Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Ciencias

Detección y clasificación de infartos mediante técnicas de Inteligencia Artificial

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias

Presenta:

Ing. Elias Gabriel Nápoles Ramos

Dirigido por:

Dr. Marcos Romo Avilés

SINODALES

Dr. Marcos Romo Avilés
Presidente

M. en C. José Luis Avendaño Juarez
Codirector

Dr. José Manuel Alvarez Alvarado
Vocal

Dr. Suresh Thenozhi
Suplente

Dr. Roberto Valentín Carrillo Serrano
Suplente

Centro Universitario
Querétaro, QRO
México.
Mayo 2026

Universidad Autónoma de Querétaro

Facultad de Ingeniería
Maestría en Ciencias

Detección y clasificación de infartos mediante técnicas de Inteligencia Artificial

Tesis

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias

Presenta:
Ing. Elias Gabriel Nápoles Ramos
Dirigido por:
Dr. Marcos Romo Avilés
Co-Director:
M. en C. José Luis Avendaño Juarez
Vocal:
Dr. José Manuel Alvarez Alvarado
Suplente
Dr. Suresh Thenozhi
Suplente
Dr. Roberto Valentín Carrillo Serrano

Querétaro, Qro. a Mayo de 2026

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Resumen

Actualmente, una de las principales causas de muerte a nivel global son las Enfermedades Cardiovasculares, dentro de estas enfermedades se tienen el Síndrome Coronario Agudo sin Elevación del Segmento ST y el Infarto Agudo al Miocardio con Elevación del Segmento ST. En el año 2021 fallecieron aproximadamente 177 mil personas a nivel nacional por estas enfermedades. El Electrocardiograma es la herramienta principal para la detección de estas enfermedades, sin embargo, la interpretación del mismo queda a las capacidades del personal de salud. Por lo anterior se desarrolló este proyecto, siendo el objetivo principal desarrollar dos modelos de Inteligencia Artificial (Máquinas de Soporte Vectorial y Red Neuronal Recurrente) para la detección de Síndrome Coronario Agudo sin Elevación del Segmento ST e Infarto Agudo al Miocardio con Elevación del Segmento ST mediante características extraídas de una señal de un Electrocardiograma. Buscando modelos ligeros con la capacidad de ser ejecutados dentro de una *Raspberry Pi B4*. Los resultados muestran una gran versatilidad en el uso de recursos computacionales y una alta taza de acertividad en la detección y clasificación de estas enfermedades, siendo una precisión de 89 % para la Máquina de Soporte Vectorial y 91.38 % para la Red Neuronal Recurrente. A pesar de los resultados se encuentran áreas de oportunidad en el tiempo de inferencia de los modelos, y el uso de memoria RAM por parte del sistema embebido.

CAPÍTULO 2

Agradecimientos

Primeramente agradeciendo a la Universidad Autónoma de Querétaro y a la Facultad de Ingeniería por las facilidades en espacio de trabajo, recursos literarios y recursos económicos para lograr este proyecto.

Posteriormente se agradece a todo el personal que conforma el cuerpo de sínodo que sin su guía, consejos, sugerencias y conocimiento el proyecto no hubiese concluido de manera satisfactoria.

Finalmente, se mantiene presente el apoyo familiar con especial dedicación a mi padre Benjamín Nápoles Hernández que fue un pilar durante toda mi trayectoria profesional, personal y educativa.

CAPÍTULO 3

Introducción

Las Enfermedades Cardiovasculares (EC) son una de las principales causas de muertes a nivel mundial, catalogadas como uno de los mayores problemas en salud pública. Están estrechamente relacionadas con otras enfermedades y malos hábitos como lo pueden ser el tabaquismo, obesidad, hipertensión arterial, entre otras [1].

Dentro de las EC se encuentra el Síndrome Coronario Agudo (SCA), mismo que tiene otras subclasificaciones de las cuales dos de gran relevancia son el Síndrome Coronario Agudo Con Elevación del segmento ST (SCACEST), también conocido como Infarto Agudo al Miocardio con Elevación del segmento ST (IMEST) y el Síndrome Coronario Agudo Sin Elevación del Segmento ST (SCA-SEST). Estas enfermedades son de importancia médica debido a que son predecesoras al paro cardiaco [2].

El término SCA es utilizado para referirse a un conjunto de síntomas provocados por una isquemia en el miocardio, misma que puede ser el resultado de un mal funcionamiento eléctrico del corazón, según sea el fallo eléctrico puede referirse a un SCA-SEST o a un SCACEST. Coloquialmente estos términos han sido generalizados y referidos a ellos como Infarto Agudo al Miocardio (IAM) [3].

Diversos estudios estiman que en el mundo ocurre un IAM cada 4 segundos englobando personas sin enfermedades relacionadas al mismo [4]. El IAM es mencionado como problema epidemiológico mayor, esto mismo conlleva a una necesidad primordial de mejorar el pronóstico de esta enfermedad [5].

En países como Estados Unidos y España, el interés de salud pública ha priorizado la detección oportuna del SCA a la población, lo cual ha generado una disminución en la tasa de mortalidad ocasionada por esta enfermedad [6]. Se han realizado estudios en España donde prevén que las tendencias de edades más vulnerables para el 2049 serán desde los 25 años hasta los 74 años [7].

Para el diagnóstico de un SCA la toma de un Electrocardiograma (ECG) de doce derivaciones es la herramienta principal para el diagnóstico de un paciente con sospecha de SCA, debido a que pueden ser apreciables las carascterísticas de las señales miocárdicas (punto Q, pico R, punto S, onda T). Existen otras herramientas como *Vancouver chest pain rule* e *INTERCHEST*, sin embargo, la más confiable es el ECG de doce derivadas debido a que cuenta con una especificidad del 91 % y sensibilidad del 32 % para el diagnóstico del SCA. El tiempo es un factor clave ya que el ECG debe tomarse en los primeros 10 minutos después de realizar contacto con personal médico o para-médico y ser interpretado por un médico debidamente calificado [8].

Hoy en día se utiliza Inteligencia Artificial (IA) para tomar decisiones como lo haría un ser

humano, a pesar de que no hay una definición exacta para IA se puede entender como la capacidad de una computadora para ejecutar algoritmos matemáticos, aprender y tomar decisiones en base a lo aprendido. Dentro de la IA tenemos el Aprendizaje Automático (Machine Learning, ML de sus siglas en inglés) y dentro del ML tenemos un subcampo llamado Aprendizaje Profundo (Deep Learning, DL de sus siglas en inglés) [9].

Las Máquinas de Soporte Vectorial (SVM, de sus siglas en inglés) es una técina de ML utilizadas en clasificación binaria, clasificación múltiple y regresión. Esta ténica cuenta con un sustento matemático basado en separadores lineales, también llamados hiperplanos, gracias a esto es una de las técnicas de ML más utilizadas en aplicaciones de clasificación [10].

La base del DL son las Redes Neuronales Artificiales (RNA), las cuales son un modelo matemático computacional que busca emular el comportamiento de aprendizaje biológico de una neurona, actualmente existen distintas estructuras de las RNA siendo de las más utilizadas las Redes Neuronales Convolucionales (CNN), Redes Neuronales Recurrentes (RNN) y Redes Generativas Antagónicas [11].

Por su parte, las RNN son bastante utilizadas cuando se busca obtener y procesar datos secuenciales como en una señal continua variable en el tiempo, es por esta razón que son utilizadas mayormente para el análisis de video, imágenes, procesamiento de audio [12].

Durante este trabajo se proponen entrenar 2 modelos de IA (SVM y RNN) para la detección y clasificación de SCA-SEST e IMEST. Para ello, se emplean características extraídas en el dominio del tiempo, obteniendo métricas de desempeño que permiten evaluar el comportamiento de cada modelo. Ambos modelos son implementados en un sistema embebido (*Raspberry pi 4B*), donde además se realizan mediciones del consumo de recursos de hardware.

3.1. Justificación

Las herramientas tecnológicas juegan un papel importante en la detección de SCA, por lo que tener el apoyo de nuevas tecnologías es una necesidad en la actualidad. Los algoritmos de IA ofrecen un apoyo para aumentar la precisión y sensibilidad en interpretación de las señales miocárdicas [3].

Estadísticas de la Secretaría de Salud de México muestran que en el año 2021 fallecieron alrededor de 220 mil personas debido a distintas enfermedades cardiovasculares, dentro de las cuales 177 mil fueron por un IAM [13].

Contextualizando en el estado de Querétaro una de las principales causas de muerte es el paro cardiorespiratorio, esta fisiopatología afecta al 17 % de la población por lo que detectar de manera oportuna es de suma importancia para la supervivencia de los pacientes [14]. Previo a un paro cardiorespiratorio se puede tener un SCA, este síndrome abarca distintas fisiopatologías como lo son Fibrilación Auricular (FV), Taquicardia Ventricular (TV), SCA-SEST e IAM. La detección oportuna de un SCA es de suma importancia para aumentar el porcentaje de supervivencia.

En la Figura 3.1 (presentada en inglés debido a las fuentes de referencia) se puede observar una red bibliométrica donde se aprecia que las CNN tienen una relación fuerte con el SCA lo que significa que es una técnica bastante utilizada en la detección de esta condición médica. Esta técnica ha otorgado resultados prometedores durante simulación, sin embargo, solo ha sido experimentada con una salida binaria. Por lo que se propone realizar un modelo de detección y multiclasificación de SCA mediante la lectura de la señal del corazón buscando obtener una mayor precisión y sensibilidad.

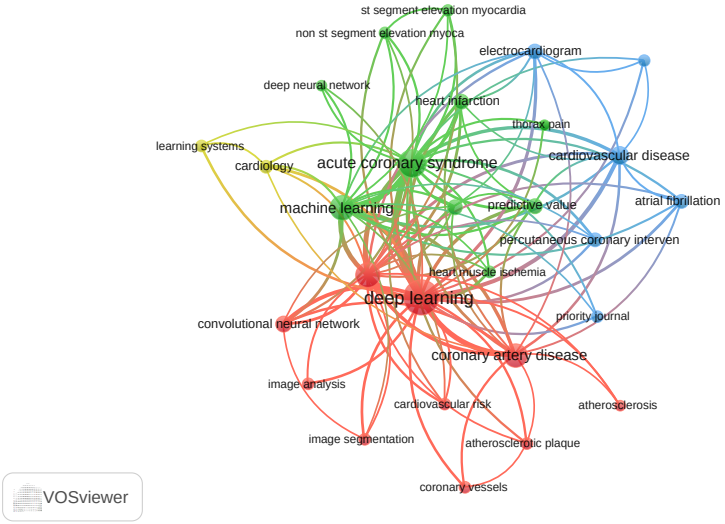

Figura 3.1: Red bibliométrica de coincidencias (presentada en inglés debido a las fuentes de referencia).

3.2. Descripción del problema

El SCA es una condición médica crítica, la cual debe ser tratada lo antes posible, sin embargo, es indispensable detectar los signos y síntomas de esta condición para realizar un diagnóstico acertado, considerando el escaso personal capacitado para detectar estos signos, aunado a factores humanos tales como emociones, estrés y experiencia muestran un área de oportunidad [15].

Para la detección de SCA se han realizado distintos estudios de manera intrahospitalaria otorgando resultados prometedores con una precisión de 93 % con el uso de las CNN. Otros estudios han realizado la evaluación de una manera simulada obteniendo resultados de 99 % en precisión. Ambos estudios han procesado las 12 derivaciones ECG realizando una clasificación binaria entre pacientes con SCA y pacientes sanos, al utilizar las 12 derivaciones del corazón implica una mayor capacidad computacional y tiempo de procesamiento [16, 17].

Debido a lo comentado en el parrafo anterior se propone embeber dos modelos de IA entrenados específicamente para detectar SCA-SEST y SCACEST, teniendo como entrada características en el dominio del tiempo obtenidas de la segunda derivada del sistema eléctrico del corazón, considerando siempre la menor carga computacional posible para el sistema embebido.

3.3. Hipótesis

Las Máquinas de Soporte Vectorial y Redes Neuronales Recurrentes, mediante los parámetros Complejo QRS, segmento ST, Frecuencia cardiaca y valor del punto T, tienen una precisión mayor del 84 % de señales provenientes de una base de datos pública, para la detección y clasificación de infartos.

3.4. Objetivo general

Desarrollar una SVM y una RNN en un sistema embebido para la detección y clasificación de infartos evaluando su precisión, sensibilidad y eficacia, mediante los parámetros Complejo QRS, segmento ST, Frecuencia Cardiaca y el valor del punto T.

3.5. Objetivos particulares

1. Obtener las características deseadas de las señales provenientes de bases de datos mediante procedimientos matemáticos para generar un vector de características.

2. Procesar las características obtenidas realizando una estandarización y normalización de los datos para homegenizar los datos.

3. Generar una nueva base de datos con la obtención de las características y su procesamiento para el entrenamiento de modelos de IA.

4. Diseñar los modelos a utilizar para SVM y RNN a partir de las características para su evaluación.

5. Evaluar el rendimiento de ambos modelos fuera de línea con el sistema embebido para comparar los modelos con un análisis estadístico de su precisión y sensibilidad.

3.6. Antecedentes

Un modelo utilizado en aplicaciones similares es la SVM la cual ha demostrado un desempeño sólido en tareas de clasificación, tanto binaria como multiclase, relacionadas con la detección de SCA [18, 19, 20]. En [21], se llevó a cabo un estudio comparando diferentes modelos de aprendizaje automático para la clasificación binaria de pacientes con IMEST sin considerar señales de ECG. El modelo con mejor rendimiento fue una SVM, alcanzando una exactitud del 99.9 %, una precisión del 99.8 % y una puntuación F1 del 99.9 %. Es importante destacar que se emplearon el muestreo de reservorio (reservoir sampling, en inglés) y la Técnica de Sobremuestreo de Minorías Sintéticas (SMOTE, de sus siglas en inglés) para mejorar el equilibrio y la calidad del conjunto de datos. Este enfoque se ha explorado utilizando bases de datos de una sola derivación, así como las doce derivaciones proporcionadas por la base de datos de ECG."

En [22] se realizó un estudio de modelo de predicción para clasificación de SCA siendo una clasificación multiple. Este estudio utilizó una base de datos pública teniendo como parámetros de entrada algunos síntomas del paciente y parámetros químicos del cuerpo humano, dando como resultado una precisión del 98.9 % exactitud 98.9 % F1-Score 98.9 %. Lo anterior utilizando la técnica de bosques aleatorios (Random Forest).

Por otro lado, [23] realizó un estudio comparativo entre las RNA y las SVM para la clasificación binaria de un paciente con posible SCA, resaltando que la SVM tuvo una mayor tasa de éxito en la clasificación siendo de 99.13 % en comparación con la RNA teniendo esta última una tasa de éxito de 90.1 %. Teniendo como valores de entrada una combinación de síntomas con información clínica. El autor muestra trabajos previos teniendo resultados similares en la comparación de estas dos técnicas de IA.

En [24], realizó un trabajo teniendo un enfoque en detección de infartos mediante SVM y el uso de las 12 señales del ECG, utilizando algoritmos genéticos (selección natural) encontró hiperparámetros adecuados para su aplicación, teniendo como mejores resultados una sensibilidad de 86.82 % una especificidad de 91.05 %. El trabajo muestra una compración entre el modelo de la SVM y la clasificación por dos cardiologos especialistas resaltando una mejor clasificación por parte de la SVM, sin embargo, el trabajo se enfoca exclusivamente en la detección de IMEST sin considerar factores externos.

Considerando que las señales ECG son bastante utilizadas para la detección de SCA [25] desarrolló una SVM con la información de las 12 derivaciones del ECG y realizando un análisis de componentes principales (PCA, por sus siglas en inglés) llegó a obtener una precisión del 99.33 %, su especificidad de 96.66 % y sensibilidad de 100 %. En el mismo trabajo se realizó otro experimento utilizando una algoritmo genético de optimización por enjambre de partículas obteniendo una precisión de 96.6 % al igual que su sensibilidad y especificidad.

En aplicaciones similares también han utilizado ténicas de DL como [26] quien utilizó RNN combinadas con algoritmos de optimización para una detección binaria de ritmos cardiacos anormales mediante las 12 derivaciones del ECG, encontrando precisiones superiores al 80 % con todos los algoritmos de optimización utilizados. También utilizó *recall* y *F1 score* como métricas de evaluación para todos los modelos entrenados.

Por su parte [27] realizó una investigación para la detección de arritmia cardiaca mediante señales ECG y modelos de DL. El estudio se enfoca en realizar una comparación entre los modelos RNN, GRU (de dsus siglas en inglés, Gate Recurrent Unit) y LSTM (de sus siglas en inglés, Long Short-Term Memory). Dentro del mismo trabajo se muestra que el modelo LSTM tiene los mejores resultados en métricas de precisión y sensibilidad, siendo 88.1 % y 92.4 % respectivamente.

El estudio realizado por [28] llevó a cabo una clasificación multiclase de enfermedades cardiovasculares utilizando una RNN, implementada dentro de un sistema embebido. A pesar de las limitaciones del hardware, los resultados fueron prometedores. Sin embargo, dicho trabajo solo utilizó intervalos de señales de ECG anotados por especialistas, sin considerar la extracción de características a partir de las señales de ECG.

En el Cuadro 3.1 se muestra una comparación rápida de los trabajos previos con objetivos similares al presente proyecto. En el mismo se puede apreciar de manera visual el modelo de IA utilizado, el tipo de clasificación utilizado para enfermedades cardiovazculares, implementación en sistema embebido y los resultados obtenidos.

Cuadro 3.1: Comparación entre el estado del arte.

Trabajo	**Modelo de IA**	**Clasificación**	**Sistema Embebido**	***Accuracy*** %
[21]	SVM	Binario	No	99.9
[24]	SVM	Binario	No	86.82
[25]	SVM	Multi clase	No	96.66
[23]	SVM	Multi clase	No	90.10
[28]	RNN (GRU)	Multi clase	Embebido (STM32)	86.64

CAPÍTULO 4

Fundamentación Teórica

En este capítulo se aterrizarán fundamentos científicos que darán soporte y comprensión de los temas cubiertos por el trabajo.

Comenzando por mencionar que el avance de la tecnología ha otorgado herramientas para el personal de salud. Dichas herramientas brindan un soporte visual para el diagnóstico de distintas enfermedades.

4.1. Anatomía del corazón

El corazón es un músculo el cual tiene un peso aproximado entre 200 y 425 gramos y se encuentra ubicado en la parte anterior de la parrilla torácica, entre los pulmones. La función principal del corazón es bombear sangre oxigenada por el torrente sanguíneo. Esto se realiza mediante contracciones musculares provocadas por un sistema eléctrico. Se puede dividir al corazón en cuatro cavidades principales, las dos primeras se llaman aurículas y se encuentran en la parte superior del corazón y los ventrículos que se encuentran en la parte inferior. Todas las cavidades están separadas mediante una pared muscular como se puede observar en la Figura 4.1 [3].

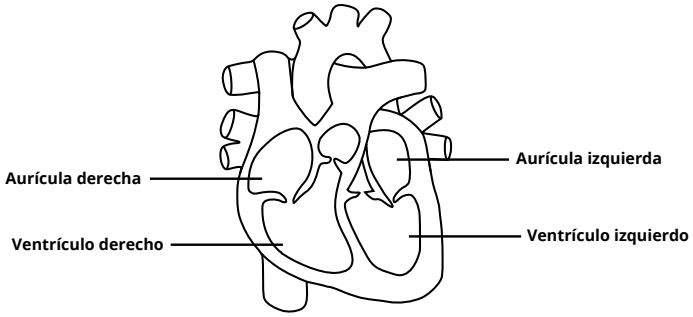

Figura 4.1: Anatomía básica del corazón.

A lo largo del corazón corre una red de tejido especializado que tiene la capacidad de conducir corriente eléctrica. El flujo de corriente eléctrica a través de esta red causa contracciones cardíacas uniformes y coordinadas, las cuales producen la acción de bombeo del corazón.

Cuando el corazón funciona con normalidad, el impulso eléctrico comienza arriba en las aurículas en el nódulo sinoauricular, después viaja al nódulo auriculoventricular y al Haz de His, finalmente

pasa a través de las fibras de Purkinje hacia los ventrículos. Cuando existe una falla eléctrica en cualquiera de los nodos eléctricos del corazón las contracciones pueden ser descoordinadas, sin ritmo o nulas [3].

4.2. Electrocardiograma

La electrocardiografía es un método de adquisición de señales ECG, el cual se encarga de registrar la actividad eléctrica del miocardio. Este método obtiene la señal por medio de electrodos ubicados en la superficie corporal a la altura de la parrilla torácica, los electrodos obtienen la despolarización y repolarización de los nodos del corazón por medio de la diferencia de potencial eléctrico [29].

El funcionamiento del corazón se puede observar mediante el registro de la actividad eléctrica de sus nodos. La representación de dicha actividad eléctrica se muestra en gráficas continuas y consiste en una línea base con distintas características que varían en el tiempo. Las características se muestran a lo largo del registro ECG teniendo en cuenta el periodo de tiempo, todas las señales miocárdicas se pueden representar mediante una onda característica a la cual se le denomina complejo QRS [30].

En la Figura 4.2 se observa la onda que muestra el ECG junto con su complejo QRS, mismo que se puede descomponer en las siguientes ondas:

- Onda P representa una despolarización de las aurículas.
- Valor Q muestra la despolarización interventricular.
- Valor R muestra la despolarización el ventrículo izquierdo.
- Valor S muestra la despolarización de las áreas basales de los ventrículos.
- Onda T representa la repolarización de los ventrículos.
- Onda U muestra la repolarización de las fibras de Purkinje.

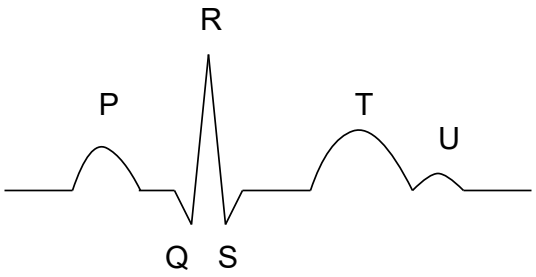

Figura 4.2: Partes de la señal ECG.

4.3. ECG y sus derivadas

Las derivaciones se obtienen según el posicionamiento de los electrodos. El ECG de 12 derivaciones es el registro más utilizado, ya que es un registro del potencial del corazón desde 12 posiciones distintas, esto es de gran importancia, ya que se puede colocar el miocardio en un espacio con base en la señal bioeléctrica [31].

Definiendo dos derivaciones:

1. Derivaciones en las extremidades (superiores e inferiores): Muestran el plano frontal del corazón y se clasifican en:

   a) Estándar (bipolares).

      1) I.
      2) II.
      3) III.

   b) Aumentadas (monopolares).

      1) aVF.
      2) aVR.
      3) aVL.

2. Las derivaciones en el tórax representan una sección transversal y son desde la derivada V1 hasta la derivada V6.

4.4. Enfermedades Cardiovasculares

Una enfermedad cardiovascular se entiende como una fisiopatología que afecta al correcto funcionamiento del sistema circulatorio en cualquiera de sus componentes y comprende una amplia familia de enfermedades [3].

El Síndrome Coronario Agudo (SCA) es ampliamente utilizado para referirse a situaciones clínicas provocadas por una isquemia miocárdica aguda. El SCA es una familia de enfermedades dentro de la misma se encuentra el Síndrome Coronario Agudo Con Elevación del Segmento ST (SCA-SEST) y el Infarto Agudo al Miocardio con Elevación del Segmento ST (IMEST). Donde un IMEST se caracteriza por presentar una elevación persistente dentro del segmento ST en la toma de señales ECG y por su parte el SCA-SEST se caracteriza por no tener una elevación del segmento ST y dentro del mismo puede tener una depreciación de este mismo segmento, a pesar de no ser obligatoria una depreción de este segmento es de suma importancia conocerlo en la toma de un ECG [32].

4.5. Procesamiento de las señales ECG

El objetivo de procesamiento de una señal es extraer la información más relevante de la misma. Es común manipular la señal utilizando diversas técnicas especializadas según las características y especificaciones deseadas. En el caso de las señales ECG se extraen patrones importantes a fin de observar con mayor claridad el complejo QRS mediante algoritmos computacionales [33].

4.5.1. Normalización

La normalización es una transformación lineal que se aplica a conjuntos de datos numéricos, esta transformación se utiliza cuando se busca disminuir el sesgo existente por la combinación de datos medidos con la cualidad de mantener la relación entre los valores numéricos del conjunto de datos original [34].

4.5.2. Normalización Min-Max

La normalización Min-Max es una de las más comunes en conjuntos de datos utilizados para el ML, esta transformación cambia el rango en el cual se encuentra y lo mantiene entre $[0, 1]$ preservando la relación entre el conjunto de datos. Si suponemos a A como un conjunto de datos numéricos de n valores con $\min A$ y $\max A$ como su valor mínimo y máximo respectivamente podemos realizar una normalización Min-Max mediante (4.1) [34]:

$$A_{inorm} = \frac{A_i - \min A}{\max A - \min A} \tag{4.1}$$

4.5.3. Estandarización

La estandarización también llamada *z-score normalización* forma parte de la normalización, sin embargo, esta transformación tiene propiedades distintas a las de normalización Min-Max. La estandarización es ampliamente utilizada en el campo de la IA. Para realizar su transformación lineal hace uso de la media (μ) y de la desviación estándar (σ). Si consideramos a X como un conjunto de datos de n valores podemos estandarizar los datos mediante la ecuación (4.2) [35]:

$$x_{i,\text{estandarizado}} = \frac{x_i - \mu}{\sigma} \tag{4.2}$$

Siendo μ y σ como la ecuación (4.3):

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i; \quad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{4.3}$$

Con la estandarización se busca obtener un conjunto de datos con desviación estándar unitaria ($\sigma = 1$) y con una media nula ($\mu = 0$). Estas características son de gran apoyo para algunos modelos de IA cuando se realiza clasificación [35].

4.6. Aprendizaje Automático

El Aprendizaje Automático (ML, de sus siglas en inglés), permite a las máquinas mejorar en una tarea específica a través de la experiencia. En esencia, es un método científico utilizado para extraer patrones y relaciones de los datos de manera autónoma. Estos patrones son útiles para realizar predicciones de comportamientos y para la toma de decisiones futuras. Todas las técnicas de aprendizaje automático se consideran parte de la IA. Estas técnicas permiten a las máquinas analizar datos históricos, conocidos como datos de entrenamiento, y crear su propio modelo para luego hacer predicciones utilizando datos de prueba [36].

4.6.1. Aprendizaje supervisado

El aprendizaje supervisado realiza su entrenamiento con base en etiquetas para los datos de entrada y salida, dicho de otra manera, este tipo de aprendizaje se apoya con datos etiquetados por el programador y tiene como objetivo mapear las entradas y salidas con los valores correctos mediante un supervisor.

4.6.2. Máquinas de soporte vectorial

Las SVM en comparación con otras técnicas de Aprendizaje Automático otorgan una precisión amplia y específicamente en los espacios de entrada no lineales, la herramienta conocida como *kernel* es bastante utilizada por las SVM. De igual manera la técnica SVM es utilizada para clasificación, basada en la idea de minimización de riesgo. Con los datos de entrenamiento la técnica genera un hiperplano óptimo (clasificador de maximización de márgenes), dicho de otro modo, el hiperplano divide un plano en dos partes donde las etiquetas se encuentran en cada lado y posteriormente mapea los datos a un espacio de dimensión mayor, si los datos cuentan con dimensión $\mathbb{R}^2$, se mapean mediante la SVM a $\mathbb{R}^3$ [37].

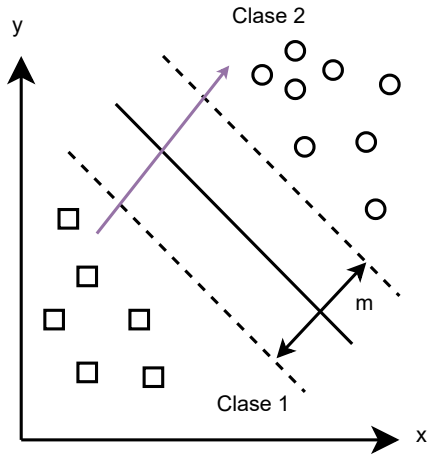

Figura 4.3: Hiperplano de una SVM.

Los elementos mostrados en la Figura 4.3 son definidos a continuación:

1. Vector de soporte: datos muy cercanos al hiperplano con los cuales se construye el clasificador.
2. Hiperplano: plano de diferenciación entre los conjuntos de objetos.
3. Margen: distancia entre las dos líneas de la clase.

4. Peso: valor del vector.

El límite está dado cuando $y = 0$, de esta manera se puede utilizar la ecuación (4.4).

$$y = \sum_{i=1}^{N} w_i x_i + b = x_i \mathbf{w} + b = 0 \tag{4.4}$$

Donde:

1. x_i es la entrada.

2. $\mathbf{w}$ es el peso.

3. b es una constante.

Las clases se identifican como ± 1 mediante el análisis de una SVM, esto reduce la matemática, gracias a esto, el valor de y tiene que ser ± 1 en los datos más aproximados al hiperplano, como subsecuente, las ecuaciones de las líneas que generan el margen tienen que ser:

$$x_i \mathbf{w} + b \geq 1 \quad y = 1 \tag{4.5}$$

$$x_i \mathbf{w} + b \leq 1 \quad y = -1 \tag{4.6}$$

Debido a lo anterior, las ecuaciones (4.5) y (4.6) se pueden expresar de la siguiente forma:

$$y_i(x_i \mathbf{w} + b) \geq 1 \tag{4.7}$$

La ecuación (4.7) fundamenta w y b realicen una clasificación apropiada en el plano dividido por el hiperplano.

Posteriormente tenemos la ecuación (4.8) que otorga la distancia al origen para separar la clase 1:

$$d_0 = \frac{(1-b)}{\|\mathbf{w}\|} \quad y_i \geq 1 \tag{4.8}$$

Por último la ecuación (4.9) otorga la distancia de separación de la clase 2 al origen:

$$d_0 = \frac{(-1-b)}{\|\mathbf{w}\|} \quad y_i \leq 1 \tag{4.9}$$

Si obtenemos la resta entre ambas ecuaciones, (4.8) y (4.9), se puede obtener el margen otorgado por la ecuación (4.10).

$$M = \frac{(1-b)}{\|\mathbf{w}\|} - \frac{(-1-b)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{4.10}$$

El máximo margen se puede obtener minimizando $\|w\|^2$.
Donde $\|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w}$ [37].

4.7. *Kernel de función radial*

Se define como *Kernel* como una similitud matemática entre diferentes instancias con diferentes dimensiones. La idea del uso de *Kernels* es que gracias a la similitud matemática podemos trabajar con espacios de características de grandes dimensiones de una manera más sencilla. Computacionalmente se utilizan los *kernels tricks* los cuales son *kernels* que permiten ser utilizados computacionalmente en una implementación de aprendizaje, mismos que son utilizados por su manejo computacional sin ser costoso. De forma matemática podemos definir un *kernel* como se muestra en la ecuación (4.11):

$$K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle \tag{4.11}$$

donde K es el *kernel* y ψ es el mapeo en el espacio [38].

El *kernel* RBF (de sus siglas en inglés, *Radial Basis Fuction*) también conocido como *Kernel Gaussiano* es definido matemáticamente por la ecuación (4.12):

$$K(\mathbf{x}, \mathbf{x}') = e^{-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma}}; \quad \sigma > 0 \tag{4.12}$$

Intuitivamente se puede observar en la ecuación (4.12) que las características en el espacio entre x, x' se acercan a cero si las instancias están alejadas y se acercan a 1 si están cerca, para definir la distancia entre ambas instancias se cuenta con σ el cual es un escalar que nos ayuda a definir que tan *cerca* están las instancias [38].

4.8. Aprendizaje Profundo

El aprendizaje profundo (DL, de sus siglas en inglés) es una derivación del ML, basado en una estructura jerárquica aprovechando las redes neuronales artificiales, estas últimas están basadas en la estructura de las neuronas cerebrales del humano, utilizando nodos para realizar una conexión entre las mismas. De esta manera esta estructura permite realizar análisis no lineales de los datos [39].

El DL es bastante utilizado en tareas de clasificación trabajando directamente con imágenes, audios y textos. Los modelos de DL son entrenados utilizando una base de datos y redes neuronales [39].

4.9. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales son una técnica de DL cuyo modelo matemático está basado en las neuronas cerebrales con una estructura similar al cerebro humano. Esta técnica tiene una gran variedad de aplicaciones gracias a su flexibilidad ya que una red neuronal tiene la capacidad de realizar distintas tareas.

Debido a que está inspirada en la estructura cerebral, los elementos básicos de funcionamiento de una estructura de una red neuronal artificial son las neuronas. Se puede describir una neurona como un dispositivo cuyo funcionamiento principal es realizar un cálculo generando un resultado teniendo como entrada un conjunto de datos [39].

La estructura de una red neuronal artificial tiene distintas capas de neuronas las cuales se pueden dividir en los siguientes tres tipos:

1. La primera es conocida como capa de entrada, la cual tiene como entrada información externa.
2. Posteriormente se tienen las capas ocultas, estas se encuentran internas en la red neuronal.
3. Por último se tiene la capa de salida, la cual transmite información al exterior.

La estructura de las Redes Neuronales Artificiales (RNAs) permite tener varias capas ocultas, al igual que permite no tener capas ocultas. Los enlaces muestran el flujo de las señales por la red neuronal, estos enlaces tienen un peso asignado. La salida de una neurona puede dirigirse a dos o más neuronas de la siguiente capa, cada neurona recibe la salida de las neuronas anteriores. El total de capas de una RNA son las sumas de la capa de salida más las capas ocultas.

Podemos observar en la Figura 4.4 la estructura de una red artificial.

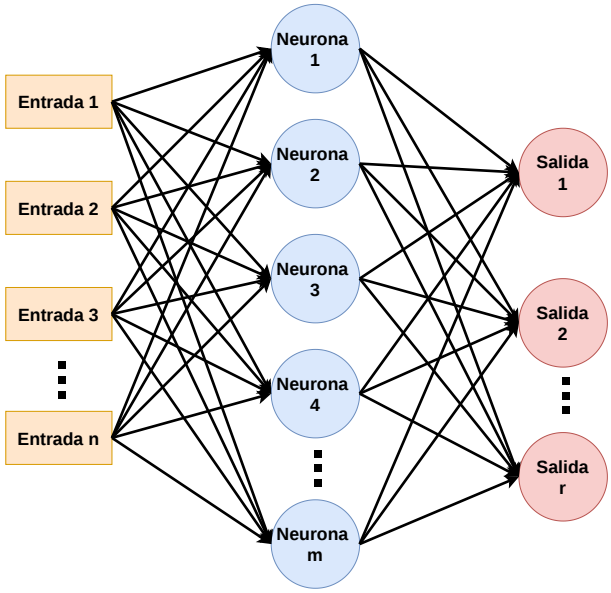

Figura 4.4: Estructura general de una Red Neuronal Artificial.

Las RNAs mantienen su funcionamiento básico mediante la ecuacion (4.13):

$$\hat{y} = \phi(\sum_{j=1}^{d} w_j x_j + b) \tag{4.13}$$

siendo $\mathbf{x} = \{x_1 ... x_d\}$ el vector de entradas, $\mathbf{w} = \{w_1 ... w_d\}$ la matriz de pesos, b el *bias* o sesgo y $\phi(\cdot)$ la función de activación [40].

4.9.1. Función de activación

En las RNAs la función de activación ($\phi(\cdot)$) es una parte crítica para el diseño de la misma, esto debido a que la función de activación puede cambiar en gran medida la predicción de la etiqueta que estará realizando la RNA [40]. Algunas funciones bastante utilizadas son las siguientes:

- ReLu.

- Sigmoid.
- Tanh.
- Identity.

Las ecuaciones (4.14) y (4.15) muestran a las funciones de activación *tanh* y $softmax$ respectivamente en su representación matemática para realizar la etiqueta del resultado [40].

$$\phi(x_j) = \frac{e^{2x_j} - 1}{e^{2x_j} + 1} \tag{4.14}$$

$$\phi(x_j) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_i}} \tag{4.15}$$

siendo K el numero total de clases de clasificación de la RNA.

4.9.2. Función de pérdida

Para las RNA se tiene otra función llamada función de pérdida la cual es la encargada de quantificar la distancia entre el valor real y el predicho por la RNA. Una función de pérdia bastante utilizada es la *Cross-entropy loss* la cual se rige bajo la ecuación (4.16) [41].

$$l(\mathbf{y}, \mathbf{y}') = -\sum_{j=1}^{q} y_j \log \hat{y}_j \tag{4.16}$$

Donde q es la cantidad de clases a predecir, $y_j = 1$ para la clase correcta, $y_j = 0$ para las otras clases y $\hat{y_j}$ es la probabilidad predicha por el modelo para la clase j [41].

4.10. Redes Neuronales Recurrentes

Dando un enfoque a las Redes Neuronales Recurrentes, estas tienen un amplio espectro de aplicaciones para tareas computacionales incluyendo el tratamiento de secuencias, la predicción no lineal de una trayectoria y modelación de la dinámica de distintos sistemas. Las RNN también son conocidas como redes espacio-temporales, debido a se procura realizar una correspondencia entre las secuencias de salida y de entrada otorgando patrones temporales [42].

La RNN más básica posible está compuesta por una neurona teniendo como entrada los inputs, otorgando outputs que regresan a la misma neurona, esto se puede apreciar en la Figura 4.5 [43].

Se puede apreciar que en cada marco temporal t, la neurona recibe ambos datos, el dato de entrada $x(t)$ y la salida del marco temporal anterior $y(t-1)$. Representando dicho proceso a través del eje x como el tiempo transcurrido, se puede obtener la parte derecha de la Figura 4.5 [43].

Todas las neuronas contienen dos pesos; el primero influye en las entradas $x(t)$ y el segundo influye en las salidas del marco temporal anterior $y(t-1)$. A dichos pesos se les nombra como w_x y w_y respectivamente. Utilizando la ecuación (4.17) podemos obtener la salida $y(t)$ de una neurona.

$$\mathbf{y}(t) = \phi(\mathbf{x}(t)^T \cdot \mathbf{W_x} + \mathbf{y}(t-1)^T \cdot \mathbf{W_y} + \mathbf{b}) \tag{4.17}$$

En la cual b se denomina como sesgo, por otro lado tenemos $\phi(\cdot)$ el cual representa la función de activación. La ecuación (4.17) se puede resumir como se muestra a continuación:

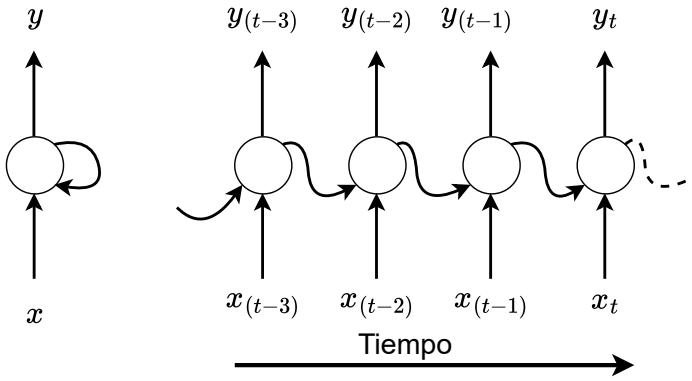

Figura 4.5: Neurona recurrente (izquierda), desarrollo en el tiempo (derecha).

$$\mathbf{Y}(t) = \phi(\mathbf{X}(t) \cdot \mathbf{W}_x + \mathbf{Y}(t-1) \cdot \mathbf{W}_y + \mathbf{b}) = \phi(\begin{bmatrix}\mathbf{X}(t) & \mathbf{Y}(t-1)\end{bmatrix} \cdot \mathbf{W} + \mathbf{b}) \tag{4.18}$$

dado que:

$$\mathbf{W} = \begin{bmatrix}\mathbf{W}_x & \mathbf{W}_y\end{bmatrix}^T \tag{4.19}$$

1. $\mathbf{Y}(t)$ se denomina como una matriz de $m \times n_{neuronas}$ dimensiones la misma tiene las salidas del marco temporal t de todos los batch, de esta forma se tiene que $m \times n_{neuronas}$ son la cantidad de neuronas de la capa.

2. $\mathbf{X}(t)$ se denomina como una matriz de dimensión de $m \times n_{neuronas}$ la misma tiene las variables de entrada de las m observaciones y $n_{entradas}$ resulta ser el número de variables de entrada.

3. Las matrices $\mathbf{W}_x$ y $\mathbf{W}_y$, son las matrices de los pesos, estas se pueden unir en una matriz $\mathbf{W}$ la cual será $(n_{entradas} + n_{neuronas}) \times n_{neuronas}$.

4. Para $\mathbf{b}$ resulta ser un vector de tamaño $n_{neuronas}$ el cual tiene la capacidad de añadir un sesgo al modelo.

$\mathbf{Y}(t)$ depende directamente de $\mathbf{X}(t)$ y de $\mathbf{Y}(t-1)$, las cuales resultan ser función de $\mathbf{X}(t-1)$ y de $\mathbf{Y}(t-2)$, que a su vez resulta ser una función de $\mathbf{X}(t-2)$ y de $\mathbf{Y}(t-3)$ y así sucesivamente. Lo cual implica que $\mathbf{Y}(t)$ es función de las entradas desde el tiempo inicial, o sea $t = 0$. Como en $t = 0$ no hay salidas previas se asumen como cero [43].

4.11. Gated Recurrent Unit (GRU)

El funcionamiento de las celdas GRU (Gated Recurrent Unit, por sus siglas en inglés), tiene un comportamiento basado principalmente en las LSTM (de sus siglas en inglés, Long Short-Term Memory), sin embargo, las GRU suelen ser más sencillas y ocupar menor capacidad computacional, lo cual las hace ideales para muchos trabajos. El principio de las GRU se basa en saber si memorizar o no memorizar resultados previos mediante compuertas. Se dividen en compuertas de reseteo y actualización de compuertas las cuales se muestran en la ecuación (4.20) [40].

$$\begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\mathbf{r}} \end{bmatrix} = \begin{pmatrix} sigm \\ sigm \end{pmatrix} \mathbf{W}^{(k)} \begin{bmatrix} \bar{\mathbf{h}}_t^{k-1} \\ \bar{\mathbf{h}}_{t-1}^{(k)} \end{bmatrix} \tag{4.20}$$

y para actualizar el estado oculto se utiliza la ecuación (4.21).

$$\bar{\mathbf{h}}_t^{(k)} = \bar{\mathbf{z}} \odot \bar{\mathbf{h}}_{t-1}^{(k)} + (1 - \bar{\mathbf{z}}) \odot tanh \mathbf{V}^{(k)} \begin{bmatrix} \bar{\mathbf{h}}_t^{(k-1)} \\ \bar{\mathbf{r}} \odot \bar{\mathbf{h}}_{t-1}^{(k)} \end{bmatrix} \tag{4.21}$$

de (4.20) y (4.21) se tiene que $\mathbf{W}^{(k)}$ y $\mathbf{V}^{(k)}$ son dos matrices de pesos, $sigm$ es la función de activación sigmoide, los vectores $\bar{\mathbf{z}}_t$ y $\bar{\mathbf{r}}_t$ son las compuertas de actualización y reset respectivamente, se cuenta con vector oculto de estados denotado por $\bar{\mathbf{h}}_t^{(k)}$ y $\odot$ representa el producto de elemento por elemento [40].

4.12. Análisis de ablación

En el campo de ML existe el análisis de ablación, inspirado en el método médico con el mismo nombre el cual se enfoca en extraer un órgano o tejido y observar el comportamiento del cuerpo con la ausencia del mismo. El análisis de ablación es similiar, debido a que se extraen características de los modelos de inteligencia artificial para observar el impacto de las características con el modelo de IA, esto nos permite utilizar arquitecturas más simples sin reducir el rendimiento del mismo [44, 45].

El principal objetivo de un análisis de ablación es obtener las características o parámetros donde las cuales su aportación es mínima o nula dentro del modelo de IA por lo tanto se pueden eliminar del modelo sin comprometer su solidez, de esta manera se puede mejorar el tamaño y la velocidad de una RNN. Para un análisis de ablación en un modelo de IA se deben de seguir los siguientes pasos para obtener el porcentaje de importancia dentro del modelo de IA [46, 47]:

- Primer paso: Obtener la sensibilidad del modelo completo.
- Segundo paso: Remover el predictor i y obtener la sensibilidad del modelo sin predictor.
- Tercer paso: Calcular el porcentaje de cambio mediante (4.22).

$$PorcentajeCambio = \frac{Y_2 - Y_1}{Y_2} \cdot 100\,\% \tag{4.22}$$

Entre mayor sea el porcentaje que tenga cada característica considera una relación directamente proporcional con el impacto dirigido al modelo de IA utilizado [47].

4.13. Análisis de desempeño de un clasificador

Cuando se entrena un modelo de IA para clasificación, se puede evaluar su comportamiento mediante diversas métricas las cuales nos ayudan a ver de una manera más amplia el comportamiento en clasificación del modelo [48].

4.13.1. Matriz de confusión

La matriz de confusión es una matriz visual en la cual se puede apreciar el desarrollo de aprendizaje de un algoritmo o modelo de IA. De manera simplificada es una matriz, la cual hace el recuento de los Falsos Positivos (FP), Falsos Negativos (FN), Verdaderos Positivos (VP) y Verdaderos Negativos (VN) de los valores clasificados como se puede observar en la Figura 4.6 [49].

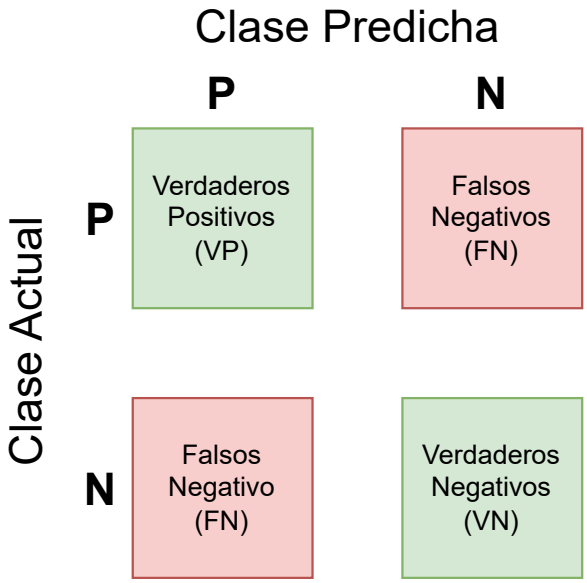

Figura 4.6: Matriz de Confusión.

4.13.2. Precisión

A pesar de que la Matriz de Confusión otorga mucha información, es necesario conocer otras métricas útiles con valores concisos. Una de estas métricas es la precisión, misma que se define como la exactitud de los valores verdaderos predichos por el clasificador y se obtiene mediante la ecuación (4.23) [48].

$$precision = \frac{VP}{VP + FP} \tag{4.23}$$

siendo VP los verdaderos positivos y FP los falsos positivos predichos por el modelo.

4.13.3. Recall

Si solo usamos la precisión como métrica de evaluación, estamos omitiendo información relevante del modelo. Es común utilizar la métrica *recall* en conjunto con la precisión para la evaluación del modelo, misma que se obtiene mediante la ecuación (4.24) [48].

$$recall = \frac{VP}{VP + FN} \tag{4.24}$$

siendo FP los falsos negativos predichos por el modelo.

4.13.4. F1-Score

La métrica F1-Score también conocida como *media armónica* es bastante utilizada para comparar distintos clasificadores ya que hace uso de tanto de la precision y el recall. También es bastante utilizado para comparar clasificadores multiclase, debido a que le otorga mayor influencia a valores bajos por lo que el F1-Score solo puede ser alto cuando la precisión y el recall son altos y se obtiene mediante la ecuación (4.25) [48, 49].

$$F1 = \frac{presicion \times recall}{presicion + recall} \tag{4.25}$$

4.13.5. Exactitud

La exactitud es otra métrica general para los clasificadores y sirve para dar una idea general de la cantidad de valores predichos correctamente y es calculada como la suma de los valores predichos de manera acertada sobre la suma total de predicciones como se muestra en la ecuación (4.26) [48].

$$accuracy = ACC = \frac{VP + VN}{FP + FN + VP + VN} \tag{4.26}$$

CAPÍTULO 5

Metodología

Durante esta sección se detallarán las etapas de desarrollo del proyecto y su integración entre las mismas. En la Figura 5.1 se muestra el diagrama de flujo general de la secuencia de etapas realizadas durante el proyecto.

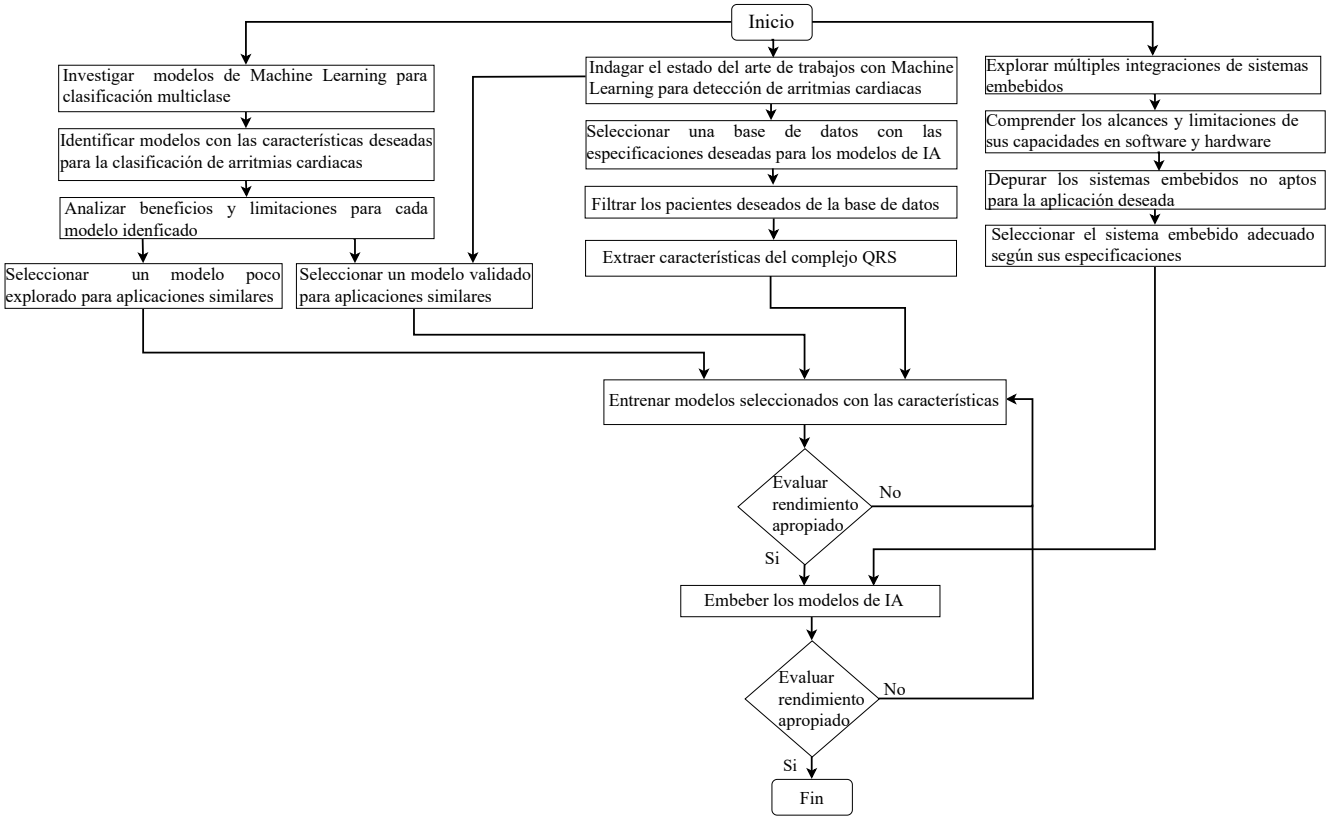

Figura 5.1: Diagrama de flujo metodología.

El desarrollo del proyecto, previo a la implementación en un sistema embebido, se realizó en un equipo con las características mostradas en el Cuadro 5.1.

Cuadro 5.1: Propiedades del equipo de cómputo.

Componente	**Detalle**
Marca	Lenovo
Año	2023
Modelo	ThinkPad
Procesador	Intel(R) Core (TM) i5-7300U CPU 2.60 GHz
Almacenamiento	238 GB SSD SAMSUNG
Memoria RAM	8 Gb
Tarjeta gráfica	Intel(R) HD Graphics 620 (128 MB)
Sistema Operativo	64 bits, procesador x64
Tipo de sistema operativo	Windows 10 Pro
Python	Colab Version Gratuita

5.1. Base de datos de entrenamiento

La base de datos [50] es utilizada para la extracción de características a fin de entrenar los modelos de IA. Esta base de datos cuenta con las características mostradas en el Cuadro 5.2. La misma cuenta con señales ECG de 25 pacientes masculinos (con un rango de edad entre 32 y 89) y 22 femeninos (con un rango de edad entre 23 y 89 años), estas señales fueron limpiadas mediante un filtro *notch* como pasa banda entre $0.1 - 100\ Hz$. Todas las señales cuentan con interpretaciones realizadas por personal especialista en cardiología, los cuales realizaron notaciones de relevancia médica como rangos donde se aprecian ciertas enfermedades como IMEST o SICA-SEST. Es importante resaltar que la base de datos no cuenta con información personal de los pacientes.

Cuadro 5.2: Características base de datos entrenamiento.

Característica	**Especificación**
Total de pacientes	47 pacientes
Cantidad de canales obtenidos	2 canales
Canal ECG	Segunda derivada (MLII)
Resolución	11 bits
Frecuencia	360 muestras por segundo
Rango de muestra	$10\ mV$
Tiempo de muestreo	30 minutos
Género de los pacientes	Masculino y femenino
Rango de edades	$23 - 89$ años
Fecha de toma de ECG	1989

Es de suma importancia obtener las características mostradas en la Figura 4.2, mismas que comprenden, el punto Q, pico R, punto S, onda T y segmento ST. Estas características deben ser extraídas para cada paciente de la base de datos por lo que para extraerlas se utilizaron herramientas de software y se siguieron los pasos del Algoritmo 1.

Algorithm 1 Extracción de características

Importar Pandas como pd
Leer señal ECG del paciente desde el archivo CSV
Convertir a *Data Frame* (df)
Normalizar los valores del df
Encontrar los picos R mediante la función *findpeaks*
Extraer Frecuencia cardíaca (FC) con la distancia entre los picos R
Extraer las demás características con base en la FC y tiempo de muestreo
Guardar los valores en un nuevo archivo CSV

Para el Algoritmo 1 es importante considerar que la información de cada paciente se encuentra en un archivo CSV diferente donde se encuentra la señal en su forma discreta y el tiempo en el que fue tomada la muestra. Bajo este argumento se debe transformar la señal en un *data frame* para realizar una normalización de la señal de la siguiente manera $df\_normalizado = (data - data.min())/(data.max() - data.min())$.

Una vez teniendo la señal normalizada se procede a extraer las características de interés de la señal, es de importancia iniciar con los los picos R utilizando la función *findpeaks*, la cual tiene como parámetros de entrada *input, height, distance* los cuales son nuestra señal, un valor mínimo que debe tener para considerarlo como pico y una distancia mínima que debe existir entre cada muestra respectivamente. Es necesario graficar la señal para realizar un ajuste de estos valores en caso de ser necesario. Posteriomente recurrimos a la frecuencia cardiaca la cual se toma al medir la diferencia de tiempo entre cada valor de cada pico y promediando el resultado cada 30 muestras del pico R. Siguiendo con el punto Q, se recorren todos los picos R detectados, se crea una ventana de 15 muestras previas a cada pico R y localiza el punto menor de cada ventana para obtener el punto Q. Continuando con los puntos S, se parten de los picos R, se crean ventanas posteriores a estos picos y se identifican puntos con menor amplitud dentro de la ventana. Para la onda T, se crean ventanas a partir de los puntos S, localizando los valores máximos de cada ventana para determinar la posición de la onda T. Por último, los valores ST se toman ventanas entre los puntos S y T para promediar todos los valores dentro de cada ventana. Una vez finalizada la extracción de características es indispensable almacenar los valores en una nueva base de datos.

5.2. Procesamiento base de datos de validación

La base de datos de validación [51] cuenta con las características mostradas en el Cuadro 5.3. La misma se encuentra revisada y con notaciones por personal del sector salud especializado. Se extrajeron las características mediante el Algoritmo 1. Cada paciente de la base de datos cuenta con los 12 canales de derivaciones del corazón, sin embargo, para este proyecto solo se considera la segunda derivación. Cada señal cuenta con anotaciones de relevancia médica como es la edad, género, diagnóstico, hemodinamia, intervenciones, entre otras. Por último se debe mencionar que la base de datos se encuentra dividida en clases de las cuales se destacan infarto al miocardio, disritmia, miceláneos y control de salud.

Cuadro 5.3: Características base de datos validación.

Característica	Especificación
Total de Pacientes	268 pacientes
Canal ECG	Segunda Derivación (MLII)
Resolución	16 bits
Frecuencia	1000 muestras por segundo
Rango de muestra	16 mV
Género de los pacientes	Masculino y Femenino

5.2.1. Extracción de características para la base de datos validación

Para la extracción de características de la base de datos de validación se realiza de la misma manera que para la base de datos de entrenamiento, sin embargo, es importante realizar un filtrado de los pacientes con las enfermedades de interés para discriminar información no relevante para este proyecto. Por la naturaleza de la base de datos no es posible mantener balanceada la cantidad de pacientes con enfermedades, lo anterior se ajusta al considerar la métrica *F1-Score* la cual es una métrica que permite tener una mejor visualización del comportamiento de los modelos conciderando el desbalance de los pacientes con las enfermedades.

Por otro lado es indispensable recortar las señales en el tiempo a fin de buscar los rangos donde son apreciables las enfermedades, bajo las anotaciones realizadas por los especialistas. Finalmente para mantener una homogeneidad durante el proyecto se deben extraer las mismas características que en la base de datos de entrenamiento, resaltando que el ventaneo se realiza considerando la mayor cantidad de muestras (previas o posteriores), debido al periodo de muestreo, en conjunto con la frecuencia cardiaca.

5.3. Entrenamiento del modelo SVM para clasificación de pacientes

El entrenamiento de un modelo de SVM contempla las características extraídas de todos los pacientes mediante el Algoritmo 1. Ya con esta nueva base de datos se utiliza el Algoritmo 2 para entrenar el modelo.

Algorithm 2 Entrenamiento modelo SVM

Importar Pandas como pd y sklearn.svm
Leer las características del CSV y convertirlas en un *DataFrame* (df)
Estandarizar los valores del df
Dividir los valores para entrenamiento y testeo
Crear modelo SVM con los hiperparámetros deseados
Entrenar el modelo
Realizar predicciones con los valores de testeo
Imprimir las métricas de evaluación
Modificar hiperparámetros y volver a entrenar hasta conseguir los mejores resultados

Siguiendo el Algoritmo 2 se realiza la estandarización con la ecuación (4.2) para mantener las

propiedades de la señal. Una vez con los datos estandarizados se realiza una división de los mismos considerando el 70 % de los valores para entrenamiento y el 30 % restante para testeo.

Comenzando con el entrenamiento de la SVM es importante ajustar los hiperparámetros *kernel*, C y σ, para realizar predicciones con los valores de testeo y así obtener la precisión *accuracy_score*. Es de suma importancia iterar los valores de los hiperparámetros a fin de encontrar los mejores resultados posibles del modelo SVM. Para este modelo se realizaron iteración con los hiperparámetros *kernel*, C y σ. Quedando como últimos valores $kernel =' rbf'$ y $C = 1000$.

5.4. Entrenamiento del modelo RNN para clasificación de pacientes

Para el entrenamiento de la RNN se utiliza un modelo GRU y se hacen uso de las mismas características que se utilizan para el entrenamiento de la SVM, el Algoritmo 3 nos muestra el procedimiento de entrenamiento de la RNN.

Algorithm 3 Entrenamiento modelo RNN

Importar librerias necesarias (Pandas, SKLearn y Tensorflow)
Importar características del CSV
Estandarizar los valores de las características
Dividir los valores para entrenamiento y testeo
Aplicar *Reshape* a las muestras de entrenamiento
Crear modelo RNN, con arquitectura GRU
Seleccionar especificaciones de la RNN (*Dropout*, neuronas, capas de entrada y salida)
Se compila y entrena el modelo
Imprimir métricas de validación
Modificar hiperparámetros y volver a entrenar hasta conseguir los mejores resultados

El Algortimo 3 utiliza una estandarización y división de los valores como se muestra en la Sección 5.3, teniendo el 70 % para el entrenamiento y 30 % para el testeo. Es importante mencionar que el modelo RNN trabaja con tensores, siendo su forma inicial $((37655, 5), (37655,))$, donde 3755 son la cantidad de valores en la base de datos y 5, la cantidad de características (Punto Q, punto R, punto S, punto T, segmento ST). Es indispensable realizar un cambio para tener el tensor de la siguiente forma $((37655, 1, 5), (37655,))$, ya que el 1 representa la serie de tiempo de los valores.

Las RNN tienen diferentes arquitecturas, en este proyecto se utiliza la arquitectura *GRU* y fue seleccionado debido a su capacidad de aprendizaje profundo en comparación con el modelo de la RNN simple sin el uso de grandes recursos computacionales en comparación la *LSTM*. Se seleccionaron 20 unidades de memoria, un *Dropout* de 10 %, una *capa densa* de 32 neuronas, una función de activación *tanh*, una *capa densa* de salida con 3 neuronas, la función de activación fue *softmax*, optimizador *Adam*, debido a la clasificación múltiple es necesario utilizar la función de pérdida *sparse_categorical_crossentropy*, 30 épocas y un $batch\_size = 16$. Se seleccionaron estos hiperparámetros para reducir la cantidad de recursos utilizados por la RNN y a su vez disminuir el consumo de *hardware* por parte del sistema embebido.

5.5. Selección de características mediante método de ablación

El método de ablación es utilizado para obtener el porcentaje de importancia de las características obtenidas del Algoritmo 1 en los modelos. Este método sigue el Algoritmo 4.

Algorithm 4 Método de Ablación

Importar base de datos de características del Algoritmo 5.2
Retirar la(s) característica(s) deseada(s)
Entrenar los modelos de IA mediante los Algoritmos 2 y 3
Comparar el rendimiento de los modelos contra los modelos entrenados con base a las métricas obtenidas

Cuando se extraen las características se utiliza la ecuación (4.22) y así, mediante el uso del *Accuracy, Precision, Recall* y el *F1-Score* de los modelos podemos obtener su porcentaje de impacto siendo $Y_2 = Y_{metrica\_total}$ y $Y_1 = Y_{metrica\_obtenida}$, Así se tiene el factor de impacto de la característica según cada métrica, posteriormente se promedian todos los factores de impacto de todas las métricas para obtener un factor de impacto total. Esta métrica es utilizada para conocer si las características seleccionadas (punto Q, punto R, punto S, punto T, segmento ST) son relevantes para cada modelo. Según sea el porcentaje de impacto se conoce si todas las características aportan al entrenamiento del modelo. Es de suma importancia realizar el Algoritmo 4 para cada modelo debido a que los modelos tienen distintas formas de aprendizaje por lo que los resultados varían por cada modelo. Debido a lo anterior este estudio se realiza depués de tener los modelos preliminares. De igual manera el estudio se realiza por duplas y ternas para conocer si existe un impacto positivo o negativo con cualquier combinación de características. Se entiende como impacto positivo si las métricas mejoran al extraer características y un impacto negativo al disminuir las métricas al extraer ciertas características. El porcentaje impacto se realiza con cada métrica debido a que son las métricas de interés en los modelos finales, lo cual otorga un panorama general de las características seleccionadas, ya que puede tener un comportamiento diferente con cada métrica.

5.6. Validación de respuesta de modelos mediante segunda base de datos

Para la base de datos de validación [51] es necesario extraer sus características siguiendo nuevamente el Algoritmo 1. Es imprecindible este paso debido a que mediante la normalización y estandarización de los datos se mantienen las propiedades de cada señal y se acotan al mismo rango todas, siendo esta la propiedad más relevante para los modelos de IA. Una vez extraídas, los modelos se someten a predicciones con los valores de esta base de datos y se obtienen las principales métricas de interés para conocer la robustez de ambos modelos. Es importante mencionar que para este paso los modelos de IA se siguen ejecutando dentro del mismo equipo de cómputo mostrado en el Cuadro 5.1, sin embargo, se mantiene presente la cantidad de recursos en *hardware* para no exceder las especificaciones del sistema embebido. En caso de no contar con resultados en métricas favorables es indispensable regresar y ajustar los hiperparámetros a fin de buscar los apropiados para conseguir resultados favorables.

5.7. Exportación de modelos a sistema embebido

En el Cuadro 5.4 se muestran las características de la *Raspberry PI 4B*, este fue el sistema embebido utilizado donde se colocaron los modelos de IA entrenados y validados. Este sistema fue seleccionado debido a que su capacidad en *hardware* permite ejecutar *Scripts* de *Python* y a su vez soporta librerias necesarias para el proyecto como lo son *SKLearn* y *tensorflow*. Otra ventaja del sistema embebido es su amplia documentación técnica siendo indispensable en el desarrollo del proyecto. Por su parte la accesibilidad al sistema, el costo y su espacio reducido resultan ventajas para los desarrolladores del proyecto.

Cuadro 5.4: Características *Raspberry Pi 4B*.

Característica	**Especificación**
CPU	Procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72
Puertos	2 puertos USB 3.0; 2 puertos USB 2.0. Conector GPIO de 40 pines estándar de *Raspberry Pi*
Conexiones	IEEE 802.11ac de 2,4 GHz y 5,0 GHz, Bluetooth 5.0, BLE
Alimentación	5 V/3 A vía USB-C, 5V vía cabezal GPIO
Expansión	Cabezal GPIO de 40 pines

Para la integración de los modelos entrenados dentro del sistema embebido, se ejecutó un procedimiento secuencial basado en el Algoritmo 5. Este proceso comenzó con una fase crítica de verificación de compatibilidad, debido a que las versiones de TensorFlow y SKLearn presentan dependencias específicas con las versiones de Python utilizadas. Una vez garantizada la estabilidad del entorno, se procedió a la exportación de los modelos en formatos .h5 (para redes neuronales) y .pkl (para modelos de aprendizaje estadístico), los cuales encapsulan la arquitectura y los pesos finales necesarios para su ejecución en dispositivos externos.

La transferencia de estos archivos hacia el sistema embebido se gestionó mediante plataformas de cloud computing, facilitando su descarga e integración remota. Para la fase de inferencia, se desarrolló un script de Python encargado de cargar los modelos y procesar un dataframe de validación como entrada para generar las predicciones correspondientes. Finalmente, la evaluación del desempeño se realizó mediante la generación de gráficas de métricas con la librería Matplotlib; estas imágenes se almacenaron localmente en el sistema y, como etapa conclusiva, se cargaron de forma manual a la nube para permitir su consulta y análisis desde cualquier dispositivo con acceso a internet.

Algorithm 5 Embeber modelos SVM y RNN

Investigar compatibilidad entre versiones de Python, Tensorflow y SKLearn
Instalar la versión apropiada de Python, Tensorflow y SKLearn
Importar los archivos con extensión *.h5* y *.plk* donde se encuentra la información de los modelos entrenados
Importar la base de datos de validación
Realizar predicciones con los modelos y la base de datos
Obtener métricas de los resultados de los modelos

Al finalizar con los modelos de IA es de interés conocer algunos parámetros de *hardware* del sistema embebido, estos parámetros pueden ser medidos mediante la arquitectura del mismo sistema embebido (mediante un *Script* dentro del Algoritmo 5) o mediante un elemento externo. Son de gran interés conocer los siguientes parámetros debido a que otorgan una idea de la exigencia de los modelos al momento de ser utilizados en un dispositivo con limitaciones en *hardware*:

- Tiempo de inferencia total.
- Tiempo de inferencia para el modelo SVM.
- Tiempo de inferencia para el modelo RNN.
- Uso promedio CPU por cada modelo.
- Máxima memoria RAM utilizada cada modelo.
- Máxima temperatura CPU alcanzada por cada modelo.

CAPÍTULO 6

Resultados y discusión

Una vez descrita la metodología del Capítulo 5, se proceden a realizar la experimentación necesaria, al realizar las pruebas correspondientes con los materiales antes mencionados se obtuvieron los resultados, mismos que se muestran y detallan a lo largo de este Capítulo.

6.1. Extracción de características

Antes de extraer las características mostradas en la Figura 4.2 fue necesario realizar una normalización Min-Max de los datos de la base de datos mediante la ecuación (4.1). Mediante este procedimiento se ajustó el rango de la señal a valores comprendidos entre 0 y 1, sin alterar sus propiedades originales. Posteriormente, haciendo uso del Algoritmo 1, se realizó una nueva base de datos procesada que contiene las características de interés, con formato CSV. En el Cuadro 6.1 se muestran algunos valores de la base de datos, siendo truncados a 4 decimales, sin embargo, para el entrenamiento de los modelos se utilizaron todas las decimales entregadas por la normalización. Las enfermedades se catalogan de la siguiente manera:

- Enfermedad 0: Paciente sano.
- Enfermedad 1: IMEST.
- Enfermedad 2: SCA-SEST.

Cuadro 6.1: Características extraídas de la base de datos de entrenamiento.

Punto Q	Punto R	Punto S	Punto T	Segmento ST	Enfermedad
0.3539	0.7602	0.3644	0.4655	0.3804	0
0.3469	0.7602	0.3705	0.4673	0.3769	0
0.3557	0.7489	0.3635	0.4646	0.3773	0
0.4982	0.6601	0.5206	0.5206	0.3670	1
0.4281	0.6832	0.4421	0.4583	0.4613	1
0.4428	0.6643	0.4414	0.4548	0.4604	1
0.4899	0.6851	0.4623	0.4660	0.4885	2
0.5087	0.7076	0.4819	0.4836	0.5000	2
0.5239	0.7169	0.4865	0.4872	0.4999	2

Es de suma importancia mencionar que este procedimiento de extracción de características fue posible por las anotaciones de los expertos en el área de salud, estas anotaciones detallan los tiempos, rangos y tipo de enfermedad. Posteriormente se realiza el procesamiento matemático presentado en el Algoritmo 1 para extraer los valores de cada característica, lo cual refiere que si se utilizan los mismos rangos de ventanas para todos los pacientes, nuestras características serán erróneas, para evitar esto se graficaron, las señales, en intervalos cortos de tiempo para una comprobación visual como se puede apreciar en la Figura 6.1 para un paciente sano. Este proceso fue el que mayor tiempo consumió por la supervisión manual de las señales.

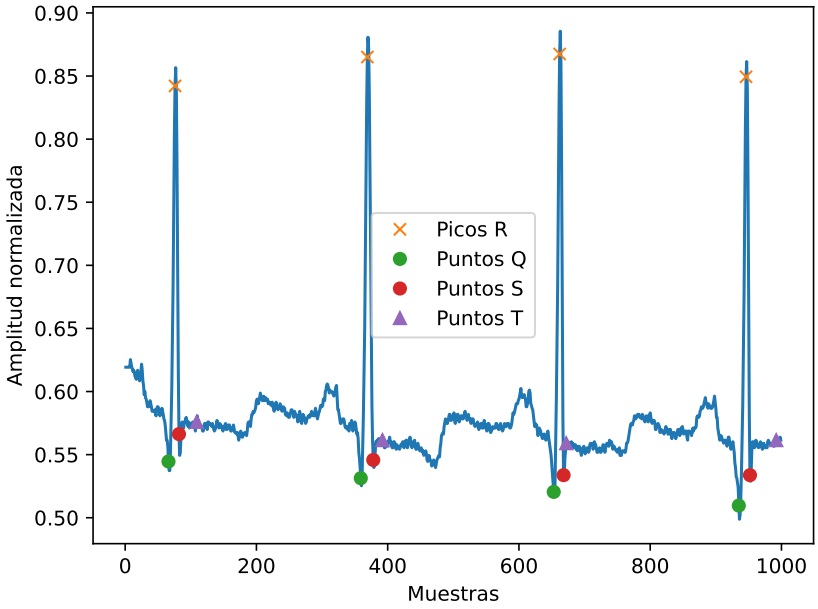

Figura 6.1: Extracción de características de paciente sano.

El proceso se realizó sin normalizar los datos, lo cual provoca una disminución en las métricas de interés, en el Cuadro 6.2 se puede apreciar las métricas obtenidas por parte del modelo SVM al realizar su entrenamiento y testeo sin normalizar los datos, sin estandarizar los datos y sin realizar

ninguno de estos métodos. Se puede realizar una comparación rápida con las métricas obtenidas al utilizar ambos métodos de procesamiento de datos, donde respalda que el uso de estos métodos de procesamiento de datos apoyan en mejorar las métricas con este modelo.

Cuadro 6.2: Métricas obtenidas por el modelo SVM sin normalización ni estandarización.

Métrica	**Sin Normalización**	**Sin Estandarización**	**Sin Normalización ni Estandarización**	**Con Normalización y Estandarización**
Accuracy	0.9154	0.9497	0.9108	0.9542
Precision	0.9145	0.9493	0.9099	0.9541
Recall	0.9154	0.9497	0.9108	0.9542
F1-Score	0.9143	0.9491	0.9096	0.9538

De la misma manera, se realizó para el modelo RNN, en el Cuadro 6.3 se observa de manera directa que al utilizar ambos procesamientos de datos antes de entrenar el modelo cuenta con mejores resultados con base en las métricas de interés.

Cuadro 6.3: Métricas obtenidas por el modelo RNN sin normalización ni estandarización.

Métrica	**Sin Normalización**	**Sin Estandarización**	**Sin Normalización ni Estandarización**	**Con Normalización y Estandarización**
Accuracy	0.9085	0.9368	0.9081	0.9398
Precision	0.9075	0.9371	0.9072	0.9398
Recall	0.9085	0.9368	0.9081	0.9398
F1-Score	0.9075	0.9366	0.9071	0.9389

Posteriormente se realizó la misma metodología para los pacientes con IMEST y con SCA-SEST. En la Figura 6.2 se muestran la comprobación visual de las características deseadas para los pacientes con IMEST. Es posible apreciar errores ligeros en la exactitud de las características deseadas esto es debido a que las ventanas seleccionadas debes ser reajustadas para mejorar la exactitud. La comprobación visual siempre es acompañada de las anotaciones realizadas por los expertos en el área de la salud y únicamente se realizaron ajustes mínimos para la comprobación de los métodos matemáticos empleados.

Para el SCA-SEST se comprueban siguiendo la misma metodología y reajustando de manera empírica teniendo como ejemplo el mostrado en la Figura 6.3.

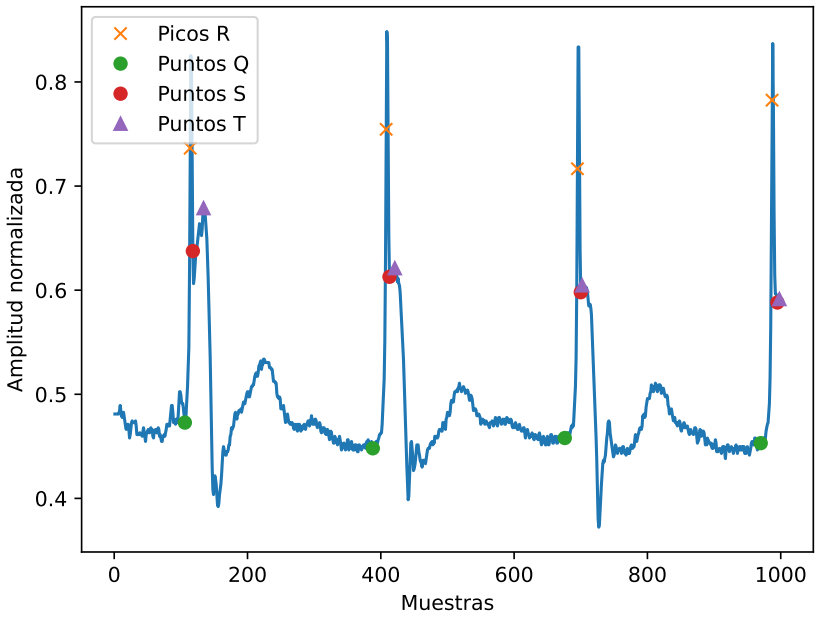

Figura 6.2: Extracción de características para pacientes con IMEST.

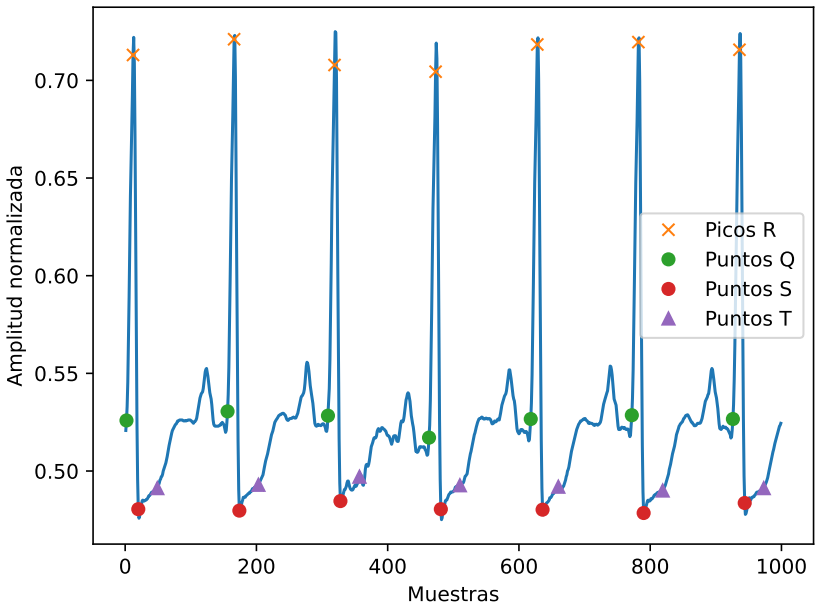

Figura 6.3: Extracción de características para pacientes con SCA-SEST.

El mismo procedimiento se realizó para la enfermedad TV, sin embargo, no fue posible la extracción de características debido a que cuenta con la forma mostrada en la Figura 6.4 y el procesamiento matemático involucra máximos y mínimos de ventanas dentro de una señal conocida. Se realizaron pruebas con la finalidad de observar el comportamiento del procesamiento matemático en este tipo de señales y los resultados se muestran en la Figura 6.5. Al realizar una comparación

visual con la Figura 4.2 es posible deducir que no existe una obtención correcta de las características por lo que se debe emplear otra metodología apropiada a esta enfermedad para extraer otras características de interés como el uso de frecuencia cardiaca, tensión arterial, amplitud de crestas y valles, detección de patrones senoidales, entre otros.

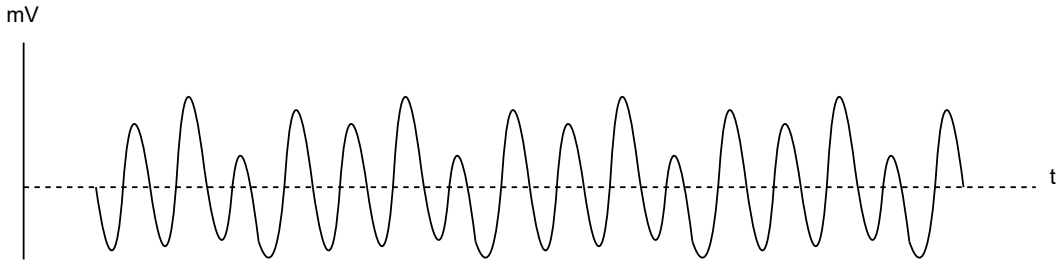

Figura 6.4: ECG de una Taquicardia Ventricular.

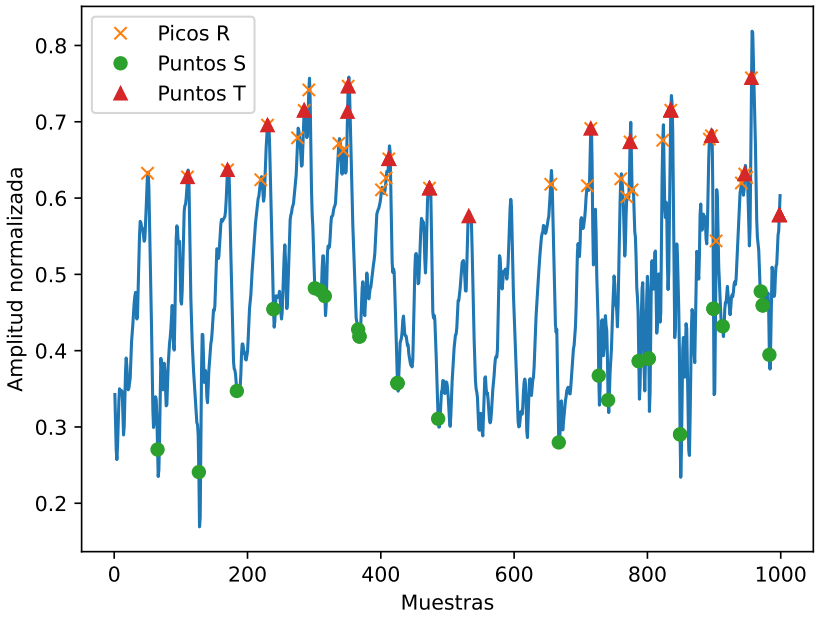

Figura 6.5: Extracción de características Taquicardia Ventricular.

6.2. Diseño y evaluación de modelos

El diseño de los modelos comprende el ajuste de los hiperparámetros y el entrenamiento de estos mediante los Algoritmos 2 y 3. Una vez entrenados, se realizó un estudio de ablación siguiendo el Algoritmo 4. El Cuadro 6.4 muestra los resultados obtenidos al extraer una característica por cada entrenamiento para el modelo SVM. Se puede apreciar que la caracterísca correspondiente al *punto Q* es la que presenta menor porcentaje de impacto y el *punto R* el que presenta mayor porcentaje de impacto. Para calcular el porcentaje de impacto primero se calcula el impacto de cada métrica

mediante la ecuación (6.1) y posteriormente se promedia el impacto de cada métrica para obtener el porcentaje de impacto final.

$$Impacto = \frac{Metrica_{total} - Metrica_{iteracion}}{Metrica_{total}} \tag{6.1}$$

Cuadro 6.4: Ablación para SVM con extracción de características únicas.

Característica	***Accuracy***	***Precision***	***Recall***	***F1-Score***	**% de impacto**
Total	0.95	0.95	0.95	0.95	-
Q	0.94	0.94	0.94	0.94	1.17
R	0.90	0.90	0.90	0.90	5.54
S	0.93	0.93	0.93	0.93	2.48
T	0.94	0.94	0.94	0.94	1.79
ST	0.93	0.93	0.93	0.93	2.37

El mismo procedimiento se realizó para modelo RNN siendo los resultados mostrados en el Cuadro 6.5 donde es posible apreciar que para este modelo nuevamente la característica *Punto Q* es la que presenta menor porcentaje de impacto y el *Punto R* es la que presenta mayor porcentaje de impacto. Al realizar una comparación rápida entre los Cuadros 6.4 y 6.5 se puede observar que la característica con menor y mayor porcentaje de impacto son las mismas, sin embargo, cuentan con diferente porcentaje de impacto lo cual es referido a la manera de aprendizaje de cada modelo.

Cuadro 6.5: Ablación para RNN con extracción de características únicas.

Característica	***Accuracy***	***Precision***	***Recall***	***F1-Score***	**% de impacto**
Total	0.94	0.94	0.94	0.94	-
Q	0.94	0.94	0.94	0.93	0.46
R	0.92	0.92	0.92	0.92	2.44
S	0.92	0.92	0.92	0.92	1.90
T	0.93	0.93	0.93	0.93	0.68
ST	0.92	0.92	0.92	0.92	1.82

Posteriormente el estudio se realizó extrayendo parejas de características, siguiendo nuevamente el Algoritmo 4. Los resultados del modelo SVM se pueden apreciar en el Cuadro 6.6, donde se observa un mayor impacto negativo al extraer dos características de manera simultánea. Resaltando la dupla *R-T*, quien fue la dupla con mayor porcentaje de impacto y la dupla *Q-T* la que mostró menor porcentaje de impacto.

Cuadro 6.6: Ablación con extracción de características por duplas para el modelo SVM.

Características	***Accuracy***	***Precision***	***Recall***	***F1-Score***	**% de impacto**
Total	0.94	0.94	0.94	0.94	-
Q-R	0.84	0.84	0.84	0.84	11.68
Q-S	0.90	0.90	0.90	0.89	6.68
Q-T	0.91	0.91	0.91	0.91	4.54
Q-ST	0.90	0.90	0.90	0.90	5.51
R-S	0.88	0.88	0.88	0.88	7.81
R-T	0.84	0.84	0.84	0.83	11.80
R-ST	0.85	0.85	0.85	0.84	10.95
S-T	0.91	0.90	0.91	0.90	5.21
S-ST	0.88	0.88	0.88	0.87	11.61

Al observar el Cuadro 6.7, donde se muestran los resultados al extraer duplas de características para el modelo RNN, se tiene que la dupla con menor porcentaje de impacto resultó ser *Q-T* y la dupla con mayor porcentaje de impacto fue *S-ST*. Si comparamos los Cuadros 6.6 y 6.7 podemos apreciar que en ambos casos la dupla *QT* es la que cuenta con menor porcentaje de impacto, sin embargo, no es posible resaltar un relación directa entre estas características y los modelos utilizados ya que se difiere en en la cantidad de impacto que ocaciona a cada modelo, aunado a que la dupla con mayor porcentaje de impacto es distinta en cada modelo. Es importante resaltar que al realizar el método de ablación por duplas existen duplas donde los modelos no alcanzan los objetivos en métricas planteadas para este proyecto.

Cuadro 6.7: Ablación con extracción de características por duplas para el modelo RNN.

Características	***Accuracy***	***Precision***	***Recall***	***F1-Score***	**% de impacto**
Total	0.94	0.94	0.94	0.94	-
Q-R	0.85	0.85	0.85	0.85	9.35
Q-S	0.88	0.88	0.88	0.88	6.31
Q-T	0.91	0.91	0.91	0.91	3.25
Q-ST	0.90	0.90	0.90	0.90	4.50
R-S	0.89	0.89	0.89	0.89	5.45
R-T	0.86	0.85	0.86	0.85	9.19
R-ST	0.86	0.86	0.86	0.86	8.55
S-T	0.91	0.91	0.91	0.91	3.49
S-ST	0.88	0.88	0.88	0.88	10.35

Finalmente, se realizó el mismo estudio con la diferencia de extraer ternas de características y evaluar el rendimiento de los modelos, dando como resultados los mostrados en los Cuadros 6.8 y 6.9, donde para el modelo SVM se observa un comportamiento muy similar en cualquiera de sus permutas. Considerando que las entradas originales de los modelos son las cinco mostradas en el Cuadro 6.1 y que durante este método se están extrayendo tres de ellas, mantener todas las métricas de interés con un valor superior al 70 % muestra la robustez del modelo SVM utilizado a pesar de no cumplir con los objetivos planteados para el proyecto. Es importante resaltar que al extraer ternas de características en la SVM se tiene prácticamente el mismo resultado en porcentaje de impacto

con un valor de 22.47 %.

Cuadro 6.8: Ablación con extracción de características por ternas para el modelo SVM.

Características	***Accuracy***	***Precision***	***Recall***	***F1-Score***	**% de impacto**
Total	0.94	0.94	0.94	0.94	-
Q-R-S	0.75	0.74	0.75	0.72	22.47
Q-R-T	0.75	0.74	0.75	0.72	22.47
Q-R-ST	0.75	0.74	0.75	0.72	22.47
Q-S-T	0.75	0.74	0.75	0.72	22.47
Q-S-ST	0.75	0.74	0.75	0.72	22.47
Q-T-ST	0.75	0.74	0.75	0.72	22.47
R-S-T	0.75	0.74	0.75	0.72	22.47
R-S-ST	0.75	0.74	0.79	0.72	21.32
R-T-ST	0.75	0.74	0.75	0.72	22.47

Para los resultados del modelo RNN mostrados en el Cuadro 6.9 se puede apreciar que se tienen porcentajes de impacto muy similares en cualquier permuta de características extraídas. Al igual que el modelo SVM extraer ternas de características nos otorga métricas superiores al 70 % donde se resalta la robustez de aprendizaje de este modelo.

Cuadro 6.9: Ablación con extracción de características por ternas para el modelo RNN.

Características	***Accuracy***	***Precision***	***Recall***	***F1-Score***	**% de impacto**
Total	0.94	0.94	0.94	0.94	-
Q-R-S	0.74	0.72	0.74	0.71	22.81
Q-R-T	0.74	0.73	0.74	0.71	22.25
Q-R-ST	0.75	0.74	0.75	0.72	20.96
Q-S-T	0.74	0.72	0.74	0.71	22.34
Q-S-ST	0.75	0.73	0.75	0.73	21.31
Q-T-ST	0.74	0.73	0.74	0.72	22.16
R-S-T	0.75	0.73	0.75	0.73	21.27
R-S-ST	0.74	0.73	0.74	0.72	22.04
R-T-ST	0.73	0.71	0.73	0.71	23.03

El estudio realizado resalta las cualidades de los modelos para realizar entrenamientos a pesar de no contar con todas las características deseadas, sin embargo, al momento de retirar cualquier característica muestra un impacto negativo en el aprendizaje de ambos modelos, esto se interpreta como una selección correcta de las características ya que no se tuvieron impactos positivos al momento de extraerlos. A pesar de que en distintos casos se sigue cumpliendo el objetivo del proyecto, cada característica individual, dupla o terna aporta positivamente en el aprendizaje de los modelos.

6.2.1. Mejores hiperparámetros SVM

En la búsqueda de la mejor respuesta por parte del modelo SVM fue necesario estar realizando entrenamientos con distintos hiperparámetros y estar observando las métricas de interés a fin de

conseguir los mejores resultados posibles, durante el proyecto se utilizó la función *gridsearch* la cual realiza el entrenamiento con todas las combinaciones posibles de los hiperparámetros que uno le introduzca. Esto se realizó con la siguiente línea de código:

- $param\_grid = 'C' : [1, 10, 100], 'gamma' : [0.01, 0.001, 0.0001], 'kernel' : ['rbf', 'poly', 'linear']$

Posteriormente se realizó el *Gridsearch* y el entrenamiento mediante:

- $grid = GridSearchCV(SVC(), param\_grid, refit = True, verbose = 3)$
- $grid.fit(X\_train, y\_train)$

Y para observar los mejores resultados se utilizaron:

- $grid.best\_params\_$
- $grid.best\_estimator\_$

De esta manera fue posible obtener los mejores parámetros de todas las combinaciones entre *gamma*, *C* y los *kernels*. Es importante mencionar que el tiempo de ejecución para estas líneas de código es bastante amplio debido a que realiza el entrenamiento completo de cada combinación posible entre los valores dados para los hiperparámetros lo cual requiere una gran capacidad computacional y tiempo de ejecución, lo cual provocó en dos ocasiones errores de ejecución.

Una vez encontrados los mejores parámetros otorgados por la función se siguieron realizando pruebas de manera empírica para observar el comportamiento del modelo SVM y al momento de extraer la función *gamma* se obtuvieron mejores resultados que los mostrados por la función. Lo cual significa que *gamma* tiene el valor *scale* que se da por la ecuación (6.2).

$$\gamma = \frac{1}{n_{\text{features}} \cdot \text{Var}(X)} \tag{6.2}$$

donde:

- n_{features} es el número de características.
- $\text{Var}(X)$ es la varianza de todos los valores.

Una vez entrenado el modelo con los mejores resultados posibles se calculó el *gamma* de manera manual dando un resultado de $gamma = 0.200365$ siguiendo la ecuación (6.2). Es de suma importancia mencionar que la función *GridSearchCV* no incluyó este valor para *gamma* debido a que esta función solo realiza combinaciones con los valores entregados y realizar comparaciones con las métricas para otorgar los mejores resultados. Se pueden utilizar otras metodologías como algoritmos metahurísticos para realizar iteraciones que puedan otorgar hiperparámetros donde se cuenten con las mejores métricas posibles en los resultados sin recursos computancionales exagerados.

6.3. Sistema Embebido

Uno de los objetivos del proyecto es tener los modelos SVM y RNN dentro de un sistema embebido, como se describió en el Capítulo 5, el sistema embebido por selección es una *Raspberry Pi 4B*. En la Figura 6.6 se puede apreciar el sistema embebido con su protección mecánica y sus extractores para evitar el sobrecalentamiento.

Figura 6.6: Sistema embebido utilizado, *Raspberry Pi 4B*.

En la Figura 6.7 se muestra una representación visual del funcionamiento de la *Raspberry Pi 4B* contemplando sus entradas, los modelos de IA y las salidas, siendo estas últimas las enfermedades predichas.

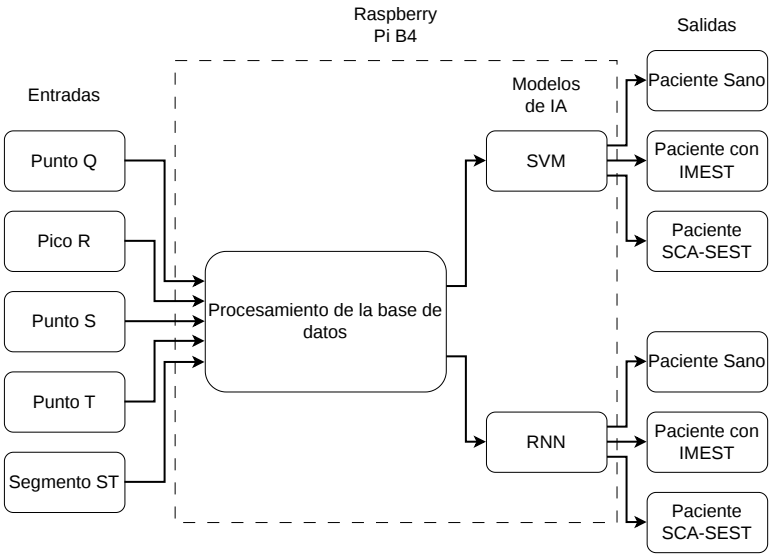

Figura 6.7: Representación gráfica del funcionamiento de la *Raspberry Pi 4B*.

Siguiendo el Algoritmo 5 se logró insertar y hacer uso de la SVM y la RNN. Realizando un Script en Geany (IDLE de Python) para la evaluación de los modelos SVM y RNN se obtuvieron las métricas mostradas en los Cuadros 6.10 y 6.11 respectivamente. En el Cuadro 6.10 se muestran

las métricas generales del modelo e individuales por enfermedad, es importante resaltar que las métricas son menores a las mostradas en el Cuadro 6.4 debido a que disminuye por las capacidades en *hardware* del sistema embebido. Otro aspecto a resaltar es que se tiene mucha variación en las métricas individuales debido a que la base de datos se encuentra desbalanceada en información para cada tipo de enfermedad es por ese motivo que es de suma importancia observar el *F1-Score* general del modelo ya que esta métrica otorga información considerando el desbalance de la base de datos. Si observamos las métricas individuales del Cuadro 6.10 se puede observar que en el caso de la enfermedad IMEST no alcanza la precisión establecida para el objetivo del proyecto, siendo resultado de tener una cantidad menor de pacientes con esta enfermedad para entrenar el modelo, esto se respalda cuando observamos el F1-Score. Cuando observamos a los pacientes sanos se cuenta con la mayor precisión y F1-Score debido a que es la enfermedad con mayor cantidad de pacientes lo cual aporta un mejor entrenamiento del modelo para este caso. Cuando revisamos las métricas generales del modelo se alcanzó y superó el objetivo del proyecto de manera satisfactoria.

Cuadro 6.10: Métricas de clasificación SVM.

SVM			
Enfermedad	***Precision***	***Recall***	***F1-Score***
0 (Paciente Sano)	97 %	85 %	91 %
1 (IMEST)	72 %	94 %	82 %
2 (SCA-SEST)	89 %	88 %	88 %
Accuracy		87.75 %	
Precision		89.73 %	
Recall		87.75 %	
F1-Score		88.11 %	

Si observamos el Cuadro 6.11 podemos ver que nuevamente existe una discrepancia entre las métricas mostradas en el Cuadro 6.5, como se mencionó antes es relacionado al cambio de capacidades en *hardware*, sin embargo, al observar las métricas generales del modelo RNN contra las mismas métricas del Cuadro 6.10 podemos observar que la RNN cuenta con un mejor aprendizaje, lo cual también se puede comprobar al observar las métricas de manera individual con las enfermedades. Nuevamente si observamos los pacientes con IMEST en precisión se obtuvo un resultado individual insatisfactorio para el proyecto, lo cual resalta que es necesario contar con más pacientes con esta enfermedad para poder realizar un entrenamiento más profundo, este desbalance se muestra igualmente en el F1-Score. Es de relevancia observar que, en métricas individuales, la enfermedad con mayor precisión es la SCA-SEST a pesar de que los pacientes sanos cuentan con mayor información, lo anterior quiere decir que el modelo RNN tuvo un mejor aprendizaje para la enfermedad con SCA-SEST.

Cuadro 6.11: Métricas de clasificación RNN.

RNN			
Enfermedad	***Precision***	***Recall***	***F1-Score***
0 (Paciente Sano)	88 %	85 %	93 %
1 (IMEST)	76 %	96 %	84 %
2 (SCA-SEST)	91 %	86 %	89 %
Accuracy		89.82 %	
Precision		91.38 %	
Recall		89.82 %	
F1-Score		90.11 %	

Si comparamos los resultados generales del modelo SVM mostrados en el Cuadro 6.10 contra los resultados generales del modelo RNN mostrados en el Cuadro 6.11, se aprecia un mejor resultado por parte de la RNN en cada una de las métricas para esta aplicación en particular. Es necesario realizar una comparación considerando los recursos de hardware utilizados para posteriormente realizar conclusiones más profundas. Ya que hasta este punto solo se puede decir que tiene una mejor taza de aprendizaje con la misma información en comparación con una SVM.

Al tener embebidos ambos modelos se obtuvieron más métricas de interés, entre ellas se encuentra la Matriz de Confusión, esta métrica se obtuvo para cada modelo de manera individual y se pueden observar en las Figuras 6.8 y 6.9. Donde el eje de las abscisas es el eje de la enfermedad clasificada y el eje vertical es la enfermedad real, entre mayor sea la relación entre la enfermedad predicha y la real se tornará de un color más oscuro, según lo anterior se desea que la enfermedad predicha y la real tengan la menor incertidumbre posible. El comportamiento de clasificación del modelo SVM se puede observar en la Figura 6.8 y muestra una predicción muy alta para los pacientes sanos, siguiendo por los pacientes con IMEST y finalizando con los pacientes SCA-SEST. De igual manera se puede apreciar que el modelo realizó multiples predicciones de IMEST cuando la clasificación real eran los pacientes sanos.

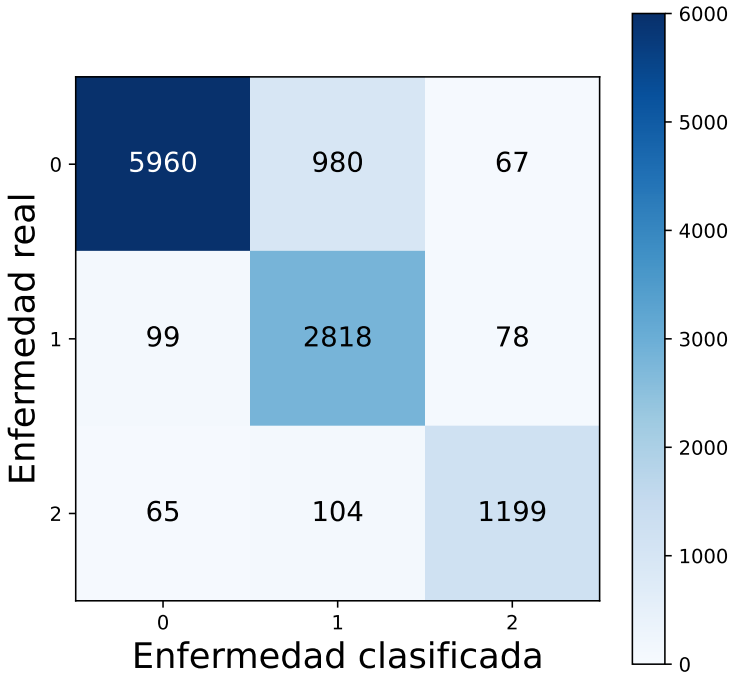

Figura 6.8: Matriz de confusión SVM.

De igual manera la Matriz de Confusión del modelo SVM mostrada en la Figura 6.8 tiene relación directa con lo interpretado en el Cuadro 6.10. Ya que para los pacientes sanos se tiene el color más oscuro haciendo referencia a que en el Cuadro la métrica de precisión es la más alta de manera individual, es importante no confundir para el caso IMEST ya que a pesar de contar con un color muy oscuro también se cuenta con una mayor cantidad de clasificaciones erroneas, lo cual disminuye su precisión es por eso que debe observarse a detalle todas las gráficas.

En la matriz de confusión de la RNN que se observa en la Figura 6.9 podemos resaltar que las predicciones de enfermedades realizadas por el modelo son muy acertadas con respecto a las enfermedades reales. Al igual que en la matriz de confusión de la SVM se puede apreciar que existen predicciones en la enfermedad IMEST cuando la clasificación real es paciente sano. Debido a que este acontecimiento se presentó en ambos modelos se puede inferir que está relacionado directamente con la base de datos utilizada en el entrenamiento de ambos modelos. Realizando una comparación de ambas matrices de confusión se puede apreciar que el modelo RNN tiene una menor tasa de error al clasificar las enfermedades IMEST y SCA-SEST, por su parte la SVM tiene una mayor acertividad al predecir a los pacientes sanos siendo confirmado por los resultados mostrados en los Cuadros 6.10 y 6.11.

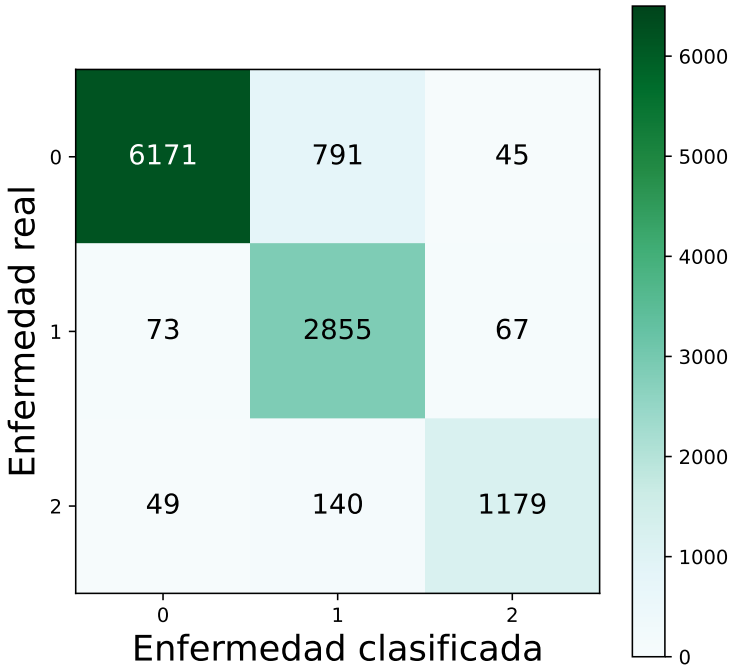

Figura 6.9: Matriz de confusión RNN.

Para el modelo SVM también se obtuvo su curva de aprendizaje ROC el cual se puede apreciar en la Figura 6.10, si realizamos una comparación rápida con la Figura 6.11 se puede observar que el modelo SVM cuenta con una mejor taza de aprendizaje contra el modelo RNN, esto sustenta lo obtenido en las matrices de confunsiones. Detallando la curva de aprendizaje para el modelo SVM se tiene que tarda más en llegar al vértice superior, sin embargo, al tener una proximidad al vértice superior implica que tiene una alta sensibilidad y una correcta clasificación de enfermedades.

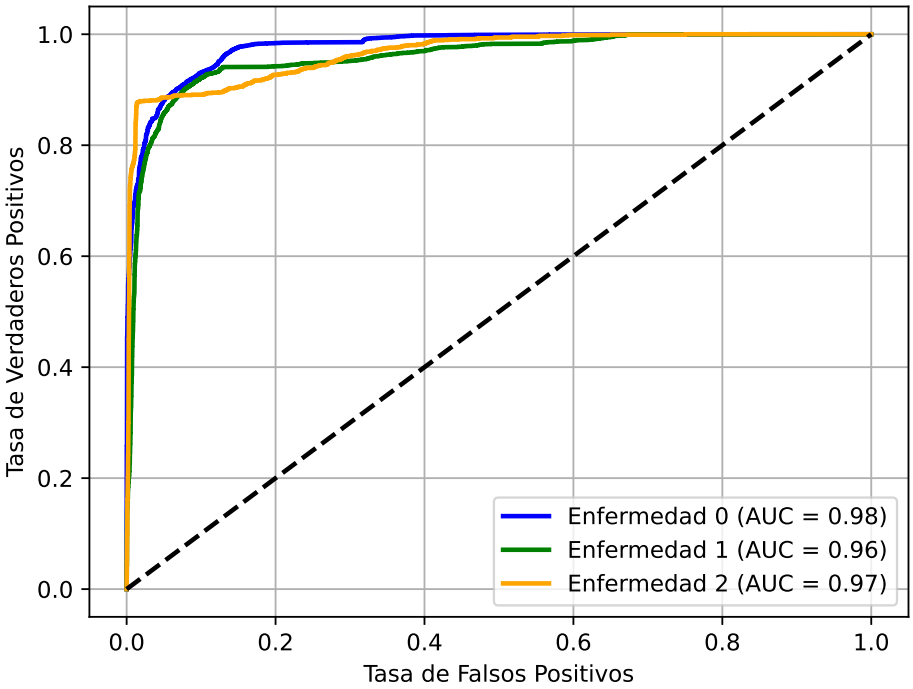

Figura 6.10: Curva de aprendizaje ROC del modelo SVM.

Por último, se obtuvo la curva ROC (por sus siglas en inglés, Receiver Operating Characteristic) del modelo RNN, la cual se presenta en la Figura 6.11. Esta gráfica permite apreciar de manera clara la tasa de aprendizaje del modelo RNN, resaltando la clase correspondiente a la enfermedad IMEST mantiene un aprendizaje más lento en comparación a las otras clases, no obstante, muestra una correcta clasificación a lo largo de las predicciones. Para los pacientes sanos y para los pacientes con SCA-SEST se mantiene una clasificación precisa y constante.

Siendo la curva AUC el área bajo la curva cercana a 1 de cada clase indica un poder discriminativo correcto de las enfermedades deseadas, lo que significa que el modelo es capaz de diferenciar de manera precisa entre las enfermedades, disminuyendo la tasa de falsos positivos como de falsos negativos. Considerando que todas las enfermedades cuentan con un AUC superior a 0.95 corrobora los resultados mostrados por el Cuadro 6.11 y por la Matriz de confusión 6.9 sobre la robustez del modelo en la clasificación de estas enfermedades, aunado a la proximidad de las curvas al vértice superior izquierdo muestra una alta sensibilidad y especificidad del modelo.

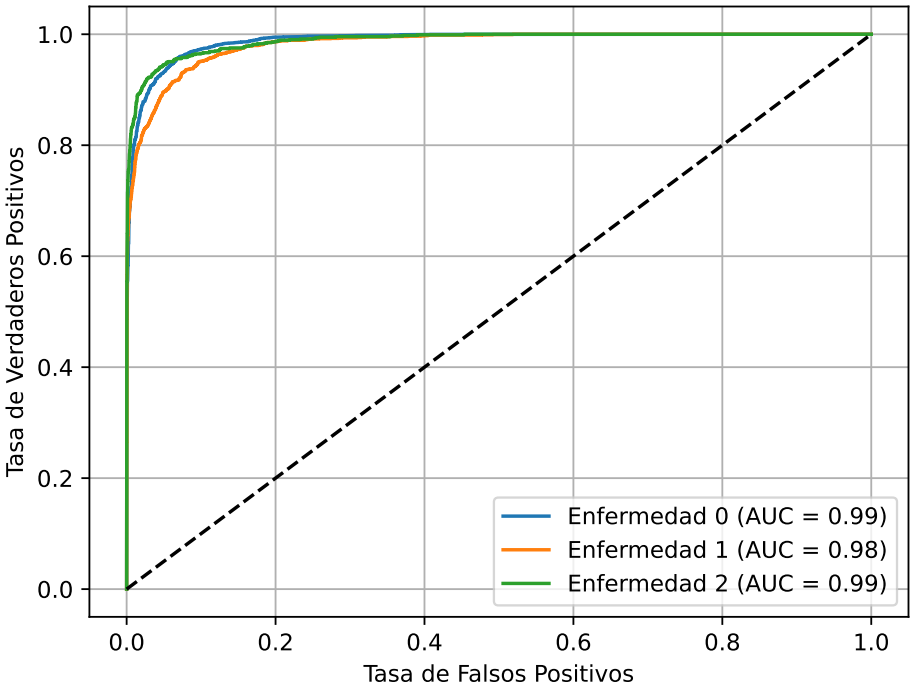

Figura 6.11: Curva de aprendizaje ROC del modelo RNN.

Una vez discutidos los modelos se procede a analizar su rendimiento en *hardware*, el Cuadro 6.12 muestras algunas métricas resultantes al ejecutar los modelos dentro del sistema embebido y promediar los resultados. El tiempo de inferencia total del *Script* fue de $10.31s$ de los cuales aproximadamente 2 terceras partes fueron consumidas por el modelo SVM ya que este cuenta con un tiempo de ejecución de $6.40s$ mientras que la RNN tuvo un tiempo de ejecución para las predicciones de $3.61s$. Esto es atribuido a que la RNN fue entrenada con la consideración mínima de capas y neuronas para su ejecución por la limitante en *hardware* mientras que con la SVM únicamente se pudieron realizar iteraciones con hiperparámetros.

Al observar el uso promedio de CPU para la SVM se cuenta con un 24.64 % y para la RNN de 25.81 % lo cual se infere en el uso completo del primer núcleo de procesamiento y un uso mínimo del segundo núcleo del procesador para ambos modelos, resaltando que el sistema embebido es capaz de soportar modelos de IA más complejos, de igual manera hace referencia a la ligereza de los modelos utilizados ya que al momento de ser entrenados y compilados se contempló el uso mínimo de recursos para el *hardware*. Se puede apreciar un mayor uso de CPU para la RNN, sin embargo, no es de gran relevancia al compararlo con la SVM y contrastarlo con las capacidades del sistema embebido.

Al observar la máxima temperatura registrada por el sistema embebido tenemos una temperatura de $56.5° \ C$ si consideramos las especificaciones del fabricante de una temperatura promedio en *ralentí* aproximada de $40.0° \ C$ y un *thermal throttling* (disminución de velocidad del procesador) en $70.0° \ C$, respalda la ligereza de los modelos ejecutados ya que aún se cuenta con rango de temperatura para hacer uso del sistema embebido. Cabe aclarar que este parámetro siempre queda a variables externas debido a que como se observó en la Figura 6.6 se cuentan con extractores de calor con la carcasa diseñada por el fabricante, sin embargo, también es de consideración el lugar

de uso, hora y clima donde se encuentre ejecutando el sistema embebido.

Por su parte el Cuadro 6.12 nos muestra el uso máximo de memoria RAM siendo de 558.80 MB para la SVM y de 579.78 MB, considerando que el modelo embebido cuenta con una memoria RAM de 8 GB se puede observar que se utilizó menos del 10 % de su capacidad total en ambos modelos, considerando ejecuciones en segundo plano y el tiempo de inferencia total de los modelos. Por lo que se puede resaltar que el sistema embebido tiene capacidad de ejecutar modelos de IA más complejos a los utilizados en este proyecto. La diferencia entre el uso de ambos modelos es mínima siendo de aproximadamente 21 MB.

Cuadro 6.12: Métricas de *hardware* de los modelos de IA.

Métrica	**Valor**
Tiempo de inferencia total	10.31 s
Tiempo inferencia SVM	6.40 s
Tiempo inferencia RNN	3.61 s
Uso promedio CPU SVM	24.64 %
Uso promedio CPU RNN	25.81 %
Máxima memoria RAM utilizada SVM	558.80 MB
Máxima memoria RAM utilizada RNN	579.78 MB
Máxima temperatura CPU	56.50° C
Consumo energético	4.90 W
Voltaje consumido	5.22 V
Máximo pico de corriente energético	0.94 A

Para el consumo energético se realizó una medición con un *USB Tester* marca *KEWEISI* modelo *KWS-V20* en el cual se tomaron mediciones durante la ejecución de los modelos de IA como se puede apreciar en la Figura 6.12 siendo los valores pico los mostrados en el Cuadro 6.12 con una potencia total de 4.90 W, si consideramos que el sistema embebido está diseñado para una máxima potencia de 15.30 W se cuenta con un uso aproximado de una tercera parte de la capacidad máxima del sistema lo cual nuevamente el bajo costo computacional de los modelos en consumo de recursos.

Figura 6.12: Medición de consumo energético por parte del sistema embebido.

Adicional se realizaron mediciones del sistema embebido en modo *ralentí* para conocer su consumo energético base, estas mediciones se pueden apreciar en la Figura 6.13, teniendo un consumo de corriente de 0.69 A dando una potencia mínima consumida de 3.60 W por lo que se realizó una diferencia entre la potencia mínima consumida por el sistema embebido y la potencia máxima al utilizar los modelos se tiene un consumo de 1.30 W por el uso de los modelos.

Figura 6.13: Mediciones *ralentí* del sistema embebido.

6.4. Discusión

El procesamiento de las señales fue el principal problema a resolver durante este proyecto debido a que sin el apoyo de los especialistas en la salud y sus anotaciones en las bases de datos no hubiese sido posible haber adquirido las características deseadas para el entrenamiento de los modelos. A pesar de contar con las anotaciones explícitas de los especialistas fue necesario realizar el procesamiento de las señales de manera individual y supervisada por pequeños tramos de cada señal para corroborar la correcta adquisición de características o reajustar las ventanas a fin de encuadrar lo obtenido con las anotaciones de los especialistas. Para el entrenamiento de los modelos se realizó con cantidad de pacientes desbalanceada lo cual influye directamente en el resultado de la predicción de los modelos por lo que utilizar una base de datos mayormente balanceada mejorará los resultados de los modelos utilizados.

Por su parte el método de ablación fue de gran utilidad para comprobar la selección de características para los modelos de IA. Este método mostró una correcta selección de características, por otro lado, a su vez mostró una gran robustez por parte de los modelos para aprender a pesar de no contar con todas las características seleccionadas.

Existen distintos modelos de IA capaces de detectar infartos, dos de estos son las SVM y las RNN, estos modelos superaron el 84 % en precisión, métrica establecida a superar basada en el estado del arte. A pesar que ambos modelos cuentan con la capacidad de realizar predicciones y clasificación múltiple de los infartos estos modelos cuentan con diferencias que se pueden observar a lo largo de los resultados. Una de las principales diferencias se presenta en el tiempo de inferencia al ejecutar el modelo, donde la RNN obtuvo un menor tiempo de inferencia lo que traduce en una predicción más rápida. Por otro lado si observamos las métricas generales de ambos modelos (*Presicion, Accuracy, Recall y F1-Score*) podemos observar que la RNN mostró una mejor respuesta en comparación con la SVM, sin embargo, la diferencia no es mayor a un 5 % lo cual muestra gran robustez por parte de ambos modelos. Si se considera la cantidad de pacientes con las características extraídas de los mismos tener una diferencia del 5 % puede ser entendible como no ser de gran relevancia, sin embargo, el contemplar una mejora en métricas y una mejora en tiempo de iteración muestra que la RNN puede ser más versatil para su implementación en sistemas embebidos.

Por parte del sistema embebido, las características de la *raspberry Pi 4B* han mostrado gran versatilidad para poder ejecutar modelos de IA al utilizar aproximadamente el 25 % de su capacidad total en CPU y un 10 % de su capacidad total en memoria RAM. Al ser un sistema embebido con gran documentación, accesible en el mercado y con los resultados mostrados en el proyecto, se vuelve una opción viable para la ejecución de proyectos con modelos de IA dedicados.

Realizando una comparación con [28] quien utilizó la misma base de datos y una RNN para la detección de infartos, se ha logrado superar en un 3.18 % el *accuracy* para una clasificación múltiple. Sin embargo, mediante su metodología de creación del modelo RNN lograron utilizar únicamente 38.48 MB a diferencia del presentado en este trabajo con $579.78MB$ de memoria RAM en su sistema embebido (*STM32*) lo cual muestra la versatilidad del modelo de IA para ser entrenado y embebido en distintos dispositivos. Por su parte [28] realizó su estudio enfocado en la comparación de los distintos modelos de RNN (RNN, GRU, LSTM, BiLSTM) utilizando la misma base de datos en contraste con el comparado en este trabajo el cual tuvo un enfoque contrastando dos modelos con distinta topología. Es importante resaltar que el trabajo presentado realiza el entrenamiento con la base de datos [50] y fue validada con [51] mientras que [28] realizó el entrenamiento y validación con la misma base de datos diviendo la base de una forma $70\,\% - 30\,\%$ siendo 70 % para el entrenamiento y el 30 % de validación, lo cual denota que la metodología utilizada en este trabajo

ofrece una versatilidad al momento de ser implementada. Gracias a lo anterior se puede inferir que los resultados muestran una metodología competitiva con lo reportado en literatura previa, si bien los sistemas embebidos cuentan con distintas capacidades en *hardware* es posible realizar ajustes para ejecutar los modelos utilizando los menores recursos posibles.

En el Cuadro 6.13 se realiza una pequeña comparación cualitativa de este trabajo respecto a trabajos previos, dentro del mismo se resaltan las aportaciones de este trabajo y proyecciones futuras de esta línea de investigación.

Cuadro 6.13: Comparación entre propiedades del estado del arte.

Trabajo	**Clasificación Múltiple**	**Sistema Embebido**	**Validado**	**Implementación Online**
[21]				
[24]				
[25]	X			
[23]	X			
[28]	X	X		
Este trabajo	X	X	X	

CAPÍTULO 7

Conclusiones

En este capítulo se detallarán las conclusiones, proyecciones e impactos del proyecto con base a lo obtenido en los resultados.

7.1. Impactos

A continuación se discutirán los principales impactos que tendrá el desarrollo de esta investigación a lo largo de distintos enfoques de interés.

7.1.1. Impacto científico

A pesar de contar con resultados prometedores mediante la exploración de esta metodología, se debe complementar con otras metodologías que refuercen los puntos de mejora de la misma, como lo puede ser el procesamiento de extracción de características en tiempo real y emplear otros sistemas embebidos dedicados a una aplicación de inteligencia artificial.

7.1.2. Impacto tecnológico

Se demostró que el sistema embebido *Raspberry Pi 4B* es capaz de ejecutar modelos de inteligencia artificial previamente entrenados, sin embargo, se detectó que se cuenta con un alto tiempo de inferencia debido a que el sistema embebido ejecuta aplicaciones en segundo plano, por lo que se recomienda utilizar un sistema embebido dedicado a la aplicación para poder disminuir el tiempo de inferencia.

7.1.3. Impacto social

Durante el desarrollo de este proyecto no fue posible tener un impato social directo, sin embargo, se detectó un apoyo indirecto explorando nuevas metodologías que pueden ser replicadas en entornos clínicos para beneficio de la sociedad.

7.2. Productos logrados

El desarrollo de la investigación culminó de manera exitosa teniendo como productos finales los siguientes:

- Preparación de artículo de divulgación científica.
- Preparación artículo indexado de contribución científica.
- Dos modelos de inteligencia artificial con la capacidad de detectar y clasificar infartos.
- Un sistema embebido capaz de ejecutar modelos de inteligencia artificial dedicados a un objetivo específico.

7.3. Conclusiones

El corazón de este proyecto se centró en el uso de Máquinas de Soporte Vectorial y Redes Neuronales Recurrentes para la detección de enfermedades cardiovasculares. El procesamiento de la señales se consolidó como el desafío principal de este proyecto debido a que el éxito de esta etapa tiene un impacto directo y en gran magnitud para las siguientes etapas del proyecto.

La metodología mostrada logró superar las metas propuestas, lo que denota el uso beneficioso de modelos de inteligencia artificial en el ambiente clínico. Este proyecto resaltó que existen maneras de utilizar modelos de inteligencia artificial con *hardware* limitado para tareas dedicadas como la clasificación de enfermedades cardiovasculares. A pesar que los resultados son prometedores, no sustituye al personal especialista por lo que se plantea como herramienta tecnológica para el apoyo de un diagnóstico.

7.4. Trabajo a futuro

A pesar de haber concluido satisfactoriamente con las metas propuestas al inicio de la investigación, este trabajo muestra una brecha importante de investigación donde se deben explorar otras metodologías capaces de detectar otras arritmias cardiacas englobadas en el infarto, como taquicardia ventricular o taquicardia auricular con el uso de distintas bases de datos que puedan aportar mayor información.

Otra área de oportunidad denotada durante esta investigación es el procesamiento de las señales en tiempo real, ya que la extracción de características tiene grandes beneficios, sin embargo, resulta ser su mayor área oportunidad por lo que se sugiere realizar trabajos enfocados a esta metodología teniendo en consideración la señal extraída sin filtros ni procesamientos matemáticos de limpieza de ruido o perturbaciones externas por lo que se sugiere utilizar redes neuronales convolucionales y redes neuronales recurrentes directamente sobre las señales para observar su comportamiento.

Bibliografía

[1] I. del Estudio, "Registro nacional de infarto de miocardio agudo ii. renima ii," *Revista Peruana de Cardiología*, vol. 39, no. 1, p. 60, 2013.

[2] S. A. Andrade Espinosa *et al.*, "Predictores de mortalidad hospitalaria en pacientes con infarto de miocardio con elevación del st en un hospital de referencia del sureste mexicano.," 2019.

[3] A. N. Pollak, *Atención Prehospitalaria Básica*. Jones and Bartlett learning, 2017.

[4] M. Rosas-Peralta and F. Attie, "Enfermedad cardiovascular: Primera causa de muerte en adultos de méxico y el mundo," *Archivos de cardiología de México*, vol. 77, no. 2, pp. 91–93, 2007.

[5] A. C. Piombo, S. Salzberg, T. Lowenberg, C. Grasso, B. Finaret, S. Golub, H. Fernández, and M. Chirico, "Epidemiología del infarto agudo de miocardio en los hospitales públicos de la capital federal," *Rev Argent Cardiol*, vol. 67, pp. 201–7, 1999.

[6] J. A. Battilana-Dhoedt, C. Cáceres-de Italiano, N. Gómez, and O. A. Centurión, "Fisiopatología, perfil epidemiológico y manejo terapéutico en el síndrome coronario agudo," 2020.

[7] A. Cordero, R. Lopez-Palop, P. Carrillo, A. Frutos, S. Miralles, C. Gunturiz, M. García-Carrilero, and V. Bertomeu-Martínez, "Cambios en el tratamiento y el pronóstico del síndrome coronario agudo con la implantación del código infarto en un hospital con unidad de hemodinámica," *Revista Española de Cardiología*, vol. 69, no. 8, pp. 754–759, 2016.

[8] C. Ricaurte-Carmona and C. A. Saldarriaga-Saldarriaga, "Diagnóstico del síndrome coronario agudo en primer nivel de atención en colombia e indicaciones de traslado emergente a mayor nivel de complejidad,¿ es posible sin enzimas cardiacas?," *Iatreia*, vol. 35, no. 4, pp. 433–446, 2022.

[9] L. Rouhiainen, "Inteligencia artificial," *Madrid: Alienta Editorial*, pp. 20–21, 2018.

[10] E. J. C. Suárez, "Tutorial sobre máquinas de vectores soporte (svm)," *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, vol. 1, pp. 1–12, 2014.

[11] J. Díaz-Ramírez, "Aprendizaje automático y aprendizaje profundo," *Ingeniare. Revista chilena de ingeniería*, vol. 29, no. 2, pp. 180–181, 2021.

[12] C. Arana, "Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales," tech. rep., Serie Documentos de Trabajo, 2021.

[13] https://www.gob.mx/salud/prensa/490-cada-ano-220-mil-personas-fallecen-debido-a-enfermedades-del-corazon.

[14] T. G. R. Lira, "Mortalidad del infarto agudo al miocardio con elevación del st acorde a la modalidad de reperfusión en el hospital general de querétaro de julio 2021 a diciembre 2022," Master's thesis, Universidad Autónoma de Queretaro, August 24th 2023.

[15] J. H. Moreno, "Estimación del tiempo puerta electrocardiograma en pacientes con síndrome coronario agudo que ingresan al servicio de urgencias del hospital general querétaro," Master's thesis, Universidad Autónoma de Queretaro, June 5th 2015.

[16] G. Raileanu, "Eelectrocardiogram interpretation using artificial intelligence: Diagnosis of cardiac and extracardiac pathologic conditions. how far has machine learning reached?," *Current Problems in Cardiology*, vol. 49, no. 1, 2023.

[17] C. Berry, "Coronary computed tomography angiography vs functional stress imaging to triage chest pain in the emergency room?," *Vascular Pharmacology*, vol. 154, March 2024.

[18] K. Subramanian and N. K. Prakash, "Machine learning based cardiac arrhythmia detection from ecg signal," in *2020 Third International Conference on smart systems and inventive technology (ICSSIT)*, pp. 1137–1141, IEEE, 2020.

[19] K. He, W. Liang, S. Liu, L. Bian, Y. Xu, C. Luo, Y. Li, H. Yue, C. Yang, and Z. Wu, "Long-term single-lead electrocardiogram monitoring to detect new-onset postoperative atrial fibrillation in patients after cardiac surgery," *Frontiers in Cardiovascular Medicine*, vol. 9, p. 1001883, 2022.

[20] C.-Y. Hsu, P.-Y. Liu, S.-H. Liu, Y. Kwon, C. J. Lavie, and G.-M. Lin, "Machine learning for electrocardiographic features to identify left atrial enlargement in young adults: Chief heart study," *Frontiers in Cardiovascular Medicine*, vol. 9, p. 840585, 2022.

[21] A. Öztekin and B. Özyılmaz, "A machine learning based death risk analysis and prediction of st-segment elevation myocardial infarction (stemi) patients," *Computers in Biology and Medicine*, vol. 188, p. 109839, 2025.

[22] S. W. A. Sherazi, H. Zheng, and J. Y. Lee, "A machine learning-based applied prediction model for identification of acute coronary syndrome (acs) outcomes and mortality in patients during the hospital stay," *Sensors*, vol. 23, no. 3, p. 1351, 2023.

[23] G. B. Berikol, O. Yildiz, and İ. T. Özcan, "Diagnosis of acute coronary syndrome with a support vector machine," *Journal of medical systems*, vol. 40, no. 4, p. 84, 2016.

[24] A. Dhawan, B. Wenzel, S. George, I. Gussak, B. Bojovic, and D. Panescu, "Detection of acute myocardial infarction from serial ecg using multilayer support vector machine," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2704–2707, IEEE, 2012.

[25] A. K. Dohare, V. Kumar, and R. Kumar, "Detection of myocardial infarction in 12 lead ecg using support vector machine," *Applied Soft Computing*, vol. 64, pp. 138–147, 2018.

[26] A. Minic, L. Jovanovic, N. Bacanin, C. Stoean, M. Zivkovic, P. Spalevic, A. Petrovic, M. Dobrojevic, and R. Stoean, "Applying recurrent neural networks for anomaly detection in electrocardiogram sensor data," *Sensors*, vol. 23, no. 24, p. 9878, 2023.

[27] S. Singh, S. K. Pandey, U. Pawar, and R. R. Janghel, "Classification of ecg arrhythmia using recurrent neural networks," *Procedia computer science*, vol. 132, pp. 1290–1297, 2018.

[28] L. Falaschetti, M. Alessandrini, G. Biagetti, P. Crippa, and C. Turchetti, "Ecg-Based Arrhythmia Classification using Recurrent Neural Networks in Embedded Systems," *Procedia Computer Science*, vol. 207, pp. 3479–3487, 2022.

[29] G. P. Garza, "El electrocardiograma y su tecnología," *AVANCES*, 2020.

[30] C. V. Silva, "Diseño e implementación de un sistema electrocardiográfico digital," *Revista Facultad de Ingeniería Universidad de Antioquia*, 2010.

[31] A. H. Association, *Soporte Vital Cardiovascular Avanzado*. Orora Visual, 2021.

[32] A. V. Martínez, *Manual de medicina de urgencias*. Editorial El Manual Moderno, 2002.

[33] "Procesamiento digital de señales."

[34] L. S. Maté, D. O. P. Trenard, and M. C. González, "Introducción a la limpieza y análisis de los datos," 2019.

[35] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Elsevier/Morgan Kaufmann, 2012.

[36] Z.-H. Zhou, *Machine Learning*. Springer, 2021.

[37] G. A. BETANCOURT, "Las máquinas de soporte vectorial (svms)," *SCIENTIA ET TECHNICA*, 2005.

[38] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2022.

[39] A. C. Franco, "Deep learning," 2019.

[40] C. C. Aggarwal, *Neural networks and deep learning: A textbook*. Springer, 2019.

[41] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into deep learning*. Cambridge University Press, 2023.

[42] I. B. Cruz, "Redes neuronales recurrentes para el análisis desecuencias," *Revista Cubana de Ciencias Informáticas*, 2007.

[43] M. C. Manchado, "Predicción demanda eléctrica española. implementación de redes neuronales recurrentes en python.," 2018.

[44] L. Wu, Y.-S. Won, D. Jap, G. Perin, S. Bhasin, and S. Picek, "Explain some noise: Ablation analysis for deep learning-based physical side-channel analysis.," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 717, 2021.

[45] S. Sheikholeslami, "Ablation programming for machine learning," 2019.

[46] M. Aviles, L.-M. Sánchez-Reyes, R. Q. Fuentes-Aguilar, D. C. Toledo-Pérez, and J. Rodríguez-Reséndiz, "A novel methodology for classifying emg movements based on svm and genetic algorithms," *Micromachines*, vol. 13, no. 12, p. 2108, 2022.

[47] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, "Ablation studies to uncover structure of learned representations in artificial neural networks," in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pp. 185–191, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2019.

[48] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, aug 4 2019.

[49] S. Raschka, Y. H. Liu, and D. Dzhulgakov, *Machine Learning with Pytorch and Scikit-Learn*. Packt Publishing, feb 25 2022.

[50] "MIT-BIH Arrhythmia Database". PhysioNet. Accedido el 15 de enero de 2025. [En línea]. Disponible: https://www.physionet.org/content/mitdb/1.0.0/.

[51] R.-D. Bousseljot, D. Kreiseler, and A. Schnabel, "The PTB Diagnostic ECG Database," 2004.