las ramas sanitarias conocidas, pueden dar lugar a un producto completo con una alta capacidad de proyección en el mercado. Este análisis nos puede servir para obtener una buena redacción a la hora de llevar a cabo la definición de requisitos necesarios para nuestra herramienta.

1.4. ELICITACIÓN DE REQUISITOS

Siguiendo lo comentado en el apartado anterior, además de entrevista con el tutor del proyecto en simulación de un cliente real, sumado a los objetivos definidos al inicio del proyecto, podemos dar lugar a una definición de requisitos que se adecuen lo máximo posible a las necesidades de nuestro sistema. Para ello, se ha elaborado una lista por cada uno de los tipos de requisitos fundamentales que componen el sistema: funcionales, no funcionales y de información.

1.4.1- REQUISITOS FUNCIONALES

Estos requisitos funcionales describen las acciones y capacidades necesarias para alcanzar los objetivos docentes y técnicos establecidos en el proyecto. En otras palabras, este tipo de requisitos describen cómo se comporta la solución a nivel técnico. Los requisitos funcionales del sistema son los siguientes:

Requisito	Descripción
RF_01	El conjunto de entrenamiento válido debe utilizar el formato .csv.
RF_02	Se podrá realizar un estudio sobre varios modelos de Inteligencia Artificial. Opcionalmente, se puede considerar representar estos resultados a través de datos estadísticos extraídos de tablas, matrices, gráficos, etc.
RF_03	El modelo seleccionado debe ser capaz de decidir qué valoración de nivel de dolor está padeciendo el paciente en ese instante, en función de las variables fisiológicas (frecuencia cardíaca, frecuencia respiratoria, temperatura corporal, respuesta galvánica).
RF_04	El sistema debe ser capaz de mostrar los resultados de la clasificación del nivel de dolor, así como las variables que están indicando el porqué de esa clasificación a partir del modelo seleccionado.

Cuadro 2: Requisitos funcionales.

1.4.2- REQUISITOS NO FUNCIONALES

Este tipo de requisitos describen características fundamentales para su correcto funcionamiento, así como el establecimiento de la calidad del sistema. Los requisitos no funcionales del sistema son los siguientes.

Requisitos no funcionales	Descripción
RNF_01	Eficiencia: El sistema debe ser capaz de adquirir y procesar las señales biomédicas definidas en el conjunto de entrenamiento de manera eficiente.
RNF_02	Rendimiento: El modelo debe ser capaz de procesar y analizar los datos biomédicos proporcionados.
RNF_03	Fiabilidad: El sistema debe implementar el modelo de Inteligencia Artificial que contenga la máxima confianza y precisión, para ello, se debe realizar un análisis y preprocesamiento de los datos.
RNF_04	Usabilidad: El sistema debe ser fácil de utilizar tanto a la hora de la recogida de datos, como la representación de la clasificación de estos.
RNF_05	Sencillez: El modelo implementado en el sistema debe ser entendible por el mismo. Para ello, la implementación debe realizarse con estructuras condicionales.
RNF_06	Mantenimiento: El sistema debe presentar un desarrollo que permita un fácil mantenimiento y actualización, de cara a mejoras futuras.
RNF_07	Escalabilidad: El modelo debe ser capaz de procesar volúmenes de datos de tamaño medio (aproximadamente 1000 muestras).
RNF_08	Portabilidad: El estudio de los modelos debe ser realizado en diferentes entornos de programación (Google Colab y Weka), permitiendo así, obtener resultados en contextos diferentes.
RNF_09	Lógica de datos: El sistema debe definir una lógica de datos similar al algoritmo escogido en el estudio.
RNF_10	Seguridad: El sistema debe garantizar la LPGD de cada persona que haga uso de este.

Cuadro 3: Requisitos no funcionales.

1.4.3- REQUISITOS DE INFORMACIÓN

Este tipo de requisitos representan qué datos necesita el sistema y qué datos va a generar. Permiten obtener conclusiones válidas y significativas a partir de ellos.

Requisitos de información	Descripción

RI_01	Almacenamiento de datos: El sistema debe ser capaz almacenar los datos biomédicos recopilados y transformados en un formato compatible con la definición de un conjunto de entrenamiento para IA. Dicho formato corresponde con el Comma-Separated-Values (csv).
RI_02	Preprocesamiento de datos: Los datos almacenados serán preprocesados necesariamente para aumentar la precisión del estudio de los algoritmos de IA. Este preprocesamiento puede incluir la limpieza de datos faltantes, la codificación de variables categóricas, entre otras técnicas.
RI_03	Análisis de modelos de IA: Se debe realizar un estudio de diferentes algoritmos que permitan clasificar el nivel de dolor en función de las variables biomédicas que se obtendrán. Esto puede incluir el cálculo de estadísticas descriptivas, la visualización de gráficos, tablas y matrices, y la identificación de patrones relevantes para decidir el algoritmo que será implementado en la lógica del sistema.
RI_04	Normativas y estándares: El sistema debe cumplir con las normativas vigentes y los estándares establecidos para la investigación científica y el manejo de datos biomédicos. Esto incluye aspectos éticos, de privacidad y de seguridad de la información.

Cuadro 4: Requisitos de información.

BLOQUE 2: EJECUCIÓN DEL PROYECTO

2.1. DISEÑO DEL SISTEMA

En general, el sistema está compuesto por dos grandes bloques, que a su vez componen varios subbloques, dando lugar a la representación de todas las estaciones de trabajo que pueden ser llevadas a cabo en él y, además, cumpliendo los requisitos que hemos definido anteriormente.

En este apartado, se irá comentando cada bloque que compone el sistema en función de la información que se puede visualizar en el diagrama de bloques.

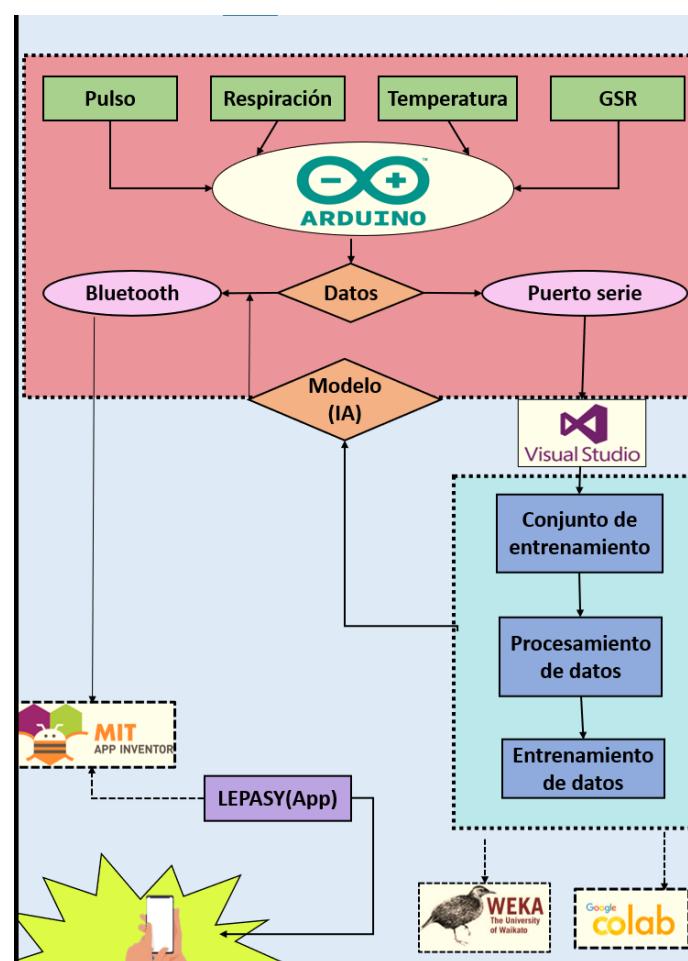


Figura 4: Diagrama de bloques del sistema

Nota. Diagrama realizado por elaboración propia.



Figura 5: Leyenda del diagrama de bloques del sistema.

Tal y como se observa en la figura anterior, los dos bloques principales que componen el sistema son los definidos como bloque instrumental y bloque inteligente.

El primero de ellos es el bloque instrumental. Este conjunto de elementos está compuesto por la suma de todos los componentes y funciones necesarias que permitan obtener las variables fisiológicas que definan el conjunto de entrenamiento del sistema. A continuación, iremos comentando cada uno de los componentes y funciones que han sido necesarios para conseguir el objetivo de este bloque.

El elemento principal del sistema es un conjunto de sensores que captan bioseñales fisiológicas que posteriormente serán enviadas en tiempo real a una placa de Arduino UNO para ser leídas, procesadas y transformadas, en el entorno de programación C++, pasando de magnitudes fisicoquímicas a magnitudes eléctricas, y, todo ello en función del fin en concreto de cada componente.

Para ello, hemos hecho empleo de un total de cuatro sensores programados para captar información del paciente en tiempo real. Este conjunto de componentes está formado por un sensor de pulso, un micrófono, un sensor de respuesta galvánica y otro de temperatura que permiten obtener la frecuencia cardíaca (BPM), frecuencia respiratoria (RPM), respuesta galvánica de la piel (GSR) y temperatura corporal ($^{\circ}\text{C}$), respectivamente, en sus unidades concretas. Cada uno de estos sensores están directamente conectados en su lugar correspondiente de la placa de Arduino UNO, permitiendo así captar las señales y poder procesarlas en función de unas librerías y funciones concretas implementadas en el entorno de desarrollo de Arduino, cuyo lenguaje de programación es en C++, obteniendo como resultado las variables fisiológicas que necesitamos en tiempo real.

Esta información será enviada por el puerto serie en un formato concreto, que será reconocida por una aplicación de interfaz sencilla diseñada a partir del lenguaje de programación C# cuyo entorno de programación permite establecer una comunicación bidireccional entre Arduino y Visual Studio necesaria para la definición de un conjunto de entrenamiento en un formato estándar (.csv). Este conjunto de entrenamiento nos permitirá recoger información de los parámetros biomédicos de serie de usuarios que han formado parte de un ensayo clínico para poder realizar el estudio.

El objetivo de la obtención de este fichero de datos separado por comas es definirlo como conjunto de entrenamiento para combinar algoritmos de inteligencia artificial que nos permitan obtener parámetros e información para realizar un pequeño estudio sobre cómo clasificar la frecuencia cardíaca, respiratoria, respuesta galvánica y temperatura corporal en nivel de dolor alto, medio o bajo. Por ello, se enviará el conjunto a dos entornos de programación que permiten realizar este estudio, Google Colab y Weka.

2.2. IMPLEMENTACIÓN

En este apartado mostramos los procedimientos que han ido dando forma a la solución propuesta en este proyecto en función de las tecnologías y la metodología utilizada y que veremos a continuación.

2.2.1- METODOLOGÍA

Cuando se está desarrollando un proyecto software de alto nivel, es muy importante seleccionar la metodología que se va a seguir a lo largo del proyecto. Nosotros hemos decidido utilizar Scrum, ya que es muy común en este tipo de proyectos.

Scrum es un conjunto de buenas prácticas para trabajar en un equipo multidisciplinar[12]. En nuestro proyecto, esta metodología está implementada en la parte de desarrollo, haciéndolo valioso desde sus primeras versiones.

Los equipos trabajan con ciclos de ejecución llamados “sprints” que permiten las versiones por las que se construye el proyecto con el fin de lograr un sistema completamente desarrollado de manera tan rápida y continua.

La metodología Scrum define diferentes roles asociados a cada uno de los componentes del equipo multidisciplinar:

- **Scrum Master:** es el equivalente a lo que comúnmente se conoce como jefe de proyecto. Su labor es definir y analizar cómo va a trabajar el equipo multifuncional, así como de ser el responsable de conseguir que se sigan los valores y las prácticas de ‘scrum’. Ayuda a los miembros del equipo para que trabajen de forma autónoma y autoorganizada. Se ocupa también de eliminar problemas y obstáculos que puedan poner en riesgo el objetivo del ‘sprint’.
- **Product owner:** Su mirada está siempre puesta en el cliente, y en lo que el equipo va a desarrollar. Es responsable de que el producto vaya incrementando su valor con cada ‘sprint’. Además, es la persona encargada de marcar el objetivo de manera clara y acordada con el resto del equipo.
- **Equipo de desarrollo:** Es el grupo de profesionales que hace el trabajo necesario para poder entregar el incremento de valor en el producto. Se autoorganizan para realizar el trabajo y han de estar disponibles a tiempo completo en el proyecto.

El personal que compone este proyecto está formado por un scrum master, asignado al tutor del trabajo, y los alumnos que lo realizan, asignados como dos integrantes pertenecientes al equipo de desarrollo.

Cada uno de los miembros del equipo deben cumplir una serie de condiciones cuando estén ejerciendo su rol correspondiente:

1. No se deben realizar cambios que pongan en peligro el objetivo.
2. El ‘Product owner’ y el equipo de desarrollo trabajan conjuntamente ajustando el detalle de las funcionalidades planificadas para el ‘sprint’.
3. Además de construir el producto, todo equipo trabaja conjuntamente como ‘Product owner’ y los miembros del equipo aclaran y negocian entre ellos a medida que se va elaborando el proyecto.

La duración máxima recomendada para un sprint será de un mes, porque si es más largo corremos el riesgo de aumentar la complejidad del producto, aumentando así el riesgo de no entregar lo que el cliente espera.

Una vez acabado el sprint será necesario llevar a cabo una labor de inspección y revisión del trabajo realizado, en la que haya un ‘feedback’ por parte del “Product owner” o incluso el propio cliente. La reunión debe cumplir una serie de condiciones:

1. Los asistentes son el equipo ‘scrum’ y otros interesados que puedan resultar clave.
2. Se identifica lo que se ha hecho y lo que no respecto a lo inicialmente acordado.
3. Se detectan los problemas y se analiza cómo se resolvieron.
4. Todo el equipo debe colaborar a la hora de decidir qué hacer a continuación.

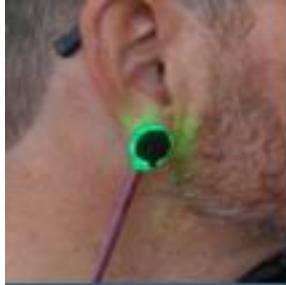
Todas las metodologías ágiles buscan mejorar de manera continua la forma en la que el equipo se relaciona durante el proceso de desarrollo. En scrum existe otra sesión específicamente definida para lograrlo: la retrospectiva. Se trata de una oportunidad para que el equipo comparta sus impresiones y recomendaciones sobre la forma en la que han trabajado durante el ‘sprint’ que acaba de finalizar, con el objetivo de identificar las lecciones aprendidas.

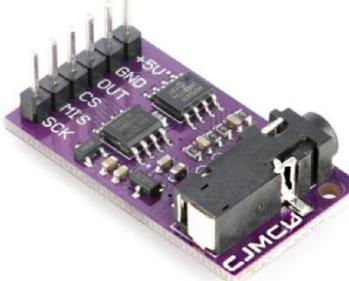
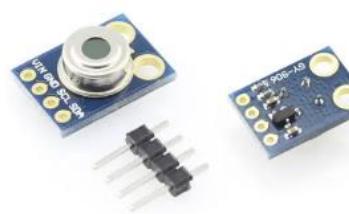
2.2.2- TECNOLOGÍAS EMPLEADAS

2.2.2.1- Bloque Instrumental

Para poder realizar el proyecto hemos necesitado algunas tecnologías que han sido esenciales para el desarrollo de este. El bloque instrumental, está recogido en la parte superior del diagrama de bloques que esta de color rojo.

2.2.2.1.1- Sensores

Frecuencia Cardíaca (PulseSensor)	 <p><i>Figura 6: Sensor de pulso.</i></p>	Mide pequeños cambios de volumen en el flujo de la sangre que atraviesa la capa epidérmica de la piel en forma de señal analógica de voltaje a través del sensor de ritmo cardíaco óptico.
Frecuencia respiratoria (KY-038)	 <p><i>Figura 7: Micrófono.</i></p>	Micrófono compuesto por un transductor que convierte las ondas sonoras en señales eléctricas
Respuesta galvánica (CJMCU-6701)		Mide la variación de la conductividad eléctrica de la piel provocada por el aumento de conductancia por la respuesta de las glándulas

	 <p><i>Figura 8: Sensor de respuesta galvánica.</i></p>	sudoríparas de la piel
Temperatura corporal (MLX90614)	 <p><i>Figura 9: Sensor de temperatura corporal.</i></p>	Mide la temperatura de un objeto a distancia a través de la emisión de radiación infrarroja captada por un chip de silicio con una fina membrana micromecanizada

Cuadro 5: Sensores del bloque instrumental.

2.2.2.1.2- Arduino

Es una plataforma de hardware y software libre que permite crear millones de proyectos.

En la actualidad hay múltiples placas que permiten realizar operaciones similares ya que

su estructura es muy parecida. En este proyecto enfocaremos el punto de atención a la placa de Arduino UNO cuya implementación será llevada a cabo dentro del sistema. Podríamos decir que es el componente más importante de todo el sistema pues la



Figura 10:Arduino.

lógica de captura y envío de datos viene definida en su entorno de programación. Dicho entorno está basado presenta una arquitectura basada en el lenguaje de programación C++ [13].

Por un lado, a nivel de hardware, es de vital importancia conocer que Arduino UNO está formado principalmente en torno al microcontrolador ATMega 328. Este componente computacional está formado, a su vez, por un Procesador AVR basado en arquitectura Harvard, con memorias y buses separados para instrucciones y datos.

Por otro lado, el aspecto software se refiere al programa que constituye el entorno de desarrollo de Arduino que permite programar la placa para darle todo tipo de utilidades a través del lenguaje de programación adaptado de C++.

El objetivo principal de esta herramienta en nuestro proyecto la lectura de las variables fisiológicas (pulso, frecuencia respiratoria, respuesta galvánica y temperatura corporal) sumado a la lógica que proporcione la clasificación del dolor en función de dichas variables y el envío por bluetooth a una aplicación móvil que permita la visualización de los resultados.

Para poder desarrollar el código del proyecto que cumpla con los objetivos y requisitos definidos inicialmente, se ha acudido a la utilización de librerías de procesamiento y transformación de señales biomédicas a sus correspondientes valores y unidades normales: pulseSensorPlayGround para el cálculo del pulso y la frecuencia cardíaca y Adafruit_MLX90614 para el procesamiento de la temperatura corporal. El resto de las variables acuden a funciones de lectura de datos analógicos simples para calcular sus resultados.

2.2.2.2- Bloque Inteligente

2.2.2.2.1- Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft. Proporciona un conjunto completo de herramientas y características para el desarrollo de software, incluyendo codificación, depuración, pruebas e implementación. Visual

Studio admite varios lenguajes de programación como C#, Visual Basic, C++, F#, JavaScript y Python, entre otros.

Este programa tiene un gran editor de código con una gran capacidad de autocompletar llamado IntelliSense que también permite la refactorización de código y navegación de código. Todas estas características lo hacen una de las herramientas más sencillas y versátiles a la hora de programar formando parte del proceso de aprendizaje informático junior.

A parte de esto, también cuenta un sistema de depuración y diagnóstico que nos ayuda a identificar y solucionar a los errores que surgen en el código con mayor facilidad utilizando técnicas de depuración, puntos de interrupción, ventanas de seguimiento e inspección de código en tiempo real.

Por otro lado, algo que también nos gusta de VisualStudio es que nos permite desarrollar para distintas plataformas móviles (iOS, Android y Windows) utilizando múltiples herramientas para elaborar este tipo de proyecto.

En nuestro caso, hemos utilizado la herramienta de creación de proyectos WindowsFormApp. Como indica su propio nombre, esta interfaz de programación gráfica permite desarrollar una aplicación utilizando la API de Windows conocida como .NET Framework. El objetivo de nuestra interfaz gráfica es recibir los datos biomédicos obtenidos por Arduino a través del puerto serie cada uno/dos segundos, y mostrarlos en una ventana que, a su vez, contiene una serie de opciones que permiten guardar esas variables en un fichero con formato .csv, indicando la clasificación del nivel de dolor que está sufriendo el usuario que está utilizando el sistema en ese instante.

2.2.2.2- Google Colab

Google Colab es un entorno gratuito desarrollado por Google que permite la apertura de un cuaderno de Jupyter Notebook y no requiere ningún tipo de instalación ya que se ejecuta en la nube con una memoria RAM de 12 GB y 124 GB de disco (para la versión gratuita).

La programación de código se lleva a cabo en cuadernos de Colab que disponen todas las características de Jupyter, incluido el lenguaje de programación: Python. Cada cuaderno está compuesto por un conjunto de celdas que pueden ser de tipo código o de tipo texto:

- Celda de tipo código: este bloque permite definir cualquier tipo de desarrollo de código en Python.

- Celda de tipo texto: este bloque permite realizar anotaciones y comentarios sobre el cuaderno. También puede servir como títulos para mantener un orden en el desarrollo.

La ventaja principal de este entorno es que es el más potente de todos los entornos que permiten desarrollar código en línea, y, por tanto, libera a nuestro equipo de tener que instalar entornos de programación que suelen ser muy pesados. En nuestro caso, preferimos utilizar este entorno que utilizar Anaconda pues nos ahorra tiempo y espacio y recursos en la memoria del equipo.

Este será el entorno en el que desarrollamos algunas de las tareas correspondientes a la programación de los algoritmos de Inteligencia Artificial que implementamos en este proyecto.

Como hemos comentado, el lenguaje de programación es Python. Python es un lenguaje que permite muchos tipos de programación como pueden ser:

- Programación orientada a objetos.
- Programación imperativa.
- Programación funcional.

Es el lenguaje más utilizado para la implementación de la Inteligencia Artificial ya que contiene catálogo gigante de librerías orientadas al procesamiento de datos, machine learning, deep learning, así como herramientas dedicadas al estudio estadístico y la representación de resultado, a través de gráficos, tablas, matrices. Por eso, también es uno de los lenguajes más sencillos a la hora del aprendizaje pues la gran mayoría de métodos están diseñados en alguna librería por lo que simplemente basta con importar dichos frameworks comenzar a programar.

A continuación, se procede a la descripción de las principales librerías que hemos utilizado en el desarrollo de este proyecto:

2.2.2.2.1- TensorFlow 2.0

TensorFlow es una plataforma de Python centrada en la implementación de modelos de aprendizaje automático de manera sencilla. Esta plataforma consiste en una librería de código abierto desarrollada por Google. TensorFlow tiene la capacidad de implementar cualquier tipo de modelo sin importar el entorno en el que se esté desarrollando (servidores, navegadores, dispositivos móviles, microcontroladores, CPU, etc.) [14].

Como ya sabemos, en nuestro caso, importaremos esta librería desde el entorno en línea de Google Colab.

Otra ventaja es la posibilidad de trabajar con grandes cantidades de datos debido la arquitectura en forma de arrays multidimensionales (tensores) provocando que el tamaño del problema pueda ser ampliarse de manera estratosférica.

2.2.2.2.2.2- Keras

Esta plataforma, al igual que TensorFlow, permite implementar modelos de aprendizaje de manera sencilla. De hecho, esta librería se suele combinar con TensorFlow permitiendo el desarrollo de modelos de la manera más completa posible.

La API de Keras hace que el código sea comprensible y fácilmente personalizable, en función de los niveles de conocimiento del usuario que lo esté utilizando. Básicamente, podemos decir que Keras es una herramienta que permite implementar modelos programados a alto nivel, desde una interfaz sencilla a bajo nivel [15]. A parte de esto, también permite la implementación de los modelos en gran variedad de productos, como dispositivos móviles o wearables, a través de interfaces que permiten definir los procesos automáticos en una infinidad de dispositivos.

2.2.2.2.2.3- Scikit-Learn

Es una biblioteca de Python que permite realizar funciones de preprocesamiento y modelado de algoritmos de Inteligencia Artificial a través de una API propia y estandarizada[16]. Hoy en día, es una de las librerías más su flexibilidad y facilidad permite que el número de problemas en los que se puede utilizar sea prácticamente innumerables. Además, es de código abierto, por lo que tanto un experto como un novato pueden acceder a su código, así como a la documentación disponible en su web. En nuestro caso, se ha utilizado esta librería para:

- Realizar el preprocesamiento de todos los datos del conjunto importando las funciones OrdinalEncoder y to_categorical.
- Establecer los parámetros de que definen los modelos de machine learning.
- Entrenar el modelo utilizando conjunto de aprendizaje proporcionando métricas como resultado que aporten información sobre el modelo.

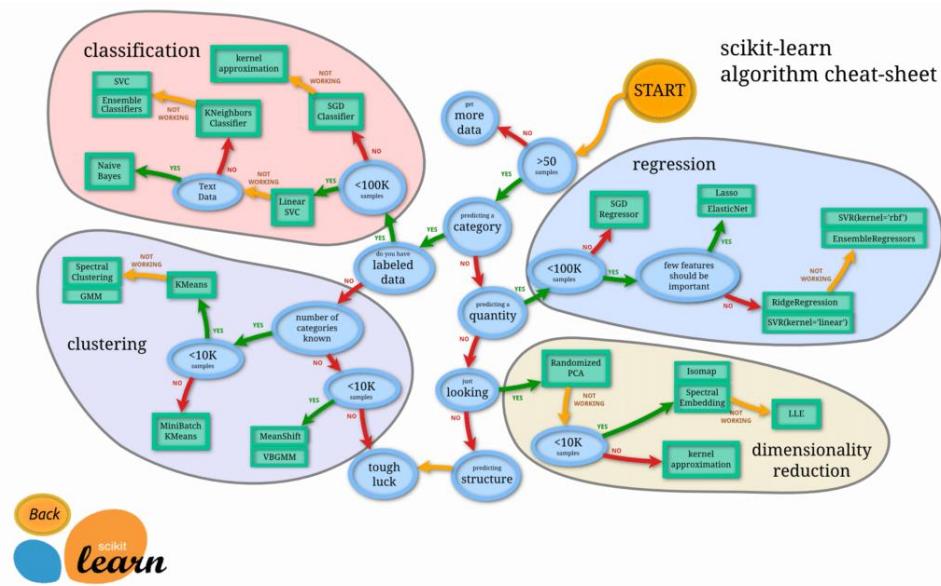


Figura 11: Cheat-sheet de scikit-learn.

Nota. Figura obtenida de la librería scikit-learn.

Las funciones que se han importado de esta librería son las siguientes:

Modelo	Función
Preprocesamiento	
Codificación ordinal	OrdinalEncoder
Escala mínimo-máximo	MinMaxScaler
Escala normalizada	Normalizer
Escala estándar	StandardScaler
Selección de características	
División del conjunto de datos	train_test_split
Métricas	
Error medio	mean_squared_error
Accuracy	accuracy_score
Matriz de confusión	confusion_matrix
Modelos	
Regresión linear	LinearRegression
Árbol de decisión	DecisionTreeClassifier
Bosque aleatorio	RandomForestClassifier
K vecinos más cercanos	KNeighborsClassifier
Perceptrón multicapa	MLPClassifier

Cuadro 6: Funciones de scikit-learn.

2.2.2.2.2.4- Pandas

Esta librería es una herramienta potente en el ámbito de análisis de datos. Permite manipular datos y operaciones a gran escala a través de una multitud de funciones que puedan representar ese manejo de los datos, en cada instante, utilizando tablas numéricas y series temporales[17].

Las características principales son:

- Manejo de valores faltantes de manera sencilla.
- Creación de objetos permitiendo la conversión de datos desiguales e indexados en estructuras más ordenadas como los DataFrame.
- Los datos se pueden alinear con un conjunto de etiquetas de manera manual o automática.
- Funciones que permiten cargar y guardar datos desde una multitud de archivos, como es el caso el caso de los archivos planos (CSV), archivos de Excel (xls) o base de datos, de manera rápida y eficiente.

2.2.2.2.5- Numpy

NumPy (Numerical Python) es una biblioteca de funciones de Python de código abierto define un estándar universal para trabajar con datos numéricos en Python. La API de esta herramienta se usa de la mano de las librerías comentadas anteriormente, como en Pandas, SciPy, Matplotlib, scikit-learn, y la mayoría de las librerías dedicadas al análisis y la ciencia de datos.

La biblioteca NumPy [18] trabaja fundamentalmente con matrices multidimensionales y estructuras de datos matriciales a través de una serie de métodos que operan con objetos de matriz n-dimensional homogéneo proporcionando así la realización de una gran cantidad de operaciones matemáticas de alto nivel sobre estos arreglos y matrices.

2.2.2.2.6- Matplotlib

Matplotlib es la librería más popular y completa para crear diagramas, matrices y otro tipo de herramientas gráficas de visualización estáticas, animadas e interactivas en Python. Las características principales de esta biblioteca son las siguientes:

- Crear gráficas de todo tipo (lineares, de puntos, de barras, etc.) tanto estáticas como interactivas. Estas últimas permiten que se pueda hacer zoom, actualizar e incluso desplazar por ellas.
- Permite exportar los resultados a muchos archivos con distinto formato (csv, xlsx, etc.).

- Se puede considerar como la rama principal de la cual se basan muchas otras librerías para implementar su funcionalidad, como es el caso de seaborn o ggplot, entre otras.

Los datos recibidos en los métodos que componen librería suelen ser mapeados y procesados para poder mostrar al usuario una interfaz gráfica con la mejor calidad posible [19].

2.2.2.2.7- App *inventor*

En este entorno de programación intuitivo y visual a través del lenguaje de programación en bloques se lleva a cabo el diseño de la interfaz del proyecto que permite mostrar la clasificación del nivel de dolor en función de las variables fisiológicas y el modelo de Inteligencia Artificial seleccionado.

2.2.2. DESARROLLO

Como hemos comentado anteriormente, el sistema completo se compone fundamentalmente de dos bloques en los cuales se aplican diversas metodologías y funcionalidades que permiten cumplir los requisitos definidos. Por ello, vamos a definir una serie de etapas que dividen los procesos que componen el sistema en base a la metodología Scrum. Estas etapas se conocen como sprints.

La decisión de definir el desarrollo del proyecto de esta manera consiste en que cada sprint está directamente relacionado con la cantidad de tecnologías y procedimientos que hay que implementar en el sistema. El objetivo de esta metodología es la construcción de versiones del sistema que vayan dando forma al proyecto hasta su consecución.

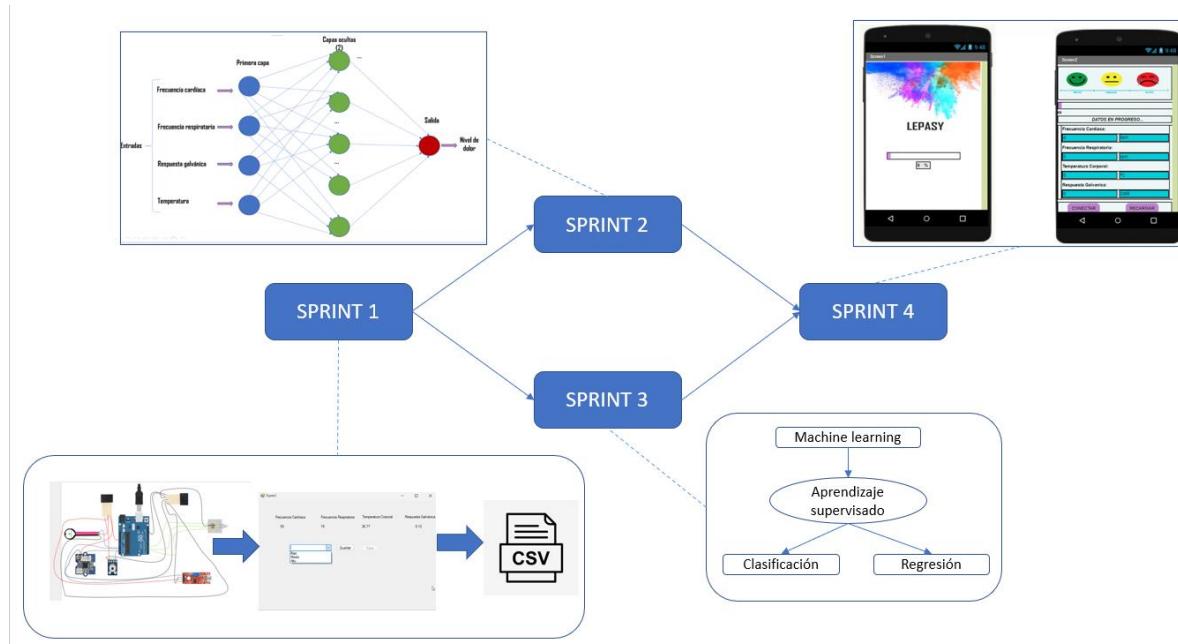


Figura 12: Esquema de los sprints.

Nota. Estudio realizado por elaboración propia.

El primer sprint consiste en un estudio de los datos biomédicos que queremos medir, para saber qué es lo que se desea obtener y cómo. Por ello, en este primer sprint se realizará la adquisición de los datos y su transformación al formato deseado, con el objetivo de poder obtener un conjunto de datos apropiados. Este primer sprint es de vital importancia para el estudio que se realice sobre los datos que se obtengan pues nos puede proporcionar una suma de mejores resultados aplicados a los procesos que se llevarán a cabo posteriormente.

En el segundo sprint se recibirán los resultados obtenidos en el primero con el objetivo de aplicar el conjunto de entrenamiento en modelos de Deep Learning para realizar un breve estudio sobre el comportamiento del conjunto de entrenamiento aplicado a esta tecnología.

El tercer sprint tendrá una estructura similar a la del sprint anterior, pero en este caso, se procesará el conjunto de entrenamiento obtenido en el primer sprint con el objetivo de aplicar los diferentes modelos de machine learning para obtener parámetros que nos permitan realizar un estudio, no sin antes pasar por varias técnicas de preprocesamiento que añadan valor añadido a los datos del conjunto. El resultado del estudio será la elección de un modelo en base a los parámetros obtenidos en la ejecución de los distintos algoritmos.

El cuarto y último sprint consistirá en la implementación del modelo obtenido como resultado del sprint anterior en una lógica que permita mostrar por pantalla la clasificación del nivel dolor de un paciente en función de los datos que proporciona el mismo en tiempo real.

El orden que se ha elegido está basado en las siguientes ideas:

- Alternar los bloques en los sprints de manera cruzada permiten que se pueda ir revisando y actualizando cada uno de los bloques en función de las necesidades que se vayan obteniendo.
- La decisión del método que implementará la lógica del sistema no será decidida hasta que se complete la ejecución de los algoritmos seleccionados para su estudio, permitiendo así obtener todos los resultados para que puedan ser analizados en forma de tablas, gráficos y matrices.
- La implementación del sistema estará completa cuando se realicen todos los sprints.

Nos centraremos en los sprints que hacen referencia al bloque inteligente, cuyo objetivo es el estudio de diferentes modelos de Inteligencia Artificial para obtener el más preciso y se adapte mejor a nuestro sistema de clasificación de niveles de dolor.

2.2.2.1-Sprint 2: Inteligencia Artificial I – Deep Learning

La Inteligencia Artificial (IA) surgió allá por los años 40-50, en una época repleta de hechos históricos como el descubrimiento de la estructura del ADN o la aparición de la televisión en color, entre otros. El primer algoritmo desarrollado en la historia de la IA se basó en un modelo matemático que utilizaba la red neuronal como lógica de cálculo en

1943. Sin embargo, no fue hasta la época del comienzo de los computadores digitales, allá por el año 1956, cuando el término Inteligencia Artificial comenzó a definirse.

En este contexto, el avance tecnológico toma gran iniciativa sobre la automatización de procesos que realizaban las personas de manera frecuente y repetida. En estos primeros años, esta tecnología comenzó a tomar forma de la mano de las reglas matemáticas y la lógica, para el diseño de algoritmos que proporcionaran un resultado concreto, sin embargo, la definición del conjunto de entrenamiento no era algo tan trivial pues se carecía de la tecnología que disponemos hoy en día. Por ejemplo, en ese entonces faltaba un cuarto de década para la invención de la primera cámara digital, por tanto, el desarrollo de un algoritmo que permitiera clasificar imágenes era imposible pues se carecían de las herramientas que proporcionaran información y variables que definieran el conjunto de entrenamiento que entrenara al modelo.

Hoy en día, la IA avanza a pasos de gigante pues cada vez se realizan más estudios sobre aspectos en los que puede entrar, como por ejemplo en el diagnóstico de enfermedades o clasificación de imágenes. Este interés e inversión sumado a las mejoras en las herramientas que componen las TIC, como las mejoras en la CPU de los computadores, establecen el deep learning con un potencial impresionante de cara a un futuro próximo.

Para entender mejor esta tecnología, es importante distinguir entre las dos ramas que la componen, el machine learning y el deep learning.

El machine learning está basado fundamentalmente en algoritmos matemáticos estadísticos que permiten tratar cierta cantidad de datos con el objetivo de dar respuesta a una serie de entradas que han sido enviadas a un modelo que aprende a reconocer patrones entre las variables.

La principal diferencia entre esta herramienta y el deep learning es que este último está basado en una representación de datos a través de capas que van aumentando su complejidad a medida que aumenta el número de capas consiguiendo una precisión inimaginable.

2.2.2.1.1- Análisis

Los algoritmos de deep learning están basados fundamentalmente en el sistema fisiológico neuronal que todas las personas contenemos en nuestro cerebro. Esta semejanza permite que el modelo vaya procesando y aprendiendo de los datos de entrada en cada capa sin importar la complejidad que presenten.

La base de este modelo consiste en que cada neurona recibe una serie de entradas y devuelve una salida a una de sus neuronas consecutivas que muestra el cálculo de una serie de elementos utilizando la ecuación:

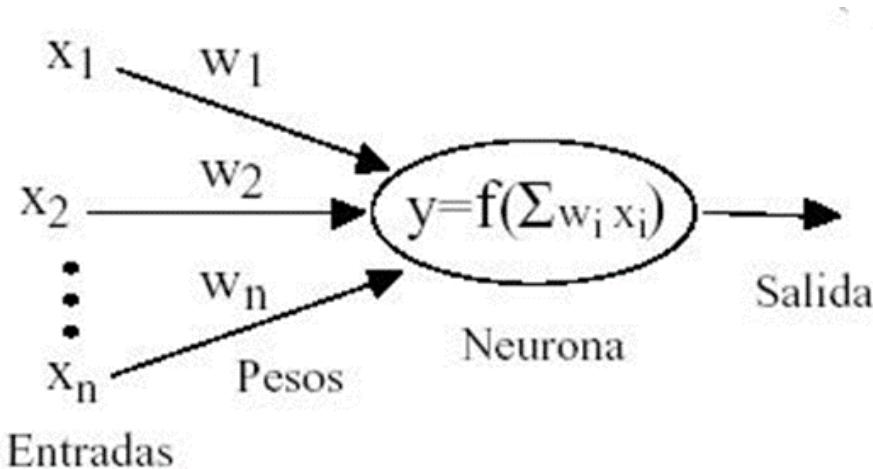


Figura 13: Estructura de red neuronal.

Nota. Figura obtenida de sitio web [I]

x_n es el valor de las entradas de la neurona.

x_1 es una entrada ficticia (inicial) cuyo valor es siempre -1.

w_n es el valor del peso numérico que determina la fuerza y el signo de las conexiones neuronales.

w_1 es un peso ficticio (inicial) que indica un umbral cuya suma de las señales de entrada debe superar para que se active.

y es una función de activación que normaliza la salida cuando el sumatorio de la multiplicación de las entradas por cada uno de sus pesos supera el valor umbral de entrada. Esta función proporciona a la neurona un comportamiento distinto al lineal. Hay una serie de funciones de activación cuyo objetivo es el mismo, pero la definición es diferente:

- **Función lineal (identity).** La salida de esta función es igual que la entrada proporcionando a la neurona un comportamiento igual al de una regresión lineal[20].
- **Función escalón (hard limiter):** función que devuelve un 0 a la salida cuando se cumple la entrada supera un umbral y 0 en otro caso. Esta herramienta es muy útil cuando la salida que se quiere clasificar es una variable categórica[21].

- **Función sigmoide (sigmoid):** esta función permite establecer una convergencia de los valores altos a 1 y los bajos a 0, permitiendo representar probabilidades de que ocurra la salida concreta. Sin embargo, presenta un inconveniente ya que su punto inicial no está centrado en el origen por lo que los resultados pueden presentar un poco de dificultad a la hora de su representación. La solución a este problema consistiría en utilizar una función similar, la tangente hiperbólica.
- **Función ReLU (ramping function):** esta función anula los valores negativos y modela los vos positivos de forma lineal. En otras palabras, la función se activa únicamente cuando los valores son positivos, proporcionando a la neurona un comportamiento lineal.

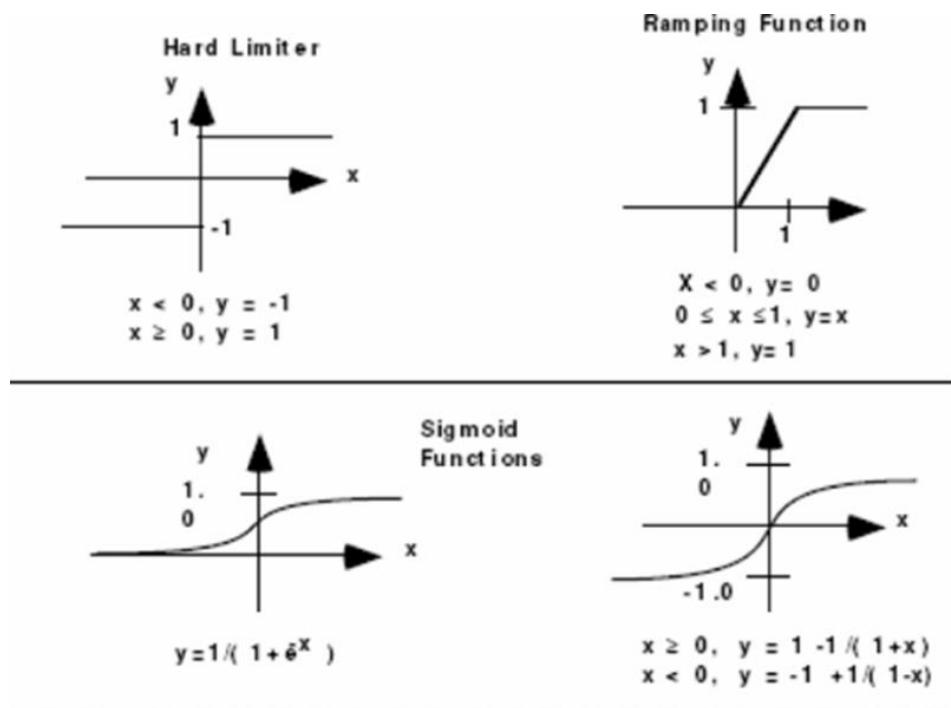


Figura 14: Funciones de activación de la neurona.

Nota. Figura obtenida de sitio web [II].

En general, cada nodo (neurona) se conecta a otros nodos a través de arcos dirigidos. Cada arco corresponde a la propagación de la salida de una neurona, dirigida a la entrada de otra y contiene un peso numérico que determina la fuerza de la información que calcula, formando una red que puede contener hasta varios centenares de capas conocida como red neuronal.

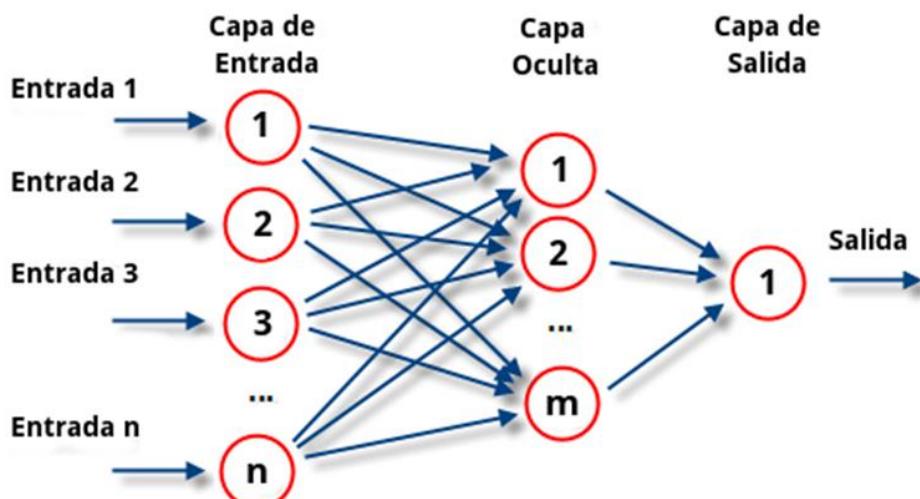


Figura 15: Ejemplo de red neuronal.

Nota. Figura obtenida de sitio web [III].

Este algoritmo está basado por un conjunto definido de neuronas agrupado por capas. En otras palabras, este modelo computacional está basado en una estructura de grafo dirigidos cuyas neuronas artificiales están fundamentadas en algoritmos de aprendizaje automático en potencia. El modelo de deep learning por excelencia consiste en las redes neuronales ya que permite “enseñar” al algoritmo a que realice una función concreta.

2.2.2.1.1.1- Captura de datos:

La captura de datos se representa la forma de cargar el conjunto de entrenamiento definido en el entorno de desarrollo que vayamos a utilizar.

El objetivo de este apartado es el almacenamiento del conjunto de entrenamiento en un DataFrame. Un DataFrame es una herramienta incluida en la librería “Pandas”. Consiste en un conjunto de datos con una estructura rectangular que almacenar y representar todo tipos de datos masivos. En otras palabras, la funcionalidad principal de esta herramienta es almacenar los datos en tablas que contengan funciones definidas que permitan ejercer acciones sobre ello.

2.2.2.1.1.2- Preprocesamiento:

2.2.2.1.1.2.1- Limpieza de valores faltantes

La limpieza de valores faltantes es una de las técnicas de procesamiento de datos más comunes. Hay varias causas que provocan que falten valores en el conjunto de entrenamiento, ya sea porque no existen o porque la recopilación de datos ha fallado en algún punto. Por ejemplo, en nuestro sistema, los sensores que se han implementado

son de bajo nivel por lo que algunas veces pueden no capturar datos por una serie de circunstancias como sobrecarga de datos o porque estén colocados de manera incorrecta. Los valores faltantes pueden aparecer representados de varias maneras: NaN, NA, null, etc.

Para responder a la cuestión sobre qué debemos hacer ahora con esos valores que no indican ningún tipo de información en el conjunto de entrenamiento, podemos referenciar varias técnicas de relleno:

- Eliminar las filas que presentan valores nulos directamente.
- Rellenar las variables nulas con el número 0.
- Rellenar las variables nulas con el número anterior o posterior de la columna adyacente.
- Rellenar las variables nulas con valores aleatorios.
- Rellenar las variables nulas con la media, la mediana o la moda de la columna en la que se encuentre el valor faltante.

La decisión de qué técnica utilizar depende del tipo de datos, la cantidad y el objetivo del conjunto de entrenamiento, de hecho, aunque se escoja una técnica u otra no indica necesariamente que sea la elección correcta, por ello, la responsabilidad de esta decisión depende del programador.

En nuestro caso, se decidió realizar un estudio sobre las variables estadísticas media, mediana y moda de cada columna que presenta valores faltantes (Frecuencia Cardíaca, Frecuencia Respiratoria, Respuesta Galvánica, en este caso) por los siguientes motivos:

- Eliminar las filas que presentan valores nulos de forma directa supondría la eliminación de cerca del 10% de los datos del conjunto de entrenamiento.
- Rellenar las variables nulas con el número 0 no aportaría ningún tipo de información al conjunto de entrenamiento, de hecho, empeoraría la correlación entre variables y el entrenamiento del modelo para la clasificación del nivel de dolor pues se supone 0 como un valor que se puede obtener y esto no es lógico pues no puede haber una frecuencia cardíaca de 0 pulsaciones por minuto, por ejemplo.
- Rellenar las variables nulas con el numero anterior o posterior de la columna o con valores aleatorios podría haber sido una buena técnica, pero la fiabilidad y la precisión de estos datos es baja. Por ejemplo, podemos encontrarnos con que la clasificación del nivel de dolor en la fila en la que se encuentra el valor nulo es

'Bajo' y la clasificación de este en la columna adyacente es 'Medio', en este caso, no tendría sentido asignar a la variable nula un valor correspondiente a un nivel de dolor 'Bajo' con lo que corresponde con un valor 'Medio'. Algo similar ocurre con los valores aleatorios ya que podemos encontrarnos que se le asigna un valor que no forma parte del rango normal de valores que le corresponde en función del nivel de dolor.

2.2.2.1.1.2.2- Codificación de las variables categóricas

Realizando un estudio del tipo de variables que tenemos en el conjunto, recogemos la siguiente información:

- Frecuencia Cardiaca: Numérica-Continua.
- Frecuencia Respiratoria: Numérica-Discreta.
- Respuesta Galvánica: Numérica-Continua.
- Temperatura Corporal: Numérica-Continua.
- Nivel de dolor: Categórica-Ordinaria.

Como podemos observar, la variable: "Nivel de dolor" es categórica-ordinaria, por lo que es necesario realizar la codificación para dicha variable. Para ello, deberíamos asignar un valor numérico a cada categoría tal que:

- "ALTO" = 3.
- "MEDIO" = 2.
- "BAJO" = 1.

Hay varios métodos de codificación de las variables categóricas, entre los que se encuentra la codificación ordinal y la codificación OneHot.

2.2.2.1.1.2.2.1- Codificación Ordinal

El objetivo principal de esta técnica consiste en sustituir el valor ordinal de la variable por un valor numérico que represente, de alguna manera, el orden, la jerarquía de cada categoría.

La ecuación matemática que lo define afirma que la k categorías de cada una de las variables X_j serán sustituidas por un solo valor numérico de tipo entero perteneciente al intervalo $[1, k]$ en función de un orden asignado.

$$\varphi_{OE}: X_j \rightarrow \mathbb{N}$$

$$x_i^j \rightarrow \varphi_{OE}(x_i^j) \in \{1, \dots, k\} \forall i = 1, \dots, n$$

En este caso, la variable categórica es el nivel de dolor cuyas categorías son: "BAJO", "MEDIO", "ALTO". Se podría asignar los valores numéricos 1, 2 y 3, respectivamente, para definir un orden creciente que reflejara la intensidad del nivel de dolor.

Sin Codificación	Codificación Ordinal
Nivel de dolor	Nivel de dolor
Bajo	0
Medio	1
Alto	2

Figura 16: Codificación ordinal del atributo nivel de dolor.

Nota. Figura obtenida por elaboración propia del cuadro utilizando PowerPoint.

La ventaja de utilizar la codificación ordinal en este tipo de variables es que se conserva la información de orden entre las categorías, lo que puede ser importante en algunos análisis estadísticos. Además, al utilizar valores numéricos en lugar de etiquetas de texto, se facilita el procesamiento y análisis de los datos. Sobre todo, es una buena opción si son equidistantes.

2.2.2.1.1.2.2.2- Codificación OneHot

Esta segunda técnica consiste en definir una variable binaria X_j por cada k-categoría de la variable que se necesita codificar para que estas variables se “activen” o “desactiven” en función de si están contenidas en la categoría, o no de la siguiente manera:

$$\varphi_{OH}: X_j \rightarrow \mathbb{Z}_2^k$$

$$x_i^j \rightarrow \varphi_{OH}(x_i^j) = (0,0,\dots,1,\dots,0) \forall i = 1,\dots,n$$

- La variable contendrá un 1 en aquellas filas de los datos que contengan la categoría.

- La variable contendrá un 0 en aquellas filas de los datos que no contengan la categoría.

Por ejemplo, en este caso, queremos que el algoritmo observe cuando se “active” la categoría “Alto”, “Medio”, “Bajo” en función del resto de variables. Por tanto, la codificación funcionaría de la siguiente manera:

Sin Codificación		Codificación OneHotEncoder		
Variables fisiológicas	Nivel de dolor	Nivel de dolor		
85	Bajo	Bajo	1	0
18		Medio	0	1
36.15		Alto	0	0
0.02				

Figura 17: Codificación OneHot del atributo nivel de dolor.

Nota. Figura obtenida por elaboración propia del cuadro utilizando PowerPoint.

La principal desventaja de esta técnica es el aumento de la dimensionalidad del conjunto de datos [22]. Es decir, las columnas del conjunto de datos aumentan en k-veces el número de k-categorías de cada variable categórica, lo que puede ser un problema en los casos en los que el tamaño del conjunto es muy pequeño, debido a un número de muestras relativamente escaso. En nuestro caso, el conjunto de datos aumenta en tres columnas, correspondiente a cada una de las categorías del nivel de dolor (alto, medio, bajo).

2.2.2.1.1.2.3- Escalado:

En la mayoría de los conjuntos de datos, las escalas de cada atributo son independientes las unas de las otras. Esto es algo que debemos evitar, ya que los algoritmos de aprendizaje no se modelan de manera correcta, aportando información un poco desconocida. Por ello, una de las principales técnicas de preprocesamiento de estos algoritmos es el escalado (comúnmente conocido como normalización o estandarización).

El escalado consiste en la transformación de la escala de los atributos dentro de un rango de valores concreto y sin perder la distribución ni la correlación entre los datos al realizar dicho proceso. Las herramientas que presenta la inteligencia artificial para realizar este objetivo son las siguientes:

- **Escala mínimo-máximo:** escala las características transformándolas a una escala entre [0-1].
- **Normalización:** escala los datos individualmente a una distribución unitaria cuya longitud sea uno. Normalmente, el rango de los datos también se transforma en un rango de entre [0-1].
- **Estandarización:** escala los datos de forma que su varianza sea igual a 1, eliminando la distribución media. El rango de la escala depende de los valores a escalar.
- **Escalado robusto:** escala los valores eliminando los valores atípicos y reduciendo la distancia de otras variables estadísticas. Estos valores hacen que la media y la desviación estándar aumenten, representando así valores anormales (valores atípicos) de la distribución.

El conjunto de datos presenta las siguientes escalas en cada atributo sin aplicar ninguna de estas técnicas:

Atributo	Frecuencia Cardíaca	Frecuencia Respiratoria	Temperatura Corporal	Respuesta Galvánica	Nivel Dolor
Escala	[41 - 238]	[6 - 28]	[35.85 - 36.81]	[0.01 - 2.25]	[0,1,2]
Unidad	BPM	RPM	°C	V	-

Figura 18: Rango de datos por cada atributo.

La elección de la técnica de escalado dependerá de la representación de los datos una vez realizada la transformación.

2.2.2.1.1.2.4- Compilación:

Para la compilación del modelo, se debe especificar una serie de conceptos bastante sencillos que permiten que el algoritmo de la red neuronal de manera más sencilla.

Función de pérdida: esta función mide la desviación entre los valores reales y las predicciones que realiza la red neuronal [23]. El resultado de este método es inversamente proporcional a la eficiencia de la red, es decir, cuanto menor es el resultado de esta función, más eficiente es la red neuronal.

Hay una cierta cantidad de funciones de pérdida desarrolladas, como puede ser el error absoluto medio, sin embargo, nosotros nos centraremos en la crosentropía categórica.

La crosentropía categórica está diseñada como un conjunto de reducciones del ruido que afectan a la arquitectura de la red y a la velocidad de procesamiento de las neuronas.

Optimizadores: son valores cambian el resultado de la salida de cada neurona y lo adapta en función. Algunos de los más conocidos:

- **Adaptive Gradient Algorithm (AdaGrad):** El algoritmo AdaGrad considera un valor uniforme para todos los pesos, se mantiene un factor de entrenamiento específico para cada uno de ellos.
- **Adam (Adaptive moment estimation):** El algoritmo Adam combina las características de varios optimizadores permitiendo que su rendimiento sea superior.

Para nuestro modelo utilizaremos el optimizador Adam pues el rendimiento es mejor utilizando esta técnica.

Historia: una de las partes del proceso de compilación del modelo consiste en la definición de una historia. En este punto, el algoritmo realiza una técnica de propagación, hacia atrás y hacia delante, que permite ajustar las ponderaciones y sesgos cada vez que calcula una salida con el objetivo de minimizar la diferencia entre el valor estimado y el valor real. Cuanto menor sea la pérdida para una red, más precisa será [24].

Cada ciclo de ajuste para reducir la diferencia se denomina época. Es cierto que este proceso consume una gran cantidad de recursos, pero el motivo de reducir la diferencia entre valores estimados y reales hace que merezca la pena realizar este procedimiento.

Podríamos pensar que cuantas más épocas menor función de pérdida del modelo, sin embargo, esto no es cierto, pues la curva que representa la evolución del error durante el entrenamiento muestra una tendencia a distanciar los valores estimados (entrenados) y los valores reales (validados). Por ello, es de vital importancia escoger la precisión del modelo en el punto en el que convergen los dos valores en la gráfica.

Después se realiza un estudio de las métricas que proporciona el modelo para observar la fiabilidad en función del conjunto de entrenamiento.

Métricas: las redes neuronales contienen una serie de datos que proporcionan información sobre el modelo, la más importante con diferencia es la accuracy.

- **Accuracy:** esta variable representa el número total de valores correctamente clasificados. Esto permite diferenciar una comparación en estudio de modelos observando en este porcentaje.

2.2.2.2. Implementación

Para implantar el modelo de Deep learning que hemos seleccionado, lo primero que hay que realizar es la carga y el preprocesamiento de datos cuyo resultado permita un aumento de la precisión del modelo una vez esté definido y compilado.

2.2.2.2.1- Preprocesamiento:

A continuación, se muestran las funciones que permiten realizar las siguientes etapas de preprocesamiento, definidas en el análisis:

Etapa	Función
Limpieza de datos faltantes	df.isna().sum
Codificación OrdinalEncoder	codification = OrdinalEncoder(categories)
Codificación OneHot	to_categorical(categories)

Cuadro 7: Funciones para las etapas de preprocesamiento.

2.2.2.2.2- Escalado

En este apartado se realiza un estudio sobre las distintas técnicas de preprocesamiento de datos que definen un cambio en la escala con el objetivo de que todos los datos comparten una escala común manteniendo la distribución general y las relaciones entre los datos originales. El entorno de programación permite implementar esta técnica importando preprocessing incluido en la librería scikit-learn.

La equivalencia de las técnicas de normalización con las funciones en esta librería son las siguientes:

Técnicas	Función
Escala mínimo-máximo	MinMaxScaler()
Normalización	Normalizer()
Estandarización	StandardScaler()
Escalado robusto	RobustScaler()

Cuadro 8: Funciones para el escalado de los datos.

2.2.2.2.3- Red neuronal:

La red neuronal de este modelo está compuesta por una capa de entrada, dos capas ocultas y una capa de salida. Es cierto que cuantas más capas ocultas, mejor precisión del modelo, pero el tamaño del conjunto de entrenamiento presenta un número de muestras adaptable a una red neuronal con dos capas ocultas.

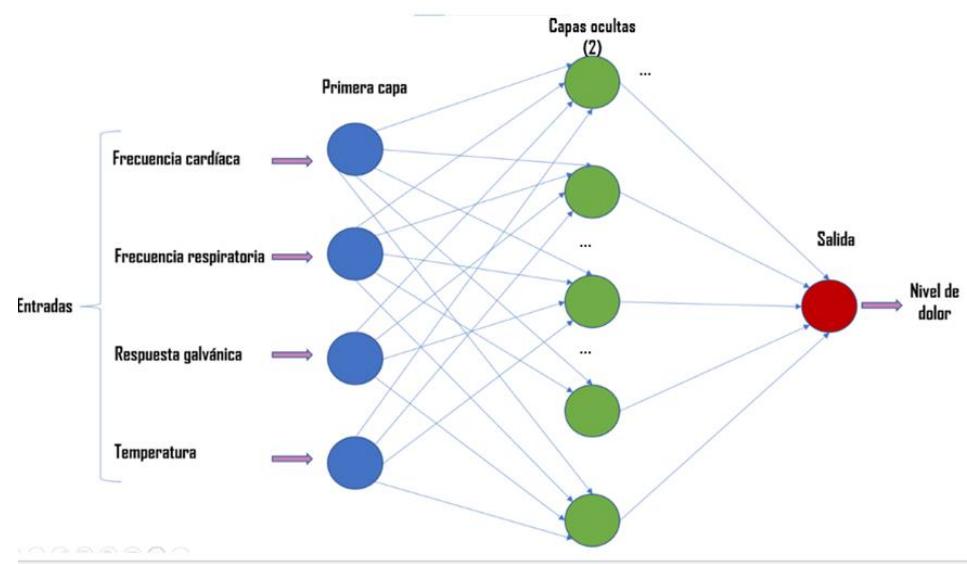


Figura 19: Red neuronal del proyecto.

Nota. Red neuronal definida por elaboración propia.

Las características fundamentales del modelo son:

Capa	Función	Número de neuronas	Función de activación
Entrada	Sequential	4	-
Oculta 1	Dense	64	ReLU
Oculta 2	Dense	64	ReLU
Salida	Dense	3	softmax

Cuadro 9: Características de la red neuronal.

Como el tamaño del conjunto de datos es de 1000 muestras, hemos decidido que la compilación del modelo se realizará con una historia de 100 épocas.

2.2.2.3- Resultados

2.2.2.3.1- Conjunto de datos:

El dataset que se carga en los diferentes entornos en los que se realizan el estudio de los modelos de Inteligencia Artificial es proporcionado tras un ensayo clínico en el que participaron cinco personas. En este ensayo se clasificó el nivel del dolor de los participantes del ensayo, en un instante determinado, en función de las medidas de las variables fisiológicas obtenidas a través del circuito que compone el sistema. Para ello, el ensayo contó con tres pacientes que padecían dolor y otros dos pacientes que no, y se les pidió que fuesen clasificando el nivel de dolor en ese instante.

Este ensayo proporcionó un conjunto de datos de mil muestras aproximadamente que fue cargado en los diferentes entornos que permiten el estudio de los modelos dando los resultados que se comentan a continuación.

2.2.2.3.2- Preprocesamiento

2.2.2.3.2.1- Limpieza de datos faltantes

Para realizar esta fase de la etapa de preprocesamiento hemos acudido a una función que permite calcular la suma de todos los valores nulos de cada una las columnas del conjunto de datos.

```
Frecuencia Cardiaca      64
Frecuencia Respiratoria  19
Respuesta Galvanica     73
Temperatura Corporal    0
Nivel Dolor              0
dtype: int64
```

Figura 20: Valores nulos del conjunto de entrenamiento

Nota. Figura obtenida del entorno Google Colab.

Observando esta imagen, podemos afirmar que el elemento sensorial más fiable a la hora de adquirir las señales fisiológicas del sistema es el sensor de temperatura corporal puesto que no presenta ningún valor faltante de las 1000 muestras que se han recogido en el conjunto de entrenamiento, seguido del micrófono. En cambio, el sensor de respuesta galvánica muestra el peor dato de todos ya que ha obtenido cerca de 80 valores faltantes, debido a un desajuste de los electrodos en el ensayo clínico. Por otra

parte, podemos decir que el sensor de frecuencia cardíaca muestra un valor confiable pues no presenta un gran número de valores faltantes.

Como definimos en el análisis, el relleno de los valores nulos se realizará en función del estudio estadístico de cada una de las columnas que presentan valores faltantes, para ello, se calculó la media, la mediana y la moda y se decidió cuál de las tres variables escoger en función de los datos que proporciona el conjunto, utilizando el método `fillna()`.

Columna	Valores faltantes	Media	Mediana	Moda
Frecuencia Cardíaca	64	112.55	103	95
Frecuencia Respiratoria	19	20.004	20	19
Respuesta Galvánica	73	0.0626	0.04	0.04
Temperatura Corporal	0	No aplica		
Nivel Dolor	0	No aplica		

Cuadro 10: Estudio estadístico de los datos faltantes.

Como se puede observar, para la variable se escogió la frecuencia cardíaca se escogió la moda porque la clasificación del nivel de dolor de las filas en las que se encuentran los valores faltantes en esta variable es “Bajo”, en la gran mayoría de ellos, dando lugar a que todos los valores faltantes que aparezcan en la columna de la variable de la frecuencia cardíaca serán sustituidos por 95 pulsaciones por minuto.

Después, en la frecuencia respiratoria se escogió la mediana por la misma justificación que en la variable anterior. En este caso, los valores faltantes se reemplazan por las 20 respiraciones por minuto que indica la mediana.

Sin embargo, para los valores faltantes de la respuesta galvánica se escogió la media de todos los valores de indicadores de respuesta galvánica del conjunto debido a que la gran mayoría de valores faltantes que aparecen en la columna que muestra la actividad electrodérmica de la piel tienen una clasificación del nivel de dolor “Alto”, por tanto, los valores faltantes se eliminarán y se incluirán los 0.062 voltios de media.

Cabe resaltar que para la temperatura corporal no se ha necesitado realizar ningún tipo de estudio pues esta variable no presenta ningún valor faltante en su registro al igual que la variable que clasifica el nivel de dolor.

2.2.2.3.1.3- Codificación de variables categóricas

2.2.2.3.1.3.1- Codificación ordinal

En este punto se realizó la importación de la función *OrdinalEncoder* disponible en la librería de sklearn que permite realizar la codificación ordinal de los datos de la columna Nivel de dolor. Como comentamos en el análisis, esta técnica permite transformar variables ordinales en variables de tipo numérico, siguiendo un orden definido por la propia herramienta. La variable ordinal Nivel Dolor se transformó a variable numérica, siguiendo un orden establecido, de la siguiente manera:

Nivel Dolor	
Categorías (Variable Ordinal)	Categorías (Variable Numérica)
Bajo	0
Medio	1
Alto	2

Cuadro 11: Categorías de la codificación OneHot.

2.2.2.3.1.3.2- Codificación OneHot

Para poder definir el modelo de red neuronal que queremos implementar, es necesario la realización de la codificación en caliente OneHot. Para ello, se importó la herramienta “to_categorical” de la librería keras. Los valores numéricos codificados anteriormente, son transformados al estándar de la codificación OneHot sobre la variable que clasifican el nivel de dolor. Este paso el aumento de la precisión del modelo con seguridad.

2.2.2.3.1.4- Escalado

Tras realizar las anteriores etapas de preprocesamiento, comienza la fase de elección de la escala. Para ello, se representan los resultados de la ejecución a través de gráficas de dispersión y barras que comparan cada una de las funciones de escalado aplicadas a cada uno de los atributos de la siguiente manera.

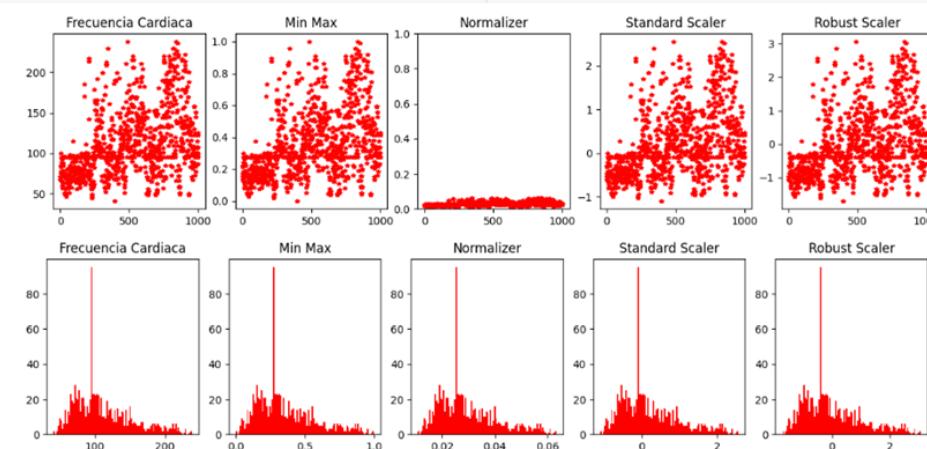


Figura 21: Gráficas del escalado de la frecuencia cardiaca.

Nota. Estudio realizado por elaboración propia.

Como podemos observar, la frecuencia cardíaca presenta una distribución de muestras similar a la distribución normal en la que las muestras se repiten en el centro de la distribución. Así como un mayor número de muestras cuando las pulsaciones por minutos son menores de 100 BPM.

El escalado mínimo-máximo muestra una representación prácticamente similar la muestra sin procesar, cambiando la escala de rango [41-238] pulsaciones por minuto al rango [0-1]. Algo similar ocurre con la estandarización y el escalado robusto, sin embargo, la escala que presentan al utilizar en estas técnicas es muy variable dependiendo del atributo, además, el rango de la escala permite encontrar valores negativos de las muestras procesadas. Esto es algo que puede provocar conflictos a la hora de que el modelo obtenga información de los datos en su entrenamiento. Mientras tanto, aunque el gráfico de barras de la variable procesada utilizando el normalizado sea similar al gráfico de barras de la variable sin procesar, con una escala en un rango compacto adecuado y que no presenta valores negativos, el gráfico de dispersión presenta que la distribución de las muestras está amontonada, lo que puede inducir fallos y confusiones a la hora del entrenamiento del modelo, bajando el porcentaje de instancias correctamente clasificadas, entre otras métricas, dando lugar a un modelo de poca confianza.

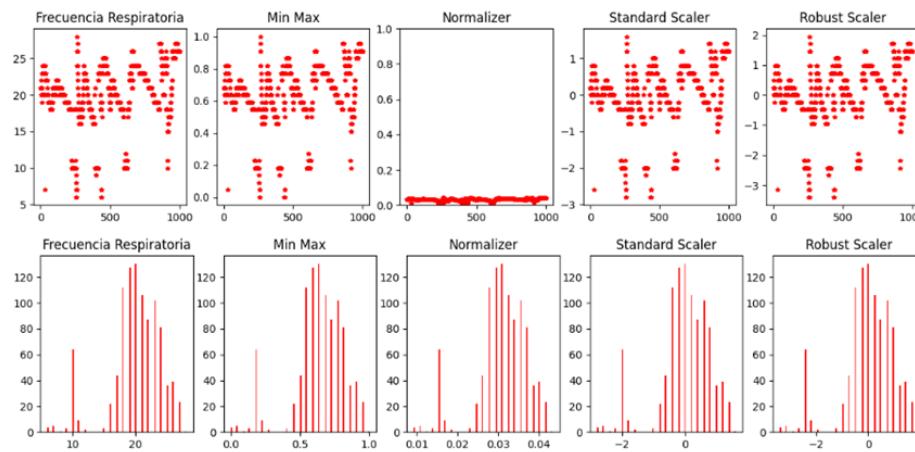


Figura 22: Gráficas del escalado de la frecuencia respiratoria.

Nota. Estudio realizado por elaboración propia.

La distribución de la variable, en este caso, se asemeja a una distribución normal, pero en este caso por la derecha de la gráfica, en todas las escalas. El comportamiento de la frecuencia respiratoria es similar al de la frecuencia cardíaca cuando se transforma su escala. Los datos se amontonan en la normalización y, tanto el escalado estándar como el escalado robusto presentan valores negativos en su escala y esta vez, más negativos que positivos. Por ello, podemos afirmar que el mejor método de escalado para esta variable también es la representación de los datos en la escala mínimo-máximo.

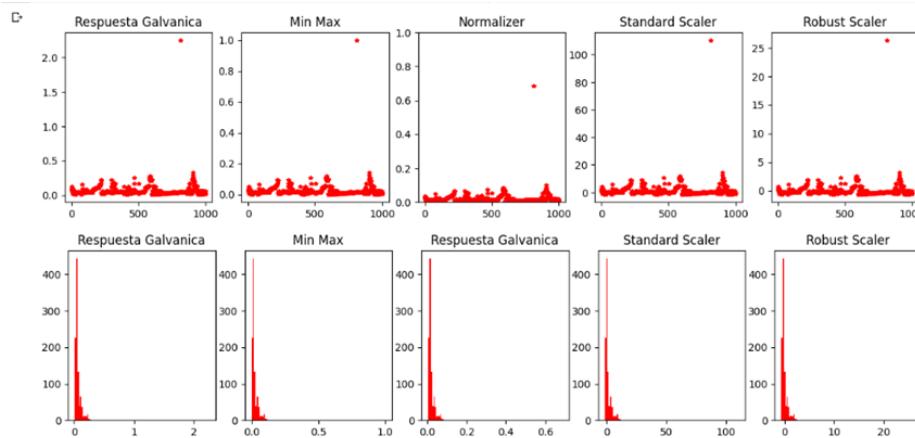


Figura 23: Gráfica del escalado de la respuesta galvánica.

Nota. Estudio realizado por elaboración propia.

La representación de la respuesta galvánica es igual en todas las escalas puesto que los valores que contiene esta variable son muy pequeños, en una escala centesimal. La normalización amontona un poco los resultados si nos fijamos bien también en este

caso. Además, cabe resaltar que la escala de las técnicas de estandarización y escalado robusto se disparan, llegando a valores superiores a las dos centenas, como es el caso de la estandarización. Por ello, podemos afirmar la mejor técnica de transformación para este caso también es el mínimo-máximo.

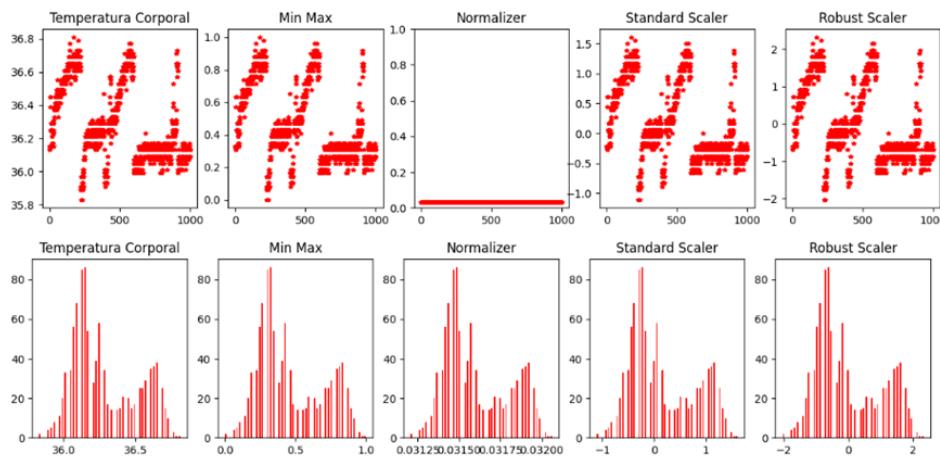


Figura 24: Gráfica del escalado de la temperatura corporal.

Nota. Estudio realizado por elaboración propia.

Como podemos observar, la temperatura corporal presenta una distribución de muestras peculiar. En este caso, la escala normal de los datos es muy pequeña, con una diferencia de aproximadamente un solo grado entre el valor máximo y el valor mínimo de los datos de temperatura corporal, por lo que, de entrada, podemos afirmar que la escala de la técnica mínimo-máximo es la más adecuada para esta variable. Además, como pasa en el resto de las variables, la estandarización y el escalado robusto presentan valores negativos dentro de su escala proporcionando información errónea al conjunto, aunque es importante recalcar que en este caso las escalas no se disparan, si no que se mantienen valores cercanos a uno. La técnica de normalización sigue agolpando los datos, de hecho, aparece como si no hubiera ningún dato, según el gráfico de dispersión, por lo que podemos clasificar esta técnica como la menos fiable.

Tras este breve análisis, hemos llegado a la conclusión que las escalas de los atributos frecuencia cardíaca, frecuencia respiratoria, temperatura corporal y respuesta galvánica presentan un comportamiento en una escala entre [0-1] similar al que presentan en la escala normal. Por ello, se ha decidido que estas variables deben pasar por un proceso de transformación utilizando la técnica de escalado mínimo-máximo ya que el resto de las técnicas no aportan información fiable al conjunto de datos, ya sea por el amontonamiento de los datos, como es el caso de la normalización, o por los valores negativos dentro de las escalas del escalado robusto y la estandarización.

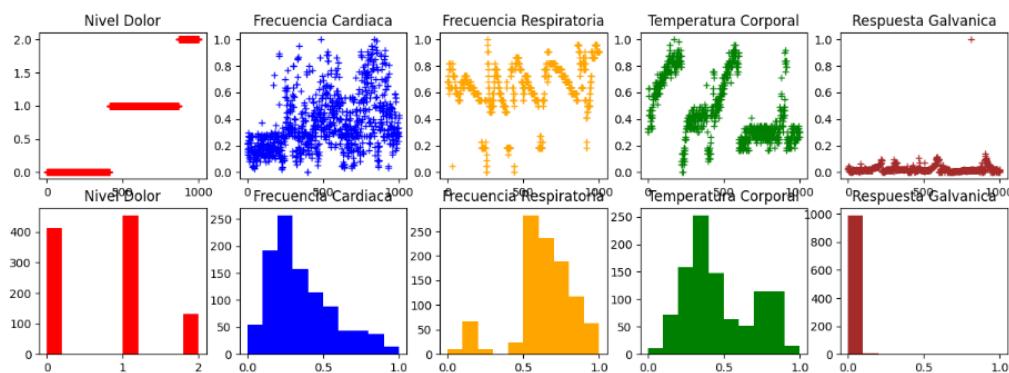


Figura 25: Normalización de los datos con escalado min-max.

Nota. Estudio realizado por elaboración propia.

En esta imagen se puede realizar una comparación de la normalización de los datos en la escala seleccionada en el estudio. Las gráficas de arriba representan gráficas de dispersión en la escala mínimo-máximo y las de abajo representan gráficos de barras en la escala normal de cada uno de los atributos que componen el conjunto de entrenamiento.

Para completar el apartado de preprocesamiento, realizamos un análisis de la correlación entre las variables.

	Frecuencia Cardiaca	Frecuencia Respiratoria	Respuesta Galvanica	Temperatura Corporal	Nivel Dolor
Frecuencia Cardiaca	1.000000	-0.022845	0.139896	-0.061336	0.327505
Frecuencia Respiratoria	-0.022845	1.000000	-0.044550	0.108603	0.357027
Respuesta Galvanica	0.139896	-0.044550	1.000000	0.216312	0.058432
Temperatura Corporal	-0.061336	0.108603	0.216312	1.000000	-0.259105
Nivel Dolor	0.327505	0.357027	0.058432	-0.259105	1.000000

Figura 26: Análisis de correlación de las variables.

Nota. Estudio obtenido por elaboración propia.

Los datos más interesantes de estudio son los valores que presentan una correlación entre el nivel de dolor y cada uno del resto de los atributos, tal y como hemos recogido en esta tabla:

	Frecuencia Cardíaca	Frecuencia Respiratoria	Respuesta Galvánica	Temperatura Corporal
Nivel de dolor	0.327505	0.357027	0.058432	-0.259105

Cuadro 12: Nivel de correlación entre las variables y el nivel de dolor.

Como podemos observar, la variable que indica una correlación significativa con el nivel de dolor es la frecuencia respiratoria, seguida de la frecuencia cardíaca. La respuesta galvánica y la temperatura corporal apenas muestran unión con el nivel de dolor, por lo

que podríamos obviarlas del conjunto de entrenamiento. Esto representaría un mayor rendimiento a la hora de entrenar el modelo, pero, teniendo en cuenta el tamaño del conjunto de datos, se decide no eliminar estas variables pues, aunque muestren poca correlación con el nivel de dolor, aportan algo de información a la estructura del modelo y al resto de atributos, por ejemplo, la respuesta galvánica presenta una correlación del 0.216312 afirmando que cuando se produce actividad electrodérmica en la piel, aumenta la sudoración, todo ello provocado por un aumento de la temperatura corporal.

En resumen, en este estudio queda reflejado que la frecuencia respiratoria y la frecuencia cardíaca son las variables más importantes para clasificar el nivel de dolor.

2.2.2.3.2- Red Neuronal

Una vez finalizada la etapa de procesamiento de los datos, cuyo objetivo era preparar el conjunto de manera que pueda aumentar su precisión el mayor número posible, se procede a la definición del modelo de la red neuronal. Como indicamos en la implementación, el modelo está compuesto por una red neuronal con una capa de entrada que recibe las variables fisiológicas, dos capas ocultas con 64 neuronas cada una y una capa de salida con 3 neuronas.

Después de entrenar el modelo, se procede a la compilación del modelo. Este proceso consiste en la configuración de ciertos parámetros que proporcionen completitud al algoritmo. Para esta red, se ha decidido compilar el proyecto con:

- Función de pérdida: categorical_crossentropy
- Optimizador: adam.
- Métricas: accuracy

Una vez compilado el modelo, se realiza un estudio de su comportamiento a través de la gráfica que representa la evolución del error durante el entrenamiento en función de las épocas muestra que el error es muy alto en un número bajo de épocas. En este punto, los valores estimados son similares a los validados, hasta llegar al punto en el que convergen, aproximadamente en la octava época. A partir de ahí, los valores comienzan a mostrar una diferencia significativa de un 0,15 de error, aproximadamente.

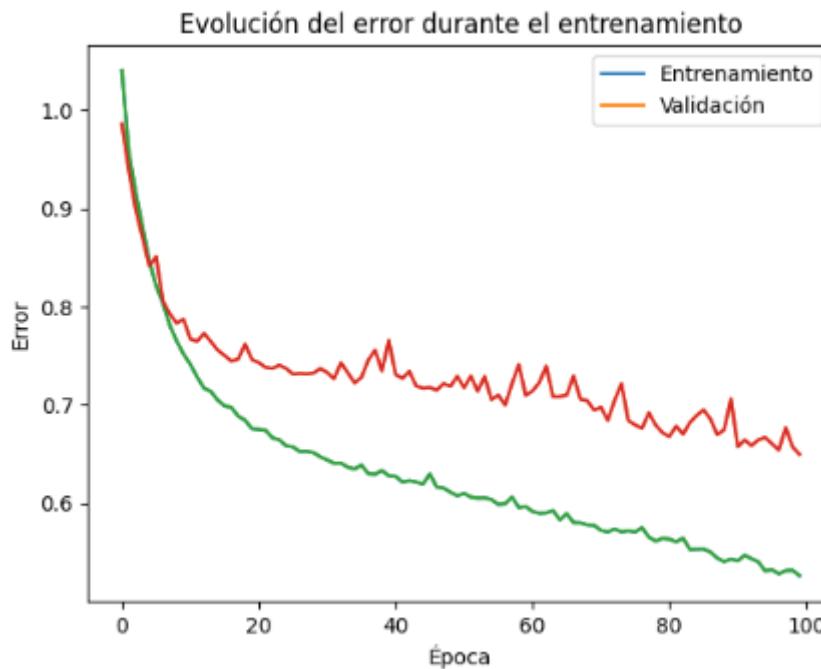


Figura 27: Evolución del error durante el entrenamiento.

Nota. Gráfica obtenida por elaboración propia.

En esa época en la que aproximadamente convergen los valores de entrenamiento y validación, aparecen las siguientes métricas como resultado:

Época (epoch)	Precisión (accuracy)	Función de pérdida (loss)
8	0.6513	0.7808

Cuadro 13: Métricas del modelo de la red neuronal.

Estas métricas representan los resultados del modelo definido informando de que la fiabilidad del modelo es aceptable, teniendo en cuenta que los valores fisiológicos han sido obtenidos a través de un prototipo del sistema.

Otra métrica de resultado es la matriz de confusión. Como podemos observar a continuación, la diagonal principal muestra que el modelo ha acertado en un alto grado de certeza, en comparación a lo que ha fallado, indicado en la diagonal secundaria.

Matriz de confusión:
[[65 12 7]
 [7 78 5]
 [2 16 9]]

Figura 28: Matriz de confusión de la red neuronal.

Nota. Matriz realizada por elaboración propia.

2.2.2.3- Sprint 3: Inteligencia Artificial II – Machine Learning:

2.2.2.3.1- Análisis:

- **Machine Learning:**

El Machine Learning se define como un conjunto de técnicas de análisis de datos, reconocimiento de patrones y obtención de predicciones a partir de la definición de modelos que realicen el aprendizaje de estas habilidades de forma autónoma [25].

Las aplicaciones de esta tecnología cada día son mayores. Desde la detección de patrones en imágenes médicas con el objetivo de elaborar un diagnóstico hasta la clasificación de procesos industriales que se repiten de manera cíclica, entre otros miles de posibilidades. En resumen, el objetivo del machine learning es aportar autonomía a procesos que contienen ciertos patrones en base al aprendizaje de los mismos mejorando la eficiencia del sistema cada vez que se realiza la tarea.

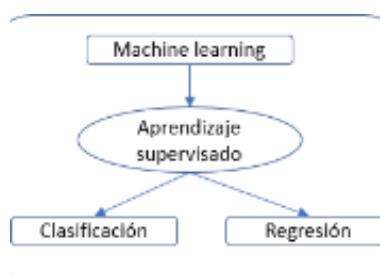


Figura 29: Esquema de tipos de aprendizaje supervisado en machine learning.

Nota. Esquema realizado por elaboración propia.

Para entender mejor el funcionamiento del sistema se procede a comentar la división del machine learning en función de la forma en la que se le enseña al algoritmo que compone el modelo:

- **Aprendizaje supervisado:**

El aprendizaje supervisado consiste en un conjunto de algoritmos que funcionan a base de un entrenamiento previo de los datos etiquetados. Se supone que conocemos el valor de cada etiquetado. Esto le permitirá al algoritmo poder “aprender” una función capaz de predecir el atributo objetivo para un juego de datos nuevo.

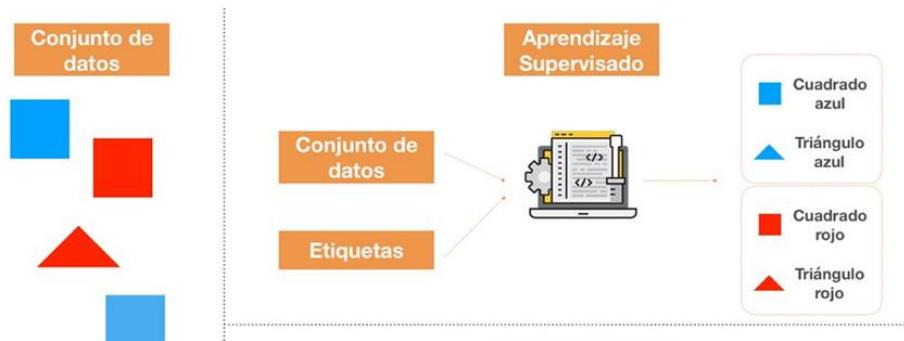


Figura 30: Esquema del aprendizaje supervisado.

Nota. Figura obtenida de sitio web [IV].

Las dos grandes familias de algoritmos supervisados son [26]:

- Los algoritmos de regresión cuando el resultado a predecir es un atributo numérico.
- Los algoritmos de clasificación cuando el resultado a predecir es un atributo categórico.

Dentro del machine learning existen otros dos tipos de aprendizaje, el aprendizaje no supervisado y el aprendizaje reforzado.

Los métodos de aprendizaje no supervisado son algoritmos que basan su proceso de entrenamiento en un conjunto de datos sin ningún valor clasificadorio, ya sea categórico o numérico, cuyo objetivo es encontrar grupos similares en el conjunto de datos.

El aprendizaje reforzado basa su proceso en un conjunto de datos de los cuales algunos se conoce la clasificación y de otros no, con el objetivo de hacer predicciones a futuro basadas en comportamientos que ya ha visto y que tiene almacenados [27].

Algunas de las características más importantes de cada conjunto de algoritmos son:

Aprendizaje supervisado	Aprendizaje no supervisado	Aprendizaje reforzado
Es la técnica más utilizada	Es la técnica más complicada	Es la técnica con mayor potencial
Los datos de salida ya son conocidos	Los datos de salida no son conocidos	Los datos de salida pueden o no pueden ser conocidos

Más intervención humana	Menos intervención humana	Algo de intervención humana
-------------------------	---------------------------	-----------------------------

Cuadro 14: Características más importantes de los algoritmos.

Técnicamente, no hay respuesta correcta o incorrecta, Machine Learning simplemente aprenderá la verdad objetiva de que ciertas formas pertenecen juntas hasta cierto grado de probabilidad. Obviamente, Machine Learning comete errores, pero, como nosotros, su fortaleza radica en su capacidad para aprender de sus errores y hacer estimaciones mejor educadas la próxima vez.

La elección de utilizar un algoritmo de Machine Learning supervisado o no supervisado generalmente depende de factores relacionados con la estructura y el volumen de sus datos y del objetivo del problema en cuestión. Un problema complejo por lo general utilizará ambos tipos de algoritmos para construir modelos de datos predictivos que ayuden a tomar decisiones sobre una variedad de desafíos comerciales [28].

En este caso, nos centraremos en algunos algoritmos de aprendizaje supervisado pues el conjunto de datos definido en el sistema contiene un atributo clasificador del resto de atributos. De hecho, para que el modelo pueda reconocer esta clasificación, es necesario realizar una buena selección de características para definir el conjunto de entrenamiento y prueba. Para ello, acudiremos al método de validación cruzada y que se comenta a continuación.

Validación cruzada:

La validación cruzada es una técnica que permite evaluar los resultados de un análisis estadístico a la hora de la división de los datos del conjunto en datos de entrenamiento y datos de prueba. Para ello, se calcula la media aritmética sobre las diferentes divisiones posibles hasta conseguir el mejor conjunto de datos y que permita validar el modelo a definir.

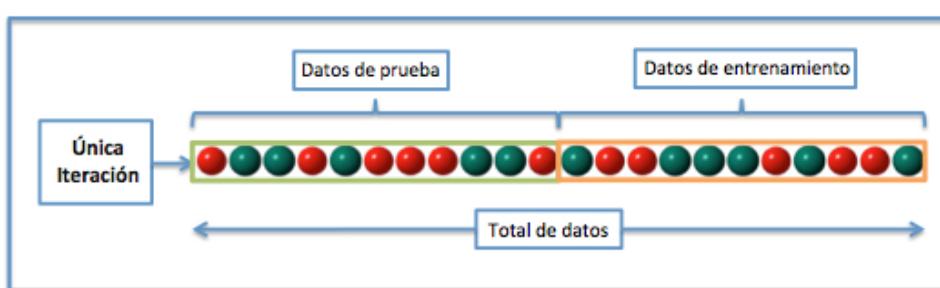


Figura 31: Ejemplo de una iteración de validación cruzada.

Nota. Figura obtenida de sitio web [V].

En cada una de las iteraciones se calcula el error obtenido en cada iteración de la siguiente manera:

$$M = \frac{1}{k} \sum_{i=1}^k M_i, \text{ donde:}$$

k Es el número de iteraciones que va a realizar la técnica.

M Es el error obtenido en cada M_k iteración.

El resultado total será la media aritmética de los errores obtenidos en las k iteraciones.

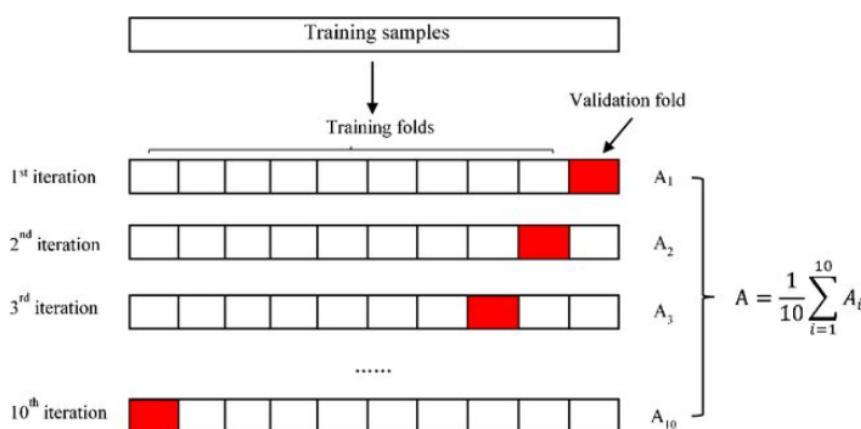


Figura 32: Algoritmo de iteración de validación cruzada.

Nota. Figura obtenida de sitio web [VI].

El número de iteraciones las define el programador con el objetivo de escoger la mejor partición del conjunto de datos posibles. El número de iteraciones elegido para la realización del proyecto se comenta en el apartado de implementación, así como en los modelos en los que se carga dicho conjunto de entrenamiento y prueba para su estudio y que son los que se muestran a continuación.

- **Regresión Lineal:**

Un modelo de regresión lineal es una técnica estadística que permite establecer predicciones que aproximan la dependencia entre una variable dependiente, múltiples variables independientes y un término aleatorio. Si el modelo se puede aplicar a un estudio, se puede decir que hay correlación entre las variables. Los resultados de esta técnica pueden servir de base a la hora de implementar otros modelos más complejos. La ecuación que define se define como:

$Y = \beta_0 + \beta_1 \cdot X_1 + \dots + \beta_m \cdot X_m + \varepsilon$, donde:

Y es la variable dependiente.

X_1, X_2, \dots, X_m son las m variables independientes.

$\beta_0, \beta_1, \dots, \beta_m$ son los parámetros del modelo que se calculan usando el método de mínimos cuadrados

ε es un término aleatorio llamado error. Este parámetro explica la independencia lineal entre X e Y .

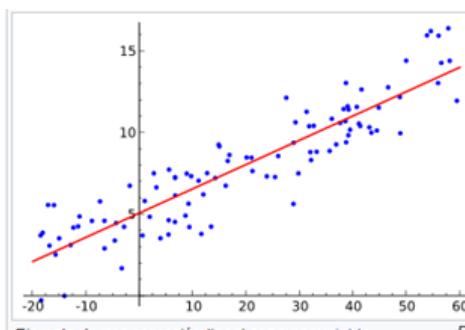


Figura 33: Ejemplo de resultado del algoritmo de regresión lineal.

En esta imagen, el modelo de regresión lineal representa la línea roja que aproxima la dependencia lineal entre el resto de las variables independientes (puntos azules).

Este modelo será implementado en el entorno de Google Colab puesto que se puede realizar en Weka, como el resto de los modelos.

- **Random Forest:**

Los árboles aleatorios es una combinación de modelos entrenados por árboles de decisión. Cada árbol definido en el modelo proporciona un resultado de manera independiente al resto, pero con la misma distribución en todo el conjunto. Dicho resultado está formado por una selección de características que divide cada nodo de cada árbol y calcula una serie de estimaciones que muestran la correlación entre ellos basadas en la regresión. (Y. Freund & R. Schapire, Machine Learning : Proceedings of the Thirteenth International conference, ***, 148–156).

Las estimaciones internas controlan el error, la fuerza y la correlación y se utilizan para mostrar la respuesta al aumento del número de características utilizadas en la división, buscando que la distribución de cada característica sea lo más homogénea posible.

A nivel funcional, el modelo construye un conjunto de múltiples árboles basados en el algoritmo CART [29].

Partiendo de la base, cada salida de un árbol se calcula con un modelo de regresión lineal. Recordamos que la salida estaba compuesta por un error calculado a través del método de los mínimos cuadrados. Como en este caso, se pretende estimar un número de k árboles, cuya función de estimación consiste en:

$$y = f(x) = \begin{cases} r_i & \text{si } x \in C_i; i=1, \dots, k \\ 0 & \text{en otro caso} \end{cases}, \text{ donde:}$$

$$r_i = \frac{\sum_{j=1}^n (y_j | x_j \in C_i, j=1, \dots, n)}{|y_j | x_j \in C_i, j=1, \dots, n|}; i = 1, \dots, k.$$

Representa la salida predicha por cada árbol

que compone el bosque aleatorio.

A partir de este valor, se calcula el error de predicción, dando como resultado una métrica que permite evaluar la precisión del modelo final, de la siguiente manera:

$$\varepsilon_{Cart} = \frac{\sum_{i=1}^n (y_{cart} - y_{verd})^2}{n}, \text{ donde:}$$

y_{cart} representa la ecuación de la regresión lineal de cada árbol.

y_{verd} también representa la ecuación lineal de cada árbol, pero sin el parámetro que indica el error, ε .

Esta ecuación representa el cálculo de los mínimos cuadrados para cada una de las regresiones calculadas en cada árbol que componen el bosque aleatorio [30].

KNN:

Este algoritmo es conocido como técnica de k -vecinos más cercanos [31]. El objetivo de este modelo es el aprendizaje en función de las propiedades y características que comparten los datos estudiados con los que los rodean.

Cada vez que se necesita clasificar un nuevo ejemplo, el algoritmo recorre el conjunto de entrenamiento para obtener los k vecinos y predecir su clase. Esto hace que el algoritmo sea computacionalmente costoso tanto en tiempo, ya que necesita recorrer la totalidad de los ejemplos en cada predicción, como en espacio, por la necesidad de mantener almacenado todo el conjunto de entrenamiento. La ecuación que define este algoritmo es:

$$C_i = i * d(e', e_i) < d(e', e_j), \forall i \neq j, \text{ donde}$$

- C_i indica la asignación de la etiqueta.

- d expresa una distancia entre los atributos vecinos. Hay varias métricas que se pueden definir en este aspecto y es primordial pues determina la precisión de la medida de los atributos que componen el modelo. Como indica este caso, la más métrica más frecuente es la distancia.
- e son los valores de los atributos en medición.

- **J48: ÁRBOLES DE DECISION**

El algoritmo J48, es una adaptación del modelo de árbol de decisión para su implementación en el entorno de desarrollo de Weka. Este algoritmo permite la clasificación de un conjunto de datos a partir de los resultados obtenidos en la definición del modelo.

Para clasificar los datos, el algoritmo calcula el árbol de decisión con el objetivo de realizar una poda de los nodos en función del rendimiento del mismo. Para ello, calcula el nivel de confianza de la poda de cada nodo y la compara con el error cometido entre la variable real y la predictora en cada una de las n iteraciones del modelo. Es decir, el algoritmo define un árbol de decisión, el cual establece un nivel de confianza concreto. En el siguiente paso, el propio algoritmo se encarga de podar el nodo del árbol cuyo error cometido sea el más bajo de todos los nodos del árbol y calcula el nivel de confianza de ese nuevo árbol que da como resultado de dicha poda. Si el nivel confianza del nuevo árbol es mayor que el del original, el algoritmo establece como el nuevo árbol como el árbol válido del modelo y repite el proceso de poda sobre el siguiente árbol con el menor error cometido hasta que el nivel de confianza de un nuevo árbol sea menor que el nivel de confianza del árbol que se está comparando.

El resultado será un árbol de decisión con el mayor nivel de confianza posible [32].

Implementación

La implementación se ha realizado utilizando tanto el entorno de Google Colab como el entorno de Weka.

En nuestro caso, importamos el conjunto de entrenamiento definido en el primer sprint: lepasyDataset.csv en los dos entornos.

En el caso del modelo de regresión linear el modelo se entrena utilizando la librería scikit_learn .

El resto de los modelos se estudian en el entorno de Weka. Una vez se cargan los datos en esta herramienta, aparece una pantalla con la descripción de cada uno de los atributos que componen el conjunto de entrenamiento.



Figura 34:Conjunto de datos en Weka.

Nota. Estudio realizado por elaboración propia.

En esta imagen se observa un breve resumen del atributo “Nivel Dolor” en el que se muestra la siguiente información sobre el tipo de datos que representa (Nominal), así como el número de muestras que aparece cada variable en el conjunto de entrenamiento a través de una tabla y un gráfico de barras. Como podemos observar, el valor del nivel de dolor que más se ha seleccionado ha sido el “Medio” (Rojo), seguido de cerca por el “Bajo” (Azul oscuro) y, con menor frecuencia, el “Alto” (Azul claro).

Para definir los modelos el entrenamiento de los modelos que queremos estudiar es necesario acceder a la pestaña “Classify”. En esta pantalla, se muestran todas las opciones disponibles para implementar en el conjunto de entrenamiento.

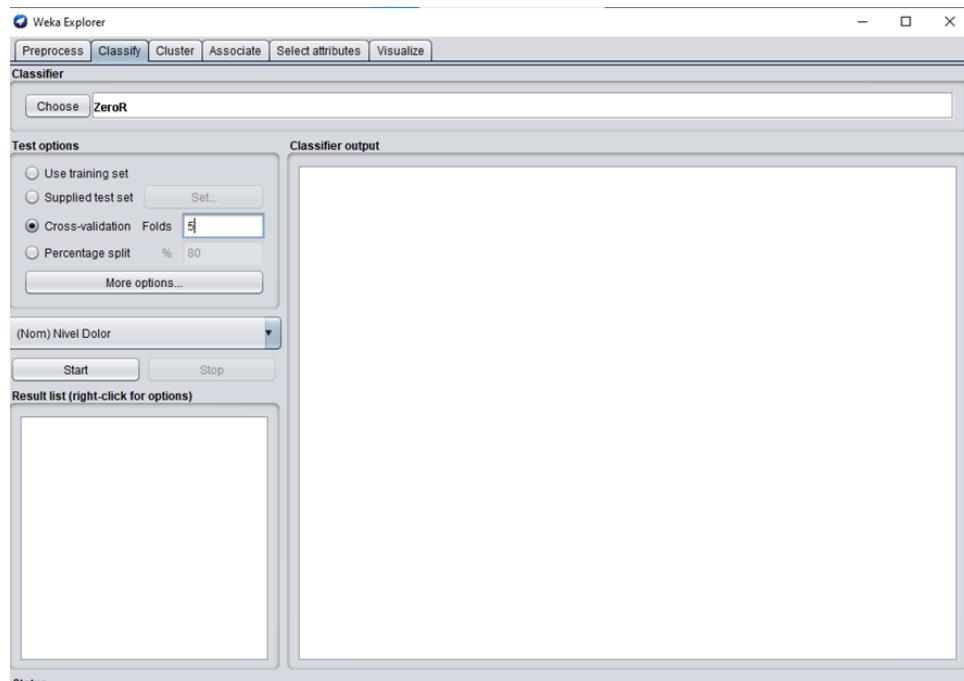


Figura 35: Opciones de clasificación en Weka.

Los pasos para definir un modelo en función de las funciones disponibles en esta pantalla son las siguientes:

- Test options: en este apartado se define cómo se va a realizar la carga de datos, utilizando el conjunto de entrenamiento al completo, utilizando la técnica de validación cruzada o seleccionando las características del modelo. En el caso del Deep learning, se utilizó la definición del modelo a través la selección de características para dividir el conjunto en un 80% aplicado a los datos de entrenamiento y en un 20% aplicado a los datos de prueba. En este caso, se realizará la técnica de validación cruzada para que todos los datos del conjunto sean utilizados tanto para el entrenamiento como para la prueba, permitiendo obtener unos resultados más fiables. Se decidió que la técnica de validación cruzada se realizará para k-folds = 5.

Justo debajo de esta opción aparece un selector de tipo combo en el que se especifica cuál es el atributo que contiene los datos sobre la clasificación. Como se puede observar, nuestra variable clasificadora es el atributo “Nivel Dolor”.

- Classifier: permite la selección del modelo que queremos implementar. Para ello, es necesario especificar un filtro cuyo nombre sea el que contiene el modelo que queremos definir y pulsar el botón “Start”. En la pantalla “Classifier output”

aparecerá el resultado de la implementación del modelo con el conjunto de entrenamiento determinado. La equivalencia de los modelos que vamos a definir con los filtros de Weka que aparecen en este menú de opciones son:

Modelo (Real)	Modelo (Weka)
Árbol de decisión	J48
Bosque aleatorio	Random Forest
K vecinos más cercanos	IBk

Cuadro 15: Equivalencia de los modelos reales en Weka.

La ejecución de los diferentes algoritmos en este entorno nos proporcionará un conjunto de métricas que nos permitirán conocer cuál es el comportamiento del modelo.

2.2.2.3.2- Resultados:

Captura de datos:

Si comparamos los valores de todos los atributos en función del nivel de dolor y el número de veces que se repite la muestra, podemos observar los siguientes gráficos:

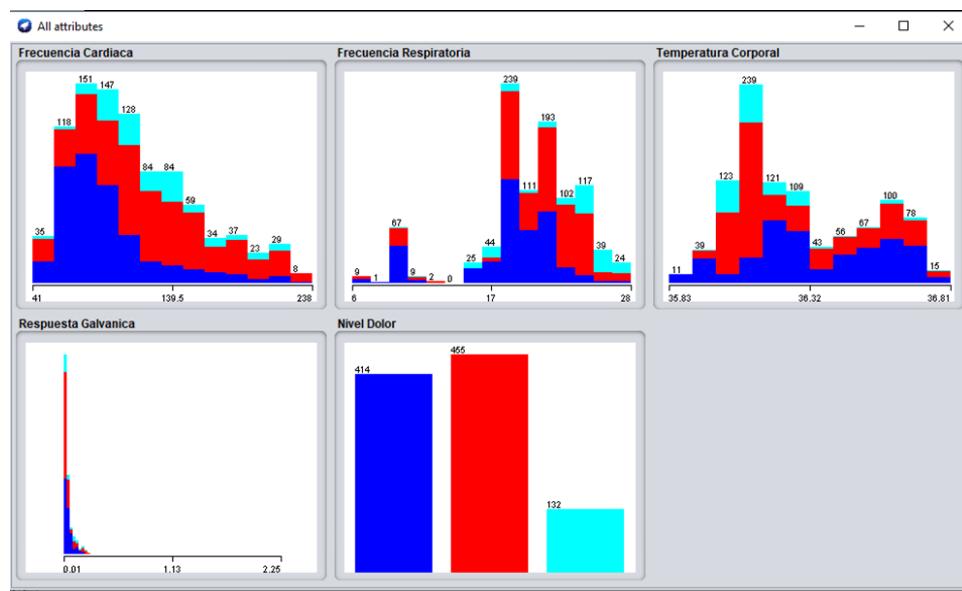


Figura 36: Comparación de los atributos en función del nivel de dolor.

Nota. Estudio realizado por elaboración propia.

La escala de cada gráfico depende de la unidad de cada atributo. Por ejemplo, la escala de la frecuencia cardiaca está entre 41 y 236 pulsaciones por minuto y presenta un mayor número de datos dentro del rango de valores [41, 129.5] pulsaciones por minuto.

A continuación, aparece una tabla con los datos más representativos de cada gráfico:

Atributo	Escala	Unidad	Muestras más frecuentes
Frecuencia cardíaca	[41 - 238]	Pulsaciones por minuto (BPM)	[41 - 129.5]
Frecuencia respiratoria	[6 - 28]	Respiraciones por minuto (RPM)	[18 - 25]
Temperatura corporal	[35.83 – 36.81]	Grado Celsius (°C)	[36 – 36.32]
Respuesta galvánica	[0.01 - 2.25]	Voltios (V)	[0.01 – 0.015]

Cuadro 16: Información sobre cada uno de los atributos.

Todos los valores muestran que los datos se han capturado correctamente a través en el bloque instrumental, obviando un par de datos sin sentido, como el de una muestra con valor frecuencia cardíaca entre [200 - 238] pulsaciones por minuto, el resto de los valores corresponden con las mediciones en la realidad o los valores que presenta la respuesta galvánica cuyos datos aparecen de forma muy agolpada dentro de un rango muy pequeño, debido a que la actividad eléctrica de la piel se mide en variaciones muy pequeñas de voltaje, proporcionando muy poca información al conjunto de entrenamiento. Esta información se podría eliminar del conjunto, sin embargo, se ha decidido mantenerla puesto que, aunque proporcione poca información sobre el comportamiento del conjunto, no es interesante eliminar un atributo de los cinco que aparecen.

- **Modelos:**

En este apartado se muestran los resultados de los diferentes modelos definidos en el análisis. Cabe resaltar que no todos los modelos han sido ejecutados en los mismos entornos de programación.

Modelo	Entorno
RegressionLinear	Google Colab
IBk	Weka
J48	Weka

RandomForest	Weka
--------------	------

Cuadro 17: Entornos de programación para cada modelo.

Comenzando con el modelo de regresión linear definido en Google Colab, aparecen los siguientes resultados:

```
PRECISIÓN DEL MODELO DE REGRESIÓN LINEAR MÚLTIPLE: 0.3433169122641547
Error cuadrático medio: 0.3470568928380528
Coeficientes: [[ 0.00521591  0.07012692  0.74913268 -0.92190983]]
Sesgo (Coeficiente W0: [32.14583155]
Coeficiente de determinación: 0.3433169122641547
```

Figura 37: Resultados del modelo de regresión lineal.

Nota. Estudio realizado por elaboración propia.

Como vemos, los resultados de este modelo afirman que presenta una fiabilidad bastante baja pues el accuracy es de un 34,33% aproximadamente, sin embargo, la precisión es relativamente buena pues el valor del error cuadrático medio es bajo.

En cuanto a los resultados que se nos muestran al ejecutar los algoritmos en Weka, cabe indicar que aparecen varios indicadores que permiten observar el comportamiento del modelo:

- Porcentaje de instancias correcta e incorrectamente clasificadas a nivel global.
- Índice Kappa de Cohen: Índice de bondad de ajuste del modelo completo. Se entiende que el modelo es apropiado si este valor supera el valor 0.7.
- Error absoluto medio: Estadístico que indica el error de estimación cometido. Valores más bajos indican que el modelo es más apropiado
- Porcentaje del área bajo la curva ROC: La curva ROC indica la relación entre la sensibilidad (verdaderos positivos entre el total de positivos) y especificidad (verdaderos negativos entre el total de negativos) en un eje de abscisas por cada atributo dentro del modelo de clasificación establecido.
- Matriz de confusión: Tabla de contingencia que relaciona la clasificación dada por el modelo con el valor real que alcanza el sujeto en la variable criterio.
- Precisión: Porcentaje de instancias bien clasificadas de entre todas las seleccionadas como positivas.

A continuación, se muestra un estudio sobre los valores obtenidos en la ejecución de cada uno de los modelos en los diferentes entornos de programación:

Modelo	Correctly classified instances	Incorrectly classified instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative square error	ROC Area	Accuracy	Confusion Matrix																
LinearRegression	-	-	-	0,3470	-	-	-	0,3433	-	-																
IBk	0,737263	0,262737	0,5602	0,1761	0,4177	0,43658	0,930269	0,787	0,734	<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>c</th><th><-- classif</th></tr> </thead> <tbody> <tr> <td>127</td><td>77</td><td>10</td><td> a = Bajo</td></tr> <tr> <td>78</td><td>346</td><td>31</td><td> b = Medio</td></tr> <tr> <td>31</td><td>36</td><td>65</td><td> c = Alto</td></tr> </tbody> </table>	a	b	c	<-- classif	127	77	10	a = Bajo	78	346	31	b = Medio	31	36	65	c = Alto
a	b	c	<-- classif																							
127	77	10	a = Bajo																							
78	346	31	b = Medio																							
31	36	65	c = Alto																							
J48	0,801199	0,198801	0,6694	0,175	0,322	0,4338	0,717144	0,821	0,801	<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>c</th><th><-- classif</th></tr> </thead> <tbody> <tr> <td>343</td><td>40</td><td>21</td><td> a = Bajo</td></tr> <tr> <td>51</td><td>374</td><td>28</td><td> b = Medio</td></tr> <tr> <td>9</td><td>40</td><td>83</td><td> c = Alto</td></tr> </tbody> </table>	a	b	c	<-- classif	343	40	21	a = Bajo	51	374	28	b = Medio	9	40	83	c = Alto
a	b	c	<-- classif																							
343	40	21	a = Bajo																							
51	374	28	b = Medio																							
9	40	83	c = Alto																							
RandomForest	0,838163	0,161838	0,7306	0,159	0,2815	0,39416	0,626813	0,944	0,836	<table border="1"> <thead> <tr> <th>a</th><th>b</th><th>c</th><th><-- classif</th></tr> </thead> <tbody> <tr> <td>370</td><td>38</td><td>6</td><td> a = Bajo</td></tr> <tr> <td>49</td><td>382</td><td>24</td><td> b = Medio</td></tr> <tr> <td>8</td><td>37</td><td>87</td><td> c = Alto</td></tr> </tbody> </table>	a	b	c	<-- classif	370	38	6	a = Bajo	49	382	24	b = Medio	8	37	87	c = Alto
a	b	c	<-- classif																							
370	38	6	a = Bajo																							
49	382	24	b = Medio																							
8	37	87	c = Alto																							

Cuadro 18: Estudio de los valores obtenidos por los modelos.

Como podemos observar, los resultados de la ejecución de todos los algoritmos presentan unas características presentan unos parámetros bastante fiables. Si nos fijamos en el porcentaje de instancias clasificadas vemos que más del 70% de las características están correctamente clasificadas en los tres modelos ejecutados con el entorno de programación de Weka. Si hablamos de las métricas que indican el error de la clasificación de cada atributo observamos que el algoritmo IBk presenta un mayor número de error respecto a los demás mientras que el algoritmo J48 y el modelo de regresión lineal presentan bajos valores de los errores cuadráticos medios. Fijándonos en el parámetro estadístico Kappa podemos afirmar que el único modelo apropiado es el modelo RandomForest ya que su valor supera el 0.7. Si a esta afirmación le sumamos que es el modelo con mayor valor de accuracy, 0.836, podemos decir que el modelo más fiable y preciso del estudio es el algoritmo RandomForest. Comentando un poco este parámetro sobre el resto de algoritmo podemos observar que el modelo de regresión linear presenta un bajo nivel de precisión. El resto de los modelos presentan una accuracy muy alta, pero el porcentaje Kappa indica que sus resultados no son tan fiables de lo que parecen.

Por ello, se llega a la conclusión de que el mejor modelo para representar la lógica que permite al sistema clasificar el nivel de dolor sea el bosque aleatorio.

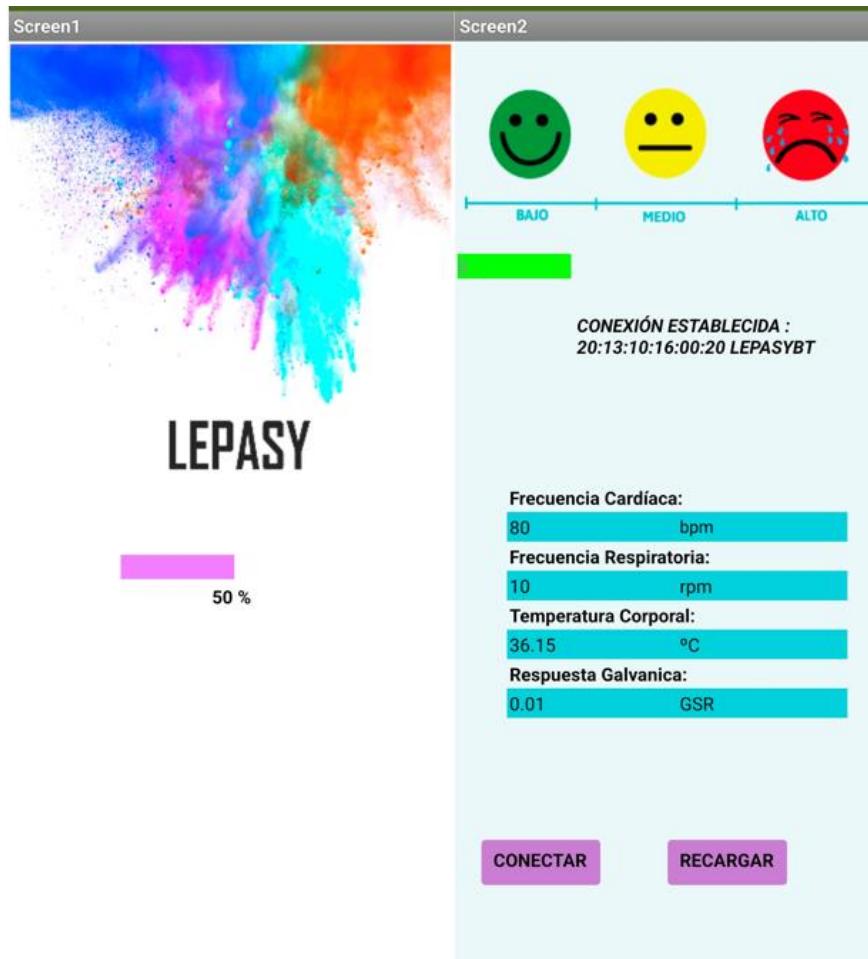
Esta lógica será implementada para ser representada en el sistema a través de una aplicación cuyo diseño y componentes es el siguiente:

Pantalla 1		
Etiqueta	Componente	Descripción
Lepasy	Etiqueta	Indica el nombre de la aplicación
Barra	Etiqueta + Barra	Muestra la carga de página hasta que se pinta al completo
0%	Etiqueta	Muestra la carga de la página desde el 0% al 99%
Pantalla 2		
Etiqueta	Componente	Descripción
Bajo, Medio, Alto	Imagen	Muestra de forma gráfica los distintos niveles de dolor
Barra	Barra	Indica la clasificación del nivel de color
Datos en progreso	Etiqueta	Muestra que la aplicación se encuentra a la espera de recibir datos. Cuando se conecta el módulo bluetooth a la aplicación, aparece la descripción del módulo conectado.
Frecuencia Cardíaca	Etiqueta	Muestra el valor que de la frecuencia cardíaca que se recibe (enviado desde el módulo bluetooth de Arduino)
Frecuencia Respiratoria	Etiqueta	Muestra el valor que de la frecuencia respiratoria que se recibe (enviado desde el módulo bluetooth de Arduino)
Temperatura Corporal	Etiqueta	Muestra el valor que de la temperatura corporal que se recibe (enviado desde el módulo bluetooth de Arduino)
Respuesta Galvánica	Etiqueta	Muestra el valor que de la respuesta galvánica que se recibe (enviado desde el módulo bluetooth de Arduino)
Conectar	Botón	Muestra el listado de redes bluetooth cercanas disponibles y permite seleccionar la conexión con una de ellas
Recargar	Botón	Muestra una recarga de la página.

Cuadro 19: Botones y etiquetas de lepasy.

El resultado fue el siguiente:

Pantalla 1:



Pantalla 2:



Figura 38: Screen1 y Screen2 de la aplicación Lepasy en funcionamiento.

Nota. Figura realizada por elaboración propia.

2.3. PRUEBAS DEL SISTEMA

En este apartado se mostrará el plan de pruebas definido para verificar el funcionamiento del sistema desarrollado al completo. Para ello, definiremos una metodología de ciclos cuyo sumatorio de como resultado el plan de pruebas total del sistema que se obtiene al integrar todos y cada uno de dichos ciclos. Para definir dicha estructura, partimos de la base de que cada funcionalidad del sistema está compuesta por una serie de circuitos que definen el comportamiento del sistema. A su vez, para comenzar cada ciclo, puede ser necesario algún tipo de prerequisito general que indique cuándo, cómo y porque debe comenzar el ciclo en función de las necesidades del sistema. Definir este ítem es opcional, pues algunas veces no será necesario definir una acción que impida o permita comenzar el ciclo.

De forma más completa, se presenta la estructura que debe llevarse a cabo para la definición del plan de pruebas del sistema:

- Prerrequisitos generales (PG): pasos necesarios para ejecutar el ciclo, de manera ordenada.
 - Ciclos: pasos necesarios para ejecutar la prueba. Cada ciclo estará compuesto por uno o varios subciclos que describirán los circuitos que debe ir pasando el usuario que realice las pruebas del sistema. Este apartado corresponderá con la definición de la prueba de usabilidad general propuesto para este sistema. Para ello, se utilizarán varias tablas compuestas por dos columnas, una que en la que se definan los criterios de validación o aceptación y otra en la que se defina el resultado esperado que confirma la realización del ciclo, por tanto, que ha pasado la prueba.

A continuación, se muestra el circuito de pruebas del bloque inteligente del sistema, siguiendo la estructura anterior:

- PG_O1: Debe haber un conjunto de entrenamiento disponible para el estudio de IA, así como una copia de este.
- PG_02: Todas las librerías deben estar importadas en el entorno de desarrollo correspondiente (Google Colab / Weka).
- PG_03: La lógica que implementa el modelo definido en el sistema debe integrarse en el entorno de Arduino IDE y no en la aplicación.
 - Ciclo 1: Carga y preprocesamiento de los datos.

Criterios de validación / aceptación	Resultado esperado
Abrir entorno Google Colab/Weka -> Cargar el conjunto de datos: <i>lepasyDataSet.csv</i> .	Los datos se cargan correctamente.
Realizar el preprocesamiento de los datos.	El conjunto de datos es más fiable que el original.

Cuadro 20: Carga y procesamiento de los datos.

- Ciclo 2: Estudio de los modelos.

Criterios de validación / aceptación	Resultado esperado
Abrir entorno Google Colab/Weka -> Cargar el conjunto de datos: <i>lepasyDataSet.csv</i> .	Los datos se cargan correctamente.
Definir los modelos de deep learning y machine learning correctamente en su respectivo entorno de programación.	Los resultados representan métricas que permiten comparar los diferentes modelos del estudio.
Escoger un modelo.	El modelo es el más fiable para implementar una lógica acorde con los objetivos del proyecto.

Cuadro 21: Estudio de los modelos.

BLOQUE 3: PLANIFICACIÓN DEL PROYECTO.

3.1- PLANIFICACIÓN TEMPORAL

Para obtener de manera sencilla la información sobre la planificación temporal, estimada (inicial) y real (final), se definió un calendario laboral que simulara las horas disponibles en las que se realizaría el proyecto en función de la disponibilidad de los participantes del proyecto. En el propio calendario, también se definieron las fechas festivas siguiendo el calendario del curso escolar.

Horario	Fechas	Número de días	Horas/día
Jornada normal (17:00-20:00)	12/10/2022 15/02/2023	126 días	3 horas/día
Jornada reducida (12:00-14:00)	16/02/2023 20/04/2023	63 días	2 horas/día
Jornada completa (12:00-14:00) (17:00-21:00)	03/05/2023 01/07/2023	59 días	6 horas/días

Cuadro 22: Horario del calendario del proyecto.

3.1.1- PLANIFICACIÓN TEMPORAL INICIAL

A continuación, aparece el diagrama de Gantt estimado para la consecución del proyecto.

La estimación de la duración de las tareas se calculó utilizando la ecuación de la estimación PERT:

$$EPERT = \frac{TO + (4 \cdot TMP) + TP}{6}, \text{ donde:}$$

TO = optimista.

TMP = tiempo más probable.

TP = tiempo pesimista.

El resultado fue que la duración del proyecto corresponde a unas 402 horas.

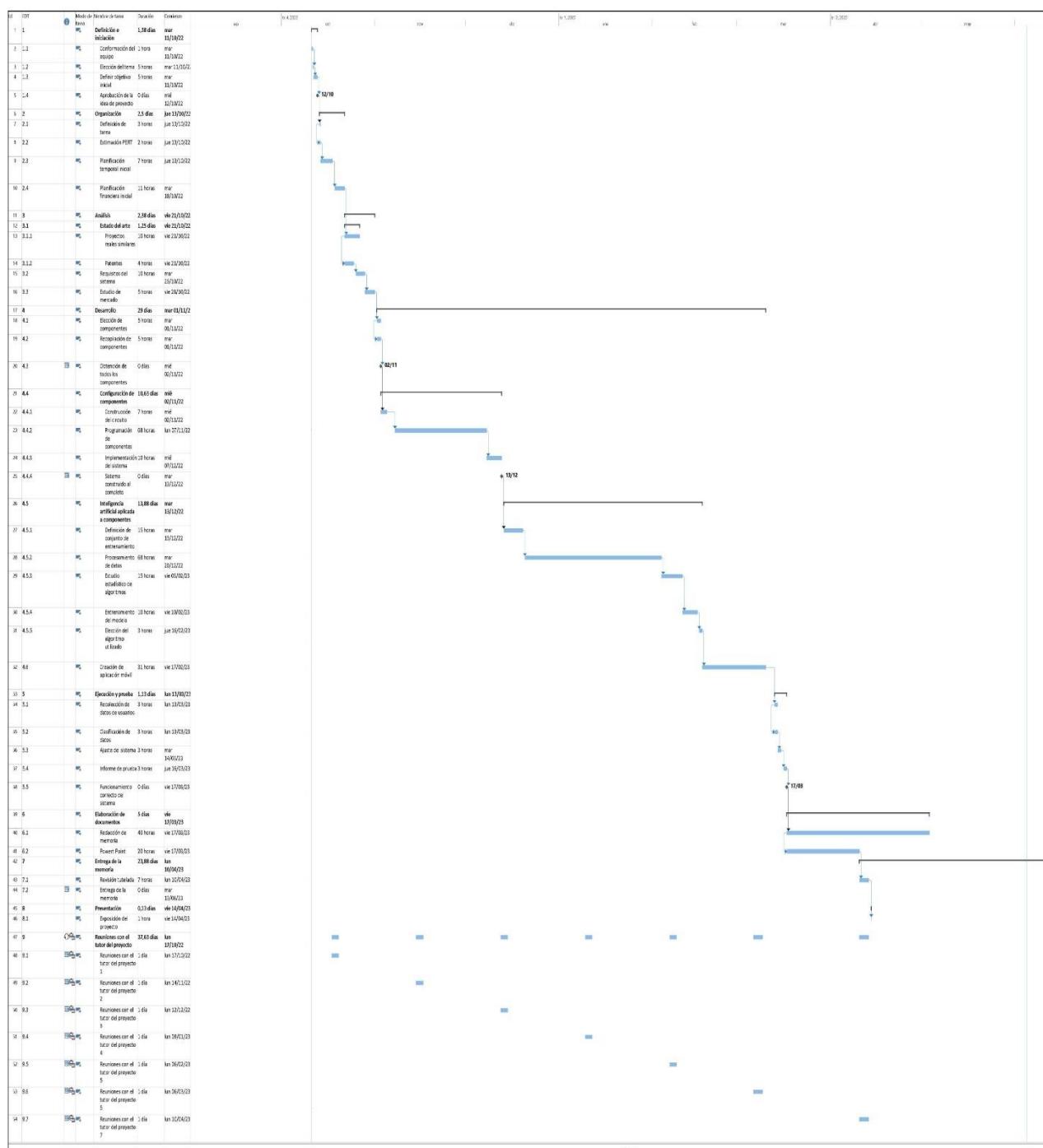


Figura 39: Diagrama de Gantt. Planificación temporal inicial.

Nota. Figura realizada por elaboración propia.

3.1. 2. PLANIFICACIÓN TEMPORAL FINAL

A continuación, aparece el diagrama de Gantt estimado para la consecución del proyecto.

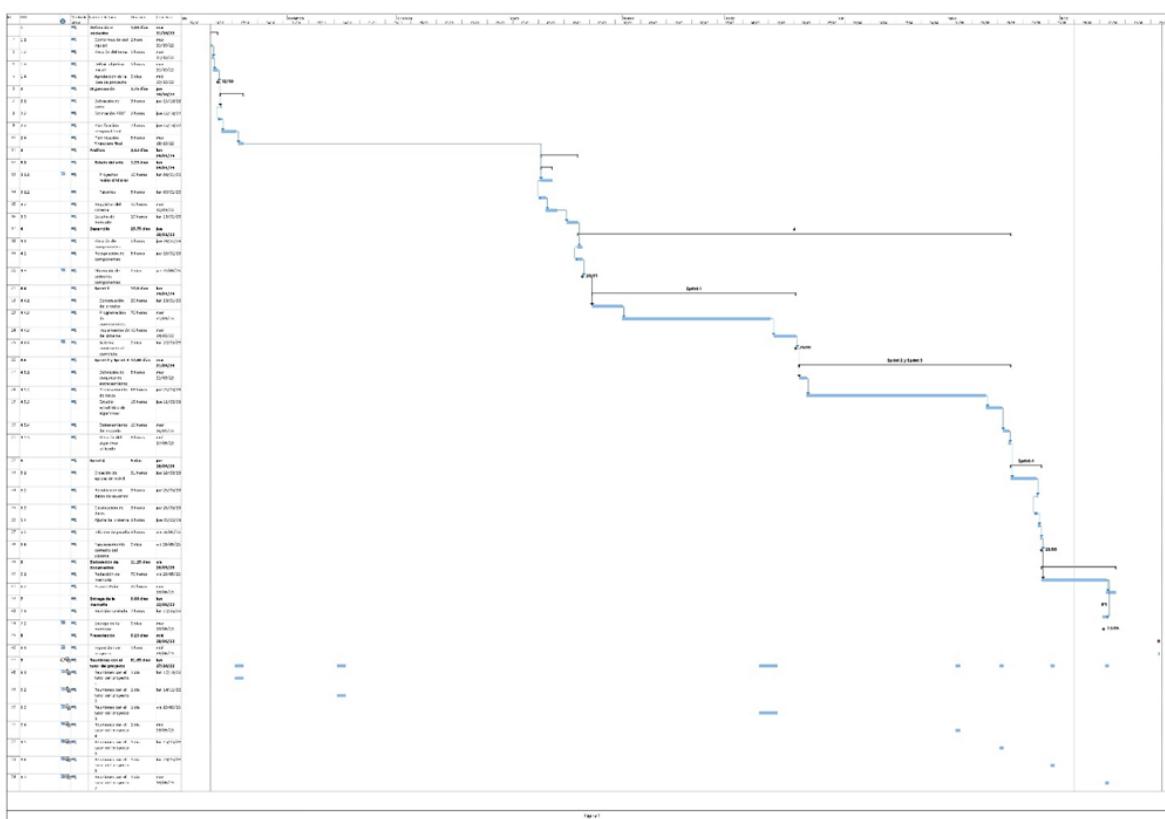


Figura 40: Diagrama de Gantt. Planificación temporal final.

Nota. Figura realizada por elaboración propia.

El resultado fue que la duración del proyecto corresponde a unas 442 horas.

3.2- PLANIFICACIÓN FINANCIERA

La estructura financiera del proyecto se compone de dos grupos de análisis financiero, el primero tiene que ver con los recursos humanos, y el segundo, recursos materiales. Estos datos, sumados a la descripción del gasto total dan como resultado una planificación financiera adecuada a las características del proyecto.

3.2.1- PLANIFICACIÓN FINANCIERA INICIAL

En este caso, los resultados que veremos a continuación representan una estimación financiera de los costes que supondría el sistema de manera aproximada.

Teniendo en cuenta que el presente proyecto se ha realizado en grupo, podemos calcular los costes para un proyecto real con el mismo número de personas que han realizado este trabajo. También, es importante mencionar que la estimación de horas de este proyecto, 402 horas, presentan un indicador de que dos es el número de personas necesarias para completar el sistema.

3.2.1.1- Recursos materiales

En este punto, aparecen los elementos que inicialmente se han decidido tener en cuenta para la estimación inicial del proyecto:

3.2.1.1.1- Hardware

MATERIAL	IMAGEN	PRECIO
Sensor de pulso		5€
Sensor de frecuencia respiratoria		8€
Arduino		30€
Felpa y cables		9€ + 6€
LED RGB y resistencias (330 Ohmios)		9€
TOTAL	—	67€

Cuadro 23: Precios de los recursos materiales empleados en la planificación financiera inicial.

3.2.1.1.2- Software

Los entornos de desarrollo software incluyen un precio base de adquisición de licencias mucho antes de la utilización de estos. Sin embargo, se estima que todos los entornos de programación forman parte de software libre y que no aplica ningún gasto de licencia o utilización. Simplemente se tendría en cuenta el precio de las licencias de Windows (259 EUR) y OpenOffice (51 EUR).

Entornos de desarrollo	Precio
Arduino IDE	0€ (incluido en la compra de la placa Arduino UNO)
Visual Studio	0€
Google Colab	0€
App Inventor	0€
Licencia Windows 10	259€
Licencia OpenOffice	51€
TOTAL	310€

Cuadro 24: Precio de licencia software.

3.2.1.2- Recursos humanos

Basándonos en XVII Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública (TIC) publicado en el BOE el 22 de febrero del 2018 podemos analizar los salarios fijos anuales en función de los grupos profesionales que se definen [33].

Considerando que las asignaturas estudiadas a lo largo de la carrera nos han permitido los conocimientos necesarios para afrontar el proyecto y la poca experiencia que los integrantes del equipo del proyecto tienen, se ha decidido clasificar a los dos miembros del equipo dentro del área 3 (consultoría, desarrollo y sistemas) nivel 3 del grupo C. Esto quiere decir que los recursos humanos de este proyecto pertenecen a un grupo profesional de tipo informático-técnico, responsables de la programación y supervisión de las tareas definidas. Además, es necesario el rol de jefe de proyecto para un trabajo de semejante magnitud que será ocupado por el tutor del proyecto.

$$\text{Salario total} = \text{Salario base} + \text{Plus convenio}$$

Como observamos en la ecuación anterior, salario total está formado por la suma del salario base más el plus convenio. Realizando una observación al convenio colectivo, el

salario total de las personas que se incluyen en dichos grupos profesionales es el siguiente:

Grupo profesional	Salario base	Plus convenio	Salario total (EUR/año)	Salario total (EUR/hora)
Jefe de Proyecto	25.035,54	1.754,77	26.790,31	14,88
Desarrollador de aplicaciones	18.222,75	1.277,25	19.500,00	10,83

Cuadro 25: Salario de los recursos humanos del proyecto.

La ecuación para calcular el salario base por hora consiste en multiplicar el salario total de un año por el número de horas laborales que el mismo presenta.

A estos costes se les ha de añadir otra serie de primas que se calculan en función de los porcentajes determinados por la seguridad social y que corren a cargo de la empresa que contrata el equipo. La base de cotización se calcula como la suma de todos los conceptos salariales de la nómina del trabajador.

La seguridad social a cargo de la empresa cubre los siguientes riesgos:

- Contingencias comunes (CC): Cubre las derivadas de enfermedad común, accidente no laboral y maternidad. Este coste se calcula aplicando un 23,6% a la base de cotización de cada trabajador.
- Desempleo (DE): Cubre el riesgo de perder el trabajo para los que se encuentran en la situación de no ejercer la profesión por cualquier tipo de causa justificada. La cotización es distinta en función del tipo de contrato. En este proceso, el porcentaje correspondería en un 7,70% ya que el tipo de contratos del personal para la elaboración de un sistema de esta índole consiste en un contrato temporal que finalizaría una vez se concluya con el cierre del proyecto.
- Fondo de Garantía Salarial (FOGASA): este tipo de riesgo cubre las faltas de pago a los trabajadores derivadas de la mala praxis de los contratadores. Se calcula aplicando un 0,2 % sobre la base de cotización.
- Riesgo laboral (RL): ofrece las coberturas del riesgo de los accidentes de trabajo y enfermedades profesionales. Estos aspectos se dividen en la cotización por la

incapacidad temporal y la de invalidez, muerte y supervivencia, cuya suma proporciona el porcentaje total cubierto:

- 0,65% por el riesgo de incapacidad temporal derivado del trabajo
- 1% riesgo de incapacidad permanente, muerte y supervivencia derivados del trabajo.
- Total: $0,65\% + 1\% = 1,65\%$
- Formación profesional (FP): hace referencia al porcentaje destinado para los cursos de formación y reciclaje personal de los trabajadores. El porcentaje se calcula aplicando el 0,6% sobre la base de cotización.

Analizando cada porcentaje de cada riesgo, el porcentaje total sobre la base de cotización de cada trabajador de la seguridad social a cargo de la empresa es el siguiente:

$$\% \text{Base Cotización} = \% \text{CC} + \% \text{DE} + \% \text{FOGASA} + \% \text{RL} + \% \text{FP}$$

$$\% \text{Base Cotización} = 23,6 + 7,7 + 0,2 + 1,65 + 0,6 = 33,75\%$$

Multiplicando el salario total en EUR/hora por el porcentaje de la base de cotización y el número de unidades, obtenemos que el coste por hora que genera cada uno de los trabajadores del proyecto es el siguiente:

Grupo profesional	Salario total (EUR/hora)	Base de cotización (%)	Total (EUR/hora)
Jefe de Proyecto	14,88	0,3375	19,90
Desarrollador de aplicaciones 1	10,83	0,3375	14,48
Desarrollador de aplicaciones 2	10,83	0,3375	14,48

Cuadro 26: Salario de recursos humanos con la cotización de la seguridad social.

El cálculo de los costes de personal se calcula por cada tarea, duración de la misma, rol de la persona que la realiza. Se obtiene el coste asociado a realizar dicha tarea y para obtener el total se hace la suma del coste de todas las tareas.

Nombre	Rol RH	Salario (EUR/hora)	Estimación PERT (horas)	Coste (EUR)
Definición e iniciación				
Conformación del equipo	Jefe de Proyecto	19,90	1	19.9
	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14,48		14.48
Elección del tema	Jefe de Proyecto	19,90	5	99.5
	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14,48		72.4
Definir objetivo inicial	Desarrollador de aplicaciones 1		5	
	Desarrollador de aplicaciones 2	14,48		72.4
Aprobación de la idea de proyecto				
Total			11	278.68
Organización				
Definición de tareas	Desarrollador de aplicaciones 1		3	43.44
	Desarrollador de aplicaciones 2	14,48		
Estimación PERT	Desarrollador de aplicaciones 1		2	28.96
	Desarrollador de aplicaciones 2	14,48		
Planificación temporal inicial	Desarrollador de aplicaciones 1		7	101.36
	Desarrollador de aplicaciones 2	14,48		
Planificación financiera inicial	Desarrollador de aplicaciones 1		11	159.28
	Desarrollador de aplicaciones 2	14,48		
Total			23	333.04
Estado del arte				
Proyectos reales similares	Desarrollador de aplicaciones 1		10	144.80
	Desarrollador de aplicaciones 2	14,48		
Patentes	Desarrollador de aplicaciones 1	14,48	4	57.92
	Desarrollador de aplicaciones 2			
Requisitos del sistema	Desarrollador de aplicaciones 1	14,48	10	144.8
	Desarrollador de aplicaciones 2			
Estudio de mercado	Desarrollador de aplicaciones 1	14,48	5	72.4
	Desarrollador de aplicaciones 2			
Total			29	622.64

Elección de componentes	Desarrollador de aplicaciones	14,48	5	72.4
	1			
Recopilación de componentes	Desarrollador de aplicaciones	14,48	5	72.4
	1			
Obtención de todos los componentes			0	0
Configuración de componentes	Desarrollador de aplicaciones	14,48	85	1230
	1			
Construcción del circuito	Desarrollador de aplicaciones	14,48	7	101.36
	1			
Programación de componentes	Desarrollador de aplicaciones	14,48	68	984.64
	1			
Implementación del sistema	Desarrollador de aplicaciones	14,48	10	144.8
	1			
Sistema construido al completo	Desarrollador de aplicaciones		0	0
	1			
Inteligencia artificial aplicada a componentes	Desarrollador de aplicaciones	14,48	111	1607.28
	1			
Definición de conjunto de entrenamiento	Desarrollador de aplicaciones	14,48	15	217.2
	1			
Procesamiento de datos	Desarrollador de aplicaciones	14,48	68	984.64
	1			
Estudio estadístico de algoritmos	Desarrollador de aplicaciones	14,48	15	217.2
	1			
Entrenamiento del modelo	Desarrollador de aplicaciones	14,48	10	144.8
	1			
Elección del algoritmo utilizado	Desarrollador de aplicaciones	14,48	3	43.44
	1			
Creación de aplicación móvil	Desarrollador de aplicaciones	14,48	31	448.88
	1			
Total			237	5067.2

Ejecución y prueba				
Recolección de datos de usuarios	Desarrollador de aplicaciones1	14,48	3	43.44
	Desarrollador de aplicaciones2			
Clasificación de datos	Desarrollador de aplicaciones1	14,48	3	43.44
	Desarrollador de aplicaciones2			
Ajuste del sistema	Desarrollador de aplicaciones1	14,48	3	43.44
	Desarrollador de aplicaciones2			
Informe de prueba	Desarrollador de aplicaciones1	14,48	3	43.44
	Desarrollador de aplicaciones2			
Funcionamiento correcto del sistema			0	0
Total		12		173.76
Elaboración de documentos				
Redacción de memoria	Desarrollador de aplicaciones1	14,48	40	579.2
	Desarrollador de aplicaciones2			
Power Point	Desarrollador de aplicaciones	14,48	20	289.6
	Desarrollador de aplicaciones2			
Total		60		868.8
Entrega de la memoria				
Revisión tutelada	Jefe de Proyecto	19,90	7	139.3
Entrega de la memoria	Jefe de Proyecto	19,90		0
	Desarrollador de aplicaciones1		0	0
	Desarrollador de aplicaciones2	14,48		
Total		7		139.3
Presentación				
Exposición del proyecto	Jefe de Proyecto	19,90		19,90
	Desarrollador de aplicaciones1		1	
	Desarrollador de aplicaciones2	14,48		14,48
Total		1		34.38

Reuniones con el tutor del proyecto				
Reuniones con el tutor del proyecto 1	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 2	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 3	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 4	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 5	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 6	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 7	Jefe de Proyecto	19,90	1	19.90
	Desarrollador de aplicaciones1			14.48
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 8	Jefe de Proyecto	19,90	5	99.5
	Desarrollador de aplicaciones1			72.4
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 9	Jefe de Proyecto	19,90	5	99.5
	Desarrollador de aplicaciones1			72.4
	Desarrollador de aplicaciones2	14,48		
Reuniones con el tutor del proyecto 10	Jefe de Proyecto	19,90	5	99.5
	Desarrollador de aplicaciones1			72.4
	Desarrollador de aplicaciones2	14,48		
Total		22		756.36
Total(HORAS)			402	8274.16

Cuadro 27: Cálculo del gasto de los recursos humanos en función de las horas estimadas.

Puede parecer que el coste por horas de un miembro del equipo sea bastante bajo. Sin embargo, este coste está estipulado por el gobierno español por lo que la variación se

lleva a cabo en función del plus convenio que cada empresa quiera aplicar, afectando al salario total de manera directamente proporcional. Por ello, podemos afirmar que el aumento o la disminución de los costes en la planificación financiera final, va a depender del número de componentes que aumenten los costes de los recursos materiales, así como del rendimiento de los miembros del equipo cuya eficiencia se verá reflejada en un mayor o menor número de horas dedicadas al proyecto, disminuyendo así los costes de los recursos humanos.

Por supuesto pensamos que, en el futuro, cuando el producto sea impulsado al mercado, tendrá numerosas empresas interesadas en el mismo y esto nos permitirá abaratar los costes totales para que el producto sea más barato para el cliente y, podrán incrementarán las ganancias de los trabajadores. Todo ello suponiendo su éxito, del que estamos bastante seguros debido a la gran utilidad de este.

3.2.2.3- Gasto total inicial

Como podemos observar, en este apartado aparece una estimación inicial de los recursos materiales y humanos necesarios para la realización del proyecto.

En esta parte hemos obviado el sensor de temperatura corporal y de sudoración y se han tenido en cuenta el precio de las licencias de los productos software que vamos a utilizar.

Recursos materiales	Recursos humanos	Gasto total
377€	8274,16€	8651.16€

Cuadro 28: Gasto total estimado del proyecto.

3.2.2- PLANIFICACIÓN FINANCIERA FINAL

En este caso, la planificación se realiza con los datos proporcionados al realizar la totalidad del proyecto, sin ningún tipo de estimación.

3.2.2.1-Recursos materiales

En esta categoría se incluye la compra de todos los materiales necesario para elaborar el sistema. Al final del proyecto, el coste total de estos recursos se ha visto incrementados hasta llegar a la cifra de 116€ debido a que se han ido incluyendo nuevos componentes a lo largo del proyecto con los que no habíamos no contado en la planificación financiera inicial. En la tabla que se muestra a continuación, se recogen los

precios de cada material, para exponer de forma más concisa el coste de cada elemento que compone el sistema realizado.

3.2.2.1.1-Hardware

Material	Imagen	Precio
Sensor de pulso		5€
Sensor de frecuencia respiratoria		8€
Sensor de temperatura corporal		19€
Sensor de sudoración		19€
Arduino		30€
Felpa y cables		9€ + 6€
Led RGB y resistencia (330 Ohmios)		9€
Módulo Bluetooth		11€

TOTAL	—	116€
--------------	---	------

Cuadro 29: Precios de los recursos materiales empleados en la planificación financiera final.

Todos los precios de los materiales están cotejados con diferentes tiendas oficiales, como por ejemplo en Amazon. Como podemos observar, al ser el primer sistema que elaboramos los costos se mantienen más o menos constantes pues se han añadido varias componentes (bluetooth, sensor GSR y sensor de temperatura corporal) de bajo coste. Pero es cierto que, si se llegara a comercializar y tuviéramos que elaborar un mayor número del producto, podríamos abaratar costes ya que normalmente al comprar productos en masa, al por mayor, baja un poco el precio por cada unidad de material. Lo cual nos permitiría comprar componentes de mayor calidad y mejorar el diseño del producto lo máximo posible.

3.2.2.1.2-Software

Los entornos de desarrollo software suelen incluir un precio base de adquisición de licencias mucho antes de la utilización de los mismos. Sin embargo, se estima que todos los entornos forman parte del material proporcionado por la empresa, ningún gasto de licencia o utilización ha sido considerado:

Entornos de desarrollo	Precio
Arduino IDE	0€ (incluido en la compra de la placa Arduino UNO)
Visual Studio	0€
Google Colab	0€
App Inventor	0€
Licencia Windows 10	0€
Licencia OpenOffice	0€
TOTAL	0€

Cuadro 30: Precio de las licencias reales.

Los entornos de desarrollo software suelen incluir un precio base de adquisición de licencias mucho antes de la utilización de los mismos. De hecho, todos los entornos que hemos utilizado forman parte de software libre cuyo uso no ha mostrado ningún gasto de licencia o derivados.

3.2.2.2-Recursos humanos

Para recordar, el coste estimado en la planificación financiera inicial por hora que genera cada uno de los trabajadores del proyecto era el siguiente:

Grupo profesional	Salario total (EUR/hora)	Base de cotización (%)	Total (EUR/hora)
Jefe de Proyecto	14,88	0,3375	19,90
Desarrollador de aplicaciones 1	10,83	0,3375	14,48
Desarrollador de aplicaciones 2	10,83	0,3375	14,48

Cuadro 31: Precio de los recursos humanos reales con la cotización en la seguridad social.

A continuación, se realiza el cálculo de los costes de personal se calcula por cada tarea final. Esto quiere decir que los valores que se obtengan presentan resultados reales ya que están basados en las tareas definidas en la planificación temporal final, obteniéndose así el coste asociado a realizar la suma de todas las tareas.

Nombre	Rol RH	Salario (EUR/hora)	Horas reales	Total
Definición e iniciación				
	Jefe de proyecto	19.9		19.9
Conformación del equipo	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	1	14.48
	Jefe de proyecto	19.9		99.5
Elección del tema	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	5	72.4
Definir objetivo inicial	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	5	72.4
Aprobación de la idea de proyecto				0
Total			11	278.68
Organización				
	Desarrollador de aplicaciones 1			
Definición de tareas	Desarrollador de aplicaciones 2	14.48	3	43.44
	Desarrollador de aplicaciones 1			
Estimación PERT	Desarrollador de aplicaciones 2	14.48	2	28.96
	Desarrollador de aplicaciones 1			
Planificación temporal inicial	Desarrollador de aplicaciones 2	14.48	7	101.36
	Desarrollador de aplicaciones 1			
Planificación financiera inicial	Desarrollador de aplicaciones 2	14.48	5	72.4
Total			17	246.16
Estado del arte				
	Desarrollador de aplicaciones 1			
Proyectos reales similares	Desarrollador de aplicaciones 2	14.48	10	144.8
	Desarrollador de aplicaciones 1			
Patentes	Desarrollador de aplicaciones 2	14.48	5	72.4
	Desarrollador de aplicaciones 1			
Requisitos del sistema	Desarrollador de aplicaciones 2	14.48	10	144.8
	Desarrollador de aplicaciones 1			
Estudio de mercado	Desarrollador de aplicaciones 2	14.48	10	144.8
Total			35	506.8
Desarrollo				
	Desarrollador de aplicaciones 1			
Elección de componentes	Desarrollador de aplicaciones 2	14.48	5	72.4
	Desarrollador de aplicaciones 1			
Recopilación de componentes	Desarrollador de aplicaciones 2	14.48	5	72.4
Obtención de todos los componentes				0 0
Total			10	144.8
Sprint 1				
	Desarrollador de aplicaciones 1			
Construcción del circuito	Desarrollador de aplicaciones 2	14.48	20	289.6
	Desarrollador de aplicaciones 1			
Programación de componentes	Desarrollador de aplicaciones 2	14.48	70	1013.6
	Desarrollador de aplicaciones 1			
Implementación del sistema	Desarrollador de aplicaciones 2	14.48	10	144.8
Sistema construido al completo				0 0
Total			100	1448

Sprint 2 y Sprint 3				
Definición de conjunto de entrenamiento	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	5	72.4
Procesamiento de datos	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	68	984.64
Estudio estadístico de algoritmos	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	15	217.2
Entrenamiento del modelo	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	10	144.8
Elección del algoritmo utilizado	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	3	43.44
Total			101	1462.48
Sprint 4				
Creación de aplicación móvil	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	31	448.88
Recolección de datos de usuarios	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	3	43.44
Clasificación de datos	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	3	43.44
Ajuste del sistema	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	3	43.44
Informe de prueba	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	3	43.44
Funcionamiento correcto del sistema	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	36	521.28
Total			79	1143.92
Elaboración de documentos				
Redacción de memoria	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	70	1013.6
Powert Point	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	20	289.6
Total			90	1303.2
Entrega de la memoria				
Revisión tutelada	Jefe de proyecto	19.9	7	139.3
	Jefe de proyecto			
	Desarrollador de aplicaciones 1			
Entrega de la memoria	Desarrollador de aplicaciones 2		0	
Total			7	139.3
Presentación				
Exposición del proyecto	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	1	14.48
Total			1	14.48
Reuniones con el tutor del proyecto				
Reuniones con el tutor del proyecto 1	Jefe de proyecto	19.9		19.9
	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	1	14.48
Reuniones con el tutor del proyecto 2	Jefe de proyecto	19.9		19.9
	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	1	14.48
Reuniones con el tutor del proyecto 3	Jefe de proyecto	19.9		19.9
	Desarrollador de aplicaciones 1			
	Desarrollador de aplicaciones 2	14.48	1	14.48

Jefe de proyecto	19.9		19.9
Reuniones con el tutor del proyecto 4	Desarrollador de aplicaciones 1		
	Desarrollador de aplicaciones 2	14.48	1 14.48
Jefe de proyecto		19.9	19.9
Reuniones con el tutor del proyecto 5	Desarrollador de aplicaciones 1		
	Desarrollador de aplicaciones 2	14.48	1 14.48
Jefe de proyecto		19.9	19.9
Reuniones con el tutor del proyecto 6	Desarrollador de aplicaciones 1		
	Desarrollador de aplicaciones 2	14.48	1 14.48
Jefe de proyecto		19.9	19.9
Reuniones con el tutor del proyecto 7	Desarrollador de aplicaciones 1		
	Desarrollador de aplicaciones 2	14.48	1 14.48
Total			7 240.66
Total(horas)			448 6928.48

Cuadro 32: Calculo del gasto real de los recursos humanos.

Como podemos observar, el número de componentes han aumentado, pues se ha añadido los módulos bluetooth, sensor de respuesta galvánica y sensor de temperatura corporal, dando lugar a un aumento de los costes relacionados con los recursos materiales. Sin embargo, el número de horas totales del proyecto han disminuido debido a la eficacia del trabajo del equipo que compone el proyecto, disminuyendo así los costes de los recursos humanos.

Por supuesto pensamos que, en el futuro, cuando el producto sea impulsado al mercado, tendrá numerosas empresas interesadas en el mismo y esto nos permitirá abaratar los costes totales para que el producto sea más barato para el cliente puesto que los componentes se irán obteniendo en masa, lo que permite que su precio se reduzca y se incrementen las ganancias del sistema. Todo ello suponiendo su éxito, del que estamos bastante seguros debido a la gran utilidad de este y novedad dentro del mercado de productos biomédicos.

3.2.2.3-Gasto total final

Como podemos observar, tanto los recursos materiales como los humanos se han disminuido con respecto al estimado en el plan inicial. Sin embargo, esta disminución de gasto no es muy significativa pues no hay mucha diferencia económica entre ellas.

Por una parte, el incluir el sensor de temperatura corporal y de sudoración ha hecho que los valores procesados sean mucho más fiables y que a la hora de procesarlos el resultado final sea lo más cercano posible a la realidad de los pacientes, que al fin y al cabo era nuestro objetivo principal del proyecto. De nada sirve todo este proyecto si ahora los parámetros no indican con certeza los resultados y se amoldan a la vida real ya que no cumpliría el objetivo de ser una herramienta de soporte y ayuda para médicos,

familiares, amigos e incluso los propios pacientes. Esta adición de componentes ha dado lugar a que el gasto en recursos materiales aumente.

Si pensamos este proyecto como un sistema a gran escala de demanda, podemos afirmar que merece la pena reducir el precio total, ya que calidad-precio del producto se correspondería de manera más aproximada a las necesidades del cliente.

Recursos materiales	Recursos humanos	Gasto total
116€	6928,48€	7044.48€

Cuadro 33: Gasto total del proyecto.

El gasto final considerado es el elaborado evaluando mi trabajo de forma exclusiva, sin incluir el coste que supondría el trabajo de mi compañero, que sería el mismo ya que hemos realizado las mismas horas de trabajo.

El gasto final considerado es el elaborado evaluando mi trabajo de forma exclusiva, sin incluir el coste que supondría el trabajo de mi compañero, que sería el mismo ya que hemos realizado las mismas horas de trabajo.

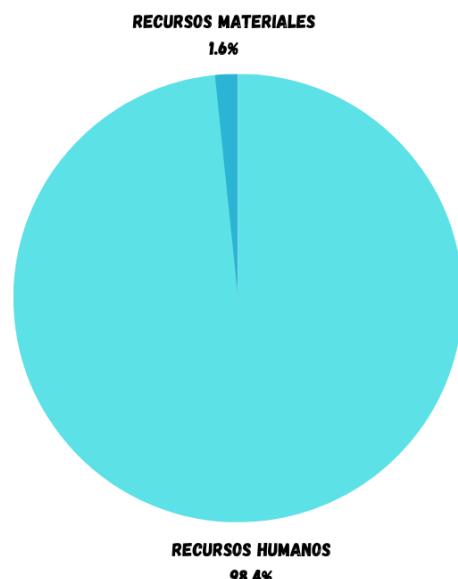


Figura 41: Comparación del gasto total que supone cada tipo de recurso.

Como podemos observar en la gráfica los recursos humanos componen casi la totalidad del gasto. A futuro, nos gustaría dar un giro a esta tabla e intentar que los recursos materiales sean de la mejor calidad posible para los pacientes.

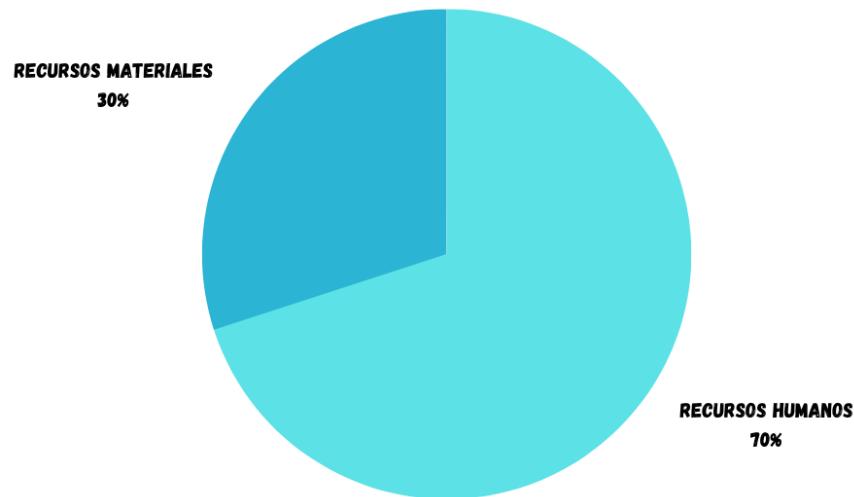


Figura 42:Meta de comparación de los gastos empleados en cada recurso a futuro.

3.3- ESTUDIO DE MERCADO

Todo proyecto necesita conocer a sus posibles clientes potenciales sobre todo cuando se trata de uno con unas características como las nuestras. Además, la falta de experiencia de los miembros del equipo en la elaboración de un proyecto de esta índole nos obliga a realizar un estudio que nos haga conocer con exactitud la posición que debe tomar en el mercado el sistema propuesto y los requisitos que los clientes desean cumplir. Por ello, es de vital importancia realizar un estudio de mercado que nos permita adaptar el sistema al máximo a las necesidades de mercado y así como la adaptación a los posibles clientes. Cabe resaltar que, si los futuros clientes desean mejorar el sistema, se podría valorar ya que, al fin y al cabo, en este proyecto, se plantea más bien un prototipo base, que está en su propio auge ya que puede mejorar mucho más sus características y elevar su valor en el mercado.

3.3.1- CLIENTES POTENCIALES

En primer lugar, en este estudio nos gustaría identificar a los clientes potenciales del proyecto. Para ello, hemos hecho una gran búsqueda, tenido en cuenta nuestros objetivos, sobre qué tipo de personas podrían estar interesadas en tener un sistema de estas características en su día a día principal. La respuesta fue sencilla los clientes incluirá tanto a algunas empresas, como a particulares que necesiten o quieran monitorizar el dolor tanto en el ámbito hospitalario correspondiente como en su casa sin necesidad de ir al hospital.

Algunas de las empresas que forman parte de este sector de diagnósticos serían:

1. Instituciones educativas: Escuelas, universidades o centros de formación que ofrecen programas de educación en campos relacionados con la bioinstrumentación, la medicina, la biotecnología, la ingeniería biomédica y disciplinas afines. Estas instituciones podrían estar interesadas en utilizar este sistema como una herramienta de aprendizaje práctico para enseñar a los estudiantes sobre la adquisición y procesamiento de señales biomédicas, así como sobre la aplicación de algoritmos de IA en la clasificación del nivel de dolor de manera técnico-práctica.
2. Centros de investigación: Organizaciones de investigación en áreas como la medicina, la salud, la neurociencia o la psicología, que llevan a cabo investigaciones relacionadas con el dolor y buscan herramientas avanzadas para medir y analizar las señales biomédicas asociadas al dolor. Este sistema

podría proporcionarles una solución accesible y de bajo costo para la monitorización y clasificación del dolor en sus estudios.

3. Empresas de tecnología médica: Compañías dedicadas al desarrollo y fabricación de dispositivos y equipos médicos podrían estar interesadas en este sistema como una solución complementaria para la monitorización del dolor en entornos clínicos o de investigación, ofreciendo a los profesionales de la salud una herramienta adicional para evaluar y tratar el dolor en pacientes.
4. Hospitales: para administrar una ayuda a su personal incluyendo a sus, médicos, enfermeros, fisioterapeutas y otros profesionales de la salud que trabajan en entornos clínicos o de rehabilitación, qué podrían encontrar útil este sistema para evaluar el nivel de dolor en sus pacientes. Les permitiría recopilar datos objetivos y cuantificables sobre el dolor, lo que podría respaldar el diagnóstico y la toma de decisiones en el tratamiento.

Es importante destacar que el éxito del sistema de monitorización del nivel de dolor dependerá de varios factores, como la calidad y precisión de las mediciones, la facilidad de uso, la confiabilidad de los resultados y la capacidad de integración con otros sistemas y herramientas existentes en el ámbito médico. Además, se deberá considerar la competencia existente en el mercado y la aceptación de nuevas tecnologías en el sector de la salud.

Un análisis más detallado y específico del mercado objetivo, incluyendo el tamaño del mercado, las tendencias del sector, la competencia y las necesidades de los clientes potenciales, sería necesario para una evaluación más precisa de las oportunidades comerciales y la estrategia de comercialización del sistema.

3.3.2. PLAN DE COMERCIALIZACIONES

En primer lugar, hemos realizado un análisis DAFO (debilidad, amenaza, fortaleza y oportunidad) para conocer la situación del proyecto sabiendo los puntos negativos, descritos como debilidades y amenazas, así como los puntos positivos que serían los de fortalezas y oportunidades. De esta misma manera, también vemos que las debilidades y las fortalezas son algo interno del proyecto que depende totalmente de nosotros, sin embargo, las amenazas y las oportunidades tienen que ver con exterior, por tanto, no depende solo de nosotros mismos, sino que también depende de los

propios clientes y de la actividad del mercado [34].

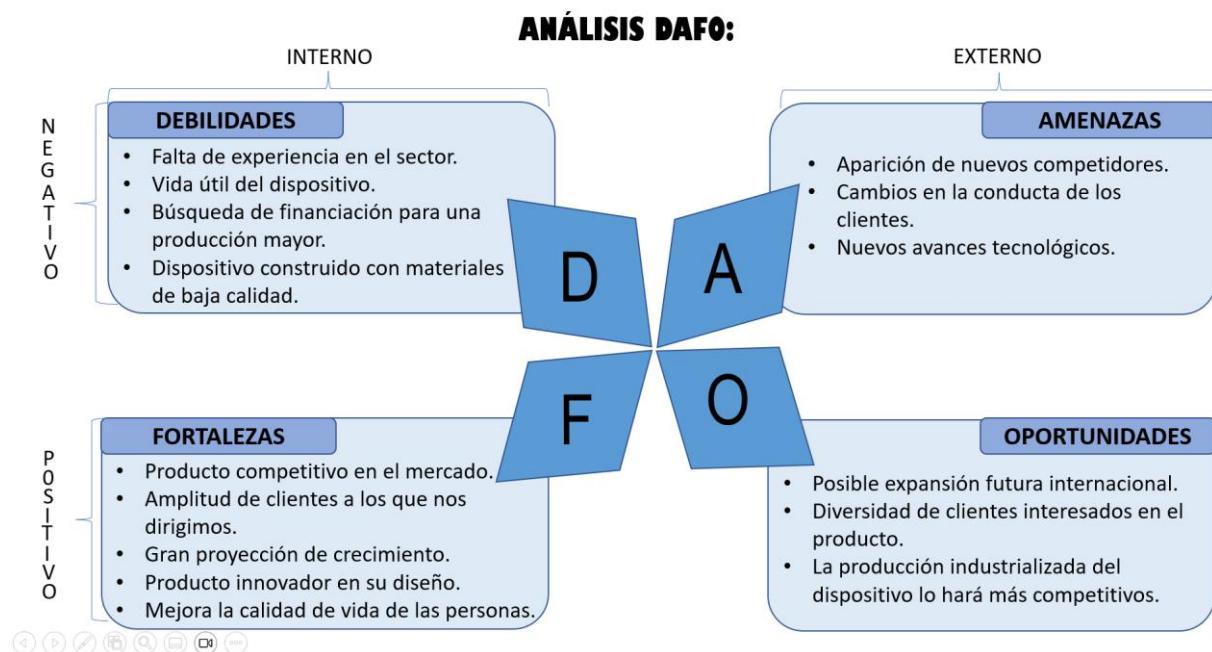


Figura 43: Análisis DAFO del proyecto.

Una vez realizado el análisis DAFO, para desarrollar el plan de comercialización hemos identificado el mercado interesado, que serían los clientes potenciales definidos en el apartado anterior.

Nos centraremos en principio en la venta a particulares, hospitales y a empresas tecnológicas, ya que no tenemos stock suficiente para abastecer a más clientes. Evidentemente, el volumen de compras del hospital y la empresa no será el mismo que el de un particular, debido a este volumen el precio de cada aparato les saldrá más económico ya que nos permite comprar en masa y obtener precios más baratos.

Uno de los objetivos a futuro es obtener el precio más económico del mercado, y de esta manera, que el producto sea accesible para todos los posibles clientes ya sean empresas o particulares y también nos gustaría incrementar la calidad de nuestros sensores, ya que como hemos indicado en el análisis DAFO, teníamos muy poco presupuesto para elaborar el proyecto completo.

Por todos estos motivos nos inquieta bastante conseguir la financiación, ya que, sin ella no podremos cumplir nada de lo dicho anterior, nos sería imposible realizar el proyecto. Hay que tener en cuenta que el precio total que hemos calculado en el plan financiero sería únicamente para el primer lote del producto, ya que para los siguientes habría muchos pasos que nos saltaríamos al estar ya hechos, por tanto, el precio sería muy

inferior, y muy competitivo en el mercado. Prevemos, que el producto será un éxito, con mucha demanda, por tanto, será esencial la industrialización del proceso, ya que nos sería imposible vender a tantos clientes sin tener el montaje adecuado en un tiempo mínimo. Creemos que con la gran proyección que tiene el proyecto alguna empresa se embarcará en ayudarnos con la financiación.

También es muy importante conocer a la competencia, solo hay un producto que se parece al propuesto en el proyecto llamado PMD_200, por tanto, vemos muy importante conseguir un precio competitivo, que haga que nuestro producto se imponga por delante en el mercado.

Nuestro producto al contrario que el PMD_200 no está orientado solamente a hospitales, tenemos también la idea de darnos a conocer en otros sectores, obteniendo de esta manera clientes de distintas disciplinas, como he indicado anteriormente, es importante ya que tenemos más oportunidad que ellos de triunfar, habrá mayor número de clientes que conozcan nuestros dispositivos.

Por otro lado, también nos diferenciamos en las características de nuestro sistema, ya que hemos cambiado el acelerómetro que ellos utilizan, por un sensor de respiración, que según diversos estudios es un parámetro que varía dependiendo del dolor. Otra ventaja es que nosotros utilizamos el sensor de pulso, sin embargo, ellos utilizan fotoplestimografía lo cual es algo menos fiable, ya que cualquier flexión del dedo puede aumentar el tamaño del músculo dando valores inciertos, no funciona bien en pieles oscuras o con tatuajes, y los valores obtenidos pueden depender mucho de la presión que ejerza la cinta en el dedo.

Nosotros hemos apostado por la comodidad del paciente ya que al ser en la cabeza les permite llevar a cabo cualquier actividad con las manos, al contrario que en PMD_200 que al tenerlo en el dedo les impide realizar cualquier tipo de actividad durante la motorización [35].

El plan de marketing de nuestro dispositivo se hará mediante sitios web, redes sociales y anuncios en línea, también daremos conferencias para hablar sobre nuestro producto y visitaremos diferentes entidades con el fin de darnos a conocer entre nuestros posibles clientes.

En cuanto a la distribución, será sencilla, nuestra intención es crear una página web en la que nuestro producto se pueda comprar de forma online, esto al menos durará hasta conseguir una buena situación en el mercado, y para ello, necesitaremos distribuidores

especializados en tecnología médica, la asociación con instituciones educativas para su inclusión en programas de estudio, y la colaboración con proveedores de servicios de salud para su implementación en entornos clínicos. Cuando consigamos cumplir todos estos propósitos, nos plantearemos crear alguna tienda física, sobre todo en las grandes ciudades.

Hasta que el cliente se adapte al producto, contaremos con un sistema de atención al cliente totalmente gratuito, en el que estaremos dispuestos a atenderlos en cualquier momento si el producto da algún tipo de error o el cliente tiene alguna duda sobre su uso. Sin embargo, al ser un wearable fácil de manejar, no será necesario incluir personal de instalación del dispositivo.

Y por supuesto lo más importante para nosotros, será adaptar el producto a los requisitos impuestos por los clientes, por lo que, en el futuro haremos diferentes versiones de este intentado amoldarlo a todas y cada una de las necesidades de nuestros clientes. Al final, en un plan comercial, lo más importante es obtener ganancias y que el cliente este satisfecho con el servicio prestado.

VII. CONCLUSIONES

La inteligencia artificial es un tema que actualmente tiene una gran demanda, ya que cada vez hay más sistemas que la utilizan y que en el futuro seguramente eliminará muchos puestos de trabajos porque existirán maquinas inteligentes capaces de hacerlo. Esto hará que se creen otro tipo de trabajos más orientados hacia la fabricación y cuidados de estas máquinas, estamos en medio de una de las revoluciones más grandes que ha podido ver el ser humano y está en nuestras manos aprovecharla sabiamente para avanzar y mejorar. Hoy en día, este es un tema muy controvertido, ya que aparecen preguntas sobre si puede llegar un momento en el que las maquinas puedan alcanzar más inteligencia que los seres humanos. Por ello, hay que tener cuidado porque al tener tanto poder se puede llegar a atentar contra la ética y la moral humana con leyes que establezcan los límites máximos a los que puede llegar esta nueva tecnología.

Este proyecto tiene una gran proyección de crecimiento en este sentido, ya que como hemos podido observar, podría ser una herramienta útil en diferentes disciplinas de trabajos médicos como el diagnóstico de enfermedades relacionadas con el nivel de dolor o la monitorización de pacientes en diferentes estados (hospitalización quirúrgica, rehabilitación,etc.).

Mediante un estudio de inteligencia artificial, hemos logrado elegir un modelo que permita entrenar los datos de un conjunto de entrenamiento, definido a partir de un ensayo clínico, de manera que el propio sistema implementa un reconocimiento de las variables fisiológicas de frecuencia cardíaca, frecuencia respiratoria y temperatura corporal y los clasifica en el nivel de dolor correspondiente con una precisión del 80%, aproximadamente, lo cual nos parece bastante elevado teniendo en cuenta los pocos recursos obtenidos para elaborar el proyecto. Con una financiación podríamos mejorar el sistema y aumentar ese porcentaje lo máximo posible, para que la maquina sea más fiable. Para ello, hemos trabajado con diferentes entornos, Google Colab, Visual Studio, Arduino y App Inventor, que nos han hecho aprender mucho sobre este tema. Por tanto, podemos decir que este proyecto no solo ha sido una mera investigación y estudio, sino que también nos ha servido de ayuda para ampliar nuestros conocimientos de cara a la vida laboral.

Es cierto que el dolor es algo subjetivo que cada persona sufre diferente, que a uno no le duela no significa que a otro sí, y es una de las causas más comunes, que dan lugar a que el paciente lleve peor cualquier tipo de enfermedad. Además, podemos afirmar

que el estudio de modelos de Inteligencia Artificial podría incluir la comparación con más algoritmos, pero pensamos que, en general, el estudio realizado es un éxito y permite demostrar la posibilidad de crear un sistema de motorización del dolor, sin necesidad de tener grandes recursos tecnológicos, sin duda, con ellos, el sistema aumentaría su fiabilidad y precisión.

Además, hemos podido comprobar que parámetros como el pulso, respiración, sudoración y temperatura permiten establecer una clasificación del nivel de dolor adecuada, aunque la correlación entre algunos de ellos sea baja. Seguro que hay más parámetros relacionados que se podrían introducir para mejorar la precisión del aparato, pero esto ya lo dejamos para estudios futuros.

En conclusión, pensamos que este proyecto tiene un gran futuro, es algo muy innovador, ya que hoy en día solamente hay un producto que se asemeje al nuestro. Además, hospitalariamente hablando, un sistema de estas características puede permitir una disminución de la congestión hospitalaria y un gran ahorro a la larga ya que evitaría muchos procesos hospitalarios derivados de un mal diagnóstico o tratamiento. Nosotros, hemos estudiado esta carrera para ayudar a las personas, pensar que con este aparato se pueda mejorar la calidad de vida de muchos ha sido lo que principalmente nos ha impulsado a elaborar el proyecto.

VIII. TRABAJO FUTURO

Tras finalizar este proyecto, hemos visto una serie de cuestiones a mejorar, al final lo que se muestra es un prototipo que con ciertos retoques en un futuro quizás se podría implantar en la vida real, y como hemos podido observar tras el estudio de mercado, sin duda sería un antes y un después en este ámbito de la medicina, con gran cantidad de clientes interesados.

Es cierto que, si hubiéramos elevado el presupuesto, simplemente con unos sensores con mayor precisión y una mejora del diseño exterior, se podría decir que más bien que prototipo se convertiría en la primera versión a la venta del producto con total seguridad.

También nos gustaría mejorar la aplicación dándole un aire más profesional, y añadiendo algunas pantallas aportando más datos a la aplicación, como por ejemplo poder acceder a gráficas que elaboren un estudio estadístico de la motorización del dolor de manera automática, para que se obtenga más datos tanto para su diagnóstico como para la aplicación de su posterior tratamiento.

Con más tiempo podríamos haber analizado más algoritmos, para conocer si el actual definitivamente es el más preciso. También a la hora de obtener los resultados del análisis de datos, al final utilizamos lógica, sin embargo, pensamos que quizás con redes neuronales se obtendría un análisis más preciso.

Evidentemente esto es un caso hipotético, por tanto, no tenemos unos requisitos definidos por los verdaderos clientes, posibles compradores del producto, en caso de implementarlo de forma real tendríamos una cita para conocer sus propuestas, y establecer con mayor exactitud los requisitos en los que estarían más interesados a la hora de usar la aplicación, lo que nos haría mejorarla y hacer que sea lo más útil posible para aquellos que al final van a trabajar con el sistema en su día a día.

Por ello definitivamente, con más tiempo y presupuesto, pensamos que nuestro sistema podría mejorar todas estas cuestiones, para poder implementarlo de forma real, y convertirse en un producto muy prometedor de cara al público.

IX. BIBLIOGRAFÍA

- [1] Ministerio de Sanidad. (s.f.). *Sanidad.gob*. Obtenido de Sanidad en un vistazo: <https://www.sanidad.gob.es/estadEstudios/sanidadDatos/home.htm>
- [2] Calsina-Berna, A., Moreno Millán, N., González-Barboteo, J., Solsona Díaz, L., & Porta Sales, J. (Noviembre de 2011). Prevalencia de dolor como motivo de consulta y su influencia en el sueño: experiencia en un centro de atención primaria. *Atención primaria*, 43(11), 568-575. doi:10.1016/j.aprim.2010.09.006
- [3] Medasense. (2023). *Medasense*. Obtenido de PMD-200: <https://medasense.com/pmd-200/>
- [4] S. Meijer, F., H. Martini, C., Broens, S., Boon, M., Niester, M., Aarts, L., . . . Dahan, A. (Mayo de 2019). Nociception-guided versus Standard Care during Remifentanil–Propofol Anesthesia: A Randomized Controlled Trial. *Anesthesiology*, 130, 745–755.
- [5] S. Meijer, F., H. Martini, C., Broens, S., Boon, M., Niester, M., Aarts, L., . . . Velzen, M. (Diciembre de 2020). Reduced postoperative pain using Nociception Level-guided fentanyl dosing during sevoflurane anaesthesia: a randomised controlled trial. *British Journal of Anaesthesia*, 125, 1070-1078.
- [6] Saunders R, W. R. (Mayo de 2020). COST-BENEFIT OF PERSONALIZING INTRAOPERATIVE PAIN MANAGEMENT. 23(5). Obtenido de <https://www.ispor.org/heor-resources/presentations-database/presentation/intl2020-3182/101924>
- [7] [6] De la Vega, R., & Miró, J. (Julio de 2014). mHealth: a strategic field without a solid scientific soul. a systematic review of pain-related apps. *PLoS One*.
- [8] Miró, J., de-la-Vega, R., Roset, R., Castarlenas, E., & Sánchez-Rodríguez, E. (Marzo de 2018). Painometer v2®: una aplicación móvil certificada para monitorizar a los pacientes con dolor. *Revista de la Sociedad Española del Dolor*, 25.
- [9] Aracena, F., & Christopher. (Julio de 2021). Diferencias en el umbral de presión de los tejidos perilaríngeos entre pacientes con odinofonía e individuos asintomáticos. *Revista de Logopedia, Foniatria y Audiología*, 41(3), 124-132.
- [10] Fisaude. (s.f.). *Algómetro Analógico FPK 60: Graduaciones dobles, medición precisa y test de tolerancia al dolor*. Obtenido de https://tienda.fisaude.com/algometro-analogico-fpk-60-graduaciones-dobles-medicion-precisa-test-tolerancia-al-dolor-p-47754.html?gclid=CjwKCAiAOJKfBhBIEiwAPhZXDzhMOJWI3bdNtKirPwa2V3k2teIvZ2A1cYtQxkhK_2TEzK1TwaY7BoCIJcQAvD_BwE
- [11] Lavigne, G., & TenBokum, L. (1996). *United States Patente nº US5533514A*.
- [12] agiles.org, p. (s.f.). *Historia de Scrum*. Obtenido de proyectos agiles.org.
- [13] Arduino, A. (s.f.). *Lenguaje de programación de Arduino, estructura de un programa*. Obtenido de Aprendiendo Arduino: <https://aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-de-arduino-estructura-de-un-programa/>

- [14] TensorFlow. (s.f.). *Introducción a TensorFlow*. Obtenido de TensorFlow: <https://www.tensorflow.org/learn?hl=es-419>
- [15] Google. (s.f.). *Keras*. Obtenido de <https://keras.io/>
- [16] DataScientest. (Septiembre de 2022). Scikit-Learn : Descubre la biblioteca de Python dedicada al Machine Learning. *DataScientest*.
- [17] pandas2.0.2. (29 de Mayo de 2023). *pandas: powerful Python data analysis toolkit*. Obtenido de <https://pypi.org/project/pandas/>
- [18] NumPy. (s.f.). *NumPy: the absolute basics for beginners*. Obtenido de https://numpy.org/doc/stable/user/absolute_beginners.html#whats-the-difference-between-a-python-list-and-a-numpy-array
- [19] Matplotlib. (s.f.). *Matplotlib: Visualization with Python*. Obtenido de <https://matplotlib.org/>
- [20] FutureSpace. (s.f.). *Redes Neuronales y Deep Learning. Capítulo 2: La Neurona*. Obtenido de <https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/>
- [21] Wikipedia. (s.f.). *Función escalón de Heaviside*. Obtenido de https://es.wikipedia.org/wiki/Funci%C3%B3n_escal%C3%B3n_de_Heaviside
- [22] InteractiveChaos. (s.f.). *One Hot Encoding*. Obtenido de <https://interactivechaos.com/es/manual/tutorial-de-machine-learning/one-hot-encoding#:~:text=Esta%20estrategia%20consiste%20en%20crear,con%20un%20valor%20de%200>
- [23] eni. (s.f.). *Funciones de pérdida (Loss function)*. Obtenido de <https://www.ediciones-eni.com/open/mediabook.aspx?idR=8dd2ca32769cb24b49648b15ef8e777e>
- [24] La La, F. (Abril de 2019). ¿Cómo aprenden las redes neuronales? *MSDN Magazine Issues*, 34(4).
- [25] Manrique Rojas, E. (6 de Febrero de 2020). Machine Learning: análisis de lenguajes de. *Revista Ibérica de Sistemas e Tecnologías de Información*, págs. 586-599.
- [26] Francisco Vallalta Rueda, J. (s.f.). *Aprendizaje supervisado y no supervisado*. Obtenido de <https://healthdataminer.com/data-mining/aprendizaje-supervisado-y-no-supervisado/>
- [27] I'MNOVATION. (s.f.). *Aprendizaje reforzado: cuando las máquinas aprenden solas*. Obtenido de https://www.imnovation-hub.com/es/transformacion-digital/aprendizaje-reforzado-cuando-las-maquinas-aprenden-solas/?_adin=02021864894
- [28] González, L. (2023). *Diferencia entre aprendizaje supervisado y aprendizaje no supervisado*. Obtenido de Aprendeia: <https://aprendeia.com/diferencia-entre-aprendizaje-supervisado-y-no-supervisado/>

- [29] Iván Cardona Alzate, N. (2019). *Predicción y selección de variables con bosques aleatorios en presencia de variables correlacionadas*. Colombia: Universidad Nacional de Colombia.
- [30] Felipe Díaz Sepúlveda, J. (2012). *Comparación entre Arboles de Regresión CART y Regresión Lineal*. Medellín: Universidad Nacional de Colombia.
- [31] Giráldez Rojo, R. (2003). *MEJORAS EN EFICIENCIA Y EFICACIA DE ALGORITMOS EVOLUTIVOS PARA APRENDIZAJE SUPERVISADO*. Sevilla: Universidad de Sevilla.
- [32] Martínez Abad, F. (s.f.). *Aplicación de técnicas de minería de datos con software Weka*. Salamanca: Universidad de Salamanca.
- [33] Estado, A. E. (6 de marzo de 2018). *Agencia Estatal Boletín Oficial del Estado*. Obtenido de [https://www.boe.es/eli/es/res/2018/02/22/\(3\)](https://www.boe.es/eli/es/res/2018/02/22/(3))
- [34] infoautónomos. (s.f.). *Guía fundamental del Análisis DAFO*. Obtenido de <https://www.infoautonomos.com/plan-de-negocio/analisis-daf/>
- [35] Polar. (s.f.). *¿CUÁLES SON LAS VENTAJAS E INCONVENIENTES DE LOS DIFERENTES MÉTODOS PARA MEDIR LA FRECUENCIA CARDÍACA?* Obtenido de <https://support.polar.com/ar-es/what-are-the-pros-and-cons-of-different-methods-for-measuring-heart-rate>
- [I] Monografias. (s.f.). *Redes Neuronales*. Obtenido de Monografias: <https://www.monografias.com/trabajos12/redneuro/redneuro2>
- [II] Elementos básicos de una Red Neuronal Artificial (Parte II) » Ejemplos de Funciones de Activación. (s.f.). Obtenido de Advanced Tech Computing Group UTPL: <https://advancedtech.wordpress.com/2007/09/26/elementos-baiscos-de-una-red-neuronal-artificialparte-ii/ejemplos-de-funciones-de-activacion/>
- [III] Wikipedia. (s.f.). *Perceptrón multicapa*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa
- [IV] González, L. (s.f.). *Diferencia aprendizaje supervisado y no supervisado 3*. Obtenido de Flickr: <https://www.flickr.com/photos/ligdieli/48054954426/>
- [V] Domenech, J. (s.f.). *Método de retención*. Obtenido de Wikipedia: https://es.m.wikipedia.org/wiki/Archivo:Metodo_de_retenci%C3%B3n.jpg
- [VI] Pu, Y., Apel, D., & Wei, C. (Octubre de 2019). Applying Machine Learning Approaches to Evaluating Rockburst Liability: A Comparison of Generative and Discriminative Models. *Pure and Applied Geophysics*, 176-181.
- [VII] Wikipedia. (s.f.). *Regresión lineal*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal

X. ANEXOS:

X.I. PROGRAMACIÓN EN ENTORNO GOOGLE COLAB.

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow
from tensorflow import keras
```

▼ 1. Recopilación y carga de datos.

```
lepsyDataset = pd.read_csv('/content/lepsyDataset.csv')
lepsyDataset.head()

      Frecuencia Cardiaca  Frecuencia Respiratoria  Temperatura Corporal  Respuesta Galvanica  Nivel Dolor
0                71.0                 21.0            36.15           0.06        Bajo
1                67.0                 21.0            36.15           0.09        Bajo
2                84.0                 21.0            36.13           0.12        Bajo
3                83.0                 21.0            36.13           0.07        Bajo
4                75.0                 22.0            36.15           0.03        Bajo

df = pd.DataFrame(lepsyDataset) #Transformación del conjunto en un DataFrame

data = df[['Frecuencia Cardiaca', 'Frecuencia Respiratoria', 'Respuesta Galvanica', 'Temperatura Corporal','Nivel Dolor']]

data

      Frecuencia Cardiaca  Frecuencia Respiratoria  Respuesta Galvanica  Temperatura Corporal  Nivel Dolor
0                71.0                 21.0            0.06            36.15        Bajo
1                67.0                 21.0            0.09            36.15        Bajo
2                84.0                 21.0            0.12            36.13        Bajo
3                83.0                 21.0            0.07            36.13        Bajo
4                75.0                 22.0            0.03            36.15        Bajo
...                  ...
996               99.0                 26.0            0.04            36.13       Alto
997               87.0                 26.0            0.03            36.09       Alto
998              122.0                 26.0            NaN            36.13       Alto
999              125.0                 26.0            0.02            36.09       Alto
1000              NaN                  26.0            0.02            36.13       Alto

1001 rows × 5 columns
```

▼ 2. Limpieza de datos faltantes.

```
data.isnull().sum() #Función para calcular la suma de los valores faltantes en cada atributo.

      Frecuencia Cardiaca    64
      Frecuencia Respiratoria   19
      Respuesta Galvanica     73
      Temperatura Corporal      0
      Nivel Dolor             0
      dtype: int64
```

2.1. Estudio de los datos faltantes de la columna "Frecuencia Cardiaca":

```
promedio = data["Frecuencia Cardiaca"].mean()
mediana = data["Frecuencia Cardiaca"].median()
moda = data["Frecuencia Cardiaca"].mode()
print(promedio, mediana, moda)

112.54855923159018 103.0  95.0
Name: Frecuencia Cardiaca, dtype: float64
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

1/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
#Como los valores de Frecuencia Cardiaca se encuentran dentro del rango del nivel de dolor bajo,  
#tiene más sentido que la Frecuencia Cardiaca sea la moda ya que su valor es 95.  
modaFC = 95 #
```

```
data["Frecuencia Cardiaca"] = data["Frecuencia Cardiaca"].fillna(modaFC)
```

2.2. Estudio de los datos faltantes de la columna "Frecuencia Respiratoria":

```
promedio = data["Frecuencia Respiratoria"].mean()  
mediana = data["Frecuencia Respiratoria"].median()  
moda = data["Frecuencia Respiratoria"].mode()  
print(promedio, mediana, moda)  
  
20.0040733197556 20.0 0 19.0  
Name: Frecuencia Respiratoria, dtype: float64
```

```
#Seleccionamos la mediana porque indica el valor normal de la frecuencia respiratoria  
medianaFR = 20
```

```
data["Frecuencia Respiratoria"] = data["Frecuencia Respiratoria"].fillna(medianaFR)
```

2.3. Estudio de los datos faltantes de la columna "Respuesta Galvanica"

```
promedio = data["Respuesta Galvanica"].mean()  
mediana = data["Respuesta Galvanica"].median()  
moda = data["Respuesta Galvanica"].mode()  
print(promedio, mediana, moda)  
  
0.06264008620689657 0.04 0 0.04  
Name: Respuesta Galvanica, dtype: float64
```

```
#Como los valores de respuesta galvánica faltantes se encuentran en el nivel de dolor medio-alto,  
#hacemos uso del valor promedio redondeado de esta variable  
promedioGSR = 0.06
```

```
data["Respuesta Galvanica"] = data["Respuesta Galvanica"].fillna(promedioGSR)
```

2.4. Estudio de los datos faltantes de la columna "Temperatura Corporal":

No es necesario realizar el estudio para esta variable porque el sensor capta todos los valores y no hay ninguno faltante.

```
data.isnull().sum()  
  
Frecuencia Cardiaca 0  
Frecuencia Respiratoria 0  
Respuesta Galvanica 0  
Temperatura Corporal 0  
Nivel Dolor 0  
dtype: int64
```

Como podemos observar, todos los valores faltantes han sido rellenados.

▼ Codificación de variables categóricas

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

Realizando un estudio de los tipo de variables que tenemos en el conjunto, recogemos lo siguiente:

- Frecuencia Cardíaca: Numérica-Continua.
- Frecuencia Respiratoria: Numérica-Discreta.
- Respuesta Galvánica: Numérica-Continua.
- Temperatura Corporal: Numérica-Continua.
- Nivel de dolor: Categórica-Ordinaria.

Como podemos observar, la variable: "Nivel de dolor" es categórica-ordinaria, por lo que es necesario realizar la codificación para dicha variable. Para ello, deberíamos asignar un valor numérico a cada categoría tal que:

- "ALTO" = 3.
- "MEDIO" = 2.
- "BAJO" = 1.

Si la variable es categórica ordinal, la técnica de codificación más apropiada a utilizar es la codificación ordinal. En este método, se asigna un valor numérico a cada categoría de la variable, de tal manera que se conserve el orden de las categorías. En este caso, si la variable es el **nivel de dolor** con las categorías "BAJO", "MEDIO", "ALTO", se podría asignar los valores numéricos 1, 2 y 3, respectivamente, para reflejar el orden creciente del nivel de dolor.

La ventaja de utilizar la codificación ordinal en este tipo de variables es que se conserva la información de orden entre las categorías, lo que puede ser importante en algunos análisis estadísticos. Además, al utilizar valores numéricos en lugar de etiquetas de texto, se facilita el procesamiento y análisis de los datos. Sobretodo, es una buena opción si son equidistantes.

```
from sklearn.preprocessing import OrdinalEncoder
category_painlevel = np.array(data[["Nivel Dolor"]])
codification = OrdinalEncoder(categories=[["Bajo", "Medio", "Alto"]])
encoder = codification.fit_transform(category_painlevel)

print(encoder)

[[0.]
 [0.]
 [0.]
 ...
 [2.]
 [2.]
 [2.]]]

data[["Nivel Dolor"]] = encoder
```

data

	Frecuencia Cardiaca	Frecuencia Respiratoria	Respuesta Galvánica	Temperatura Corporal	Nivel Dolor	
0	71.0	21.0	0.06	36.15	0.0	
1	67.0	21.0	0.09	36.15	0.0	
2	84.0	21.0	0.12	36.13	0.0	
3	83.0	21.0	0.07	36.13	0.0	
4	75.0	22.0	0.03	36.15	0.0	
...	
996	99.0	26.0	0.04	36.13	2.0	
997	87.0	26.0	0.03	36.09	2.0	
998	122.0	26.0	0.06	36.13	2.0	
999	125.0	26.0	0.02	36.09	2.0	
1000	95.0	26.0	0.02	36.13	2.0	

1001 rows × 5 columns

Selección de características

▼ Normalización

```
fig = plt.figure(figsize=(15,5))
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

3/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```

ax1 = fig.add_subplot(2,5,1)
ax1.set_title("Nivel Dolor")
ax1.plot(data["Nivel Dolor"], linewidth=0, marker = "+", color = "red")

ax2 = fig.add_subplot(2,5,2)
ax2.set_title("Frecuencia Cardiaca")
ax2.plot(data["Frecuencia Cardiaca"], linewidth=0, marker = "+", color = "blue")

ax3 = fig.add_subplot(2,5,3)
ax3.set_title("Frecuencia Respiratoria")
ax3.plot(data["Frecuencia Respiratoria"], linewidth=0, marker = "+", color = "orange")

ax4 = fig.add_subplot(2,5,4)
ax4.set_title("Temperatura Corporal")
ax4.plot(data["Temperatura Corporal"], linewidth=0, marker = "+", color = "green")

ax5 = fig.add_subplot(2,5,5)
ax5.set_title("Respuesta Galvánica")
ax5.plot(data["Respuesta Galvánica"], linewidth=0, marker = "+", color = "brown")

ax11 = fig.add_subplot(2,5,6)
ax11.set_title("Nivel Dolor")
ax11.hist(data["Nivel Dolor"], color = "red")

ax22 = fig.add_subplot(2,5,7)
ax22.set_title("Frecuencia Cardiaca")
ax22.hist(data["Frecuencia Cardiaca"], color = "blue")

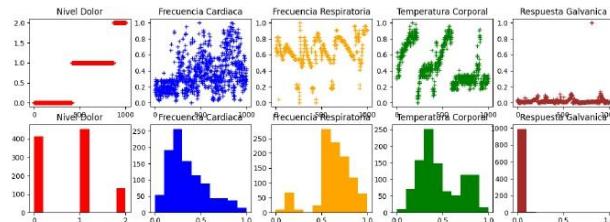
ax33 = fig.add_subplot(2,5,8)
ax33.set_title("Frecuencia Respiratoria")
ax33.hist(data["Frecuencia Respiratoria"], color = "orange")

ax44 = fig.add_subplot(2,5,9)
ax44.set_title("Temperatura Corporal")
ax44.hist(data["Temperatura Corporal"], color = "green")

ax55 = fig.add_subplot(2,5,10)
ax55.set_title("Respuesta Galvánica")
ax55.hist(data["Respuesta Galvánica"], color = "brown")

plt.show()

```



Haz doble clic (o pulsa Intro) para editar

```
from sklearn import preprocessing
```

Escala en función del mínimo y el máximo

```
data_min_max=preprocessing.MinMaxScaler().fit_transform(data[["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvánica", ""]])
```

Normalización

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

4/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
data_normalizer=preprocessing.Normalizer().transform(data[["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica", "Temperatura"])
data_normalizer=data_normalizer.T
# normalizado= X/raiz_cuadrada(X_1^2 + X_2^2 + X_3^2 +...)
data_normalizer
```

```
array([[ 0.01887961,  0.03250003,  0.01823725,  0.03148697],
       [ 0.01781597,  0.03250003,  0.02735588,  0.03148697],
       [ 0.02233644,  0.03250003,  0.0364745 ,  0.03146955],
       ...,
       [ 0.03244103,  0.04023813,  0.01823725,  0.03146955],
       [ 0.03323876,  0.04023813,  0.00607908,  0.03143471],
       [ 0.02526146,  0.04023813,  0.00607908,  0.03146955]])
```

Estandarización

```
data_standard_scaler = preprocessing.StandardScaler().fit_transform(data[["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica"]])
## estandarizado = (X-media)/std
data_robust_scaler= preprocessing.RobustScaler().fit_transform(data[["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica"]])
#estandarizado=(X-rango_intercuartilico)/std
data_standard_scaler,data_robust_scaler

(array([[-0.97686631,  0.24201497, -0.02943652, -0.60800696],
       [-0.107352217,  0.24201497,  0.33137112, -0.60800696],
       [-0.66273476,  0.24201497,  0.69217876, -0.69671341],
       ...,
       [ 0.25549591,  1.4569447 , -0.02943652, -0.69671341],
       [ 0.3279878 ,  1.4569447 , -0.51051337, -0.87412632],
       [-0.39693115,  1.4569447 , -0.51051337, -0.69671341]]),
array([[-0.53703704,  0.2         ,  1.         , -0.22222222],
       [-0.61111111,  0.2         ,  2.5        , -0.22222222],
       [-0.2962963 ,  0.2         ,  4.         , -0.27777778],
       ...,
       [ 0.40740741,  1.2         ,  1.         , -0.27777778],
       [ 0.46296296,  1.2         , -1.         , -0.38888889],
       [-0.09259259,  1.2         , -1.         , -0.27777778]]))
```

Comparación de métodos

```
#convierte vectores de numpy a Data Frames para graficarlos
data_min_max = pd.DataFrame(data_min_max, columns=["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica", "Temperatura"])
data_normalizer=pd.DataFrame(data_normalizer, columns=["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica", "Temperatura"])
data_standard_scaler=pd.DataFrame(data_standard_scaler, columns=["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica", "Temperatura"])
data_robust_scaler=pd.DataFrame(data_robust_scaler, columns=["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvanica", "Temperatura"])
#crea una figura con 5 subfiguras para comparar los metodos
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos

ax1.set_title("Frecuencia Cardiaca")
ax1.plot(data["Frecuencia Cardiaca"], linewidth=0, marker="*", color="red", markersize=4)

ax2.set_title("Min Max")
ax2.plot(data_min_max["Frecuencia Cardiaca"], linewidth=0, marker="*", color="red", markersize=4)

ax3.set_title("Normalizer")
ax3.plot(data_normalizer["Frecuencia Cardiaca"], linewidth=0, marker="*", color="red", markersize=4)
ax3.set_ylim(0,1)

ax4.set_title("Standard Scaler")
ax4.plot(data_robust_scaler["Frecuencia Cardiaca"], linewidth=0, marker="*", color="red", markersize=4)

ax5.set_title("Robust Scaler")
ax5.plot(data_standard_scaler["Frecuencia Cardiaca"], linewidth=0, marker="*", color="red", markersize=4)

#crea una figura con 5 subfiguras para comparar los metodos con los histogramas
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos de los histogramas
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

5/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
ax1.set_title("Frecuencia Cardiaca")
ax1.hist(data["Frecuencia Cardiaca"], color="red", bins=100)

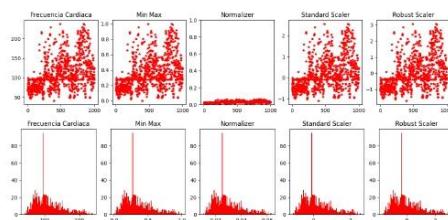
ax2.set_title("Min Max")
ax2.hist(data_min_max["Frecuencia Cardiaca"], color="red", bins=100)

ax3.set_title("Normalizer")
ax3.hist(data_normalizer["Frecuencia Cardiaca"], color="red", bins=100)

ax4.set_title("Standard Scaler")
ax4.hist(data_robust_scaler["Frecuencia Cardiaca"], color="red", bins=100)

ax5.set_title("Robust Scaler")
ax5.hist(data_standard_scaler["Frecuencia Cardiaca"], color="red", bins=100)

plt.show()
```



Podemos observar que los datos son prácticamente iguales, independientemente de los transformadores de normalización y escala. Sin embargo, la principal diferencia que encontramos es la escala sin afectar la información. En el caso de min_max, la escala está entre [0, 1] En el caso de normalizer, la escala está entre [0, 0.12] lo que puede provocar un agolamiento de los datos. En el caso de standar_scaler, la escala está entre [-1, 1.5], evitando los outliers que aparecen en el robust_scaler cuya escala está entre [-1.5,2]. Por ello, utilizaremos el transformador min_max.

Haz doble clic (o pulsa Intro) para editar

```
#crea una figura con 5 subfiguras para comparar los métodos
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos

ax1.set_title("Frecuencia Respiratoria")
ax1.plot(data["Frecuencia Respiratoria"], linewidth=0, marker="*", color="red", markersize=4)

ax2.set_title("Min Max")
ax2.plot(data_min_max["Frecuencia Respiratoria"], linewidth=0, marker="*", color="red", markersize=4)

ax3.set_title("Normalizer")
ax3.plot(data_normalizer["Frecuencia Respiratoria"], linewidth=0, marker="*", color="red", markersize=4)
ax3.set_ylim(0,1)
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfnB8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

6/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```

ax4.set_title("Standard Scaler")
ax4.plot(data_robust_scaler["Frecuencia Respiratoria"], linewidth=0, marker="*", color="red", markersize=4)

ax5.set_title("Robust Scaler")
ax5.plot(data_standard_scaler["Frecuencia Respiratoria"], linewidth=0, marker="*", color="red", markersize=4)

#crea una figura con 5 subfiguras para comparar los métodos con los histogramas
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#create y personaliza series de datos de los histogramas

ax1.set_title("Frecuencia Respiratoria")
ax1.hist(data["Frecuencia Respiratoria"], color="red", bins=100)

ax2.set_title("Min Max")
ax2.hist(data_min_max["Frecuencia Respiratoria"], color="red", bins=100)

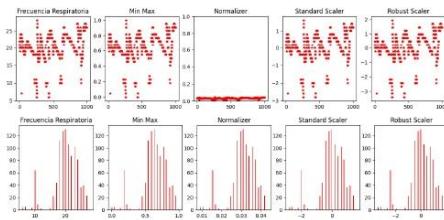
ax3.set_title("Normalizer")
ax3.hist(data_normalizer["Frecuencia Respiratoria"], color="red", bins=100)

ax4.set_title("Standard Scaler")
ax4.hist(data_robust_scaler["Frecuencia Respiratoria"], color="red", bins=100)

ax5.set_title("Robust Scaler")
ax5.hist(data_standard_scaler["Frecuencia Respiratoria"], color="red", bins=100)

plt.show()

```



La distribución de la variable, en este caso, se asemeja a una distribución normal, pero en este caso por la derecha de la gráfica, en todas las escalas. El comportamiento de la frecuencia respiratoria es similar al de la frecuencia cardíaca cuando se transforma su escala. Los datos se amontonan en la normalización y, tanto el escalado estándar como el escalado robusto presentan valores negativos en su escala y esta vez, más negativos que positivos. Por ello, podemos afirmar que el mejor método de escalado para esta variable también es la representación de los datos en la escala mínimo-máximo.

```

#crea una figura con 5 subfiguras para comparar los métodos
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)

```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9FQfD&printMode=true>

7/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```

ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos

ax1.set_title("Respuesta Galvanica")
ax1.plot(data["Respuesta Galvanica"], linewidth=0, marker="*", color="red", markersize=4)

ax2.set_title("Min Max")
ax2.plot(data_min_max["Respuesta Galvanica"], linewidth=0, marker="*", color="red", markersize=4)

ax3.set_title("Normalizer")
ax3.plot(data_normalizer["Respuesta Galvanica"], linewidth=0, marker="*", color="red", markersize=4)
ax3.set_ylim(0,1)

ax4.set_title("Standard Scaler")
ax4.plot(data_robust_scaler["Respuesta Galvanica"], linewidth=0, marker="*", color="red", markersize=4)

ax5.set_title("Robust Scaler")
ax5.plot(data_standard_scaler["Respuesta Galvanica"], linewidth=0, marker="*", color="red", markersize=4)

#crea una figura con 5 subfiguras para comparar los métodos con los histogramas
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos de los histogramas

ax1.set_title("Respuesta Galvanica")
ax1.hist(data["Respuesta Galvanica"], color="red", bins=100)

ax2.set_title("Min Max")
ax2.hist(data_min_max["Respuesta Galvanica"], color="red", bins=100)

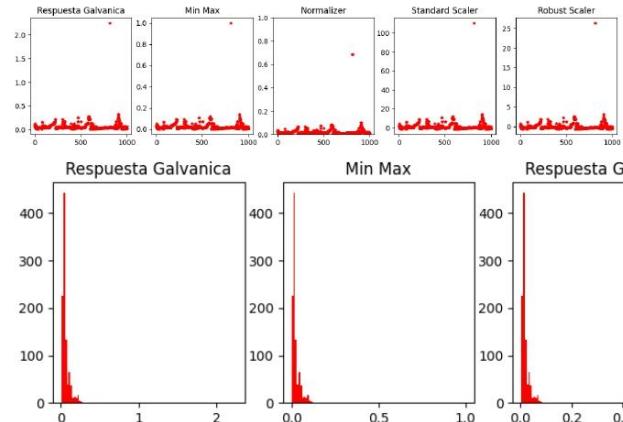
ax3.set_title("Respuesta Galvanica")
ax3.hist(data_normalizer["Respuesta Galvanica"], color="red", bins=100)

ax4.set_title("Standard Scaler")
ax4.hist(data_robust_scaler["Respuesta Galvanica"], color="red", bins=100)

ax5.set_title("Robust Scaler")
ax5.hist(data_standard_scaler["Respuesta Galvanica"], color="red", bins=100)

plt.show()

```



8/6/23, 21:48

Lepasy.ipynb - Colaboratory

La representación de la respuesta galvánica es igual en todas las escalas puesto que los valores que contiene esta variable son muy pequeños, en una escala centesimal. La normalización amontona un poco los resultados si nos fijamos bien también en este caso. Además, cabe resaltar que la escala de las técnicas de estandarización y escalado robusto se disparan, llegando a valores superiores a las dos centenas, como es el caso de la estandarización. Por ello, podemos afirmar la mejor técnica de transformación para este caso también es el mínimo-máximo.

```
#crea una figura con 5 subfiguras para comparar los metodos
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos

ax1.set_title("Temperatura Corporal")
ax1.plot(data["Temperatura Corporal"], linewidth=0, marker="*", color="red", markersize=4)

ax2.set_title("Min Max")
ax2.plot(data_min_max["Temperatura Corporal"], linewidth=0, marker="*", color="red", markersize=4)

ax3.set_title("Normalizer")
ax3.plot(data_normalizer["Temperatura Corporal"], linewidth=0, marker="*", color="red", markersize=4)
ax3.set_ylim(0,1)

ax4.set_title("Standard Scaler")
ax4.plot(data_robust_scaler["Temperatura Corporal"], linewidth=0, marker="*", color="red", markersize=4)

ax5.set_title("Robust Scaler")
ax5.plot(data_standard_scaler["Temperatura Corporal"], linewidth=0, marker="*", color="red", markersize=4)

#crea una figura con 5 subfiguras para comparar los metodos con los histogramas
fig=plt.figure(figsize=(15,3))
ax1=fig.add_subplot(1,5,1)
ax2=fig.add_subplot(1,5,2)
ax3=fig.add_subplot(1,5,3)
ax4=fig.add_subplot(1,5,4)
ax5=fig.add_subplot(1,5,5)

#crear y personaliza series de datos de los histogramas

ax1.set_title("Temperatura Corporal")
ax1.hist(data["Temperatura Corporal"], color="red", bins=100)

ax2.set_title("Min Max")
ax2.hist(data_min_max["Temperatura Corporal"], color="red", bins=100)

ax3.set_title("Normalizer")
ax3.hist(data_normalizer["Temperatura Corporal"], color="red", bins=100)

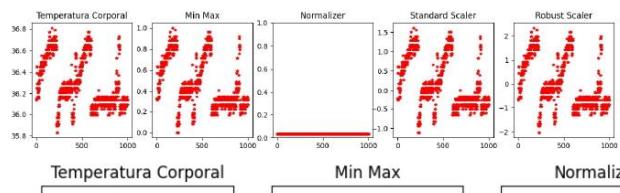
ax4.set_title("Standard Scaler")
ax4.hist(data_robust_scaler["Temperatura Corporal"], color="red", bins=100)

ax5.set_title("Robust Scaler")
ax5.hist(data_standard_scaler["Temperatura Corporal"], color="red", bins=100)

plt.show()
```

8/6/23, 21:48

Lepasy.ipynb - Colaboratory



Como podemos observar, la temperatura corporal presenta una distribución de muestras peculiar. En este caso, la escala normal de los datos es muy pequeña, con una diferencia de aproximadamente un solo grado entre el valor máximo y el valor mínimo de los datos de temperatura corporal, por lo que, de entrada, podemos afirmar que la escala de la técnica mínimo-máximo es la más adecuada para esta variable. Además, como pasa en el resto de las variables, la estandarización y el escalado robusto presentan valores negativos dentro de su escala proporcionando información errónea al conjunto, aunque es importante recalcar que en este caso las escalas no se disparan, si no que se mantienen valores cercanos a uno. La técnica de normalización sigue agolpando los datos, de hecho, aparece como si no hubiera ningún dato, según el gráfico de dispersión, por lo que podemos clasificar esta técnica como la menos fiable.

```
data[["Frecuencia Cardiaca", "Frecuencia Respiratoria", "Respuesta Galvánica", "Temperatura Corporal"]] = data_min_max
```

	Frecuencia Cardiaca	Frecuencia Respiratoria	Respuesta Galvanica	Temperatura Corporal	Nivel Dolor
0	0.152284	0.681818	0.022321	0.326531	0.0
1	0.131980	0.681818	0.035714	0.326531	0.0
2	0.218274	0.681818	0.049107	0.306122	0.0
3	0.213198	0.681818	0.026786	0.306122	0.0
4	0.172589	0.727273	0.008929	0.326531	0.0

Análisis de correlación de las variables

```
data.corr('pearson')
```

	Frecuencia Cardíaca	Frecuencia Respiratoria	Respuesta Galvánica	Temperatura Corporal	Nivel Dolor
Frecuencia Cardíaca	1.000000	-0.022845	0.139896	-0.061336	0.327508
Frecuencia Respiratoria	-0.022845	1.000000	-0.044550	0.108603	0.357022
Respuesta Galvánica	0.139896	-0.044550	1.000000	0.216312	0.058432

▼ Deep Learning

data

	Frecuencia Cardíaca	Frecuencia Respiratoria	Respuesta Galvánica	Temperatura Corporal	Nivel Dolor
0	0.152284	0.681818	0.022321	0.326531	0.0
1	0.131980	0.681818	0.035714	0.326531	0.0
2	0.218274	0.681818	0.049107	0.306122	0.0
3	0.213198	0.681818	0.026786	0.306122	0.0
4	0.172589	0.727273	0.008929	0.326531	0.0
...
996	0.294416	0.909091	0.013393	0.306122	2.0
997	0.233503	0.909091	0.008929	0.265306	2.0
998	0.411168	0.909091	0.022321	0.306122	2.0
999	0.426396	0.909091	0.004464	0.265306	2.0
1000	0.274112	0.909091	0.004464	0.306122	2.0

1001 rows × 5 columns

<https://colab.research.google.com/drive/1VY4UZFMcq8be3XwYO54NfNb8sV6oHptx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

10/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

Red neuronal

```
from keras.models import Sequential
from keras.layers import Dense, Activation
https://colab.research.google.com/drive/1VY4U1FMcq8bF3XwYO54NfNb8sV6ohIPtx?hl=es#scrollTo=GvDOJ9lFQfD&printMode=true
```

11/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```

from keras.layers import Dense, Activation
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(64, input_shape=(4,), activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(3, activation='softmax')])
# Compilación del modelo
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train,
                      batch_size=32, epochs=100,
                      validation_data=[X_test, y_test])

Epoch 1/100
25/25 [=====] - 1s 8ms/step - loss: 1.0402 - accuracy: 0.4787 - val_loss: 0.9856 - val_accuracy: 0.5572
Epoch 2/100
25/25 [=====] - 0s 2ms/step - loss: 0.9575 - accuracy: 0.5975 - val_loss: 0.9390 - val_accuracy: 0.5821
Epoch 3/100
25/25 [=====] - 0s 3ms/step - loss: 0.9170 - accuracy: 0.6137 - val_loss: 0.8991 - val_accuracy: 0.5771
Epoch 4/100
25/25 [=====] - 0s 2ms/step - loss: 0.8818 - accuracy: 0.6087 - val_loss: 0.8707 - val_accuracy: 0.5721
Epoch 5/100
25/25 [=====] - 0s 2ms/step - loss: 0.8463 - accuracy: 0.6137 - val_loss: 0.8413 - val_accuracy: 0.5771
Epoch 6/100
25/25 [=====] - 0s 2ms/step - loss: 0.8212 - accuracy: 0.6413 - val_loss: 0.8509 - val_accuracy: 0.5871
Epoch 7/100
25/25 [=====] - 0s 2ms/step - loss: 0.8032 - accuracy: 0.6525 - val_loss: 0.8054 - val_accuracy: 0.6020
Epoch 8/100
25/25 [=====] - 0s 2ms/step - loss: 0.7808 - accuracy: 0.6513 - val_loss: 0.7925 - val_accuracy: 0.6219
Epoch 9/100
25/25 [=====] - 0s 2ms/step - loss: 0.7649 - accuracy: 0.6612 - val_loss: 0.7832 - val_accuracy: 0.6219
Epoch 10/100
25/25 [=====] - 0s 2ms/step - loss: 0.7517 - accuracy: 0.6687 - val_loss: 0.7872 - val_accuracy: 0.6070
Epoch 11/100
25/25 [=====] - 0s 2ms/step - loss: 0.7412 - accuracy: 0.6837 - val_loss: 0.7670 - val_accuracy: 0.6368
Epoch 12/100
25/25 [=====] - 0s 2ms/step - loss: 0.7283 - accuracy: 0.6812 - val_loss: 0.7647 - val_accuracy: 0.6318
Epoch 13/100
25/25 [=====] - 0s 2ms/step - loss: 0.7169 - accuracy: 0.6913 - val_loss: 0.7728 - val_accuracy: 0.6318
Epoch 14/100
25/25 [=====] - 0s 3ms/step - loss: 0.7136 - accuracy: 0.6875 - val_loss: 0.7643 - val_accuracy: 0.6368
Epoch 15/100
25/25 [=====] - 0s 2ms/step - loss: 0.7048 - accuracy: 0.6963 - val_loss: 0.7557 - val_accuracy: 0.6517
Epoch 16/100
25/25 [=====] - 0s 2ms/step - loss: 0.6990 - accuracy: 0.6950 - val_loss: 0.7502 - val_accuracy: 0.6517
Epoch 17/100
25/25 [=====] - 0s 2ms/step - loss: 0.6973 - accuracy: 0.6975 - val_loss: 0.7445 - val_accuracy: 0.6617
Epoch 18/100
25/25 [=====] - 0s 3ms/step - loss: 0.6884 - accuracy: 0.6988 - val_loss: 0.7466 - val_accuracy: 0.6617
Epoch 19/100
25/25 [=====] - 0s 2ms/step - loss: 0.6844 - accuracy: 0.6988 - val_loss: 0.7621 - val_accuracy: 0.6468
Epoch 20/100
25/25 [=====] - 0s 2ms/step - loss: 0.6757 - accuracy: 0.7050 - val_loss: 0.7461 - val_accuracy: 0.6368
Epoch 21/100
25/25 [=====] - 0s 3ms/step - loss: 0.6750 - accuracy: 0.7088 - val_loss: 0.7430 - val_accuracy: 0.6468
Epoch 22/100
25/25 [=====] - 0s 2ms/step - loss: 0.6741 - accuracy: 0.7025 - val_loss: 0.7382 - val_accuracy: 0.6567
Epoch 23/100
25/25 [=====] - 0s 2ms/step - loss: 0.6664 - accuracy: 0.7100 - val_loss: 0.7370 - val_accuracy: 0.6517
Epoch 24/100
25/25 [=====] - 0s 2ms/step - loss: 0.6644 - accuracy: 0.7063 - val_loss: 0.7404 - val_accuracy: 0.6617
Epoch 25/100
25/25 [=====] - 0s 3ms/step - loss: 0.6583 - accuracy: 0.7150 - val_loss: 0.7370 - val_accuracy: 0.6617
Epoch 26/100
25/25 [=====] - 0s 3ms/step - loss: 0.6572 - accuracy: 0.7188 - val_loss: 0.7312 - val_accuracy: 0.6716
Epoch 27/100
25/25 [=====] - 0s 3ms/step - loss: 0.6524 - accuracy: 0.7212 - val_loss: 0.7319 - val_accuracy: 0.6816
Epoch 28/100
25/25 [=====] - 0s 2ms/step - loss: 0.6527 - accuracy: 0.7312 - val_loss: 0.7314 - val_accuracy: 0.6816
Epoch 29/100
25/25 [=====] - 0s 2ms/step - loss: 0.6514 - accuracy: 0.7212 - val_loss: 0.7323 - val_accuracy: 0.6915

import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Evolución del error durante el entrenamiento')
plt.ylabel('Error')
plt.xlabel('Época')
plt.legend(['Entrenamiento', 'Validación'], loc='upper right')
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Evolución del error durante el entrenamiento')
plt.ylabel('Error')
plt.xlabel('Época')
plt.legend(['Entrenamiento', 'Validación'], loc='upper right')

```

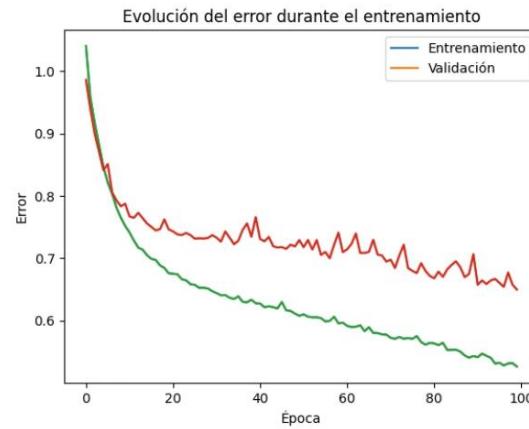
<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

12/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
plt.show()  
plt.show()
```



```
# Evaluacion del rendimiento del modelo en los datos de prueba
loss, accuracy = model.evaluate(X_test, y_test)
print('Función de pérdida:', loss)
print('Exactitud:', accuracy)

7/7 [=====] - 0s 1ms/step - loss: 0.6497 - accuracy: 0.7562
Función de pérdida: 0.6497092843055725
Exactitud: 0.7562189102172852

from sklearn.metrics import confusion_matrix

# Predicciones del modelo en los datos de prueba
y_pred = model.predict(X_test)

# Conversión las probabilidades a etiquetas de clase (alto, medio, bajo)
y_pred_labels = np.argmax(y_pred, axis=1)
y_test_labels = np.argmax(y_test, axis=1)

# Generación de la matriz de confusión
conf_matrix = confusion_matrix(y_test_labels, y_pred_labels)
print('Matriz de confusión:\n', conf_matrix)

7/7 [=====] - 0s 1ms/step
Matriz de confusión:
[[65 12  7]
 [ 7 78  5]
 [ 2 16  9]]

model.save('neuronalNetworkLevelPainSystem')

WARNING:absl:Found untraced functions such as _update_step_xla while saving (showing 1 of 1). These functions will not be directly
```

Machine Learning

Limpieza de datos faltantes

```
modaFC = 95
medianaFR = 20
promedioGSR = 0.06
data["Frecuencia Cardiaca"] = data["Frecuencia Cardiaca"].fillna(modafC)
data["Frecuencia Respiratoria"] = data["Frecuencia Respiratoria"].fillna(medianaFR)
data["Respuesta Galvanica"] = data["Respuesta Galvanica"].fillna(promedioGSR)
data.isnull().sum()

Frecuencia Cardiaca      0
Frecuencia Respiratoria  0
Respuesta Galvanica      0
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

13/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
Temperatura Corporal      0
Nivel Dolor               0
dtype: int64
```

Codificación de variables categóricas

```
from sklearn.preprocessing import OrdinalEncoder
category_painlevel = np.array(data[["Nivel Dolor"]])
codification = OrdinalEncoder(categories=[[ "Bajo", "Medio", "Alto"]])
encoder = codification.fit_transform(category_painlevel)
data[["Nivel Dolor"]] = encoder
data
```

	Frecuencia Cardíaca	Frecuencia Respiratoria	Respuesta Galvánica	Temperatura Corporal	Nivel Dolor
0	71.0	21.0	0.06	36.15	0.0
1	67.0	21.0	0.09	36.15	0.0
2	84.0	21.0	0.12	36.13	0.0
3	83.0	21.0	0.07	36.13	0.0
4	75.0	22.0	0.03	36.15	0.0
...
996	99.0	26.0	0.04	36.13	2.0
997	87.0	26.0	0.03	36.09	2.0
998	122.0	26.0	0.06	36.13	2.0
999	125.0	26.0	0.02	36.09	2.0
1000	95.0	26.0	0.02	36.13	2.0

1001 rows × 5 columns

Correlación de variables

```
data.corr('pearson')
```

	Frecuencia Cardíaca	Frecuencia Respiratoria	Respuesta Galvánica	Temperatura Corporal	Nivel Dolor
Frecuencia Cardíaca	1.000000	-0.022845	0.139896	-0.061336	0.327505
Frecuencia Respiratoria	-0.022845	1.000000	-0.044550	0.108603	0.357027
Respuesta Galvánica	0.139896	-0.044550	1.000000	0.216312	0.058432

Selección de características

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
X=data[["Frecuencia Cardíaca","Frecuencia Respiratoria","Respuesta Galvánica","Temperatura Corporal"]]
y=data[["Nivel Dolor"]]
y = y.astype(int);
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Modelo de Regresión Lineal

```
from sklearn.linear_model import LinearRegression
reg= LinearRegression().fit(X_train, y_train)
```

```
precision = reg.score(X_train, y_train)
```

```
# Hacer predicciones sobre el conjunto de prueba
y_pred = reg.predict(X_test)
```

```
#PRECISIÓN DEL MODELO
print("PRECISIÓN DEL MODELO DE REGRESIÓN LINEAR MÚLTIPLE: ",precision)
# Error cuadrático medio
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfnNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

14/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
mse = mean_squared_error(y_test, y_pred)
print("Error cuadrático medio:", mse)
# Coeficientes y el sesgo - Valores de los pesos del modelo de regresión
coef = reg.coef_
print("Coeficientes:", coef)
print("Sesgo (Coeficiente W0:", reg.intercept_)
#Coeficientes de determinación
print("Coeficiente de determinación:", reg.score(X_train, y_train))

PRECISIÓN DEL MODELO DE REGRESIÓN LINEAR MÚLTIPLE:  0.3433169122641547
Error cuadrático medio: 0.3470568928380528
Coeficientes: [[ 0.00521591  0.07012692  0.74913268 -0.92190983]]
Sesgo (Coeficiente W0: [32.14583155]
Coeficiente de determinación: 0.3433169122641547
```

Árbol de decisión:

```
from sklearn import tree
arbol_decision = tree.DecisionTreeClassifier(criterion="entropy")
arbol = arbol_decision.fit(X_train, y_train)
accuracy = arbol_decision.score(X_test, y_test)
print("Accuracy:",accuracy)
tree.plot_tree(arbol)
```

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
Accuracy: 0.8159203980099502
[Text(0.5669055451127819, 0.9705882352941176, 'x[1] <= 22.5\nentropy =
1.428\nsamples = 800\nvalue = [330, 365, 105']),
Text(0.3107377819548872, 0.9117647058823529, 'x[0] <= 101.5\nentropy =
1.275\nsamples = 572\nvalue = [307, 226, 39']),
Text(0.11395676691729323, 0.8529411764705882, 'x[3] <= 36.16\nentropy =
0.801\nsamples = 307\nvalue = [246, 56, 51']),
Text(0.05263157894736842, 0.7941176470588235, 'x[2] <= 0.035\nentropy =
1.239\nsamples = 94\nvalue = [49, 38, 51]),
Text(0.0225256390977443608, 0.7352941176470589, 'x[3] <= 36.0\nentropy =
0.406\nsamples = 37\nvalue = [34, 3, 0']),
Text(0.015037593984962405, 0.6764705882352942, 'entropy = 0.0\nsamples = 28\nvalue =
[28, 0, 0']),
Text(0.03007518796992481, 0.6764705882352942, 'x[2] <= 0.025\nentropy =
0.918\nsamples = 9\nvalue = [6, 3, 0']),
Text(0.0225256390977443608, 0.6176470588235294, 'x[0] <= 87.0\nentropy =
1.0\nsamples = 6\nvalue = [3, 3, 0']),
Text(0.015037593984962405, 0.5588235294117647, 'x[2] <= 0.015\nentropy =
0.811\nsamples = 4\nvalue = [1, 3, 0']),
Text(0.007518796992481203, 0.5, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2, 0']),
Text(0.0225256390977443608, 0.5, 'x[0] <= 74.0\nentropy = 1.0\nsamples = 2\nvalue =
[1, 1, 0']),
Text(0.015037593984962405, 0.4411764705882353, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0, 0']),
Text(0.03007518796992481, 0.4411764705882353, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1, 0']),
Text(0.03007518796992481, 0.5588235294117647, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0, 0']),
Text(0.03759398496240601, 0.6176470588235294, 'entropy = 0.0\nsamples = 3\nvalue =
[3, 0, 0']),
Text(0.08270676691729323, 0.7352941176470589, 'x[2] <= 0.08\nentropy =
1.241\nsamples = 55\nvalue = [15, 35, 5']),
Text(0.06766917293233082, 0.6764705882352942, 'x[1] <= 21.5\nentropy =
0.811\nsamples = 44\nvalue = [11, 33, 0']),
Text(0.06015037593984962, 0.6176470588235294, 'x[3] <= 35.98\nentropy =
0.938\nsamples = 31\nvalue = [11, 20, 0']),
Text(0.05263157894736842, 0.5588235294117647, 'entropy = 0.0\nsamples = 3\nvalue =
[3, 0, 0']),
Text(0.06766917293233082, 0.5588235294117647, 'x[3] <= 36.11\nentropy =
0.863\nsamples = 28\nvalue = [8, 20, 0']),
Text(0.05263157894736842, 0.5, 'x[2] <= 0.045\nentropy = 0.337\nsamples =
16\nvalue = [1, 15, 0']),
Text(0.045112781954887216, 0.4411764705882353, 'x[1] <= 18.5\nentropy =
0.592\nsamples = 7\nvalue = [1, 6, 0']),
Text(0.03759398496240601, 0.38235294117647056, 'x[1] <= 14.0\nentropy =
1.0\nsamples = 2\nvalue = [1, 1, 0']),
Text(0.03007518796992481, 0.3235294117647059, 'entropy = 0.0\nsamples = 1\nvalue =
[0, 1, 0']),
Text(0.045112781954887216, 0.3235294117647059, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0, 0']),
Text(0.05263157894736842, 0.38235294117647056, 'entropy = 0.0\nsamples = 5\nvalue =
[0, 5, 0']),
Text(0.06015037593984962, 0.4411764705882353, 'entropy = 0.0\nsamples = 9\nvalue =
[0, 9, 0']),
Text(0.08270676691729323, 0.5, 'x[1] <= 20.5\nentropy = 0.98\nsamples = 12\nvalue =
[7, 5, 0']),
Text(0.07518796992481203, 0.4411764705882353, 'x[0] <= 69.5\nentropy =
1.0\nsamples = 10\nvalue = [5, 5, 0']),
Text(0.06766917293233082, 0.38235294117647056, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0, 0']),
Text(0.08270676691729323, 0.38235294117647056, 'x[0] <= 76.0\nentropy =
0.954\nsamples = 8\nvalue = [3, 5, 0']),
Text(0.07518796992481203, 0.3235294117647059, 'entropy = 0.0\nsamples = 2\nvalue =
[0, 2, 0]),
Text(0.09022556390977443, 0.3235294117647059, 'x[1] <= 18.5\nentropy =
1.0\nsamples = 6\nvalue = [3, 3, 0']),
Text(0.08270676691729323, 0.264705882352941, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0, 0']),
Text(0.09774436090225563, 0.264705882352941, 'x[0] <= 81.0\nentropy =
0.811\nsamples = 4\nvalue = [1, 3, 0']),
Text(0.09022556390977443, 0.20588235294117646, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0, 0']),
Text(0.10526315789473684, 0.20588235294117646, 'entropy = 0.0\nsamples = 3\nvalue =
[0, 3, 0']),
Text(0.09022556390977443, 0.4411764705882353, 'entropy = 0.0\nsamples = 2\nvalue =
[2, 0, 0']),
Text(0.07518796992481203, 0.6176470588235294, 'entropy = 0.0\nsamples = 13\nvalue =
[0, 13, 0']),
Text(0.09774436090225563, 0.6764705882352942, 'x[1] <= 19.5\nentropy =
1.495\nsamples = 11\nvalue = [4, 2, 5']),
Text(0.09022556390977443, 0.6176470588235294, 'x[1] <= 14.0\nentropy =
0.863\nsamples = 7\nvalue = [0, 2, 5']),
Text(0.08270676691729323, 0.5588235294117647, 'entropy = 0.0\nsamples = 2\nvalue =
[0, 2, 0]),
Text(0.09774436090225563, 0.5588235294117647, 'entropy = 0.0\nsamples = 5\nvalue =
[0, 0, 5]),
Text(0.10526315789473684, 0.6176470588235294, 'entropy = 0.0\nsamples = 4\nvalue =
[4, 0, 0]),
Text(0.17528195488721804, 0.7941176470588235, 'x[0] <= 76.5\nentropy =
0.415\nsamples = 215\nvalue = [197, 18, 0])
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

16/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
Text(0.16776315789473684, 0.7352941176470589, 'entropy = 0.0\nsamples = 86\nvalue = [86, 0, 0]',  
Text(0.18280075187969924, 0.7352941176470589, 'x[3] <= 36.51\nentropy = 0.583\nsamples = 129\nvalue = [111, 18, 0]',  
Text(0.135338334586466165, 0.6764705882352942, 'x[1] <= 21.5\nentropy = 0.314\nsamples = 53\nvalue = [50, 3, 0]',  
Text(0.12030075187969924, 0.6176470588235294, 'x[3] <= 36.19\nentropy = 0.144\nsamples = 49\nvalue = [48, 1, 0]',  
Text(0.11278195488721804, 0.5588235294117647, 'x[1] <= 18.5\nentropy = 0.65\nsamples = 6\nvalue = [5, 1, 0]',  
Text(0.10526315789473684, 0.5, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0, 0]',  
Text(0.12030075187969924, 0.5, 'x[2] <= 0.03\nentropy = 0.811\nsamples = 4\nvalue = [3, 1, 0]',  
Text(0.11278195488721804, 0.4411764705882353, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0, 0]',  
Text(0.12781954887218044, 0.4411764705882353, 'x[1] <= 20.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1, 0]',  
Text(0.12030075187969924, 0.38235294117647056, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]',  
Text(0.135338334586466165, 0.38235294117647056, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0, 0]',  
Text(0.12781954887218044, 0.5588235294117647, 'entropy = 0.0\nsamples = 43\nvalue = [43, 0, 0]',  
Text(0.15037593984962405, 0.6176470588235294, 'x[0] <= 90.5\nentropy = 1.0\nsamples = 4\nvalue = [2, 2, 0]',  
Text(0.14285714285714285, 0.5588235294117647, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2, 0]',  
Text(0.15789473684210525, 0.5588235294117647, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0, 0]',  
Text(0.23026315789473684, 0.6764705882352942, 'x[0] <= 97.5\nentropy = 0.717\nsamples = 76\nvalue = [61, 15, 0]',  
Text(0.212406015037594, 0.6176470588235294, 'x[2] <= 0.225\nentropy = 0.602\nsamples = 68\nvalue = [58, 10, 0]',  
Text(0.20488721804511278, 0.5588235294117647, 'x[2] <= 0.075\nentropy = 0.569\nsamples = 67\nvalue = [58, 9, 0]',  
Text(0.17669172932330826, 0.5, 'x[2] <= 0.045\nentropy = 0.764\nsamples = 36\nvalue = [28, 8, 0]',  
Text(0.15789473684210525, 0.4411764705882353, 'x[0] <= 89.5\nentropy = 0.605\nsamples = 27\nvalue = [23, 4, 0]',  
Text(0.15037593984962405, 0.38235294117647056, 'entropy = 0.0\nsamples = 11\nvalue = [11, 0, 0]',  
Text(0.16541353383458646, 0.38235294117647056, 'x[0] <= 95.5\nentropy = 0.811\nsamples = 16\nvalue = [12, 4, 0]',  
Text(0.15789473684210525, 0.3235294117647059, 'x[3] <= 36.63\nentropy = 0.918\nsamples = 12\nvalue = [8, 4, 0]',  
Text(0.15037593984962405, 0.2647058823529412, 'x[0] <= 94.5\nentropy = 0.971\nsamples = 10\nvalue = [6, 4, 0]',  
Text(0.135338334586466165, 0.20588235294117646, 'x[3] <= 36.56\nentropy = 1.0\nsamples = 4\nvalue = [2, 2, 0]',  
Text(0.12781954887218044, 0.14705882352941177, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0, 0]',  
Text(0.14285714285714285, 0.14705882352941177, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2, 0]',  
Text(0.16541353383458646, 0.20588235294117646, 'x[3] <= 36.56\nentropy = 0.918\nsamples = 6\nvalue = [4, 2, 0]',  
Text(0.15789473684210525, 0.14705882352941177, 'x[1] <= 20.5\nentropy = 0.918\nsamples = 3\nvalue = [1, 2, 0]',  
Text(0.15037593984962405, 0.08823529411764706, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]',  
Text(0.16541353383458646, 0.08823529411764706, 'x[1] <= 21.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1, 0]',  
Text(0.15789473684210525, 0.029411764705882353, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0, 0]',  
Text(0.17293233082706766, 0.029411764705882353, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]',  
Text(0.17293233082706766, 0.14705882352941177, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0, 0]',  
Text(0.16541353383458646, 0.2647058823529412, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0, 0]',  
Text(0.17293233082706766, 0.3235294117647059, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0, 0]',  
Text(0.19548872180451127, 0.4411764705882353, 'x[0] <= 78.5\nentropy = 0.991\nsamples = 9\nvalue = [5, 4, 0]',  
Text(0.18796992481203006, 0.38235294117647056, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]',  
Text(0.20300751879699247, 0.38235294117647056, 'x[0] <= 96.0\nentropy = 0.954\nsamples = 8\nvalue = [5, 3, 0]',  
Text(0.19548872180451127, 0.3235294117647059, 'x[1] <= 20.5\nentropy = 0.863\nsamples = 7\nvalue = [5, 2, 0]',  
Text(0.18796992481203006, 0.2647058823529412, 'x[0] <= 82.0\nentropy = 1.0\nsamples = 4\nvalue = [2, 2, 0]',  
Text(0.18045112781954886, 0.20588235294117646, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0, 0]',  
Text(0.19548872180451127, 0.20588235294117646, 'x[3] <= 36.6\nentropy = 0.918\nsamples = 3\nvalue = [1, 2, 0]',  
Text(0.18796992481203006, 0.14705882352941177, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]',  
Text(0.20300751879699247, 0.14705882352941177, 'entropy = 1.0\nsamples = 2\nvalue = [1, 1, 0]',  
Text(0.20300751879699247, 0.2647058823529412, 'entropy = 0.0\nsamples = 3\nvalue =
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

17/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
[3, 0, 0]',  
Text(0.21052631578947367, 0.3235294117647059, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.23308270676691728, 0.5, 'x[3] <= 36.7\nentropy = 0.206\nsamples = 31\nvalue  
= [30, 1, 0]',  
Text(0.22556390977443608, 0.4411764705882353, 'entropy = 0.0\nsamples = 25\nvalue  
= [25, 0, 0]',  
Text(0.24060150375939848, 0.4411764705882353, 'x[0] <= 83.0\nentropy =  
0.65\nsamples = 6\nvalue = [5, 1, 0]',  
Text(0.23308270676691728, 0.38235294117647056, 'x[2] <= 0.125\nentropy =  
1.0\nsamples = 2\nvalue = [1, 1, 0]',  
Text(0.22556390977443608, 0.3235294117647059, 'entropy = 0.0\nsamples = 1\nvalue =  
[1, 0, 0]',  
Text(0.24060150375939848, 0.3235294117647059, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.24812030075187969, 0.38235294117647056, 'entropy = 0.0\nsamples = 4\nvalue  
= [4, 0, 0]',  
Text(0.2199248120300752, 0.5588235294117647, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.2199248120300752, 0.5588235294117647, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 0, 1]',  
Text(0.24812030075187969, 0.6176470588235294, 'x[0] <= 99.5\nentropy =  
0.954\nsamples = 8\nvalue = [3, 5, 0]',  
Text(0.24060150375939848, 0.5588235294117647, 'entropy = 0.0\nsamples = 3\nvalue =  
[0, 3, 0]',  
Text(0.2556390977443609, 0.5588235294117647, 'x[1] <= 19.5\nentropy =  
0.971\nsamples = 5\nvalue = [3, 2, 0]',  
Text(0.24812030075187969, 0.5, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0, 0]',  
Text(0.2631578947368421, 0.5, 'x[0] <= 100.5\nentropy = 0.918\nsamples = 3\nvalue  
= [1, 2, 0]',  
Text(0.2556390977443609, 0.4411764705882353, 'entropy = 0.0\nsamples = 1\nvalue =  
[1, 0, 0]',  
Text(0.270676691729323, 0.4411764705882353, 'entropy = 0.0\nsamples = 2\nvalue =  
[0, 2, 0]',  
Text(0.5075187969924813, 0.8529411764705882, 'x[3] <= 36.36\nentropy =  
1.279\nsamples = 265\nvalue = [61, 170, 34]',  
Text(0.4323308270676692, 0.7941176470588235, 'x[3] <= 36.19\nentropy =  
1.44\nsamples = 181\nvalue = [58, 94, 29]',  
Text(0.37218045112781956, 0.7352941176470589, 'x[2] <= 0.055\nentropy =  
1.323\nsamples = 140\nvalue = [25, 88, 27]',  
Text(0.31954887218045114, 0.6764705882352942, 'x[1] <= 18.5\nentropy =  
1.022\nsamples = 101\nvalue = [22, 74, 5]',  
Text(0.2932330827067669, 0.6176470588235294, 'x[0] <= 191.0\nentropy =  
0.967\nsamples = 33\nvalue = [20, 13, 0]',  
Text(0.2857142857142857, 0.5588235294117647, 'x[2] <= 0.015\nentropy =  
0.894\nsamples = 29\nvalue = [20, 9, 0]',  
Text(0.2781954887218045, 0.5, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4, 0]',  
Text(0.2932330827067669, 0.5, 'x[0] <= 104.5\nentropy = 0.722\nsamples = 25\nvalue  
= [20, 5, 0]',  
Text(0.2857142857142857, 0.4411764705882353, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.3007518796992481, 0.4411764705882353, 'x[0] <= 118.0\nentropy =  
0.65\nsamples = 24\nvalue = [20, 4, 0]',  
Text(0.2932330827067669, 0.38235294117647056, 'entropy = 0.0\nsamples = 9\nvalue =  
[9, 0, 0]',  
Text(0.3082706766917293, 0.38235294117647056, 'x[0] <= 123.0\nentropy =  
0.837\nsamples = 15\nvalue = [11, 4, 0]',  
Text(0.3007518796992481, 0.3235294117647059, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.3157894736842105, 0.3235294117647059, 'x[0] <= 140.5\nentropy =  
0.75\nsamples = 14\nvalue = [11, 3, 0]',  
Text(0.3082706766917293, 0.2647058823529412, 'entropy = 0.0\nsamples = 3\nvalue =  
[3, 0, 0]',  
Text(0.3233082706766917, 0.2647058823529412, 'x[3] <= 36.08\nentropy =  
0.845\nsamples = 11\nvalue = [8, 3, 0]',  
Text(0.3082706766917293, 0.20588235294117646, 'x[2] <= 0.025\nentropy =  
0.544\nsamples = 8\nvalue = [7, 1, 0]',  
Text(0.3007518796992481, 0.14705882352941177, 'x[0] <= 168.0\nentropy =  
1.0\nsamples = 2\nvalue = [1, 1, 0]',  
Text(0.2932330827067669, 0.08823529411764706, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.3082706766917293, 0.08823529411764706, 'entropy = 0.0\nsamples = 1\nvalue =  
[1, 0, 0]',  
Text(0.3157894736842105, 0.14705882352941177, 'entropy = 0.0\nsamples = 6\nvalue =  
[6, 0, 0]',  
Text(0.3383458646616541, 0.20588235294117646, 'x[0] <= 158.0\nentropy =  
0.918\nsamples = 3\nvalue = [1, 2, 0]',  
Text(0.3308270676691729, 0.14705882352941177, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.3458646616541353, 0.14705882352941177, 'x[0] <= 171.0\nentropy =  
1.0\nsamples = 2\nvalue = [1, 1, 0]',  
Text(0.3383458646616541, 0.08823529411764706, 'entropy = 0.0\nsamples = 1\nvalue =  
[1, 0, 0]',  
Text(0.3533834586466165, 0.08823529411764706, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 1, 0]',  
Text(0.3007518796992481, 0.5588235294117647, 'entropy = 0.0\nsamples = 4\nvalue =  
[0, 4, 0]',  
Text(0.3458646616541353, 0.6176470588235294, 'x[2] <= 0.035\nentropy =  
0.567\nsamples = 68\nvalue = [2, 61, 5]',  
Text(0.3308270676691729, 0.5588235294117647, 'x[1] <= 20.5\nentropy =  
1.352\nsamples = 9\nvalue = [1, 3, 5]',  
Text(0.3308270676691729, 0.5588235294117647, 'entropy = 0.0\nsamples = 1\nvalue =  
[0, 0, 1]')
```

8/6/23, 21:48

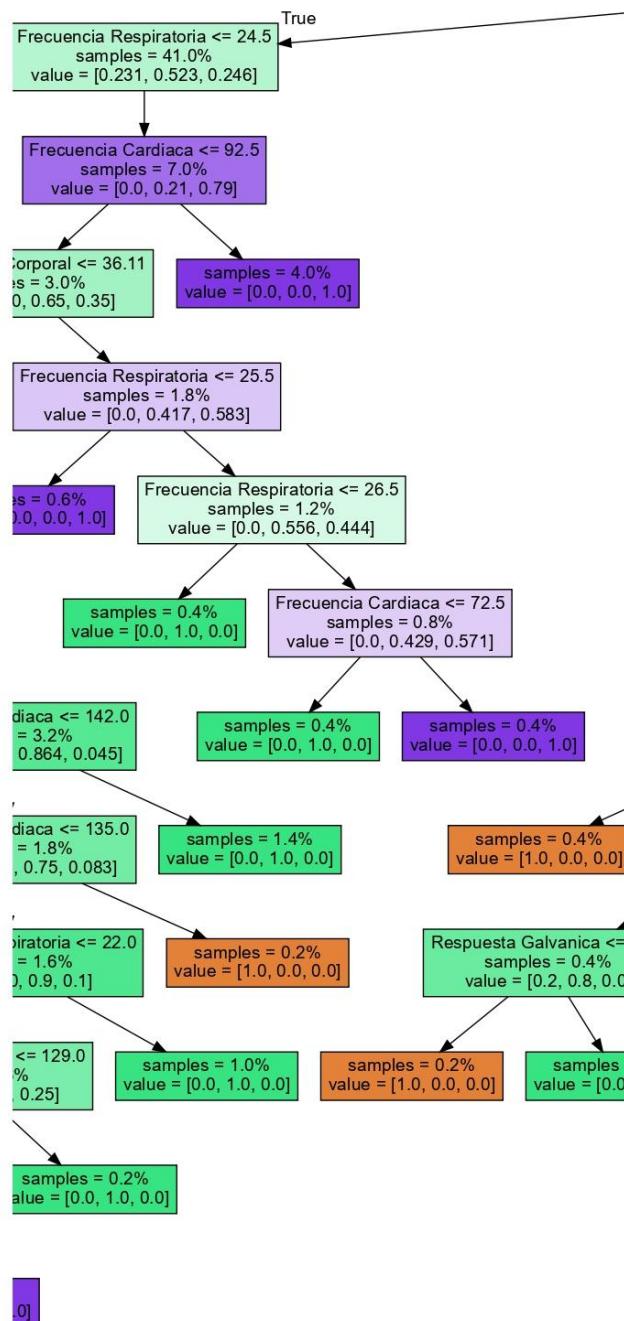
Lepasy.ipynb - Colaboratory

RandomForest

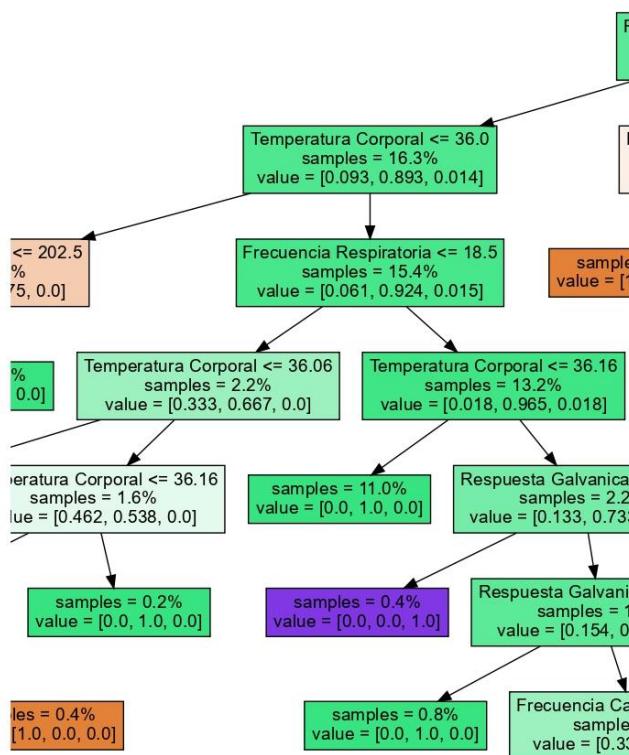
```
Text(0.3308270676691729, 0.4411764705882353, 'entropy = 0.0\nsamples = 5\nvalue =\nfrom sklearn.ensemble import RandomForestClassifier\nfrom sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay\n    0.124\nsamples = 59\nvalue = [1. 58. 0]\').\nrf = RandomForestClassifier()\nrft(X_train, y_train)\n<ipython-input-80-efede872133b>:2: DataConversionWarning: A column-vector y was pass\n    rf.fit(X_train, y_train)\n      RandomForestClassifier()\n      RandomForestClassifier()\n\ny_pred = rf.predict(X_test)\n\naccuracy = accuracy_score(y_test, y_pred)\nprint("Accuracy:", accuracy)\n\nAccuracy: 0.8656716417910447\n\nimport graphviz\nfrom sklearn.tree import export_graphviz\nfor i in range(3):\n    tree = rf.estimators_[i]\n    dot_data = export_graphviz(tree,\n        feature_names=X_train.columns,\n        filled=True,\n        max_depth=20,\n        impurity=False,\n        proportion=True)\ngraph = graphviz.Source(dot_data)\ndisplay(graph)
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

19/24

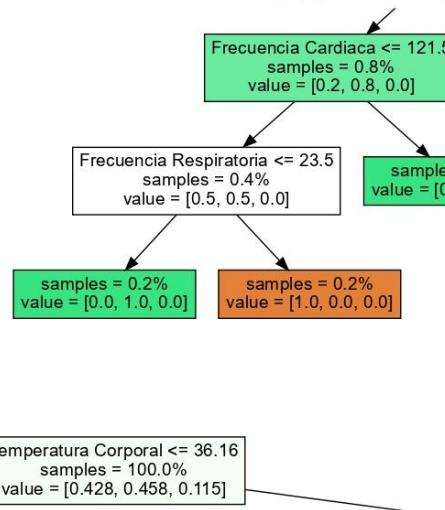


```
Frecuencia Cardiaca <= 96.5
samples = 100.0%
value = [0.396, 0.482, 0.121]
```

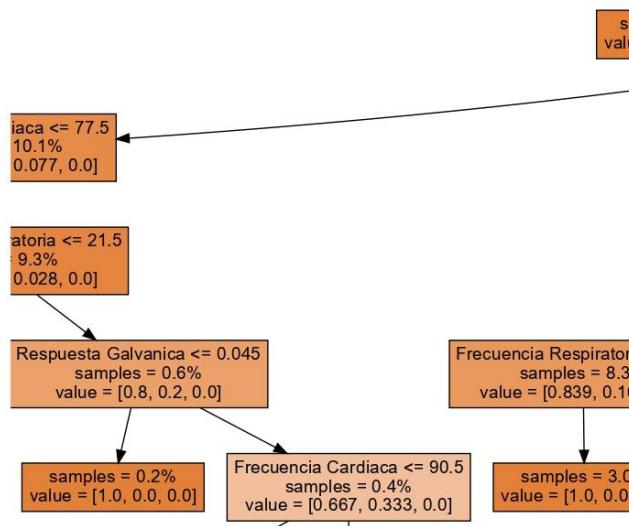


8/6/23, 21:48

Lepasy.ipynb - Colaboratory



Temperatura Corporal <= 36.16
samples = 100.0%
value = [0.428, 0.458, 0.115]



```

from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
  
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfnNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

22/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
from sklearn.neural_network import MLPClassifier
import graphviz

cart = DecisionTreeClassifier(random_state=99)
cart

DecisionTreeClassifier
DecisionTreeClassifier(random_state=99)

cart.fit(X_train, y_train)

DecisionTreeClassifier
DecisionTreeClassifier(random_state=99)

cart.score(X_train, y_train)

0.99875

cart.score(X_test, y_test)

0.8159203980099502

Validación cruzada

validacioncruzada1=cross_val_score(cart,X_train, y_train,cv=5)
validacioncruzada1.mean()

0.80625

validacioncruzada2=cross_val_score(cart, X_test, y_test, cv=5)
validacioncruzada2.mean()

0.786219512195122
```

Algoritmo KNN

```
knn=KNeighborsClassifier(n_neighbors=5, metric='hamming')
knn.fit(X_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:215: Da
    return self._fit(X, y)
KNeighborsClassifier
KNeighborsClassifier(metric='hamming')

knn.score(X_train, y_train)

0.76625

knn.score(X_test,y_test)

0.6567164179104478
```

Algoritmo Perceptrón MultiCapa

```
mlp=MLPClassifier(random_state=99, hidden_layer_sizes=(100,),max_iter=200)
mlp.fit(X_train, y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:110: UserWarning: The input data contains NaN values which will be removed before training.
y = column_or_1d(y, warn=True)
MLPClassifier
MLPClassifier(random_state=99)

mlp.score(X_train, y_train)

0.68
```

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

23/24

8/6/23, 21:48

Lepasy.ipynb - Colaboratory

```
mlp.score(X_test, y_test)
0.6417910447761194

mlpsig=MLPClassifier(random_state=99, hidden_layer_sizes=(100,), max_iter=200,activation='logistic')
mlpsig.fit(X_train, y_train)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py: column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py: warnings.warn(
    MLPClassifier
MLPClassifier(activation='logistic', random_state=99)

mlpsig.score(X_train, y_train)
0.7075

mlpsig.score(X_test, y_test)
0.6915422885572139
```

✓ 1 s completado a las 19:10

<https://colab.research.google.com/drive/1VY4UZFMcq8bE3XwYO54NfNb8sV6oHPtx?hl=es#scrollTo=GgvDOJ9tFQfD&printMode=true>

24/24

X.II. PROGRAMACIÓN EN ENTORNO WEKA

Weka Explorer

Preprocess Classify Cluster Associate Selected attributes Visualize Auto-WEKA

Classifier

Choose: `IBK-K1-W0-A\weka.core.neighboursearch.LinearNNGeearch-A\weka.core.EuclideanDistance -R firstIco?"`

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 5
- Percentage split % 66
- More options...

(Nom) Nivel Dolor

Start Stop

Result list (right-click for options)

- 19:39:24 - trees.J48
- 19:39:45 - trees.RandomForest
- 19:39:59 - lazy.IBK

Classifier output

```
==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances      802      80.1190 %
Incorrectly Classified Instances   198      19.8801 %
Kappa statistic                   0.6694
Mean absolute error               0.175
Root mean squared error           0.332
Relative absolute error            43.3801 %
Root relative squared error      71.7144 %
Total Number of Instances        1000

==== Detailed Accuracy By Class ====

```

	IF Rate	FF Rate	Precision	Recall	F-Measure	MCC	ROC Area	FRC Area	Class
0,829	0,162	0,851	0,829	0,840	0,729	0,916	0,682		Bajo
0,826	0,153	0,790	0,826	0,805	0,641	0,571	0,651		Medio
0,629	0,045	0,680	0,629	0,654	0,604	0,596	0,589		Alto
Weighted Avg.	0,801	0,131	0,801	0,801	0,673	0,893	0,621		

```
==== Confusion Matrix ====

```

a	b	c	-- classified as
541	40	11	a = Bajo
51	376	28	b = Medio
9	40	63	c = Alto

Weka Explorer

Preprocess Classify Cluster Associate Selected attributes Visualize Auto-WEKA

Classifier

Choose: `IBK-K1-W0-A\weka.core.neighboursearch.LinearNNGeearch-A\weka.core.EuclideanDistance -R firstIco?"`

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 5
- Percentage split % 66
- More options...

(Nom) Nivel Dolor

Start Stop

Result list (right-click for options)

- 19:39:24 - trees.J48
- 19:39:45 - trees.RandomForest
- 19:39:59 - lazy.IBK

Classifier output

```
==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances      738      73.7263 %
Incorrectly Classified Instances   262      26.2737 %
Kappa statistic                   0.5602
Mean absolute error               0.1761
Root mean squared error           0.4177
Relative absolute error            43.4581 %
Root relative squared error      93.0269 %
Total Number of Instances        1001

==== Detailed Accuracy By Class ====

```

	IF Rate	FF Rate	Precision	Recall	F-Measure	MCC	ROC Area	FRC Area	Class
0,740	0,186	0,755	0,740	0,769	0,606	0,809	0,698		Bajo
0,760	0,207	0,754	0,760	0,757	0,553	0,783	0,702		Medio
0,492	0,047	0,613	0,492	0,546	0,490	0,736	0,592		Alto
Weighted Avg.	0,737	0,177	0,734	0,737	0,734	0,664	0,707	0,659	

```
==== Confusion Matrix ====

```

a	b	c	-- classified as
327	77	10	a = Bajo
78	346	31	b = Medio
31	36	65	c = Alto

Weka Explorer

Preprocess Classify Cluster Associate Selected attributes Visualize Auto-WEKA

Classifier

Choose: `IBK-K1-W0-A\weka.core.neighboursearch.LinearNNGeearch-A\weka.core.EuclideanDistance -R firstIco?"`

Test options

- Use training set
- Supplied test set Set...
- Cross-validation Folds 5
- Percentage split % 66
- More options...

(Nom) Nivel Dolor

Start Stop

Result list (right-click for options)

- 19:39:24 - trees.J48
- 19:39:45 - trees.RandomForest
- 19:39:59 - lazy.IBK

Classifier output

```
==== Stratified Cross-Validation ====
==== Summary ====
Correctly Classified Instances      639      63.8162 %
Incorrectly Classified Instances   162      16.1838 %
Kappa statistic                   0.7006
Mean absolute error               0.159
Root mean squared error           0.2815
Relative absolute error            39.4158 %
Root relative squared error      62.6613 %
Total Number of Instances        1001

==== Detailed Accuracy By Class ====

```

	IF Rate	FF Rate	Precision	Recall	F-Measure	MCC	ROC Area	FRC Area	Class
0,894	0,097	0,867	0,894	0,880	0,793	0,958	0,641		Bajo
0,840	0,137	0,836	0,840	0,838	0,702	0,928	0,921		Medio
0,659	0,035	0,744	0,659	0,659	0,650	0,557	0,600		Alto
Weighted Avg.	0,838	0,107	0,836	0,838	0,837	0,734	0,544	0,613	

```
==== Confusion Matrix ====

```

a	b	c	-- classified as
370	38	6	a = Bajo
49	382	24	b = Medio
8	37	67	c = Alto