ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERIA DE SISTEMAS

RECONOCIMIENTO DE EMOCIONES USANDO CARACTERÍSTICAS EN LOS DOMINIOS DEL TIEMPO, FRECUENCIA Y TIEMPO-FRECUENCIA EXTRAÍDAS DE UN DATASET DE EEG USANDO TÉCNICAS DE APRENDIZAJE AUTOMÁTICO

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENICIAS DE LA COMPUTACIÓN

AUTOR: TEJADA BUSTILLOS JONATHAN MOISÉS

jonathan.tejada@epn.edu.ec

DIRECTOR: MYRIAM HERNÁNDEZ ÁLVAREZ

myriam.hernandez@epn.edu.ec

CERTIFICACIONES

Yo, Jonathan Moisés Tejada Bustillos declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Jonathan Tejada

Certifico que el presente trabajo de integración curricular fue desarrollado por Jonathan Tejada, bajo mi supervisión.

Dra. Myriam Hernández Álvarez

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Jonathan Moisés Tejada Bustillos

Dra. Myriam Hernández Álvarez

DEDICATORIA

Dedico este documento a mis padres Moisés Tejada y Susana Bustillos que han sabido enseñarme y mostrarme lo valioso del esfuerzo y el mérito propio, que han estado presentes en cada una de las etapas de mi vida y más que eso que me han apoyado incondicionalmente.

Jonathan Moisés Tejada Bustillos

AGRADECIMIENTO

Agradecimiento

Mi agradecimiento va dirigido a Dios que es el dador de vida y me ha permitido culminar esta etapa de vida con excelencia y tengo la certeza que será el principio de los muchos éxitos que tendré.

A mis padres que me han inculcado en el amor, el respeto, la perseverancia, la responsabilidad y la honestidad y me han encaminado en alcanzar cada una de mis metas con humildad y esfuerzo. Este camino ha sido largo, de muchos tropiezos, sin embargo, he podido levantarme de cada uno de ellos y continuar hacia mi meta que hoy por hoy la veo cumplirse. Mis padres me han demostrado que siempre puedo contar con su apoyo incondicional y que la familia es lo más importante.

Agradezco tanto, que en este proceso de vida llamado universidad, no hayan faltado los amigos que hicieron de los días malos un mejor día, siempre me sentiré feliz de saber que mis padres se sienten orgullosos del hijo en el que me he convertido.

A mis hermanas les agradezco por ser esa parte dulce y frágil que me ha mostrado un equilibrio en mi vida. Gracias por la paciencia, por el apoyo, el amor y sobre todo por alegrar mis días con su existencia.

Agradezco a mis maestros que me han compartido el conocimiento con su sabiduría y prudencia, y muchas veces con firmeza formando así un profesional de bien.

Jonathan Moisés Tejada Bustillos

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presenta documento describe a detalle el proceso de extracción de características para el estudio de señales de electroencefalogramas a través de procesos y algoritmos de aprendizaje automático. Desde el procesamiento de las señales a través de su análisis matemático en los distintos dominios: tiempo, frecuencia y tiempo frecuencia, representado por un conjunto de datos bien organizados por columnas o características; pasando por la reducción de estas características a través de técnicas o algoritmos como el algoritmo PCA o el análisis de la Matriz de Correlación; hasta la definición y comparación de distintos modelos de aprendizaje automático que estudien esas características. Todo el proceso anterior descrito permite encontrar modelos inteligentes de clasificación que permitan el reconocimiento de emociones de las personas, las cuales tienen una gran y extendida importancia en distintos campos relacionados con el estudio del hombre y su interior.

PALABRAS CLAVE: EEG, Dominio del tiempo, Dominio de la frecuencia, Dominio de la tiempo-frecuencia, Extracción de características, PCA, Matriz de correlación, Aprendizaje Automático.

ABSTRACT

This document describes in detail the process of feature extraction used for electroencephalogram signals analysis and study using machine learning techniques and algorithms. From signal processing and their mathematical analysis in different domains such as: Time domain, Frequency domain and Time-Frequency domain, represented by a well-organized dataset over columns and features; through the reduction of these features using different techniques such as PCA Algorithm or the Correlation Matrix analysis; until the definition and comparison of distinct machine learning models that study those features. All the previous process allows finding new intelligent classification models for human emotion recognition, which have a great and spread importance in many fields related with mankind study and its inside.

KEYWORDS: EEG, Time domain, Frequency domain, Time-Frequency domain, Features extraction, PCA, Correlation Matrix, Machine Learning.

1. INTRODUCCIÓN

Las emociones juegan un rol en la capacidad en la toma de decisiones, percepción, interacción e inteligencia humanas. Algunas emociones no pueden ser reflejadas mediante señales fisiológicas como el habla, entonación de la voz, expresión facial y lenguaje corporal. Sin embargo, los análisis de las variaciones en los voltajes de las señales eléctricas recogidas en la superficie del cráneo como resultado de la actividad eléctrica de las neuronas en el cerebro son llamados EEG, nos permiten, tener una alternativa para detectar las emociones que no muestran señales fisiológicas externas.

Los datos obtenidos del dispositivo BCI Open-BCI que permite la extracción de señales EEG de manera no invasiva, se realizaron mediante etapas de calibración donde no se induce ninguna emoción en el individuo y etapas con emociones inducidas por intervalos.

A partir de estas señales tomadas durante un intervalo de tiempo de un individuo construimos los modelos usando características en distintos dominios. Los modelos fueron entrenados, probados y evaluados.

Se ha determinado que el patrón de señales tomados de un individuo varia de uno a otro e incluso los tomados en un mismo individuo varían en determinados intervalos de tiempo, por lo que un modelo no predecirá de manera precisa de un individuo a otro o incluso si se prueba con el mismo individuo de un momento a otro. Esta variabilidad de resultados determina la necesidad de crear aplicaciones de rápida respuesta para reconocimiento de emociones que puedan ejecutarse en tiempo real y que puedan aplicarse a pesar de las diferencias en cada individuo.

El tiempo empleado en desarrollar y entrenar un modelo usando los datos obtenidos del dispositivo son altos, debido a la cantidad de la información, lo cual dificulta implementar una aplicación en tiempo real con los datos crudos. Por esta razón, es necesario extraer características relevantes para disminuir la complejidad computacional en el proceso de reconocimiento de emociones con vistas a una posible ejecución en tiempo real.

En este contexto, se plantea que el presente trabajo desarrolle algoritmos de reconocimiento de emociones usando técnicas de aprendizaje de máquina realizando extracción y selección de características en los dominios del tiempo, frecuencia y tiempo-frecuencia para señales EEG etiquetadas en los cuadrantes del espacio valencia-excitación. De esta manera, se reduce el tamaño del dataset procesado y se pueden obtener modelos en tiempos mucho menores que incluso pueden permitir ejecuciones en tiempo real.

Para evaluar los algoritmos se elegirán métricas que permitan valorar y ajustar los modelos obtenidos. Para resolver el problema planteado, una métrica importante además de exactitud, precisión y F1, será el tiempo de ejecución.

Las emociones juegan un rol en la capacidad en la toma de decisiones, percepción, interacción e inteligencia humanas. Mientras algunas emociones no puedan ser reflejadas mediante señales fisiológicas como el habla, entonación de la voz, expresión facial y lenguaje corporal. Los análisis de las variaciones en los voltajes de las señales eléctricas recogidas en la superficie del cráneo como resultado de la actividad eléctrica de las neuronas en el cerebro son llamados EEG, nos permiten, tener una alternativa para detectar las emociones que no muestran señales fisiológicas externas.

Los datos obtenidos del dispositivo BCI Open-BCI que permite la extracción de señales EEG de manera no invasiva, se realizaron mediante etapas de calibración donde no se induce ninguna emoción en el individuo y etapas con emociones inducidas por intervalos.

A partir de estas señales tomadas durante un intervalo de tiempo de un individuo construimos los modelos usando características en distintos dominios. Los modelos serán entrenados, probados y evaluados.

Se ha determinado que el patrón de señales tomados de un individuo varia de uno a otro e incluso los tomados en un mismo individuo varia en determinados intervalos de tiempo, por lo que un modelo no predecirá de manera precisa de un individuo a otro o incluso si se prueba con el mismo individuo de un momento a otro [2]. Esta variabilidad de resultados determina la necesidad de crear aplicaciones de rápida respuesta para reconocimiento de emociones que puedan ejecutarse en tiempo real y que puedan aplicarse a pesar de las diferencias en cada individuo.

El tiempo empleado en desarrollar y entrenar un modelo usando los datos obtenidos del dispositivo son altos, debido a la cantidad de la información, lo cual dificulta implementar una aplicación en tiempo real con los datos crudos. Por esta razón, es necesario extraer características relevantes para disminuir la complejidad computacional en el proceso de reconocimiento de emociones con vistas a una posible ejecución en tiempo real.

En este contexto, se plantea que el presente trabajo desarrolle algoritmos de reconocimiento de emociones usando técnicas de aprendizaje de máquina realizando extracción y selección de características en los dominios del tiempo, frecuencia y tiempo-frecuencia para señales EEG etiquetadas en los cuadrantes del espacio valencia-excitación.

Para evaluar los algoritmos se elegirán métricas que permitan valorar y ajustar los modelos obtenidos. Para resolver el problema planteado, una métrica importante además de exactitud, precisión y F1, será el tiempo de ejecución.

1.1 Objetivo general

Reconocer emociones usando características en los dominios del tiempo, frecuencia y tiempo-frecuencia extraídas de un dataset de EEG, usando técnicas de aprendizaje automático.

1.2 Objetivos específicos

OE1: Mitigar el impacto del ruido y los artefactos contenidos en el dataset. Balancear el número de entradas que existen por cada clase.

OE2: Extraer las características en los dominios del tiempo, frecuencia y tiempo-frecuencia del dataset y seleccionar las características de mayor relevancia.

OE3: Implementar los modelos de Aprendizaje de Máquina a través del entrenamiento y prueba utilizando las características relevantes.

OE4: Evaluar la precisión, exactitud y métrica F1 de los modelos.

1.3 Alcance

En el presente proyecto el alcance contempla el desarrollo de un sistema de reconocimiento de emociones usando como entrada señales EEG obtenidas de un dataset que está etiquetado con emociones en el espacio valencia-excitación. Las fases que se van a seguir son: preprocesamiento, extracción de características, balanceo de los datos, selección de características, entrenamiento, prueba y evaluación del rendimiento.

La primera etapa es la eliminación del ruido, que puede provenir de distintas fuentes tanto internas como externas. Aquí se eliminan estos artefactos que inciden en el correcto desempeño de los algoritmos.

A partir de este dataset se extraerán las características en los dominios del tiempo, frecuencia y tiempo-frecuencia. A continuación, se realiza el balanceo de carga aquí se iguala el número

de registros que existen por cada clase que vamos a reconocer para evitar que exista sesgo hacia determinada clase.

A partir del set de características obtenidas se utilizará el análisis de componentes principales para extraer las características más relevantes y utilizar las más representativas con el fin de optimizar el rendimiento y reducir el número de entradas del modelo.

Usando las características principales, se entrenarán y probarán los modelos desarrollados, los mismos que se ajustan y afinan para obtener un rendimiento apropiado.

Finalmente se evalúa el rendimiento de los modelos con el determinado conjunto de características utilizadas.

1.4 Marco teórico

1.4.1 Señales de EEG

Un electroencefalograma es un método que permite cuantificar la cantidad de corriente que fluye en la corteza cerebral dentro de un periodo de tiempo determinado [1]. Este crea una especie de función o señal tal como se puede observar en la figura 1.4.1.1, la cual expresa, justamente, el valor de la amplitud o de la corriente en función del tiempo.

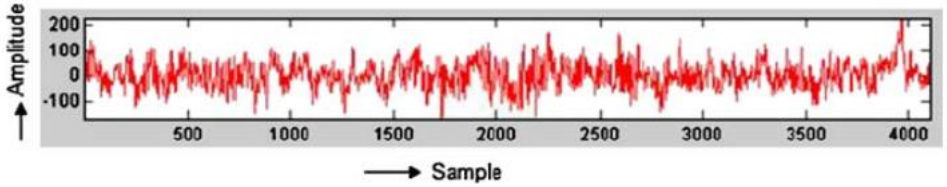


Figura 1.4.1.1. Ejemplo de una señal de un EEG.

Esta información incluida dentro de un EEG es extraída gracias al uso de electrodos que son ubicados de una forma específica en la cabeza del sujeto sometido al EEG, tal que cada uno de ellos obtiene una señal tal como la mostrada en la figura anterior, correspondiente con la zona a la cual fueron asignados [2]. Así, al colocar, por ejemplo, 8 electrodos sobre un paciente, el resultado completo final del EEG es parecido al mostrado en la figura 1.4.1.2.

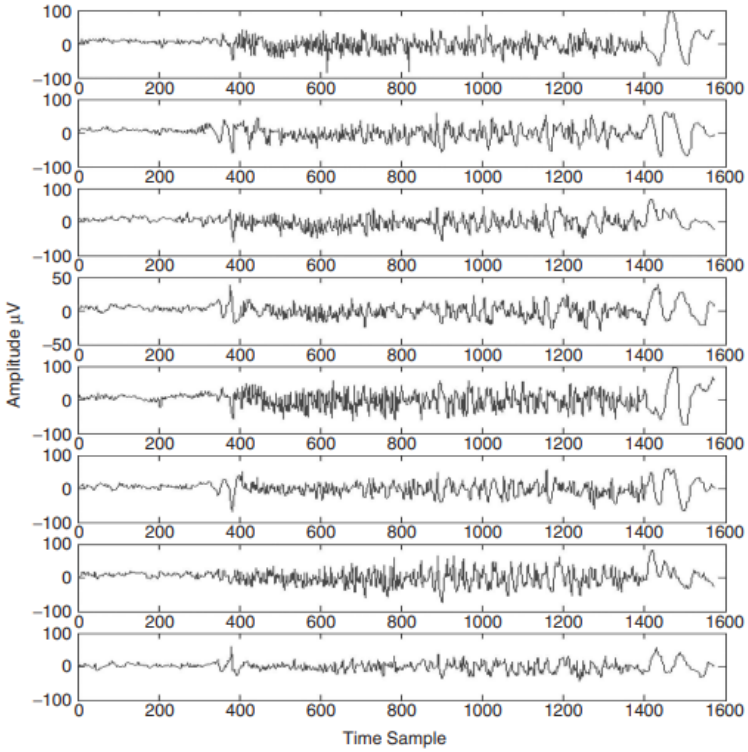


Figura 1.4.1.2. Ejemplo de un EEG de 8 electrodos [1].

Dentro de la extracción y análisis de estas señales, existe un componente o instrumento muy utilizado el cual permite conectar el cerebro directamente con un dispositivo, en el cual los datos serán almacenados para los procesos posteriores que sean necesarios; este es conocido como Interfaz Cerebro-Computador o BCI por sus siglas en inglés (Brain-Computer Interface) [3].

A través de un BCI, es posible tomar directamente los datos que son recolectados por los electrodos, que por sí solos no pueden retener información, de modo que esta sea almacenada dentro de otro dispositivo o sistema, usualmente un computador [3].

1.4.2 Análisis de señales

Si bien los BCI descritos en el subcapítulo anterior pueden ser utilizados para almacenar la información de los EEG, el mayor uso que se le da hoy en día es para el análisis de señales en tiempo real, es decir, es posible enviar la información en tiempo real de las señales del cerebro de una persona a un computador, tal que el computador sea capaz de realizar acciones basadas en la información recibida [4].

Dicho lo anterior, si bien es posible permitir que las acciones sean realizadas en base a la información que llega, la mayor parte de los casos necesitará de información derivada de la señal que recibe, por ejemplo, en base a un rango de frecuencias, a la amplitud promedio, entre muchas otras, por lo que es necesario realizar un análisis y procesamiento de la señal antes de realizar cualquier acción o estudio posterior [4].

Estos análisis se hacen en base a un estudio más profundo de la señal recibida que, generalmente, se basa en la transformación y operación de la señal por medio de herramientas matemáticas especializadas en el estudio de estas. Una de las más usadas es

el uso del Dominio de la Frecuencia por medio de la Transformada de Fourier, a partir de la cual se pueden analizar características más específicas de la señal [5].

Aun así, no son las únicas, por lo que tanto las anteriores como algunas más serán descritas posteriormente en el presente documento.

Dominio

En el contexto de las matemáticas, el dominio es la representación del conjunto de valores numéricos independientes pertenecientes a una etiqueta, de las cuáles depende una función [6].

Para entender este concepto, se describe el siguiente ejemplo:

- Un grupo de entrevistadores debe realizar una investigación para conocer cuál es el promedio de ingresos de las personas de una localidad respecto a su edad.

Independientemente de cualquier posible resultado para dicha investigación, es posible determinar dos cosas a partir de ello:

- Los ingresos, que son los valores de salida o esperados, dependen de otra variable o valor, que en este caso será la edad.
- La edad, representa el conjunto de valores independientes a partir de los cuáles se obtendrá la información del promedio de los ingresos.

Con el segundo enunciado, podemos deducir que el dominio del ejemplo propuesto es la edad, la cual tendrá un rango aproximado de 18 años hasta 100 años.

Si bien cualquier función se puede representar en cualquier dominio según las necesidades o las propiedades del objeto de estudio, dentro del campo del procesamiento de señales se han descrito 3 dominios básicos sobre el cuál una señal puede ser construida y analizada:

- Dominio del Tiempo.
- Dominio de la Frecuencia.
- Dominio del Tiempo-Frecuencia.

Cada uno de ellos será descrito a continuación.

1.4.2.1 Dominio del Tiempo

El dominio más común para la mayoría de los sucesos del mundo real representa el cambio o el valor medido de un determinado objeto de estudio respecto al tiempo [7].

Dentro del estudio de las señales en relación con su procesamiento y análisis dentro de computadores, lo más usual es trabajar con un dominio del tiempo discreto, lo cual significa que, en lugar de tomar absolutamente cada instante de tiempo, se toman solamente puntos exactos de tiempo de forma arbitraria, así como las correspondientes mediciones para dicho punto [7].

Lo anterior es de gran importancia, pues cualquier computador en el que se requiera analizar una señal no puede procesar de forma infinita cada punto de ella. Más allá de las herramientas

matemáticas que pueden utilizarse que estudian este campo de forma continua, la inclusión de componentes computacionales fuerza la creación y uso de nuevas funciones o algoritmos que puedan estudiar una señal de forma discreta.

Con ello, y dada la naturaleza de la presente investigación, se referirá a la versión discreta de cualquier algoritmo, método o función en lugar de su usual definición matemática continua.

1.4.2.2 Dominio de la Frecuencia

Frecuencia

En el contexto de las matemáticas, la frecuencia representa la cantidad de veces que ocurre un cierto fenómeno en una unidad de tiempo, usualmente medida en segundos [8].

La frecuencia está relacionada principalmente con funciones o señales que tienen un comportamiento repetitivo o periódico, la cual permite caracterizar cuántas veces un punto de la función se repite a lo largo del dominio del tiempo en un segundo [8].

Si bien lo anterior dicho es lo más común, también es posible relacionar la frecuencia a funciones no periódicas o funciones más complejas, cuya frecuencia varía de acuerdo con el punto del dominio en el que se estudie dicha función; para ello, existen herramientas como la Transformada de Fourier.

El dominio de la frecuencia permite descomponer una señal inicialmente descrita en función del tiempo en distintas componentes, cada una con una frecuencia asociada. Más detalladamente, después de realizar una serie de operaciones a través de lo que se conoce como Transformada de Fourier, una señal se representa como la suma de todas las frecuencias que componen dicha señal si se la trata como una ecuación periódica o, al menos, como la suma de funciones periódicas conocidas (senos o cosenos) que más se asemejan a la señal original [9].

Así, la figura 1.4.2.2.1 muestra una representación de una señal cualquiera, como la descomposición en las frecuencias que la representan.

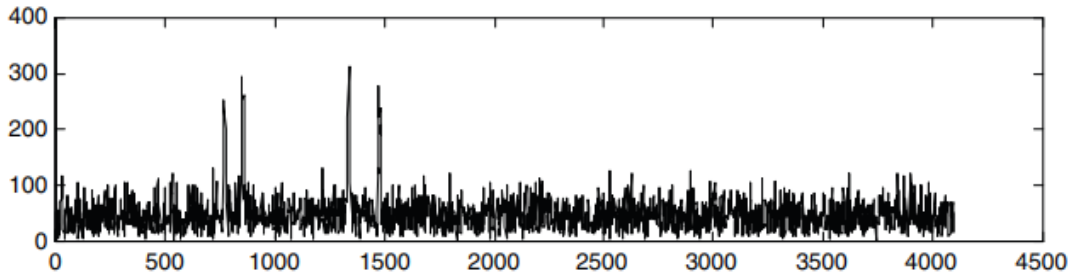


Figura 1.4.2.2.1. Ejemplo de una señal en el dominio de la frecuencia [10].

Transformada de Fourier

La Transformada de Fourier es una transformación matemática, la cual permite transformar una señal o función desde el dominio del tiempo hacia el dominio de la frecuencia, o viceversa, aplicando la Transformada Inversa de Fourier [11].

Si bien esta transformación se describe usualmente de forma continua en el ámbito de la matemática, cuando se trabaja dentro de equipos computacionales se trabaja con lo que se

conoce como la Transformada Rápida de Fourier, la cual se define tal como muestra la ecuación 1.

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} \mathrm{x_n} e^{-\frac{2\pi i}{N} kn} \quad (1)$$

Donde:

- $\mathbf{X_k}$: El k-ésimo componente en el dominio de la frecuencia.
- $\mathbf{x_n}$: El n-ésimo componente en el dominio del tiempo.
- **N**: El número de muestras de la señal que está siendo transformada.
- $\boldsymbol{e}$: El número de Euler, aproximadamente igual a 2.71828182.
- $\boldsymbol{i}$: La unidad imaginaria, equivalente a $\sqrt{-1}$.
- $\boldsymbol{k}$: Variable en el componente de la frecuencia.

Con la información descrita anteriormente, es posible describir el dominio de la frecuencia como la representación de una señal a través de su descomposición en distintas frecuencias y sus correspondientes amplitudes.

1.4.2.3 Dominio del Tiempo-Frecuencia

Como su nombre lo indica, se trata de un dominio orientado al análisis de señales, el cuál abarca a la señal en ambos dominios: tiempo y frecuencia [12].

Tanto el dominio del tiempo como el dominio de la frecuencia se centran en trabajar con la señal dentro de su respectivo dominio con sus propias características, sin embargo, estos no son capaces de abarcar por sí solos toda la información y características de una señal.

Por ello, nace el dominio de tiempo-frecuencia, que intenta extraer información de una señal desde ambos dominios, permitiendo la descripción de dicha señal dentro de un solo dominio, pero caracterizándolo más que en cada dominio individualmente [12].

Transformada de Wavelet

La transformada de Wavelet es una transformación matemática, la cual permite transformar una señal o función desde el dominio del tiempo hacia el dominio de tiempo-frecuencia, o viceversa, aplicando la Transformada Inversa de Wavelet [12].

En comparación con la Transformada de Fourier, se pudo observar en la ecuación 1 que la transformada se obtiene a través de la multiplicación de valores por una ecuación características y única (exponencial). En cambio, la Transformada de Wavelet no depende exclusivamente de una función, sino que puede trabajarse a partir de una familia de funciones, cada una con sus propias particularidades de acuerdo el objeto de estudio, que pueden definirse a través de dos variables o componentes.

- **Componente de escala:** como su nombre lo indica, este valor permite realizar una escala de la información o punto de la señal que está siendo procesada. En la figura 1.4.3.2.1, se observa un ejemplo de una función con diferentes valores de escala.

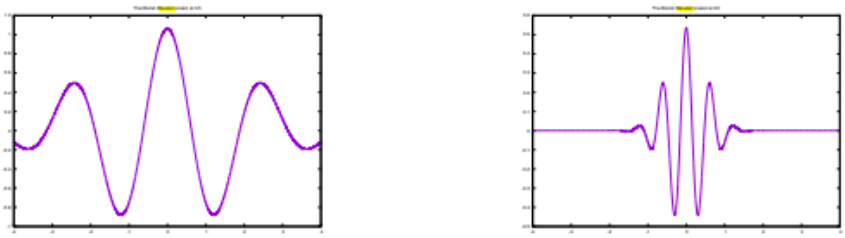


Figura 1.4.2.3.1. Wavelet a diferentes escalas [12].

- **Componente de frecuencia:** esta función permite modificar la frecuencia de la función wavelet escogida. En la figura 1.4.2.3.2, se observa un ejemplo de una función con diferentes valores de frecuencia.

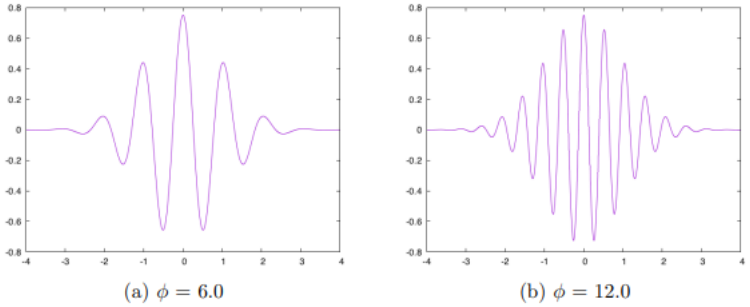


Figura 1.4.2.3.2. Wavelet a diferentes frecuencias [12].

Establecer estas dos componentes o variables de forma específica forman lo que se conoce como una Función Wavelet, que no es más que una función que cumple con ciertos criterios para que puedan ser utilizadas para ejecutar la transformación.

Así, existe una familia de funciones de Wavelet, cada una de ellas siendo más útil cuando se analizan ciertas funciones en comparación con otras, sin embargo, podemos describir la transformación de Wavelet de forma general tal que (ver Ecuación 2) [13]:

$$X_k(a,b) = \sum_{n=0}^{N-1} \mathrm{x_n} \Psi_{\mathrm{a,b}}(x_n) \qquad (2)$$

Donde:

- $\mathbf{X_k}$: El k-ésimo componente en el dominio de tiempo-frecuencia.
- $\mathbf{x_n}$: El n-ésimo componente en el dominio del tiempo.
- **N**: El número de muestras de la señal que está siendo transformada.

- $\mathbf{a}$: Variable que modifica la escala del n-ésimo punto.
- $\mathbf{b}$: Variable que modifica la traslación del n-ésimo punto.
- $\boldsymbol{\Psi_{a,b}}$: Función de Wavelet usada en la transformación**.**

La figura 1.4.2.3.3, muestra un ejemplo de una función Wavelet común en el dominio discreto, conocida como la función “Haar”, cuya distribución es semejante a la descrita en dicha imagen, pero que puede variar conforme la modificación de las componentes previamente descritas. Así, la función “Haar” se comporta como una señal con dos ventanas, una hacia arriba del eje X, y otra hacia abajo [13].

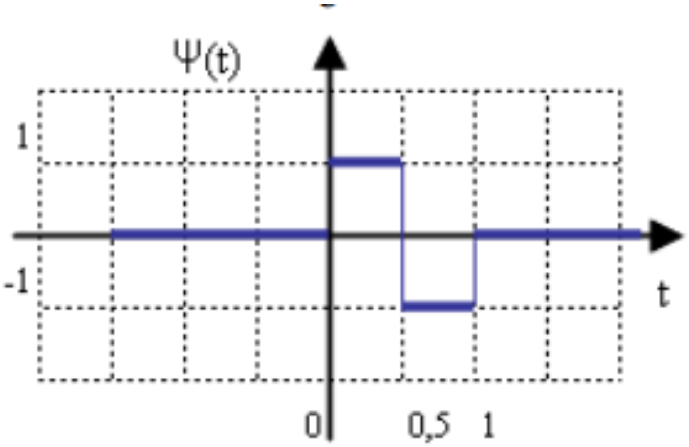


Figura 1.4.2.3.3. Función Wavelet Haar.

1.4.3 Análisis en el dominio del tiempo

1.4.3.1 Parámetros estadísticos

MEDIA

En estadística, la media es una medida de tendencia central que representa el valor promedio de un conjunto de valores, en el caso de un conjunto discreto [15]. Independientemente del conjunto de datos, siempre que sean numéricos, se puede calcular su valor mediante la ecuación 3:

$$\bar{x}_S = \frac{1}{N}\sum_{n=0}^{N-1} \mathrm{x_n} \tag{3}$$

Donde:

- $\boldsymbol{\bar{x}_S}$: La media de la señal.
- $\mathbf{x_n}$: El n-ésimo componente en el dominio del tiempo.
- $\mathbf{N}$: El número de muestras de la señal que está siendo transformada.

VARIANZA

En estadística, la varianza es una medida de tendencia central que representa a través de un valor numérico la dispersión o distribución de la información a lo largo de todo el conjunto de datos, en otras palabras, describe que tan alejados se encuentran todos los datos de la media [16]. Este valor puede calcularse a través de la ecuación 4:

$$v_S = \frac{1}{N}\sum_{n=0}^{N-1}(\mathrm{x_n} - \bar{x})^2 \quad (4)$$

Donde:

- $\boldsymbol{v_S}$: La varianza de la señal.

  $\boldsymbol{\bar{x}}$: La media de la señal.
- $\mathbf{x_n}$: El n-ésimo componente en el dominio del tiempo.
- $\mathbf{N}$: El número de muestras de la señal que está siendo transformada.

DESVIACIÓN ESTÁNDAR

En matemática, la desviación estándar es una medida de tendencia central que, al igual que la varianza, representa la separación existente entre los datos y la media de su conjunto, pero de una forma más normalizada o centralizada [16]. Así, existe una relación entre la varianza y la desviación estándar, representada por la ecuación 5:

$$s_S = \sqrt{v_S} \quad (5)$$

Donde:

- $\boldsymbol{s_S}$: La media de la señal.
- $\boldsymbol{v_S}$: La varianza de la señal.

1.4.3.2 Parámetros de Hjorth

Los parámetros de Hjorth son valores derivados de medidas estadísticas utilizados en el procesamiento de señales para la extracción de sus características. Los parámetros de Hjorth son 3:

- Actividad.
- Movilidad.
- Complejidad.

Los cuáles serán descritos con más detalle a continuación [17].

ACTIVIDAD

La actividad es una medida que describe el espectro de poder de una señal en el ámbito de la frecuencia, así, muestra el espectro o conjunto de frecuencias en el cuál la señal tiene un impacto significativo. Así (ver Ecuación 6) [17]:

$$Actividad = v_S \tag{6}$$

Donde:

- $\boldsymbol{Actividad}$: La actividad de la señal.
- $\boldsymbol{v_S}$: La varianza de la señal.

MOBILIDAD

La movilidad es una medida que representa el cambio de la frecuencia de la señal en comparación con una onda sinusoidal con las mismas características de la señal analizada, es decir, analiza que tan parecida es la señal a una función común seno cuya frecuencia es igual a la más representativa de esta señal [17]. Esta puede encontrarse a través de la ecuación 7:

$$Mobilidad = \sqrt{\frac{v_{S'}}{v_S}} \tag{7}$$

Donde:

- $\boldsymbol{Mobilidad}$: La complejidad de la señal.
- $\boldsymbol{v_{S'}}$: La varianza de la primera derivada de la señal.
- $\boldsymbol{v}$: La varianza de la señal.

COMPLEJIDAD

La complejidad es una medida que representa la proporción de desviación estándar que se encuentra dentro del espectro de poder de la señal, es decir, representa la proporción de señal con una potencia significativa en la señal [17], la cual puede ser calculada a través de la ecuación 8:

$$Complejidad = \frac{\sqrt{v_S}\ \sqrt{v_{S''}}}{v_{S'}} \tag{8}$$

Donde:

- $\boldsymbol{Complejidad}$: La complejidad de la señal.
- $\boldsymbol{v}$: La varianza de la señal.
- $\boldsymbol{v_{S'}}$: La varianza de la primera derivada de la señal.

- $v_{s''}$: La varianza de la segunda derivada de la señal.

1.4.4 Análisis en el dominio de la frecuencia

1.4.4.1 Bandas de frecuencia

Dentro de la medicina, más específicamente en el estudio de las ondas cerebrales, estas se han dividido en 5 etiquetas de acuerdo con las frecuencias a las que se detecta que está funcionando el cerebro, cada una de ellas asociada a una característica o comportamiento de la persona [18]. Siendo estas:

- Delta.
- Theta.
- Alpha.
- Beta.
- Gamma.

Delta

Se trata de la banda de frecuencia más baja [0.5 Hz a 4 Hz], y está relacionada con las señales más bajas del cerebro, principalmente, asociadas al momento de sueño profundo de las personas [18].

Theta

La segunda banda de frecuencia más baja [4 Hz a 8 Hz], y está se relacionada con los momentos más tempranos del sueño, así como con los momentos juntos antes de despertarse, donde el cerebro sigue inconsciente, pero empieza a recuperarla [18].

Alpha

La banda de frecuencia central [8 Hz a 12 Hz], asociado a la consciencia calmada de la persona, así, cuando la persona se encuentra totalmente consciente, pero en un estado más tranquilo y primitivo [18].

Beta

La penúltima banda de frecuencia [12 Hz a 30 Hz], está relacionada con una actividad del cerebro más alerta, donde el cerebro se encuentra en un estado de concentración o relacionado con la toma de decisiones o solución de problemas [18].

Gamma

La banda de frecuencia más alta [30 Hz a 100 Hz], está asociada a los procesos más activos del cerebro, cuando el cerebro necesita una mayor concentración y capacidad, como lo pueden ser procesos de memorización o gran concentración [18].

Todas estas bandas de frecuencia tienen un comportamiento normal asociado tal como se muestra en la figura 1.4.4.1.1, sin embargo, este comportamiento puede variar debido a diversas razones, principalmente relacionadas con las emociones que presentan las personas en un instante determinado, o por problemas asociados al cerebro. En el presente trabajo, se abordará el primer caso.

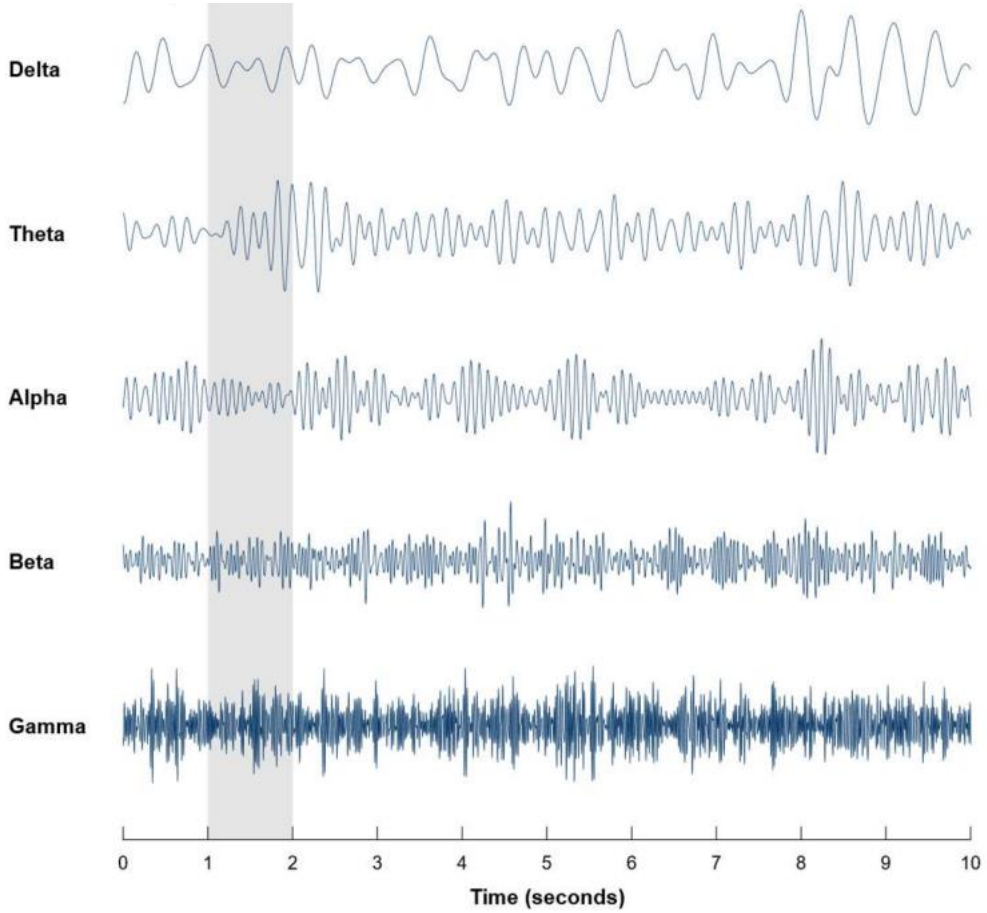


Figura 1.4.4.1.1. Bandas de Frecuencia EEG [19].

1.4.4.2 Entropía diferencial

La entropía diferencial es un método de análisis de información utilizado principalmente en estadística y en la teoría de información, cuyo propósito es describir la “sorpresa” de encontrar un punto de información dentro de una distribución de datos dada [20].

En otras palabras, dado un conjunto de datos, en este caso, una señal, la entropía diferencial calcula un valor que representa qué tan fácil es de encontrar esa señal de entre todas las posibles señales que pueden existir, de alguna forma, indicando la normalidad o qué tan fácil o común es encontrarla.

La entropía diferencial se deriva directamente de la Teoría de la Información de Shannon, a partir de la cual nace también su definición tal como se muestra a continuación [21] (ver Ecuación 9):

$$H = -\sum_{n=0}^{N-1} x_n \log(x_n) \quad (9)$$

Donde:

- $\boldsymbol{H}$: La entropía diferencial de la señal.
- $\mathbf{x_n}$: El n-ésimo componente en el dominio del tiempo.
- $\mathbf{N}$: El número de muestras de la señal que está siendo transformada.

1.4.5 Análisis en el dominio del tiempo-frecuencia

1.4.5.1 Coeficientes de Wavelet

A partir de la transformación de una señal al dominio del tiempo frecuencia por medio de la Transformada de Wavelet, es posible analizar una señal al analizar su comportamiento en relación con otra señal de interés, cuyo propósito será comparar estas dos señales y poder describir qué tan parecida es la señal que se está analizando con la señal de interés [22].

Coeficiente de Aproximación

Los coeficientes de aproximación son valores resultado de comparar las dos señales (la señal a analizar y la señal Wavelet escogida) y determinar las componentes de alta escala y baja frecuencia de la señal.

Coeficiente de Detalle

Los coeficientes de detalles son valores resultado de comparar las dos señales (la señal a analizar y la señal Wavelet escogida) y determinar las componentes de baja escala y de alta frecuencia de la señal [22].

Para poder calcular estos coeficientes, es necesario ejecutar un proceso iterativo a través el cuál se evalúan posibles valores de escala y de desplazamiento de la señal hasta encontrar aquellos coeficientes que asemejan más a la señal que está siendo transformada con la señal Wavelet usada, tal que (ver Ecuación 10):

$$c_k = ASF\left(c_{k-1}, \mathrm{x_n}, \Psi_{\mathrm{a,b}}(x_n)\right) \quad (10)$$

Donde:

- $\boldsymbol{c_k}$: El k-ésimo coeficiente de aproximación o detalle.
- $\boldsymbol{c_{k-1}}$: El (k-1)-ésimo coeficiente de aproximación o detalle.
- $\mathbf{x_n}$: El n-ésimo componente en el dominio del tiempo de la señal.
- $\boldsymbol{\Psi_{a,b}(x_n)}$: El n-ésimo componente de la función de Wavelet.

- $\boldsymbol{ASF}$**:** Representa el proceso iterativo de análisis de semejanza de las funciones, en relación con los coeficientes ya conocidos.

1.4.6 Extracción de características

La extracción de características hace referencia a un proceso dentro del campo de la analítica de datos cuya finalidad es de determinar un conjunto de etiquetas o valores que representen un objeto de estudio, tal que, al ser analizado por un modelo o una persona posterior a dicho análisis, estas sean consideradas las mínimas, pero suficientes para entender dicho concepto [23].

Usualmente, la extracción de características está relacionada con la generación de modelos o algoritmos de clasificación, donde estas características servirán como la información base de aprendizaje para dichos modelos, y su importancia recae principalmente en la eficiencia computacional y la propia precisión de la clasificación a posteriori del modelo [23].

Existen diversos métodos que permiten la extracción y definición de dichas características, desde un análisis humano y manual realizado a través de análisis matemáticos como lo puede ser la matriz de correlación las características, hasta el uso de algoritmos especializados como el Análisis de Componentes Principales.

Matriz de Covarianza o Matriz de Correlación

La matriz de covarianza o matriz de correlación es una matriz que almacena información respecto a la relación que existe entre cada una de las características de entrada con las demás. Así, la matriz de correlación se define como la ecuación 11 [24]:

$$Corr = \begin{pmatrix} \sigma_{11}^2 & \dots & \sigma_{1N} \\ \dots & \dots & \dots \\ \sigma_{N1} & \dots & \sigma_{NN}^2 \end{pmatrix} \tag{11}$$

Donde:

- $\mathbf{Corr}$: Es la matriz de correlación de las características.

Y cada valor de σ_{ij} se representa a través de la ecuación 12:

$$\sigma_{ij} = \frac{1}{N} \sum_{k=0}^{N-1} (x_{ki} - \bar{x_i})(x_{kj} - \bar{x_j}) \tag{12}$$

Donde:

- $\boldsymbol{\sigma_{ij}}$: Representa la covarianza entre la i-ésima columna y la j-ésima columna.
- $\mathbf{N}$: Es el número de características total.
- $\mathbf{x_{ki}}$: Representa el valor de la característica de la i-ésima columna.
- $\boldsymbol{\bar{x_i}}$: Representa la media de la i-ésima columna.
- $\mathbf{x_{kj}}$: Representa el valor de la característica de la j-ésima columna.

- $\bar{x_j}$: Representa la media de la j-ésima columna.

Esta matriz de correlación permite analizar la cantidad de características que están fuertemente relacionadas entre sí. Así, mientras mayor sea la covarianza calculada, implica una mayor relación entre dos características, mientras que menor sea existe una menor relación entre ellas [24].

Según estos valores de covarianza es posible decidir eliminar o combinar una o más características del total inicial a través de un análisis simple. Por ejemplo, si tenemos dos características con una correlación muy alta, significa que ambas se relacionan directamente, tal que combinarlas o eliminar una de las dos no modificaría el resultado final, mientras que disminuye la cantidad de características a procesar por la máquina.

Análisis de Componentes Principales

El Análisis de Componentes Principales o PCA por sus siglas en inglés (Principal Component Analysis), se trata de un algoritmo dentro del Aprendizaje de Máquina el cual permite reducir el número de columnas o características a ser procesadas por el computador [24].

La diferencia respecto a la matriz de covarianza es que el algoritmo PCA reduce de forma automática el número de columnas, pero igual que la anterior a través del análisis de valores numéricos y estadísticos.

Si bien existen varias formas de implementar el PCA, su uso recae en dos posibles valores que se pueden describir para su uso:

- **Número de componentes final:** que representa el número de componentes que se desean como los componentes finales después de su uso. Por ejemplo, si tenemos un número de componentes inicial de 50, y definimos un número de componentes final como 30, entonces el algoritmo eliminará internamente 20 de los 50 componentes iniciales para terminar en los 30 descritos.
- **Umbral de aceptación:** un valor que permite describir un número de componentes no estático, sino dependiente de la información de entrada. Este umbral de aceptación permite describir el número de columnas que serán eliminadas con base a la varianza de dicha columna.

1.4.7 MODELOS DE APRENDIZAJE AUTOMÁTICO

Los modelos de aprendizaje automático son herramientas derivadas de la Inteligencia Artificial, los cuales son capaces de identificar patrones y relaciones de información que han adquirido, que le permitirán realizar predicciones o clasificaciones de nuevos datos.

Dentro del aprendizaje automático, existen muchos modelos con capacidades y propósitos diferentes: algunos siendo útiles para predecir fenómenos, otros para analizar tendencias y predecir patrones, muchos otros para realizar una clasificación de información, entre otras funciones [24].

A continuación, se explicarán algunos de los modelos de aprendizaje automático a ser utilizados en el presente trabajo, relacionados con la clasificación.

1.4.7.1 K-Nearest Neighbors

Es un algoritmo de aprendizaje automático supervisado utilizado en clasificación.

Para ello, este algoritmo es suministrado con información previa conocida cuya clasificación o etiqueta es también ya conocida, tal que, en breves rasgos, resume toda aquella información como un punto en el espacio con su respectiva clasificación.

A partir de allí, cada nuevo conjunto de datos de entrada que quiera ser clasificado será comparado con estos puntos ya existentes a través de la medida de la distancia entre el punto que se le ha asignado en el espacio, y los K puntos existentes más cercanos (Siendo K un número arbitrario que representa la cantidad de vecinos a analizar), tal que, aquella etiqueta que cuenta con más puntos cercanos a la entrada, le otorgará su propia clasificación [23].

La ecuación 13, muestra el principio que utiliza este algoritmo para etiquetar la información:

$$d(x_i, f_i) = \sqrt{\sum_{k=0}^{N-1} (x_i - f_i)^2} \tag{13}$$

Donde:

- $\boldsymbol{d}$: Representa la distancia entre un valor conocido y una entrada a etiquetar.
- $\mathbf{N}$: Representa las dimensiones de las características.
- $\mathbf{x_i}$: Representa todos los componentes del valor conocido.
- $\mathbf{f_i}$: Representa todos los componentes de la entrada a etiquetar.

Esta última ecuación se calcula tantas veces como datos conocidos existan previamente registrados en el algoritmo. Una vez conocidas todas las distancias, estas son ordenas de menor a mayor donde, según el valor K escogido previamente, se verificarán cuáles son las etiquetas de los primeros K valore de la nueva lista.

Con esto, el algoritmo verifica cual es la etiqueta más repetida dentro de esos K valores, para asignar dicha etiqueta al valor de entrada que se deseaba clasificar. Así, su ejecución se puede apreciar de mejor manera en la figura 1.4.7.1.1.

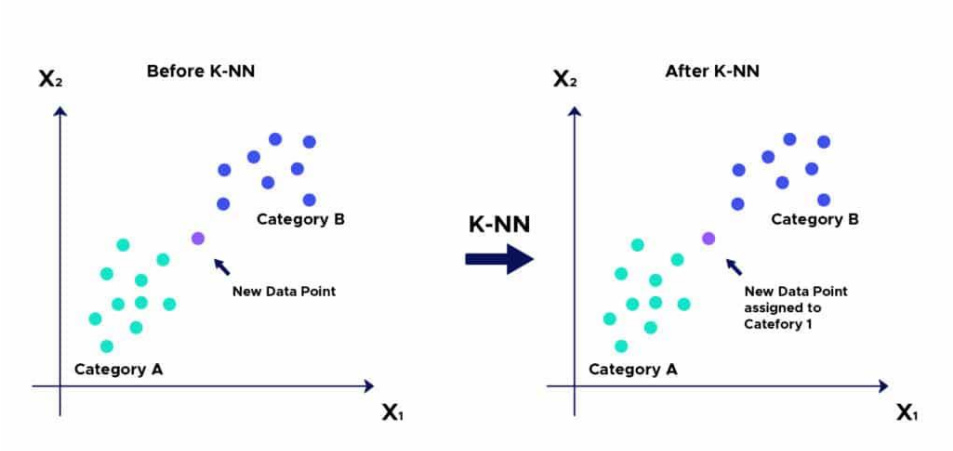


Figura 1.4.7.1.1. Representación del algoritmo KNN.

1.4.7.2 Random Forest Classifier

Es un algoritmo de aprendizaje automático supervisado utilizado en clasificación.

Para ello, este algoritmo utiliza lo que se conocen como Árboles de Decisión, que no son más que estructuras de decisión que otorgan un resultado de acuerdo con un criterio generado por el propio algoritmo en relación con los datos que ha analizado [24]. Un ejemplo de árbol de decisión se puede observar en la figura 9.

Figura 1.4.7.2.1. Representación de un Árbol de Decisión.

El algoritmo de clasificación de Random Forest lo que hace es generar un conjunto de Árboles de decisión, cuya cantidad como estructura puede ser descrita en muchas implementaciones, o generarse de forma aleatoria. A partir de ese conjunto, el algoritmo lo que hace es enviar el conjunto de datos de entrada a cada uno de esos árboles, tal que cada uno de ellos haga una predicción independiente de todos los demás [23].

Una vez realizadas las predicciones de cada Árbol, la respuesta o etiqueta final se basa en un criterio de “votación”, tal que la elección de su clasificación de Random Forest se basa en darle la etiqueta que más veces se repitió en ese conjunto dado por los Árboles de Decisión.

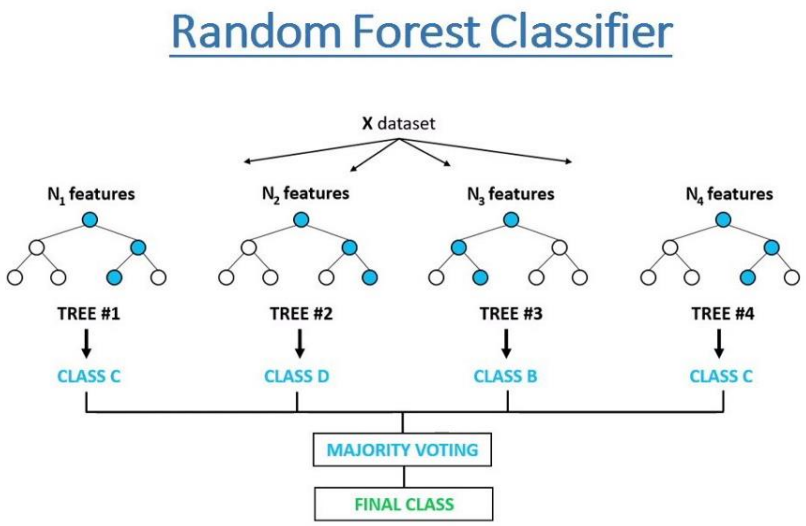


Figura 1.4.7.2.2. Representación del algoritmo Random Forest Classifier.

En la figura 1.4.7.2.2, se observa más claramente el funcionamiento del algoritmo Random Forest, tal que la información se envía a un conjunto de árboles de decisión, cada uno otorgando una clasificación, para que la clasificación o etiqueta final sea decidida por la mayor cantidad de clases o etiquetas contabilizadas en esa lista.

1.4.7.3 Deep Neural Network

Es un algoritmo de aprendizaje automático supervisado utilizado en clasificación.

Para esto, este tipo de algoritmo es trabajado a través de un esquema como si se tratase de un mapa, definido por distintas capas y neuronas, las cuáles a través de una función definida, serán capaces de modificarse a sí mismas para encontrar un punto en el cuál sean capaces de realizar una predicción [24] [25].

Con esto, cuando un nuevo conjunto de datos de entrada ingresa en el modelo y cada característica pasa por cada capa y neurona, las cuáles interactúan con dicha información antes de enviarla a un nivel más cercano a la salida, hasta llegar al final donde se encuentra el valor con el que será etiquetada la información. Una mejor visualización de este algoritmo se puede ver en la siguiente figura (ver Figura 1.4.7.3.1):

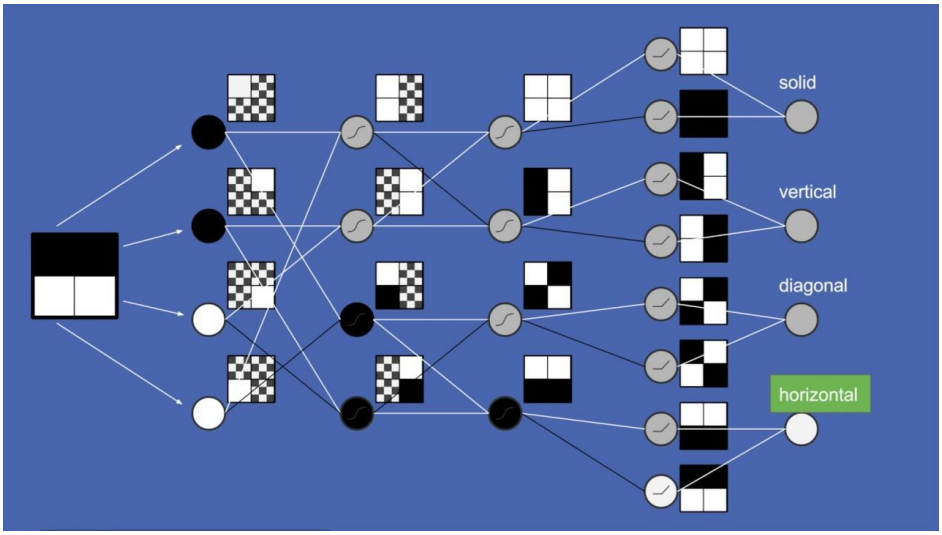


Figura 1.4.7.3.1. Representación de una Red Neuronal Profunda.

Observando la figura, vemos como la red neuronal toma características de la información que se ha ingresado, y a partir de funciones internas dicha información es transformada y transferida a la parte más a la derecha, hasta determinar un resultado.

1.4.8 Parámetros de evaluación de modelos de aprendizaje automático

1.4.8.1 Accuracy

El parámetro accuracy, exactitud en español, es una métrica que permite evaluar el funcionamiento de modelos de clasificación de aprendizaje automático, al analizar la proporción de valores evaluados correctamente respecto al total de predicciones [26] [27].

Este parámetro puede ser calculado a través de la ecuación (ver Ecuación 14) mostrada a continuación:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

Donde:

- $\boldsymbol{Accuracy}$: El valor de exactitud asociado a los resultados del modelo.
- $\boldsymbol{TP}$: El número de verdaderos positivos.
- $\boldsymbol{TN}$: El número de verdaderos negativos.
- $\boldsymbol{FP}$: El número de falsos positivos.
- $\boldsymbol{FN}$: El número de falsos negativos.

1.4.8.2 Precision

El parámetro precision, o precisión en español, es una métrica que permita evaluar modelos de clasificación, al analizar la proporción de valores positivos correctamente acertados [26] [27], así, este puede calcularse a través de la siguiente ecuación (ver Ecuación 15):

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

Donde:

- $\boldsymbol{Precision}$: El valor de precisión asociado a los resultados del modelo.
- $\boldsymbol{TP}$: El número de verdaderos positivos.
- $\boldsymbol{FP}$: El número de falsos positivos.

1.4.8.3 F1-Score

El parámetro F1-score, o puntuación F1 en español, es una métrica que permite la evaluación de modelos de aprendizaje automático de clasificación, al ejecutar una media harmónica de otras métricas como lo son la precisión y exhaustividad [26] [27].

Independientemente de la definición de las métricas anteriores, es posible calcular esta métrica a partir de la ecuación 16:

$$F1 - Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{16}$$

Donde:

- $\boldsymbol{F1 - Score}$: El valor de exactitud asociado a los resultados del modelo.
- $\boldsymbol{TP}$: El número de verdaderos positivos.
- $\boldsymbol{FP}$: El número de falsos positivos.
- $\boldsymbol{FN}$: El número de falsos negativos.

2. Revisión Sistemática de la literatura

Una revisión sistemática de la literatura es un método riguroso y estructurado para identificar, evaluar y sintetizar la evidencia existente sobre un tema de investigación específico. Esta técnica se utiliza en la investigación académica para obtener una visión general y completa de un campo de estudio y proporcionar una base para la toma de decisiones basada en evidencia.

La revisión sistemática de la literatura implica la identificación exhaustiva de todas las publicaciones relevantes en el tema de interés, incluyendo artículos de revistas, libros y otros recursos. Estas publicaciones son evaluadas y seleccionadas utilizando criterios predefinidos

de inclusión y exclusión. Luego, se realiza una síntesis crítica de la información de las publicaciones seleccionadas utilizando técnicas estadísticas y de análisis de datos.

Los componentes de la Revisión Sistemática de la literatura son los siguientes:

- Preguntas de investigación.
- Selección de palabras clave.
- Cadenas de búsqueda.
- Selección y reducción del conjunto de documentos.
- Resultados y discusión

2.1 Preguntas de investigación

Definir las preguntas de investigación definen el alcance y los objetivos de la revisión, guían la búsqueda y selección de la literatura relevante, permiten evaluar críticamente la calidad de los estudios y sintetizar los resultados, y ayudan a identificar la calidad global de la evidencia disponible.

De acuerdo con el alcance del proyecto se han definido las siguientes preguntas de investigación con el fin de alcanzar los objetivos específicos.

RQ01.- ¿Permite La extracción de características en el dominio del tiempo, frecuencia y tiempo-frecuencia de señales EEG predecir emociones relacionadas con los cuadrantes del espacio valencia – arousal usándolos como vector de entrada en algoritmos de aprendizaje automático?

RQ02.- ¿Cuáles son los algoritmos más comunes usados para los modelos de predicción?

RQ03.- ¿Cuáles son las métricas de evaluación más comúnmente usadas para evaluar los modelos de predicción?

RQ04.- ¿Cuál es el desempeño de los modelos de predicción?

RQ05.- ¿Qué porcentaje de los datos es usado en los modelos para entrenamiento validación y pruebas?

RQ06.- ¿Cuáles son las características en el dominio del tiempo de señales EEG más usadas por los modelos de predicción?

RQ07.- ¿Cuáles son las características en el dominio de la frecuencia de señales EEG más usadas por los modelos de predicción?

RQ08.- ¿Cuáles son las características en el dominio del tiempo-frecuencia de señales EEG más usadas por los modelos de predicción?

2.2 Palabras clave

La selección adecuada de palabras clave en una revisión sistemática de la literatura es esencial para identificar y recuperar de manera eficiente la información relevante para la pregunta de investigación. Las palabras clave permiten realizar búsquedas en bases de datos bibliográficas y otros recursos de información de manera sistemática y reproducible, mejorando la eficiencia de la búsqueda y la calidad de los resultados. Una selección

cuidadosa y bien fundamentada de las palabras clave puede garantizar que la revisión sea rigurosa, exhaustiva y sistemática, y que pueda ser replicada y actualizada en el futuro.

Para el diseño de las cadenas de búsqueda se han seleccionado las siguientes palabras clave las cuales resumen el objetivo general del proyecto.

EEG (electroencefalograma), emociones, detección, aprendizaje, características tiempo, frecuencia y tiempo-frecuencia, reconocimiento, preprocesamiento.

2.3 Cadenas para la búsqueda.

Las cadenas de búsqueda son una herramienta fundamental en la revisión sistemática de la literatura, ya que permiten identificar y recuperar de manera eficiente la información relevante para la pregunta de investigación.

Una cadena de búsqueda es una combinación de términos y operadores booleanos que se utilizan para buscar información en bases de datos bibliográficas y otros recursos de información. Al utilizar términos específicos y estandarizados, y combinarlos con operadores booleanos, se puede mejorar la precisión y la exhaustividad de la búsqueda, reduciendo la probabilidad de omitir información importante.

Las cadenas de búsqueda SS01, SS02, SS03 mostradas a continuación se utilizaron en el motor de búsqueda Semantic Scholar, del cual se obtuvo los resultados mostrados en la tabla 2.3.1.

SS1.- Feature extraction AND (time domain OR frequency domain OR time-frequency domain) AND emotion recognition AND valence – arousal space

SS2.- Feature extraction AND (time domain OR frequency domain OR time-frequency domain) AND Machine learning algorithms AND emotion recognition AND valence –arousal space AND EEG AND BCI.

SS3.- Feature extraction AND (time domain OR frequency domain OR time-frequency domain) AND Machine learning algorithms AND emotion recognition AND valence –arousal space AND EEG AND BCI AND evaluation metrics.

Tabla 2.3.1. Cadenas de búsqueda en Semantic Scholar.

Database Search Engine	ID	Command search	Search Date	Total
Semantic Scholar	SS01	Feature extraction AND (time domain OR frequency domain OR time-frequency domain) AND emotion recognition AND valence – arousal space	29/06/2022	152
	SS02	Feature extraction AND (time domain OR frequency domain OR time-frequency domain) AND Machine learning algorithms AND emotion recognition AND valence –arousal space AND EEG AND BCI		10
	SS03	Feature extraction AND (time domain OR frequency domain OR time-frequency domain)		1

		AND Machine learning algorithms AND emotion recognition AND valence –arousal space AND EEG AND BCI AND evaluation metrics		

2.4 Selección y reducción del conjunto de documentos.

Con el fin de hacer una selección rigurosa y objetiva de los estudios que se incluirán en la investigación se han incluido los siguientes criterios de inclusión y exclusión.

2.4.1 Criterios de inclusión

IC01. Publish in science, technology, and EEG-BCI related journals and proceedings
IC02. Peer-reviewed research papers with SCOPUS, SCImago, Journal Citation Index JCR.
IC03. Articles proposing emotion recognition models using EEG y BCI.

2.4.2 Criterios de exclusión

EC01 Published in health, psychology, or medical journals and proceedings;
EC02. Literature reviews, chapters in books, theses, technical reports, research; proposals, lectures notes, or handbooks;
EC03. Published in preprint platforms;
EC04. Articles without full text;
EC05. Articles proposing emotion prediction models using EEG;
EC06. Articles published before 2017.

A continuación, se muestra en la tabla 2.4.1 los valores actualizados luego de aplicar los criterios de inclusión y exclusión.

Tabla 2.4.1. Cadenas de búsqueda en Semantic Scholar con criterios.

Database Search Engine	ID	Command search	Search Date	Total
Semantic Scholar	SS01	Feature extraction AND (time domain OR frequency domain OR time-frequency domain) AND emotion recognition AND valence – arousal space	29/06/2022	54
	SS02	Machine learning algorithms AND emotion recognition AND valence –arousal space AND EEG AND BCI.		5
	SS03	Machine learning algorithms AND emotion recognition AND valence –arousal space AND EEG AND BCI AND evaluation metrics		1

En la Imagen 2.4.1 se muestra un diagrama del proceso que se llevó a cabo durante la revisión sistemática de la literatura donde se muestra: la planificación de la revisión, realizar la revisión y reportar la revisión.

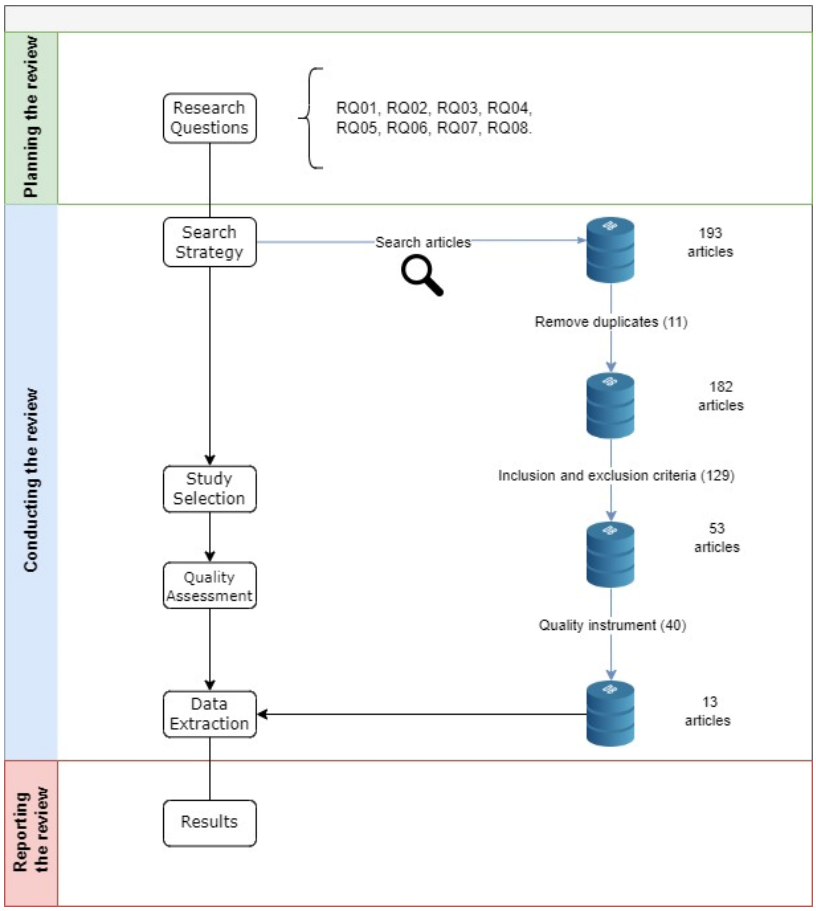


Figura 2.4.1. Proceso de la revisión.

En el diagrama se observa como a partir de la planeación las preguntas de investigación dan el pie para iniciar la revisión, luego en la revisión se extraen los artículos en cada uno de los pasos, por duplicados, criterios de inclusión y exclusión y al final se seleccionaron los que tenían mayor número de referencias.

De los 13 artículos seleccionados se muestra en la tabla 2.4.2 el nombre, las características en el dominio del tiempo, características en el dominio de la frecuencia, características en el dominio del tiempo-frecuencia y los métodos utilizados para la selección de características.

Tabla 2.4.2. Características de los artículos seleccionados.

#	Nombre del Articulo	Características en el dominio del tiempo	Caracteristi_cas en el dominio de la frecuencia	Características en el dominio del tiempo-frecuencia	Selección de características
P01	Inter-Brain EEG Feature Extraction and Analysis for Continuous Implicit Emotion Tagging During Video Watching	i-Amplitude, Power, i-Phase, ISC	Frecuency Bands	N/A	N/A

P02	Electroencephalogram Emotion Recognition Based on Empirical Mode Decomposition and Optimal Feature Selection	Lempel-Ziv Complexity (LZC)	Spectrum Centroid	Empirical Mode Decomposition	Sequential Backward Selection for EEG Feature Selection
P03	Emotion Recognition from EEG Using Rhythm Synchronization Patterns with Joint Time-Frequency-Space Correlation	mean, root mean, root amplitude, absolute mean, skew, kurtosis, variance, index, kurtosys index	Power Spectral Density, Frecuency bands,	discrete wavelet transform,	Principal component analisys
P04	An Efficient Approach to EEG-Based Emotion Recognition using LSTM Network	N/A	band power	N/A	N/A
P05	AI inspired EEG-based spatial feature selection method using multivariate empirical mode decomposition for emotion classification	DNN	DNN	DNN, Complex Continuous Wavelet Transform	Empirical Mode Decomposition, Euclidean distance formula
P06	EEG-based Emotion Recognition with Feature Fusion Networks	Hjorth, Differential Entropy, Sample Entropy	Power Spectral Density	N/A	CNN Feature Fusion Networl Model
P07	Exploring EEG Characteristics to Identify Emotional Reactions under Videogame Scenarios	standard deviation, complexity, mobility, kurtosis, skewness	power spectral density—PDS, and differential entropy—DE from theta, alpha, beta, gamma, and all EEG frequency spectrum	N/A	N/A
P08	EEG-Based Emotion Recognition by Convolutional Neural Network with Multi-Scale Kernels	differential entropy, mean, mean of the first difference, variance, standard deviation	Frecuency bands, power spectrum density and differential entropy	wavelet transform (Mean, standard deviation, variance, and entropy were extracted from the time-frequency domain to	Pearson

				evaluate the performance)	
P09	GTSception: A Deep Learning EEG Emotion Recognition Model Based on Fusion of Global, Time Domain and Frequency Domain Feature Extraction	Temporal Convolution	N/A	N/A	N/A
P10	A Comparative Analysis of Time-frequency Feature Extraction Techniques for Large Scale Electroencephalogram Data	N/A	Frecuency Bands.	discrete wavelet transform, Discrete Wavelet Packet Transform	N/A
P11	The Back-propagation Neural Network Classification of EEG Signal Using Time Frequency Domain Feature Extraction	Peak to peak, Mean square value, variance, Hjort	Fourier transform, maximum power spectral	N/A	Principal component analisys
P12	Emotion recognition using time–frequency ridges of EEG signals based on multivariate synchrosqueezing transform	mean, variance	Frecuency Bands,	N/A	N/A
P13	Exploring the Feature Selection of the EEG Signal Time and Frequency Domain Features for k - NN and Weighted k-NN	Peak to peak, Mean square value, variance, Hjort	Maximum power spectral density, Maximum power spectral frequency, Power sum,	N/A	PCA, Chi square, L1 Regresion, Random Forest, Recursive Feature Elimination
P14	Subject-independent Emotion recognition based on Entropy of EEG Signals	entropy	N/A	wavelet entropy	N/A
P15	Classification of Emotional States Inparkinson's Disease Patients Using Time,Frequency and Time-Frequency Analysis	entropy, energy entropy, Teager energy entropy	Spectral Entropy, spectral energy-entropy, spectral teagen energy entropy,	short time fourier transformed Entropy, stft energy-entropy, stft teager energy entropy	N/A

P16	Electroencephalography-based classification of human emotion: a hybrid strategy in machine learning paradigm	mean, standard deviation, Entropy, Hjort, variance	power, frecuency band	wavelet	N/A

De los 13 artículos seleccionados se muestra en la tabla 2.4.3 el nombre, los algoritmos de aprendizaje de máquina, las métricas de evaluación, numero de emociones clasificadas y valores de rendimiento.

Tabla 2.4.3. Modelos de ML y métricas de evaluación.

#	Algoritmos de aprendizaje de máquina	Métricas de evaluación	Número de emociones clasificadas	Valores de rendimiento (accuracy)
P01	Regresion	accuracy	4	arousal (0.61±0.01) valence (0.70±0.01)
P02	KNN, SVM,	accuracy	4	arousal 84,90% valence 86,46%
P03	ANN	accuracy	4	arousal 67% valence 76,0%
P04	LSTM (Long Short-term Memory)	accuracy	4	arousal 93,13% valence 96,49%
P05	SVM, KNN	accuracy	4	96,3% 83,1%
P06	SVM	accuracy	4	valence 80,52% arousal 75,22%
P07	Bayesian Ridge Regression Model,	accuracy	4	91%
P08	CNN	accuracy	4	arousal 08,27% valence 98,36%

P09	CNN	accuracy	4	aousal 87,6% valence 91,51
P10	SVM, KNN	accuracy	4	valence 80,0% arousal 85,00%
P11	Back-propagation Neural Network	acuuracy	4	63,75%
P12	KNN, SVM, ANN	accuracy	4	arousal 71,55% valence 70,02%
P13	KNN, Weighted KNN	accuracy	4	60,68%
P14	SVM	accuracy	4	valence 70,1% arousal 64,2%
P15	KNN, PNN, SVM	accuracy	6	above 90%
P16	KNN, BPANN, Convined model	accuracy	4	78,33%

En la figura 2.4.2 se muestra la precisión de que obtuvieron los algoritmos evaluados para el reconocimiento de emociones en relación con el número de emociones a reconocer. De acuerdo con la gráfica gran parte de los artículos usaron 4 emociones y obtuvieron un rendimiento entre 0,96 y 0,6. Adicionalmente artículo en el que se utilizó 6 emociones obtuvo 0,9.

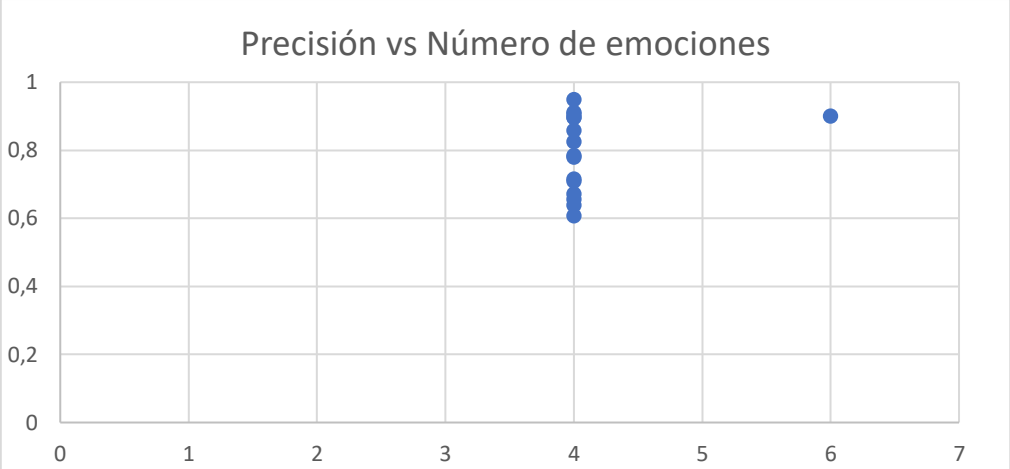


Figura 2.4.2. Precisión vs Número de emociones.

Dentro de los artículos analizados, se encontraron que se utilizaban las siguientes características en el dominio del tiempo, siendo la ordenada el número de coincidencias que una característica fue usada en un artículo (ver figura 2.4.3).

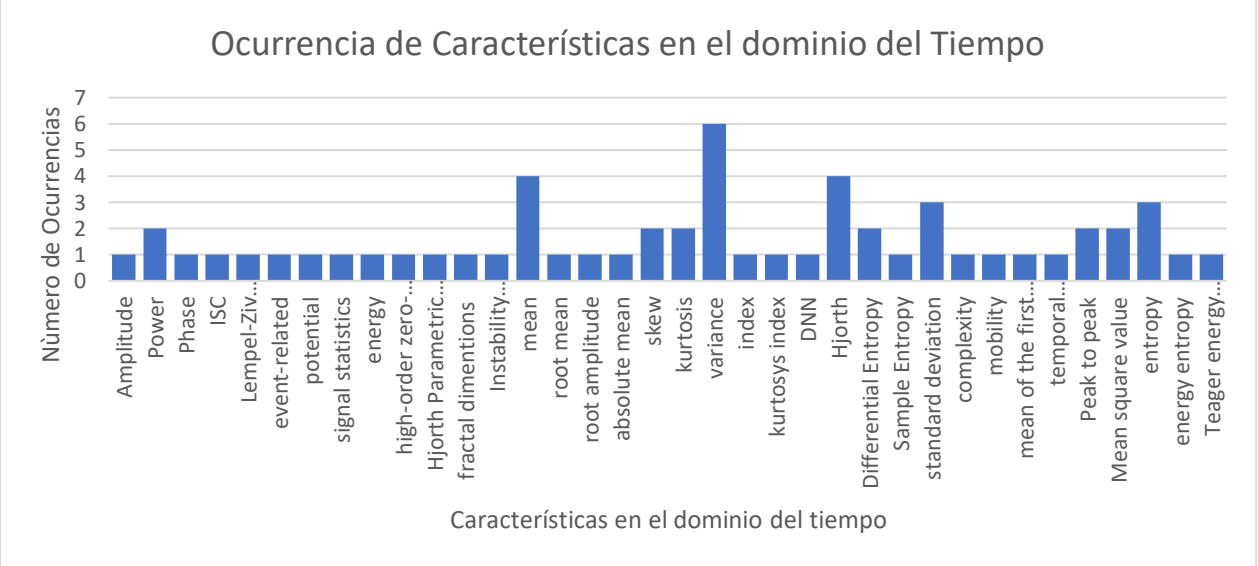


Figura 2.4.3. Ocurrencia de características en el dominio del tiempo.

En el dominio de la frecuencia, las siguientes características (ver figura 2.4.4) tienen las siguientes ocurrencias en los artículos seleccionados.

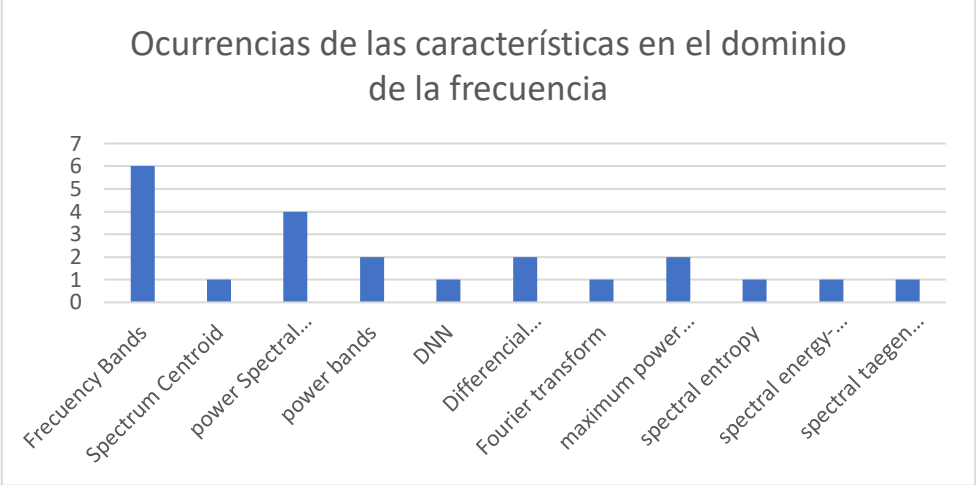


Figura 2.4.4. Características en el dominio de la frecuencia.

En los artículos seleccionados en la revisión sistemática de la literatura, se encontró la siguiente distribución de coincidencias de las características en el dominio de del tiempo-frecuencia (ver figura 2.4.5).

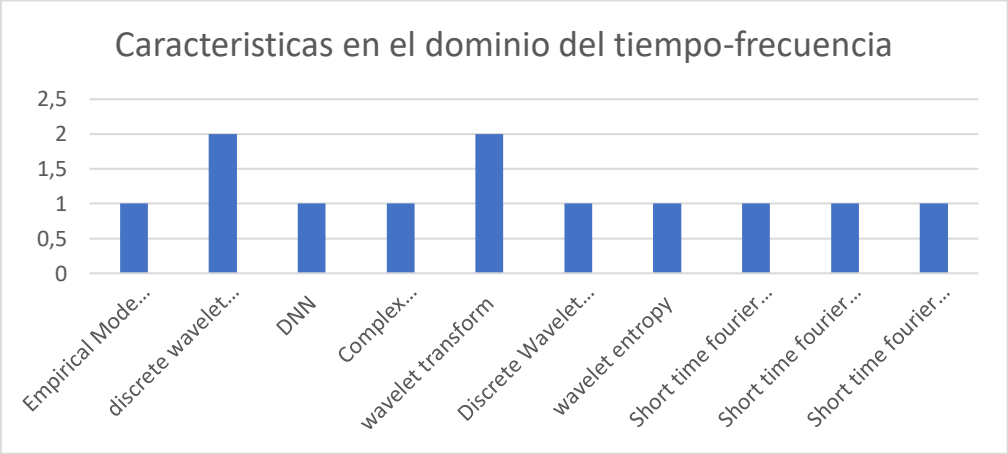


Figura 2.4.5. Características en el dominio del tiempo-frecuencia.

Para la selección de características el algoritmo que destaca es Análisis de Componentes Principales dentro del conjunto de artículos analizados de acuerdo con la revisión sistemática de la literatura como se muestra en la figura 2.4.6.

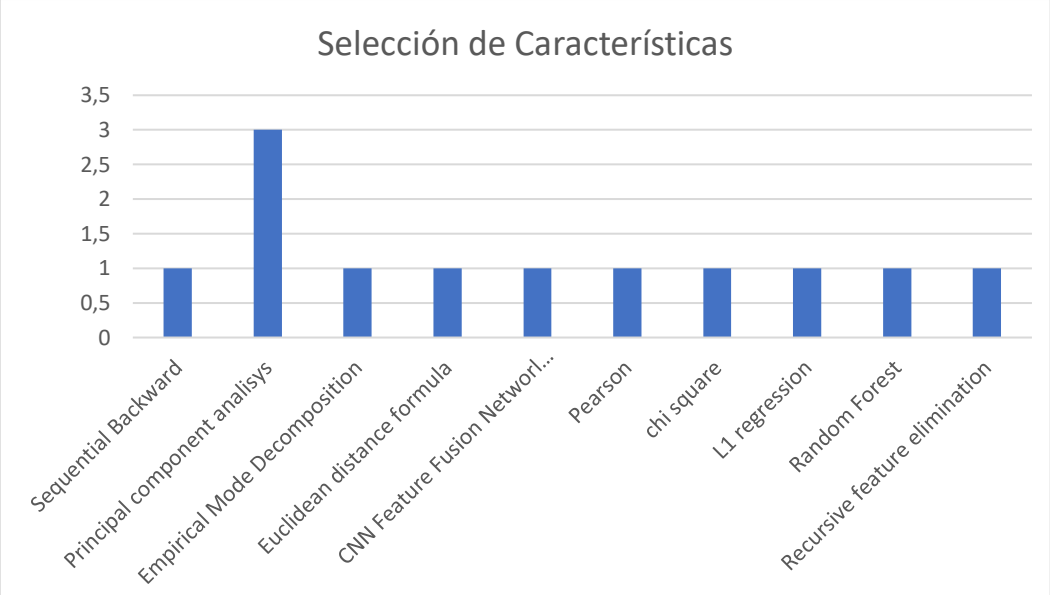


Figura 2.4.6. Selección de características.

Como se observa en la figura 2.4.7 Los modelos de Aprendizaje de Máquina que destacan son KNN, SVN, de entra los modelos utilizados en los artículos.

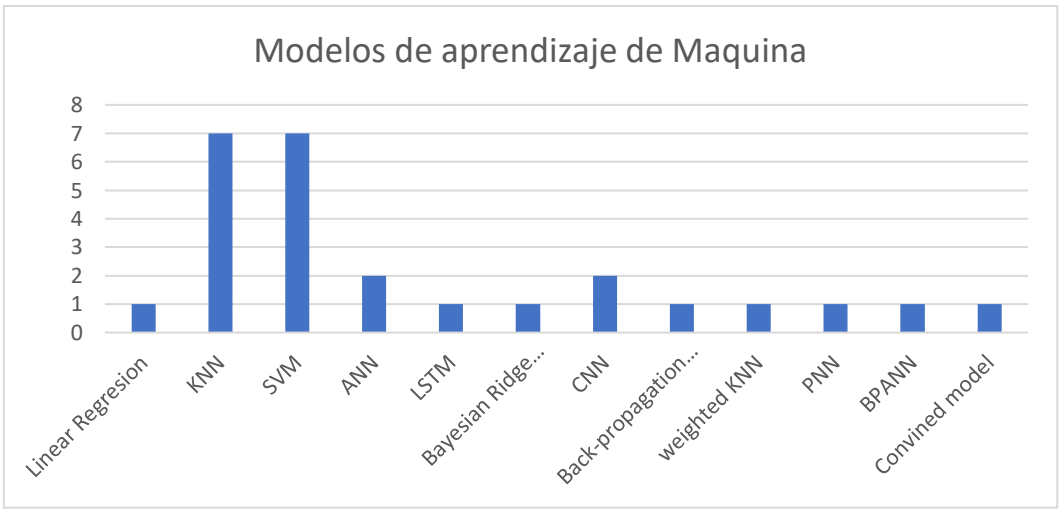


Figura 2.4.7. Modelos de aprendizaje de Máquina.

3. Metodología

En la Figura X, se presentan los componentes que se desarrollaron para obtener el sistema de reconocimiento de emociones usando características extraídas en los dominios del tiempo, frecuencia y tiempo-frecuencia.

Se parte del dataset explicado en el punto anterior, se aplican los algoritmos para extracción de características, se realiza la selección de características relevantes y con estos atributos se alimentan los algoritmos que producen los modelos probados.

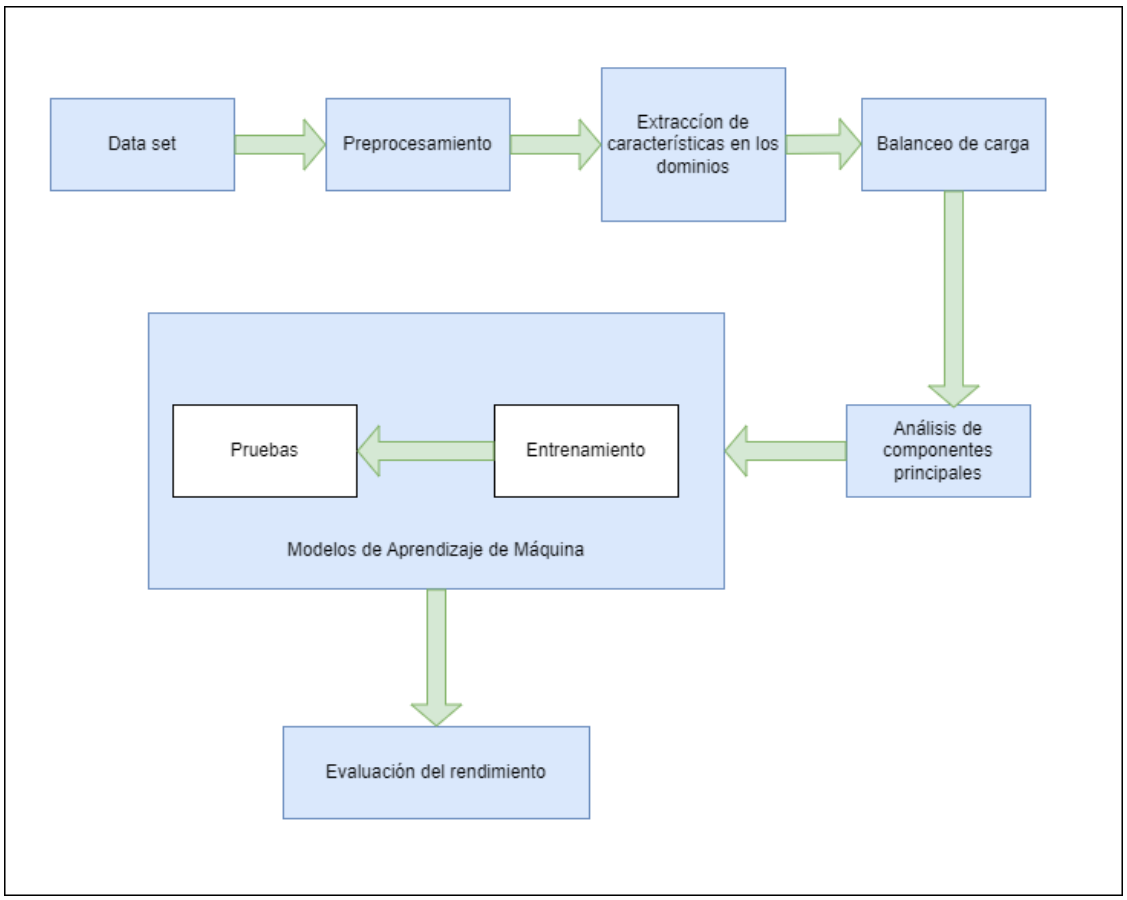


Figura 3.1. Componentes del sistema de reconocimiento de emociones.

3.1 Data set

El conjunto de datos tiene las siguientes columnas.

- Canales del 1-8: que corresponden a los nodos del dispositivo,
- S1, S2, S3: corresponden a la información en X, Y & Z del acelerómetro del dispositivo.
- Hora: La hora en la que se realizó la medición.
- Valence, Arousal: Los valores de valencia y excitación acorde a la escala antes mostrada.

indice	channel1	channel2	channel3	channel4	channel5	channel6	channel7	channel8	s1	s2	s3	time	const	valence	arousal
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.138	0.940	0.006	12:14:00,111	1.620000e+12	5	3
1	-79865.59	-67339.79	-79986.66	-81832.46	-79933.12	-74381.88	-79506.29	-78767.14	0.000	0.000	0.000	12:14:00,111	1.620000e+12	5	3
1	-79865.06	-67337.35	-79986.66	-81833.13	-79930.53	-74382.27	-79500.50	-78767.55	0.000	0.000	0.000	12:14:00,113	1.620000e+12	5	3
1	-79866.34	-67337.04	-79988.77	-81834.31	-79929.74	-74382.96	-79497.38	-78768.38	0.000	0.000	0.000	12:14:00,113	1.620000e+12	5	3
1	-79866.36	-67338.41	-79988.38	-81832.93	-79932.18	-74382.05	-79502.72	-78767.16	-0.102	0.946	0.018	12:14:00,114	1.620000e+12	5	3

Figura 3.1.1. Data set correspondiente al EEG.

El dataset que se utilizará en este trabajo de investigación fue previamente obtenido en el trabajo de Integración curricular “Obtención de un dataset para reconocimiento de emociones usando técnicas de aprendizaje automático.”, en el que se categorizaron los datos utilizando el maniquí de autoevaluación (SAM), en los dominios de valencia y excitación.

La valencia emocional puede ser negativa o positiva, la imagen 3.1.2.1 muestra una persona que está claramente afligida mientras que la 3.1.2.9 muestra una persona claramente exaltada.

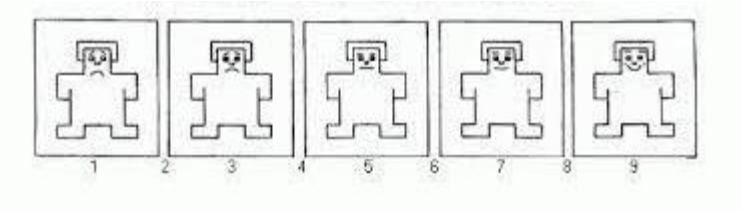


Figura 3.1.2. Escala de valencia.

Se ha separado en dos emociones según el siguiente criterio:

Tabla 3.1.1. Clasificación de la emoción según la valencia.

Valencia (V)	Emoción
V<5	Valencia baja (LV)
5<=V	Valencia alta (HV)

La intensidad de una experiencia se hace comparando la tranquilidad con la excitación, en la imagen 3.1.3.1 se puede observar a una persona que está muy calmada, en su lugar en la imagen 3.1.3.9 se observa una persona que está estallando de excitación.

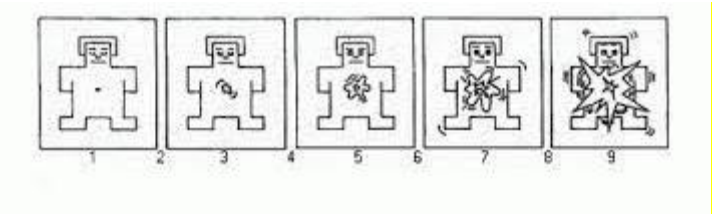


Figura 3.1.3. Escala de excitación.

Se ha separado en dos emociones según el siguiente criterio:

Tabla 3.1.2. Clasificación de la emoción según la excitación.

Excitación (A)	Emoción
A<5	Excitación baja (LA)
5<=A	Excitación alta (HA)

3.2 Preprocesamiento

Se dejaron exclusivamente los 8 canales con las señales y su valor en valencia y excitación para la extracción de características (ver figura 3.2.1).

```
del df['indice']
del df['s1']
del df['s2']
del df['s3']
del df['const']
del df['time']
del df['ev']
del df['ea']
```

```
df.head()
```

	channel1	channel2	channel3	channel4	channel5	channel6	channel7	channel8	valence	arousal
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5	3
1	-79865.59	-67339.79	-79986.66	-81832.46	-79933.12	-74381.88	-79506.29	-78767.14	5	3
2	-79865.06	-67337.35	-79986.66	-81833.13	-79930.53	-74382.27	-79500.50	-78767.55	5	3
3	-79866.34	-67337.04	-79988.77	-81834.31	-79929.74	-74382.96	-79497.38	-78768.38	5	3
4	-79866.36	-67338.41	-79988.38	-81832.93	-79932.18	-74382.05	-79502.72	-78767.16	5	3

Figura 3.2.1. Eliminación de parámetros innecesarios del dataset.

Para realizar la extracción de características es necesario agrupar los datos, ya que en todas las características se debe tener el mismo número de datos, se realizó una agrupación que fue conveniente para todas las características. Con el dominio de la frecuencia se debe tener al menos dos períodos de la frecuencia más baja que se quiere extraer, es este caso la frecuencia más baja se encuentra en la banda de frecuencia Delta, donde el rango inferior es de 0.5 Hz para tener dos ciclos es necesario de 4 segundos de medición. Adicionalmente según la documentación del fabricante el producto tiene una frecuencia de muestreo de 250 Hz. Esto fue creado como una constante (ver figura 3.2.2).

```
eeg_bands = {'Delta': (0.5, 4),
             'Theta': (4, 8),
             'Alpha': (8, 12),
             'Beta': (12, 30),
             'Gamma': (30, 45)}

# Constantes
frecuencia=250
tiempomin=4
intervalo=frecuencia*tiempomin
delta = 0.000000001
```

Figura 3.2.2. Definición de constantes para cálculos.

Para procesar la información de una mejor forma, se dividió el conjunto de datos en intervalos correspondientes al producto de la frecuencia por el tiempo de muestro, tal que se analicen intervalos de cada 3 segundos, tal como muestra la figura 3.2.3.

```
# División de los datos en intervalos antes de procesarse
i=0
vector=list()
while i+1 <= df.shape[0]/intervalo:
  vector.append(pd.DataFrame(df.iloc[i*intervalo:(i+1)*intervalo,:]))
  i=i+1
```

Figura 3.2.3. División de la información.

Se reemplazo los valores nulos con cero (ver figura 3.2.).

```
df.fillna(0,inplace=True)
```

Figura 3.2.4. Reemplazo de valores nulos.

3.3 Extracción de características.

Para la extracción de características se envía los datos agrupados a cada una de las funciones las mismas que devuelven el valor de la característica correspondiente.

3.3.1 Características en el dominio del tiempo.

Se definición una función (time_statistics) para el cálculo de media, varianza y desviación estándar por cada conjunto de datos, por cada uno de los 8 canales, tal como muestra la figura 3.3.1.1.

```
# DOMINIO: TIEMPO
# MÉTODO: MEDIA, VARIANZA, DESVIACION
def time_statistics(row):
  vals = list(row.iloc[:,:].mean())
  vals.extend(list(row.iloc[:,:-2].var(ddof=0)))
  vals.extend(list(row.iloc[:,:-2].std(ddof=0)))
  return vals
```

Figura 3.3.1.1. Función para cálculo de media, varianza y desviación estándar.

Del mismo modo, se definió una función (hjorth_params) para el cálculo de los parámetros de Hjorth para cada uno de los canales, tal como muestra la figura 3.3.1.2.

```
# DOMINIO: TIEMPO
# MÉTODO: PARÁMETROS DE HJORTH
def hjorth_params(row):
  vals = list()
  for i in range(1,9):
    vals.append(eeg.hjorthActivity(row['channel'+str(i)]))
    vals.append(eeg.hjorthComplexity(row['channel'+str(i)]))
    vals.append(eeg.hjorthMobility(row['channel'+str(i)]))
  return vals
```

Figura 3.3.1.2. Función para el cálculo de parámetros de Hjorth.

3.3.2 Características en el dominio de la frecuencia.

En el dominio de la frecuencia, la primera función (frequency_statistics) se definió para el cálculo de los promedios de cada canal, para cada una de las bandas de frecuencia asociadas a un EEG (ver figura 3.3.2.1).

```
# DOMINIO: FRECUENCIA
# MÉTODO: MEDIA DE BANDAS DE FRECUENCIA
def frequency_statistics(row):
  fft_vals = list()
  fft_freqs = list()
  for i in range(1,9):
    fft_vals.append(np.absolute(np.fft.rfft(row['channel'+str(i)])))
    fft_freqs.append(np.fft.rfftfreq(len(row['channel'+str(i)]), 1.0/frecuencia))
  vals = list()
  for i in range(0,8):
    vals_aux=list()
    for band in eeg_bands:
      freq_ix = np.where((fft_freqs[i] >= eeg_bands[band][0]) &
                         (fft_freqs[i] <= eeg_bands[band][1]))[0]
      vals_aux.append(np.mean(np.array(fft_vals[i])[freq_ix]))
    vals.extend(vals_aux)
  return vals
```

Figura 3.3.2.1 Función para el cálculo de media de bandas de frecuencia.

También, se definió la función (differrential_entropy) que permité el cálculo de la entropía diferencial por cada uno de los 8 canales (ver figura 3.3.2.2).

```
# DOMINIO: FRECUENCIA
# MÉTODO: ENTROPÍA DIFERENCIAL
def differential_entropy(row):
  vals = list()
  for i in range(1,9):
    vals.append(sci.stats.differential_entropy(row['channel'+str(i)], method='vasicek'))
  return vals
```

Figura 3.3.2.2. Función para el cálculo de la entropía diferencial.

3.3.3 Características en el dominio del tiempo-frecuencia.

Finalmente, dentro del dominio de la frecuencia, se creó una función (wavelet_analysis) para el cálculo de coeficientes de wavelet, tal como muestra la figura 3.3.3.1.

```
# DOMINIO: TIEMPO-FRECUENCIA
# MÉTODO: MEDIA DE WAVELET
def wavelet_analysis(row):
  vals = list()
  wavelet_function = 'haar'
  for i in range(1,9):
    vals_aux = wlt.dwt(row['channel'+str(i)], wavelet_function)
    vals.append(sum(vals_aux[0])/len(vals_aux[0]))
    vals.append(sum(vals_aux[1])/len(vals_aux[1]))
  return vals
```

Figura 3.3.3.1. Función para el cálculo de coeficientes de Wavelet.

3.3.4 Generación del vector de características.

Aquí se genera el vector de características y además de añade la etiqueta correspondiente para dicho vector, dado que los nuevos datos son generados se calculó del subconjunto de datos para el valor de valencia y excitación se usó el promedio de estos valores para dicho subconjunto. Según la siguiente tabla.

Tabla 3.3.4.1. Clasificación de emoción según los valores de Valencia y Excitación.

Valencia	Excitación	Cuadrante
HV	HA	HVHA
LV	HA	LVHA
HV	LA	HVLA
LV	LA	LVLA

```
vals = list()
for row in vector:
  aux_vals = list()
  aux_vals.append(time_statistics(row))
  aux_vals.append(hjorth_params(row))
  aux_vals.append(frequency_statistics(row))
  aux_vals.append(differential_entropy(row))
  aux_vals.append(wavelet_analysis(row))
  # Definición de la categoría
  if aux_vals[0][8]<=4 and aux_vals[0][9]<=4:
    aux_vals.append([1]) # LVLA
  elif aux_vals[0][8]>4 and aux_vals[0][9]<=4:
    aux_vals.append([2]) # HVLA
  elif aux_vals[0][8]<=4 and aux_vals[0][9]>4:
    aux_vals.append([3]) # LVHA
  elif aux_vals[0][8]>4 and aux_vals[0][9]>4:
    aux_vals.append([4]) # HVHA
  vals.append(aux_vals)

features = list()
for row in vals:
  aux_features = list()
  for feature_vec in row:
    aux_features.extend(feature_vec)
  features.append(aux_features)
```

Figura 3.3.4.1. Generación del vector de características.

Como se observa en la figura anterior (ver figura 3.3.4.1), el vector de características consta de 112 variables provenientes de la extracción de cada una de las características por cada canal.

Luego de la obtención de la data es necesario realizar el escalado de datos (ver figura 3.3.4.2) ya que las características con valores con escala más alta pueden generar que el modelo se centre en dichas variables y descuide a las otras.

```
mm = MinMaxScaler()
#ss = StandardScaler()
X_DATA_FEATURES = mm.fit_transform(X_DATA_FEATURES)
```

Figura 3.3.4.2. Escalado de datos.

3.4 Balanceo de la carga

Se realiza el conteo de los datos en los distintos cuadrantes, como se observa existe un desbalanceo importante de la carga (ver figura 3.4.1).

```
print(auxDF.groupby('cuadrante').size())
```

```
cuadrante
1     564
2    1809
3     488
4    1083
dtype: int64
```

Figura 3.4.1. Conteo de etiquetas antes del balanceo de carga.

Se realizará el balanceo de la carga, con el fin de evitar que el modelo priorice la clase mayoritaria, debido a que su objetivo principal es minimizar el error, ya que, con la diferencia en las clases, el modelo evitará predecir las clases minoritarias y de todas formas obtener puntajes altos en entrenamiento y testeo (ver figura 3.4.2).

```
oversample = ibl.over_sampling.SMOTE()
X, y = oversample.fit_resample(X_DATA_FEATURES, Y_DATA_FEATURES)

X_DATA_FEATURES = pd.DataFrame(X)
Y_DATA_FEATURES = pd.DataFrame(y)
```

Figura 3.4.2. Balanceo de carga de los datos.

Reconteo de clases luego del balanceo de la carga (ver figura 3.4.3).

```
print(Y_DATA_FEATURES.groupby('cuadrante').size())
```

```
cuadrante
1    1809
2    1809
3    1809
4    1809
dtype: int64
```

Figura 3.4.3. Conteo de etiquetas después del balanceo de carga.

3.5 Análisis de componentes principales

Se extrae la matriz de correlación para observar la relación que existe entre las clases extraídas (ver figura 3.5.1).

```
correlation = X_DATA_FEATURES.corr()
corr_matrix = np.array(correlation)
```

```
fig = plt.figure(figsize=(15,15))
corr_matrix_plot = sns.heatmap(corr_matrix, cmap='crest')
```

Figura 3.5.1. Cálculo de la matriz de correlación antes del uso de PCA.

Así, podemos ver su resultado dentro de la figura 3.5.2, que muestra el nivel de relación de cada par de características dentro de los datos.

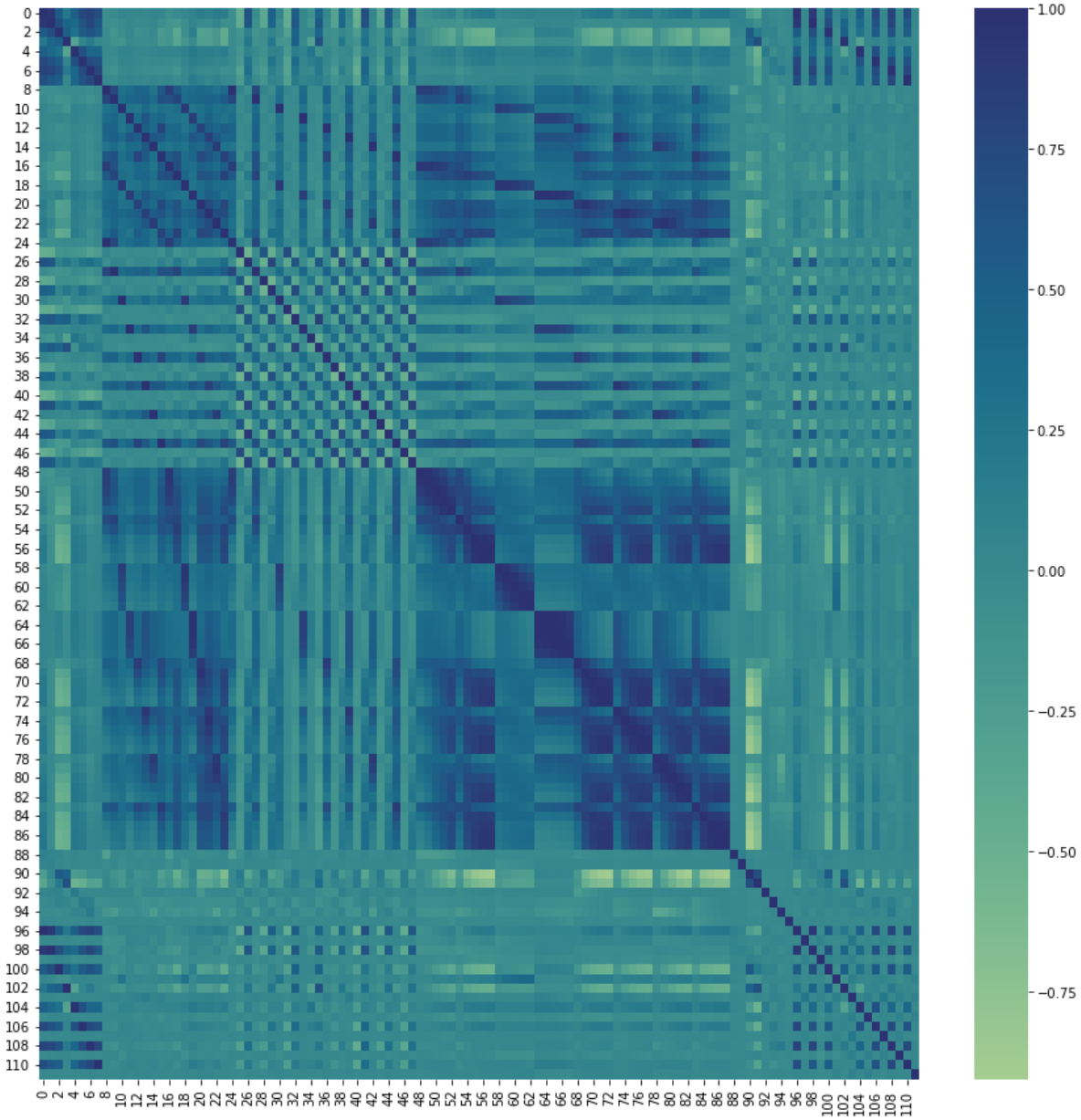


Figura 3.5.2. Visualización de la matriz de correlación antes del uso de PCA.

Luego se realizó la reducción de variables utilizando análisis de componentes principales y se seleccionaron las que tengan más de un 95 por ciento de varianza, algo que se puede observar en la primera línea de la figura 3.5.3.

Finalmente, el data set se redujo a 17 variables.

```
pca = PCA(n_components=0.95)
reduced_X = pca.fit_transform(X_DATA_FEATURES)
X_DATA_FEATURES_PCA = pd.DataFrame(reduced_X)
print(len(X_DATA_FEATURES_PCA.columns))
```

```
17
```

Figura 3.5.3. Reducción de características mediante PCA.

Aquí se realizará el análisis de correlación para validar el resultado después de la reducción de características (ver figura 3.5.4).

```
correlation = X_DATA_FEATURES_PCA.corr()
corr_matrix = np.array(correlation)
fig = plt.figure(figsize=(25,25))
corr_matrix_plot = sns.heatmap(corr_matrix, cmap='crest')
```

Figura 3.5.4. Cálculo de la matriz de correlación después del uso de PCA.

El resultado de la matriz de correlación después del uso de PCA, se puede observar en la siguiente figura (ver figura 3.5.5).

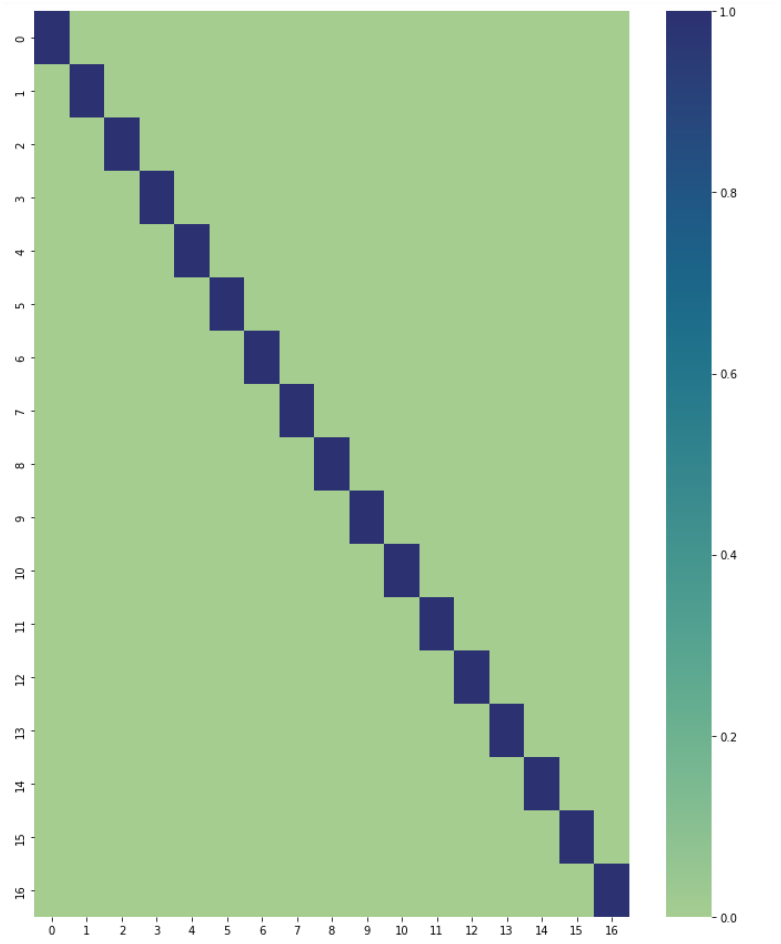


Figura 3.5.5. Visualización de la matriz de correlación.

3.6 Modelos de Aprendizaje de máquina

Para el entrenamiento y testeo se separó los datos de 70% serán destinados para el entrenamiento y 30 % para el testeo. Se entrenaron y testearon los modelos con el conjunto de datos previo al análisis de componentes principales.

3.6.1 Entrenamiento.

Durante el entrenamiento los modelos son alimentados con el conjunto de datos de prueba, tanto los calores de las variables, así como el vector que contiene las clases correspondientes.

K-Nearest Neighbors

Se implemento el modelo de KNN y se lo configuró para catalogar de acuerdo con los 3 vecinos más cercanos (ver figura 3.6.1.1).

```
model_KNN_pca = KNeighborsClassifier(n_neighbors=3)
model_KNN_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA)
```

Figura 3.6.1.1. Creación y entrenamiento del modelo KNN.

Random Forest Classifier.

Se implementó el modelo de RFC que trabaje con una profundidad máxima de 20 ramas (ver figura 3.6.1.2).

```
model_RFC_pca = RandomForestClassifier(max_depth=20, random_state=0)
model_RFC_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA)
```

Figura 3.6.1.2. Creación y entrenamiento del modelo RFC.

Artificial Neural Neteworks

La red neuronal es de tipo “Sequential” se declara en la capa de entrada con 17 neuronas debido a que este es el tamaño del conjunto de datos luego del análisis de componentes principales.

De acuerdo con la arquitectura presentada la red consta de 5 capas ocultas con el siguiente número de neuronas 20, 40, 60, y 8 con la función de activación “relu” finalmente se usa una capa de dimensión 4 ya que es el número de clases y de con la función de activación de tipo “softmax” (ver figura 3.6.1.3).

Como optimizador se utilizó Adam y en la función de pérdida el error cuadrático medio.

Los datos previamente al entrenamiento y testeo del modelo se realizó el método “one hot encoding” que permite separar en variables cada una de las clases. Dando la siguiente codificación. Permitiendo así compatibilizar estos datos con la salida del Softmax.

Tabla 3.6.1.1. Clasificación de emoción según el cuadrante Valencia-Excitación.

Categórica	One Hot Encoding			
Cuadrante	1 (LVLA)	2 (HVLA)	3 (LVHA)	4 (HVHA)
1 (LVLA)	1	0	0	0
2 (HVLA)	0	1	0	0
3 (LVHA)	0	0	1	0

4 (HVHA)	0	0	0	1

```
model_ANN_pca = Sequential()
model_ANN_pca.add(Dense(20,input_dim=17,activation='relu'))
model_ANN_pca.add(Dense(40,activation='relu'))
model_ANN_pca.add(Dense(64,activation='relu'))
model_ANN_pca.add(Dense(32,activation='relu'))
model_ANN_pca.add(Dense(8,activation='relu'))
model_ANN_pca.add(Dense(4,activation='softmax'))

model_ANN_pca.compile(optimizer='adam', loss='mean_squared_error', metrics = ['accuracy'])
```

```
y_train_FEATURES_PCA=pd.get_dummies(y_train_FEATURES_PCA['cuadrante'])
y_test_FEATURES_PCA=pd.get_dummies(y_test_FEATURES_PCA['cuadrante'])
```

Figura 3.6.1.3. Creación y entrenamiento del modelo ANN.

Dentro de la implementación específica de Python, el entrenamiento del modelo entrega valores conforme se están ejecutando de acuerdo a alguna métrica que quiera ser analizada, en este caso, el valor de accuracy, como se observó en la imagen anterior.

```
datos_modelo_pca=model_ANN_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA, validation_data=(x_test_FEATURES_PCA, y_test_FEATU
0.7854
Epoch 295/300
127/127 [==============================] - 0s 2ms/step - loss: 0.0534 - accuracy: 0.8571 - val_loss: 0.0784 - val_accuracy:
0.7877
Epoch 296/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0537 - accuracy: 0.8541 - val_loss: 0.0804 - val_accuracy:
0.7771
Epoch 297/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0546 - accuracy: 0.8531 - val_loss: 0.0835 - val_accuracy:
0.7738
Epoch 298/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0551 - accuracy: 0.8511 - val_loss: 0.0843 - val_accuracy:
0.7632
Epoch 299/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0547 - accuracy: 0.8521 - val_loss: 0.0758 - val_accuracy:
0.8033
Epoch 300/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0528 - accuracy: 0.8578 - val_loss: 0.0790 - val_accuracy:
0.7913
```

Figura 3.6.1.4. Resultado del entrenamiento del modelo ANN.

3.6.2 Testeo.

Durante la etapa de testeo se realizó las predicciones del conjunto de dato de pruebas y se almacenan en una variable para luego realizar la comparación con los valores originales.

K-Nearest Neighbors

En la figura 3.6.2.1 se muestra el uso del modelo KNN para predecir un conjunto de datos que permitirá posteriormente analizar su modelo.

```
pred_KNN_pca = model_KNN_pca.predict(x_test_FEATURES_PCA)
```

Figura 3.6.2.1. Prueba del modelo KNN.

Random Forest Classifier

En la figura 3.6.2.2 se muestra el uso del modelo RFC para predecir un conjunto de datos que permitirá posteriormente analizar su modelo.

```
pred_RFC_pca = model_RFC_pca.predict(x_test_FEATURES_PCA)
```

Figura 3.6.2.2. Prueba del modelo RFC.

Artificial Neural Neteworks

En la figura 3.6.2.3 se muestra el uso del modelo ANN para predecir un conjunto de datos que permitirá posteriormente analizar su modelo.

```
pred_ANN_pca = model_ANN_pca.predict(x_test_FEATURES_PCA)
```

Figura 3.6.2.3. Prueba del modelo ANN.

3.7 Evaluación del rendimiento.

Para la evaluación del rendimiento se usará el Accurracy, Precision y el F1 - Score.

K-Nearest Neighbors

Analizando la matriz de confusión se puede observar que las clase que más clasificaciones erróneas tiene es la HVLA(2) lo mismo que se ve reflejado en los valores de accurracy y f1-score (ver figura 3.7.1).

```
print(confusion_matrix(y_test_FEATURES_PCA, pred_KNN_pca))
print(classification_report(y_test_FEATURES_PCA, pred_KNN_pca))
```

```
[[486  26  11  15]
 [ 71 279  82  95]
 [  6  12 542   4]
 [ 25  60  12 445]]
              precision    recall  f1-score   support

           1       0.83      0.90      0.86       538
           2       0.74      0.53      0.62       527
           3       0.84      0.96      0.90       564
           4       0.80      0.82      0.81       542

    accuracy                           0.81      2171
   macro avg       0.80      0.80      0.80      2171
weighted avg       0.80      0.81      0.80      2171
```

Figura 3.7.1. Medidas de rendimiento del algoritmo KNN.

Random Forest Classifier

En este algoritmo se observa también que la clase HVLA(2), este modelo tiene un resultado similar a KNN (ver figura 3.7.2).

```
[[488  36   6   8]
 [ 50 328  68  81]
 [  3  36 513  12]
 [ 17  66   9 450]]
              precision    recall  f1-score   support

           1       0.87      0.91      0.89       538
           2       0.70      0.62      0.66       527
           3       0.86      0.91      0.88       564
           4       0.82      0.83      0.82       542

    accuracy                           0.82      2171
   macro avg       0.81      0.82      0.81      2171
weighted avg       0.82      0.82      0.82      2171
```

Figura 3.7.2. Medidas de rendimiento del algoritmo RFC.

Artificial Neural Networks

Comparación de la pérdida y el accuracy durante el entrenamiento, para validar, debido a que los valores no defieren en gran medida no se considera que existe overfitting en el modelo (ver figura 3.7.3).

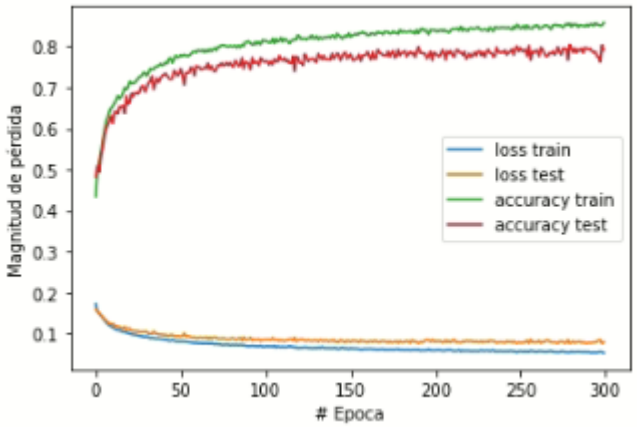


Figura 3.7.3. Análisis de pérdida y acurracy durante el entrenamiento.

Los resultados se mantienen similares para que, en los modelos anteriores, con puntajes bajos para la clase HVLA(2) (ver figura 3.7.4).

```
print(accuracy_score(y_test_FEATURES_PCA['cuadrante'], pred_ANN_pca1['cuadrante']))
print(classification_report(y_test_FEATURES_PCA['cuadrante'], pred_ANN_pca1['cuadrante']))
```

```
0.7913403961308153
              precision    recall  f1-score   support

           1       0.82      0.83      0.83       538
           2       0.72      0.50      0.59       527
           3       0.83      0.93      0.88       564
           4       0.76      0.89      0.82       542

    accuracy                           0.79      2171
   macro avg       0.79      0.79      0.78      2171
weighted avg       0.79      0.79      0.78      2171
```

Figura 3.7.4. Medidas de rendimiento del algoritmo ANN.

4. RESULTADOS

Los datos obtenidos se muestran consistentes en cuanto a valores de precisión por emoción entre un modelo y otro a demás los valores no difieren entre los modelos a los que se les ingreso los datos que fueron reducidos usando análisis de componentes principales.

Tabla 4.1. Comparación de rendimiento de los modelos con y sin uso del PCA.

Datos	Modelo	Precision				F1 Score				Accuracy
		HVLA	LVLA	LVHA	HVHA	HVLA	LVLA	LVHA	HVHA	
Datos luego de Análisis de componentes principales	KNN	0.83	0.74	0.84	0.8	0.86	0.62	0.90	0.81	0.81
	Random Forest Classifier	0.87	0.70	0.86	0.82	0.89	0.66	0.88	0.82	0.82
	ANN	0.82	0.72	0.83	0.76	0.83	0.59	0.88	0.82	0.79
Datos sin análisis de componentes principales	KNN	0.79	0.74	0.83	0.79	0.85	0.62	0.87	0.79	0.79
	Random Forest Classifier	0.93	0.78	0.90	0.86	0.93	0.75	0.93	0.87	0.87
	ANN	0.85	0.80	0.79	0.76	0.86	0.67	0.86	0.79	0.80

Para KNN se tiene que los mejores resultados se obtuvieron con los datos que de salida del Análisis de componentes principales, en Accuracy se tiene 0.81.

Igualmente, en la precisión por emoción se observan variaciones mínimas. Y en F1-Score los valores para el data set de PCA es superior.

En el caso del Random Forest Classifier los resultados superaron ampliamente con el uso de la data set que proviene directamente de las características extraídas de las Señales EEG, con un valor de Accuracy igual a 0.87 en el más alto y 0.82 en el otro caso.

En cuanto a el modelo de ANN se obtuvieron resultados que difieren muy poco, en ambos casos los resultados se encuentran cerca de 0.80 para Accuracy, aunque en el resultado de F1-Score para la emoción LVLA es donde más difieren.

Los algoritmos se desempeñan de forma similar haciendo uso del Análisis de componentes principales, en comparación a cuando se usa el vector de características de las señales. Lo que demuestra que la reducción de variables de 112 a 17 fue bastante acertada, debido a que se representó la información del conjunto de datos en ese número de reducido de variables.

El algoritmo con mejor desempeño fue el Random Forest Classifier que tuvo la puntuación más alta en las distintas métricas con los dos conjuntos de datos.

Los algoritmos KNN Y ANN tienen una puntuación bastante estable, por lo que se resalta los tiempos cortos de entrenamiento del KNN.

Ya que las características se calcularon de subconjuntos compuestos de 1000 filas del conjunto de datos inicial, por lo que teniendo en cuenta que el dispositivo toma 250 muestras por segundo en 8 canales. Se redujo a un vector de características cada 4 segundos de 112 variables las mismas que luego con el análisis de componentes principales se reducen a 17, se resalta el rendimiento de los algoritmos con el conjunto de dato mucho más reducido que el inicial.

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- La extracción de características en el dominio del tiempo, frecuencia y tiempo-frecuencia, contribuyo a la extracción de información relevante en los distintos dominios para permitir el reconociendo de emociones en el cuadrante valencia-excitación.
- La extracción de características en los dominios del tiempo, frecuencia y tiempo-frecuencia permitió la reducción del conjunto de datos pasando de 250 muestras de 8 canales por segundo a un vector de características de tamaño 112 cada 4 segundos el mismo que se redujo a 17 luego del análisis de componentes principales.
- Aunque se redujo el conjunto de datos debido a la extracción de características en el dominio del tiempo, frecuencia y tiempo-frecuencia, el nuevo conjunto de datos aún mantiene información relevante que permitió el reconocimiento de emociones por parte de los algoritmos de aprendizaje de máquina.
- El balanceo de la carga permitió que las predicciones de los algoritmos no sean sesgadas hacia las clases mayoritarias.

- En los modelos ANN y KNN el uso de análisis de componentes principales permitió reducir la cantidad de variables del vector de entrada de 112 a 17 y se mantuvieron los resultados de reconocimiento de emociones.
- En el modelo Random Forest Clasifier el uso de análisis de componentes principales permitió reducir la cantidad de variables del vector de entrada de 112 a 17 y aunque el valore de las métricas de evaluación disminuyeron siguen siendo superiores al resto de modelos en cuanto al reconocimiento de emociones.
- La segmentación del conjunto de datos en entrenamiento y prueba permitió entrenar los modelos y luego obtener su precisión accuracy y F1-score, permitiendo evaluar el rendimiento de los algoritmos.
- La extracción de características a partir de EEG permitió reducir el conjunto de datos, y el número de reconocimientos que un algoritmo debe realizar en intervalos de tiempo, esto con respecto a una posible implementación en tiempo real.
- Los resultados obtenidos para los modelos implementados mostraron resultados positivos en relación con trabajos semejantes, donde los valores de exactitud mostrados por los modelos oscilaban entre el 70% y poco más del 85%, en comparación con valores entre 79% y 87% obtenidos en el presente trabajo.

5.2 Recomendaciones

- El uso de un conjunto de datos más extenso y diverso en cuanto al número de personas del que provienen las mediciones es preferible, debido a que con la reducción en el conjunto de datos se puede generalizar más la información, lo que permitirá mejorar el reconocimiento de los algoritmos.
- Dado que los resultados, dependen de la calidad del conjunto de datos iniciales, se recomienda enfatizar en la calidad y volumen de datos recolectados, así como en el etiquetado de las clases.
- Debido a que se está tratando una función o una señal cuyo origen es dependiente del tiempo, es recomendable que los datos mantengan el orden temporal, pues una posible mezcla o toma aleatoria de las muestras en realidad no representa una señal original y fiel a la realidad.
- Se recomienda no balancear la carga antes de la extracción del vector de características, debido a que los datos generados no mantendrán correlación unos con otros como si sucede en un EEG con las mediciones a lo largo del tiempo. Aunque el balanceo de la carga sobre las señales EEG no genere inconvenientes con la extracción de características en función del tiempo, si los tendrá en la función de la frecuencia y tiempo-frecuencia.

6. BIBLIOGRAFÍA

[1] Puthankattil, Subha Dharmapalan & Joseph, Paul & Acharya, U Rajendra & Lim, Choo. "*EEG signal analysis: a survey. Journal of medical systems".* 34. 195-212. 10.1007/s10916-008-9231-z, 2010.

[2] Saeid Sanei and J.A. Chambers. "*EEG SIGNAL PROCESSING*." Centre of Digital Signal Processing Cardiff University, UK .

[3] Mohammad-Mahdi Moazzami, "*EEG Signal Processing in Brain-Computer interface"*. A THESIS Submitted to Michigan State University in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE Computer Science. 2012.

[4] Silvia Chiappa, "*Analysis and Classification of EEG Signals using Probabilistic Models for Brain Computer Interfaces"*, 2006.

[5] Hewlett Packard. "*The Fundamentals of Signal Analysis"*, 23 de febrero de 2023, [https://www.hpmemoryproject.org/an/pdf/an_243.pdf].

[6] Robert G. Bartle, Donald R. Sherbert, "*Introduction to real analysis",* fourth Edition. University of Illinois, Urbana-Champaign, 2011.

[7] Andrew CA (2016) Signals in the time-domain, 23 de febrero de 2023 [https://mast.queensu.ca/~andrew/teaching/pdf/334-notes-volume2.pdf].

[8] «Diccionario de la lengua española» "*frecuencia* | *Diccionario de la lengua española*". 23 de febrero de 2023, Edición del Tricentenario. [https://dle.rae.es/frecuencia].

[9] Getachew Admassie Ambaye *"Time and Frequency Domain Analysis of Signals: A Review."* International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181http://www.ijert.org IJERTV9IS120127 (This work is licensed under a Creative Commons Attribution 4.0 International License.) Published by : www.ijert.org Vol. 9 Issue 12, December-2020.

[10] "*CHAPTER 9 Frequency-Domain Signal Analysis"*, 23 de febrero de 2023 [http://dsp-book.narod.ru/SATFSS3.pdf].

[11] "*Analysis and Design of Feedback Control Systems",* MASSACHUSETTS INSTITUTE OF TECHNOLOGY DEPARTMENT OF MECHANICAL ENGINEERING: 23 de febrero de 2023 [https://web.mit.edu/2.14/www/Handouts/FreqDom.pdf].

[12] Vinay Yuvashankar, *"APPLICATIONS IN TIME-FREQUENCY DOMAIN ANALYSIS*" McMaster University Hamilton, Ontario, Canada, 2017.

[13] Praveen Kumar, *Wavelet Analysis and Its Applications*. 23 de febrero de 2023 [https://efi.eng.uci.edu/papers/efg_149.pdf].

[14] JIMMY CORTÉ, HUGO CANO, JOSÉ CHAVES. *"Implementation of the wavelet HAAR to reconstruct the function f (t) = t On the range [-3.3] different degrees of resolution*." Scientia et Technica Año XIV, No 38, Junio 2008. Universidad Tecnológica de Pereira ISSN 0122-1701, 2008.

[15] Joseph C. Watkins. "*An Introduction to the Science of Statistics: From Theory to Implementation"* 23 de febrero de 2023 [https://www.math.arizona.edu/~jwatkins/statbook.pdf]

[16] Roxy Peck, Chris Olsen, Jay Devore, "*Introduction to Statistics and Data Analysis"* Thomson Higher Education 10 Davis Drive Belmont, CA 94002-3098 USA.

[17] Seung-Hyeon Oh, Yu-Ri Lee and Hyoung-Nam Kim. "*A Novel EEG Feature Extraction Method using Hjorth Parameter*".

[18] Luis E. Morillo, *ANÁLISIS VISUAL DEL ELECTROENCEFALOGRAMA*. Guía neurológica 7. 23 de febrero de 2023 [https://www.acnweb.org/guia/g7cap17.pdf]

[19] ZILAN ISIK, LOU GRIPENTOFT, "*EEG Signal Analysis in the Frequency Domain An Examination of Abnormalities During the Gait Cycle",* 23 de febrero de 2023 [Microsoft Word - 16 EEG signal analysis in frequency domain Ver 8 (diva-portal.org)], 2022.

[20] Antonio Di Crescenzo, Luca Paolillo, Abstract Alfonso Suárez-Llorens, *Stochastic comparisons, differential entropy and varentropy for distributions induced by probability density functions*∗ arXiv:2103.11038v1, 2021.

[21] Robert M. Gray, "*Entropy and Information Theory First Edition"*, Corrected Information Systems Laboratory Electrical Engineering Department Stanford University Springer-Verlag New York, 2006.

[22] R.J.E. Merry, "*Wavelet Theory and Applications A literature study"* Eindhoven University of Technology Department of Mechanical Engineering Control Systems Technology Group, 2006.

[23] Shai Shalev-Shwartz and Shai Ben-David, *"Understanding Machine Learning: From Theory to Algorithms*", 2014.

[24] Giuseppe Bonaccorso, *"Machine Learning Algorithms",* First published: July 2017 Production reference: 1200717 Published by Packt Publishing Ltd, 2017.

[25] Neural networks and deep learning, *"Neural networks and deep learning"*, 23 de febrero de 2023 [http://neuralnetworksanddeeplearning.com/].

[26] Hossin, Mohammad & M.N, Sulaiman. "*A Review on Evaluation Metrics for Data Classification Evaluations"*. International Journal of Data Mining & Knowledge Management Process. 5. 01-11. 10.5121/ijdkp.2015.5201, 2015

[27] Željko Đ. Vujović, "*Classification Model Evaluation Metrics"* (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 6, 2021

7. ANEXOS

INSTALACIÓN E INCLUSIÓN DE LIBRERIAS

Inclusión de librerías de manipulación de datos

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Inclusión de librerías para el procesamiento de datos

In [2]:

```
import eeglib.eeg as eeg
import scipy as sci
import pywt as wlt
import seaborn as sns
```

Inclusión de librerias de tratamiento de señales

In [3]:

```
import imblearn as ibl
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

Inclusión de librerías para procesamiento en modelos de ML

In [4]:

```
import tensorflow as tf
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

DEFINICIÓN DE FUNCIONES PARA LA EXTRACCIÓN DE CARACTERÍSTICAS

Funciones para la extracción de características en el dominio del tiempo

In [5]:

```
# DOMINIO: TIEMPO
# MÉTODO: MEDIA, VARIANZA, DESVIACION
def time_statistics(row):
  vals = list(row.iloc[:,:].mean())
  vals.extend(list(row.iloc[:,:-2].var(ddof=0)))
  vals.extend(list(row.iloc[:,:-2].std(ddof=0)))
  return vals
```

In [6]:

```
# DOMINIO: TIEMPO
# MÉTODO: PARÁMETROS DE HJORTH
def hjorth_params(row):
  vals = list()
  for i in range(1,9):
    vals.append(eeg.hjorthActivity(row['channel'+str(i)]))
    vals.append(eeg.hjorthComplexity(row['channel'+str(i)]))
    vals.append(eeg.hjorthMobility(row['channel'+str(i)]))
  return vals
```

Funciones para la extracción de características en el dominio de la frecuencia

In [7]:

```
# DOMINIO: FRECUENCIA
# MÉTODO: MEDIA DE BANDAS DE FRECUENCIA
def frequency_statistics(row):
  fft_vals = list()
  fft_freqs = list()
  for i in range(1,9):
    fft_vals.append(np.absolute(np.fft.rfft(row['channel'+str(i)])))
    fft_freqs.append(np.fft.rfftfreq(len(row['channel'+str(i)]), 1.0/frecuencia))
  vals = list()
  for i in range(0,8):
    vals_aux=list()
    for band in eeg_bands:
      freq_ix = np.where((fft_freqs[i] >= eeg_bands[band][0]) &
                         (fft_freqs[i] <= eeg_bands[band][1]))[0]
      vals_aux.append(np.mean(np.array(fft_vals[i])[freq_ix]))
    vals.extend(vals_aux)
  return vals
```

In [8]:

```
# DOMINIO: FRECUENCIA
# MÉTODO: ENTROPÍA DIFERENCIAL
def differential_entropy(row):
  vals = list()
  for i in range(1,9):
    vals.append(sci.stats.differential_entropy(row['channel'+str(i)], method='vasicek'))
  return vals
```

Funciones para la extracción de características en el dominio de tiempo-frecuencia

In [9]:

```
# DOMINIO: TIEMPO-FRECUENCIA
# MÉTODO: MEDIA DE WAVELET
def wavelet_analysis(row):
  vals = list()
  wavelet_function = 'haar'
  for i in range(1,9):
    vals_aux = wlt.dwt(row['channel'+str(i)], wavelet_function)
    vals.append(sum(vals_aux[0])/len(vals_aux[0]))
    vals.append(sum(vals_aux[1])/len(vals_aux[1]))
  return vals
```

DEFINICIÓN DE CONSTANTES

In [10]:

```
eeg_bands = {'Delta': (0.5, 4),
             'Theta': (4, 8),
             'Alpha': (8, 12),
             'Beta': (12, 30),
             'Gamma': (30, 45)}

# Constantes
frecuencia=250
tiempomin=4
intervalo=frecuencia*tiempomin
delta = 0.000000001

# Vector con los índices para el vector de características
indices = ['time_mean_channel1', 'time_mean_channel2', 'time_mean_channel3', 'time_mean_c
           'time_variance_channel1', 'time_variance_channel2', 'time_variance_channel3',
           'time_stddev_channel1', 'time_stddev_channel2', 'time_stddev_channel3', 'time_
           'hjorth_activity_channel1', 'hjorth_complexity_channel1', 'hjorth_mobility_cha
           'hjorth_activity_channel2', 'hjorth_complexity_channel2', 'hjorth_mobility_cha
           'hjorth_activity_channel3', 'hjorth_complexity_channel3', 'hjorth_mobility_cha
           'hjorth_activity_channel4', 'hjorth_complexity_channel4', 'hjorth_mobility_cha
           'hjorth_activity_channel5', 'hjorth_complexity_channel5', 'hjorth_mobility_cha
           'hjorth_activity_channel6', 'hjorth_complexity_channel6', 'hjorth_mobility_cha
           'hjorth_activity_channel7', 'hjorth_complexity_channel7', 'hjorth_mobility_cha
           'hjorth_activity_channel8', 'hjorth_complexity_channel8', 'hjorth_mobility_cha
           'delta_band_channel1', 'theta_band_channel1', 'alpha_band_channel1', 'betha_ba
           'delta_band_channel2', 'theta_band_channel2', 'alpha_band_channel2', 'betha_ba
           'delta_band_channel3', 'theta_band_channel3', 'alpha_band_channel3', 'betha_ba
           'delta_band_channel4', 'theta_band_channel4', 'alpha_band_channel4', 'betha_ba
           'delta_band_channel5', 'theta_band_channel5', 'alpha_band_channel5', 'betha_ba
           'delta_band_channel6', 'theta_band_channel6', 'alpha_band_channel6', 'betha_ba
           'delta_band_channel7', 'theta_band_channel7', 'alpha_band_channel7', 'betha_ba
           'delta_band_channel8', 'theta_band_channel8', 'alpha_band_channel8', 'betha_ba
           'differential_entropy_channel1', 'differential_entropy_channel2', 'differentia
           'approx_coefficient_channel1', 'detail_coefficient_channel1',
           'approx_coefficient_channel2', 'detail_coefficient_channel2',
           'approx_coefficient_channel3', 'detail_coefficient_channel3',
           'approx_coefficient_channel4', 'detail_coefficient_channel4',
           'approx_coefficient_channel5', 'detail_coefficient_channel5',
           'approx_coefficient_channel6', 'detail_coefficient_channel6',
           'approx_coefficient_channel7', 'detail_coefficient_channel7',
           'approx_coefficient_channel8', 'detail_coefficient_channel8',
           'cuadrante']
```

IMPORTACIÓN LOS DATOS

Lectura del archivo desde Google Drive

In [11]:

```
df = pd.read_csv('Dataset_Completo.csv')
```

In [12]:

```
df.head()
```

Out[12]:

	indice	channel1	channel2	channel3	channel4	channel5	channel6	channel7	channel
0	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0
1	1	-79865.59	-67339.79	-79986.66	-81832.46	-79933.12	-74381.88	-79506.29	-78767.1
2	1	-79865.06	-67337.35	-79986.66	-81833.13	-79930.53	-74382.27	-79500.50	-78767.5
3	1	-79866.34	-67337.04	-79988.77	-81834.31	-79929.74	-74382.96	-79497.38	-78768.3
4	1	-79866.36	-67338.41	-79988.38	-81832.93	-79932.18	-74382.05	-79502.72	-78767.1

Exclusión de columnas

In [13]:

```
del df['indice']
del df['s1']
del df['s2']
del df['s3']
del df['const']
del df['time']
del df['ev']
del df['ea']
```

In [14]:

```
df.head()
```

Out[14]:

	channel1	channel2	channel3	channel4	channel5	channel6	channel7	channel8	valer
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
1	-79865.59	-67339.79	-79986.66	-81832.46	-79933.12	-74381.88	-79506.29	-78767.14	
2	-79865.06	-67337.35	-79986.66	-81833.13	-79930.53	-74382.27	-79500.50	-78767.55	
3	-79866.34	-67337.04	-79988.77	-81834.31	-79929.74	-74382.96	-79497.38	-78768.38	
4	-79866.36	-67338.41	-79988.38	-81832.93	-79932.18	-74382.05	-79502.72	-78767.16	

Reemplazar los valore NaN

In [15]:

```
df.fillna(0,inplace=True)
```

In [16]:

```
# División de los datos en intervalos antes de procesarse
i=0
vector=list()
while i+1 <= df.shape[0]/intervalo:
  vector.append(pd.DataFrame(df.iloc[i*intervalo:(i+1)*intervalo,:]))
  i=i+1
```

GENERACIÓN DEL VECTOR DE CARACTERÍSTICAS

In [17]:

```
vals = list()
for row in vector:
  aux_vals = list()
  aux_vals.append(time_statistics(row))
  aux_vals.append(hjorth_params(row))
  aux_vals.append(frequency_statistics(row))
  aux_vals.append(differential_entropy(row))
  aux_vals.append(wavelet_analysis(row))
  # Definición de la categoría
  if aux_vals[0][8]<=4 and aux_vals[0][9]<=4:
    aux_vals.append([1]) # LVLA
  elif aux_vals[0][8]>4 and aux_vals[0][9]<=4:
    aux_vals.append([2]) # HVLA
  elif aux_vals[0][8]<=4 and aux_vals[0][9]>4:
    aux_vals.append([3]) # LVHA
  elif aux_vals[0][8]>4 and aux_vals[0][9]>4:
    aux_vals.append([4]) # HVHA
  vals.append(aux_vals)

features = list()
for row in vals:
  aux_features = list()
  for feature_vec in row:
    aux_features.extend(feature_vec)
  features.append(aux_features)
```

```
E:\Anaconda\lib\site-packages\scipy\stats\_entropy.py:288: RuntimeWarning:
divide by zero encountered in log
  logs = np.log(n/(2*m) * differences)
E:\Anaconda\lib\site-packages\eeglib\features.py:289: RuntimeWarning: inva
lid value encountered in double_scalars
  return np.sqrt(np.var(np.gradient(data)) / np.var(data))
```

In [18]:

```
auxDF = pd.DataFrame(features, columns=indices)
del auxDF['time_mean_arousal']
del auxDF['time_mean_valence']
print(auxDF.groupby('cuadrante').size())
```

```
cuadrante
1     564
2    1809
3     488
4    1083
dtype: int64
```

In [19]:

```
Y_DATA_FEATURES = auxDF['cuadrante']
del auxDF['cuadrante']
```

In [20]:

```
X_DATA_FEATURES = auxDF
```

Eliminación de nulos y valores no numéricos

In [21]:

```
X_DATA_FEATURES.fillna(X_DATA_FEATURES.mean(), inplace=True)
X_DATA_FEATURES.replace([float("-inf"), float("inf")], [-1e100, 1e100], inplace=True)
np.all(np.isinf(X_DATA_FEATURES))
```

Out[21]:

```
False
```

Escalado de datos.

In [22]:

```
mm = MinMaxScaler()
#ss = StandardScaler()
X_DATA_FEATURES = mm.fit_transform(X_DATA_FEATURES)
```

Balanceo de la carga.

In [23]:

```
oversample = ibl.over_sampling.SMOTE()
X, y = oversample.fit_resample(X_DATA_FEATURES, Y_DATA_FEATURES)

X_DATA_FEATURES = pd.DataFrame(X)
Y_DATA_FEATURES = pd.DataFrame(y)
```

In [24]:

```
print(Y_DATA_FEATURES.groupby('cuadrante').size())
```

```
cuadrante
1    1809
2    1809
3    1809
4    1809
dtype: int64
```

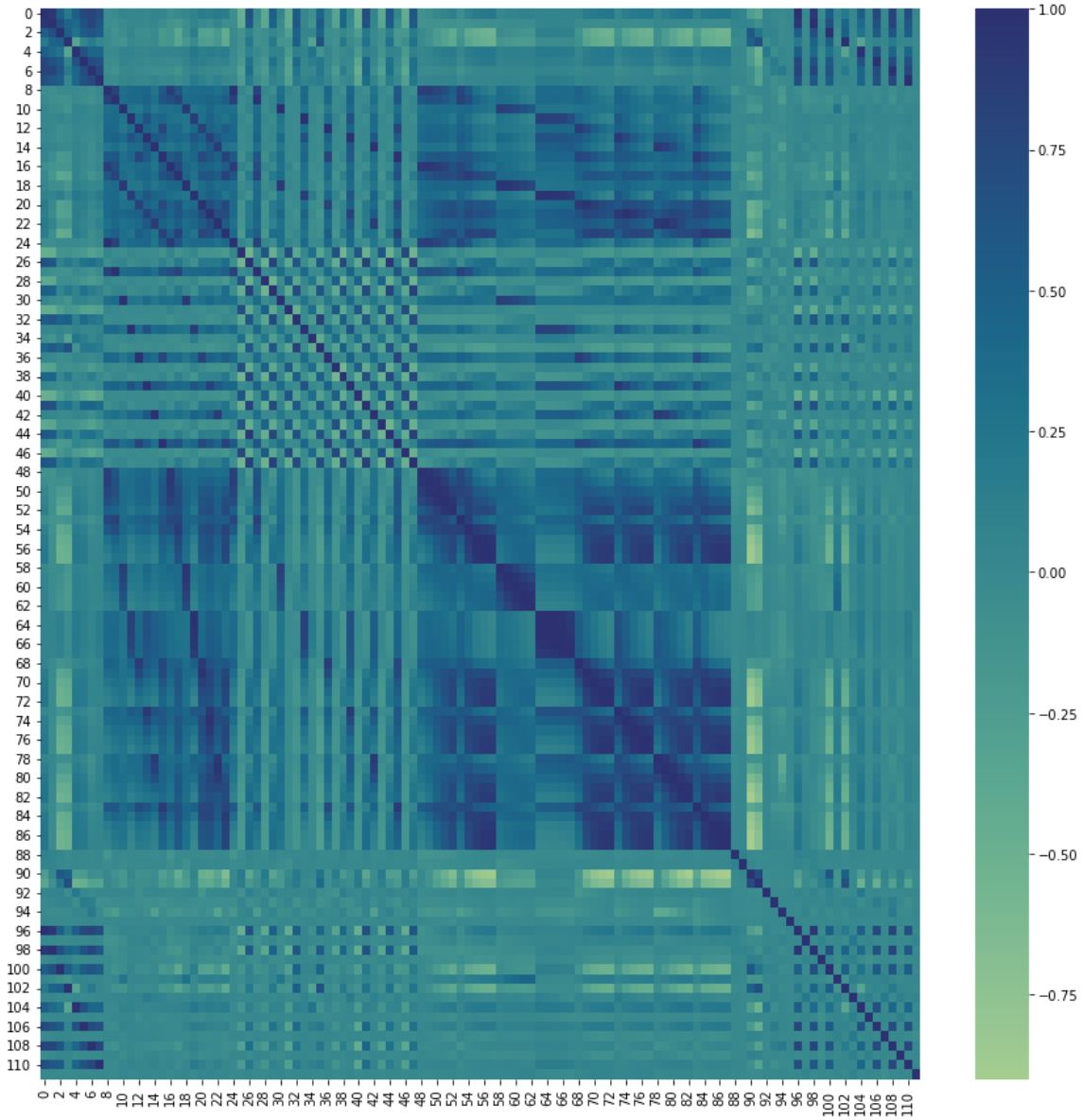
Matriz de Correlación

In [25]:

```
correlation = X_DATA_FEATURES.corr()
corr_matrix = np.array(correlation)
```

In [26]:

```
fig = plt.figure(figsize=(15,15))
corr_matrix_plot = sns.heatmap(corr_matrix, cmap='crest')
```



Análisis de componentes principales

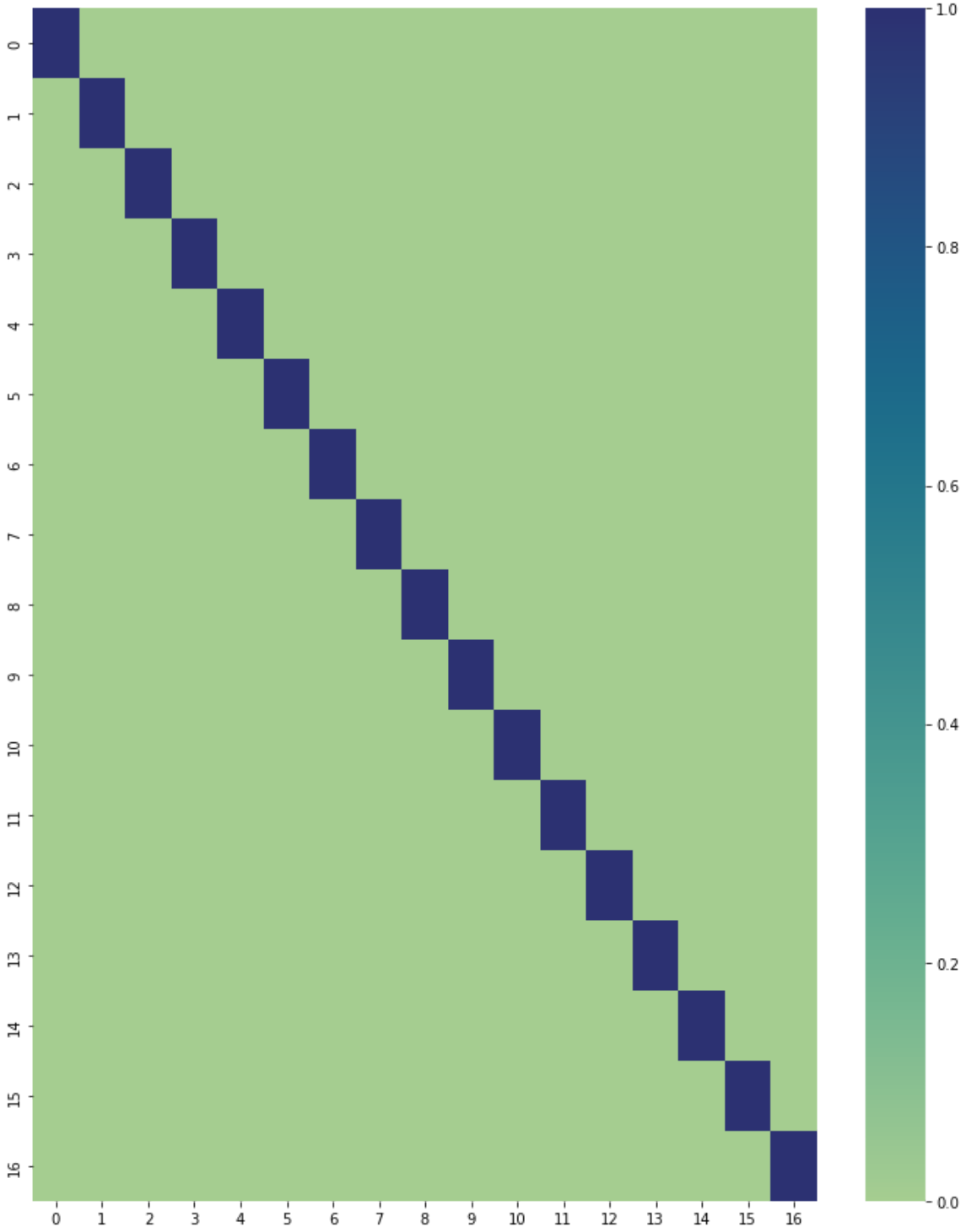
In [27]:

```
pca = PCA(n_components=0.95)
reduced_X = pca.fit_transform(X_DATA_FEATURES)
X_DATA_FEATURES_PCA = pd.DataFrame(reduced_X)
print(len(X_DATA_FEATURES_PCA.columns))
```

```
17
```

In [28]:

```
correlation = X_DATA_FEATURES_PCA.corr()
corr_matrix = np.array(correlation)
fig = plt.figure(figsize=(12,15))
corr_matrix_plot = sns.heatmap(corr_matrix, cmap='crest')
```



Separación de los datos en entrenamiento y testeo

In [29]:

```
x_train_FEATURES_PCA, x_test_FEATURES_PCA, y_train_FEATURES_PCA, y_test_FEATURES_PCA = tr
x_train_FEATURES, x_test_FEATURES, y_train_FEATURES, y_test_FEATURES = train_test_split(X
```

KNN

PCA

In [30]:

```
model_KNN_pca = KNeighborsClassifier(n_neighbors=3)
model_KNN_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA)
```

```
E:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.py:198: Da
taConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples,), for example using rave
l().
  return self._fit(X, y)
```

Out[30]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [31]:

```
pred_KNN_pca = model_KNN_pca.predict(x_test_FEATURES_PCA)
```

In [32]:

```
print(confusion_matrix(y_test_FEATURES_PCA, pred_KNN_pca))
print(classification_report(y_test_FEATURES_PCA, pred_KNN_pca))
```

```
[[517  16   7   6]
 [ 70 265  88 101]
 [ 22  14 484   9]
 [ 29  45  37 461]]
              precision    recall  f1-score   support

           1       0.81      0.95      0.87       546
           2       0.78      0.51      0.61       524
           3       0.79      0.91      0.85       529
           4       0.80      0.81      0.80       572

    accuracy                           0.80      2171
   macro avg       0.79      0.79      0.78      2171
weighted avg       0.79      0.80      0.79      2171
```

NO PCA

In [33]:

```
model_KNN = KNeighborsClassifier(n_neighbors=3)
model_KNN.fit(x_train_FEATURES, y_train_FEATURES)
```

```
E:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.py:198: Da
taConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples,), for example using rave
l().
  return self._fit(X, y)
```

Out[33]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [34]:

```
pred_KNN = model_KNN.predict(x_test_FEATURES)
```

In [35]:

```
print(confusion_matrix(y_test_FEATURES, pred_KNN))
print(classification_report(y_test_FEATURES, pred_KNN))
```

```
[[503  14   9   6]
 [ 73 290  71 121]
 [ 16  10 528   8]
 [ 39  43  21 419]]
              precision    recall  f1-score   support

           1       0.80      0.95      0.87       532
           2       0.81      0.52      0.64       555
           3       0.84      0.94      0.89       562
           4       0.76      0.80      0.78       522

    accuracy                           0.80      2171
   macro avg       0.80      0.80      0.79      2171
weighted avg       0.80      0.80      0.79      2171
```

Random Forest Classifier

PCA

In [36]:

```
model_RFC_pca = RandomForestClassifier(max_depth=20, random_state=0)
model_RFC_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA)
```

```
C:\Users\jonat\AppData\Local\Temp\ipykernel_18656\547018914.py:2: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  model_RFC_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA)
```

Out[36]:

```
RandomForestClassifier(max_depth=20, random_state=0)
```

In [37]:

```
pred_RFC_pca = model_RFC_pca.predict(x_test_FEATURES_PCA)
```

In [38]:

```
print(confusion_matrix(y_test_FEATURES_PCA, pred_RFC_pca))
print(classification_report(y_test_FEATURES_PCA, pred_RFC_pca))
```

```
[[491  37   6  12]
 [ 46 314  77  87]
 [ 16  24 480   9]
 [ 24  61  16 471]]
              precision    recall  f1-score   support

           1       0.85      0.90      0.87       546
           2       0.72      0.60      0.65       524
           3       0.83      0.91      0.87       529
           4       0.81      0.82      0.82       572

    accuracy                           0.81      2171
   macro avg       0.80      0.81      0.80      2171
weighted avg       0.80      0.81      0.80      2171
```

NO PCA

In [39]:

```
model_RFC = RandomForestClassifier(max_depth=20, random_state=0)
model_RFC.fit(x_train_FEATURES, y_train_FEATURES)
```

```
C:\Users\jonat\AppData\Local\Temp\ipykernel_18656\4095416588.py:2: DataCon
versionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  model_RFC.fit(x_train_FEATURES, y_train_FEATURES)
```

Out[39]:

```
RandomForestClassifier(max_depth=20, random_state=0)
```

In [40]:

```
pred_RFC = model_RFC.predict(x_test_FEATURES)
```

In [41]:

```
print(confusion_matrix(y_test_FEATURES, pred_RFC))
print(classification_report(y_test_FEATURES, pred_RFC))
```

```
[[500  23   4   5]
 [ 39 392  53  71]
 [  5  14 540   3]
 [ 10  48   4 460]]
              precision    recall  f1-score   support

           1       0.90      0.94      0.92       532
           2       0.82      0.71      0.76       555
           3       0.90      0.96      0.93       562
           4       0.85      0.88      0.87       522

    accuracy                           0.87      2171
   macro avg       0.87      0.87      0.87      2171
weighted avg       0.87      0.87      0.87      2171
```

Artificial Neural Networks

PCA

In [42]:

```
model_ANN_pca = Sequential()
model_ANN_pca.add(Dense(20,input_dim=17,activation='relu'))
model_ANN_pca.add(Dense(40,activation='relu'))
model_ANN_pca.add(Dense(64,activation='relu'))
model_ANN_pca.add(Dense(32,activation='relu'))
model_ANN_pca.add(Dense(8,activation='relu'))
model_ANN_pca.add(Dense(4,activation='softmax'))

model_ANN_pca.compile(optimizer='adam', loss='mean_squared_error', metrics = ['accuracy']
```

In [43]:

```
y_train_FEATURES_PCA=pd.get_dummies(y_train_FEATURES_PCA['cuadrante'])
y_test_FEATURES_PCA=pd.get_dummies(y_test_FEATURES_PCA['cuadrante'])
```

In [44]:

```
print(y_test_FEATURES_PCA)
```

```
      1  2  3  4
4287  1  0  0  0
5336  0  0  1  0
1234  0  1  0  0
5927  0  0  1  0
6118  0  0  1  0
...  .. .. .. ..
2960  0  1  0  0
6440  0  0  1  0
7     0  1  0  0
1251  0  1  0  0
1729  0  1  0  0

[2171 rows x 4 columns]
```

In [45]:

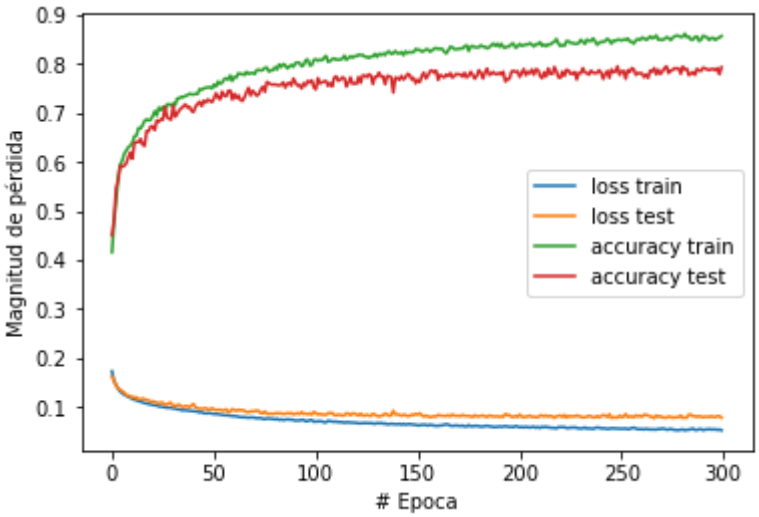
```
datos_modelo_pca=model_ANN_pca.fit(x_train_FEATURES_PCA, y_train_FEATURES_PCA, validation
```

```
127/127 [==============================] - 0s 3ms/step - loss: 0.1010 -
accuracy: 0.7106 - val_loss: 0.1077 - val_accuracy: 0.6817
Epoch 25/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1012 -
accuracy: 0.7054 - val_loss: 0.1093 - val_accuracy: 0.6863
Epoch 26/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1004 -
accuracy: 0.7123 - val_loss: 0.1086 - val_accuracy: 0.6859
Epoch 27/300
127/127 [==============================] - 0s 2ms/step - loss: 0.1000 -
accuracy: 0.7050 - val_loss: 0.1045 - val_accuracy: 0.7126
Epoch 28/300
127/127 [==============================] - 0s 2ms/step - loss: 0.0987 -
accuracy: 0.7169 - val_loss: 0.1052 - val_accuracy: 0.6914
Epoch 29/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0986 -
accuracy: 0.7175 - val_loss: 0.1083 - val_accuracy: 0.6877
Epoch 30/300
127/127 [==============================] - 0s 3ms/step - loss: 0.0987 -
accuracy: 0.7145 - val_loss: 0.1086 - val_accuracy: 0.6859
```

In [46]:

```
pred_ANN_pca = model_ANN_pca.predict(x_test_FEATURES_PCA)
_, train_acc_pca=model_ANN_pca.evaluate(x_train_FEATURES_PCA, y_train_FEATURES_PCA, verbo
_, test_acc_pca=model_ANN_pca.evaluate(x_test_FEATURES_PCA, y_test_FEATURES_PCA, verbose=
plt.xlabel("# Epoca")
plt.ylabel("Magnitud de pérdida")
plt.plot(datos_modelo_pca.history["loss"],label='loss train')
plt.plot(datos_modelo_pca.history["val_loss"],label='loss test')
plt.legend()
plt.plot(datos_modelo_pca.history["accuracy"],label='accuracy train')
plt.plot(datos_modelo_pca.history["val_accuracy"],label='accuracy test')
plt.legend()
plt.show()
```

```
68/68 [==============================] - 0s 1ms/step
```



In [47]:

```
y_test_FEATURES_PCA1=pd.DataFrame(y_test_FEATURES_PCA)
pred_ANN_pca1=pd.DataFrame(pred_ANN_pca,columns=[1,2,3,4])
y_test_FEATURES_PCA1['cuadrante']=pd.DataFrame(y_test_FEATURES_PCA1).idxmax(1)
pred_ANN_pca1['cuadrante']=pred_ANN_pca1.idxmax(1)
print(y_test_FEATURES_PCA)
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, conf
```

```
      1  2  3  4  cuadrante
4287  1  0  0  0           1
5336  0  0  1  0           3
1234  0  1  0  0           2
5927  0  0  1  0           3
6118  0  0  1  0           3
...  .. .. .. ..         ...
2960  0  1  0  0           2
6440  0  0  1  0           3
7     0  1  0  0           2
1251  0  1  0  0           2
1729  0  1  0  0           2

[2171 rows x 5 columns]
```

In [48]:

```
print(accuracy_score(y_test_FEATURES_PCA['cuadrante'], pred_ANN_pca1['cuadrante']))
print(classification_report(y_test_FEATURES_PCA['cuadrante'], pred_ANN_pca1['cuadrante']))
```

```
0.7927222478120681
              precision    recall  f1-score   support

           1       0.80      0.91      0.85       546
           2       0.78      0.54      0.64       524
           3       0.81      0.87      0.84       529
           4       0.78      0.84      0.81       572

    accuracy                           0.79      2171
   macro avg       0.79      0.79      0.78      2171
weighted avg       0.79      0.79      0.79      2171
```

Artificial Neural Networks

NO PCA

In [49]:

```
model_ANN = Sequential()
model_ANN.add(Dense(112,input_dim=112,activation='relu'))
model_ANN.add(Dense(128,activation='relu'))
model_ANN.add(Dense(64,activation='relu'))
model_ANN.add(Dense(32,activation='relu'))
model_ANN.add(Dense(8,activation='relu'))
model_ANN.add(Dense(4,activation='softmax'))

model_ANN.compile(optimizer='adam', loss='mean_squared_error', metrics = ['accuracy'])
```

In [50]:

```
y_train_FEATURES=pd.get_dummies(y_train_FEATURES['cuadrante'])
y_test_FEATURES=pd.get_dummies(y_test_FEATURES['cuadrante'])
```

In [51]:

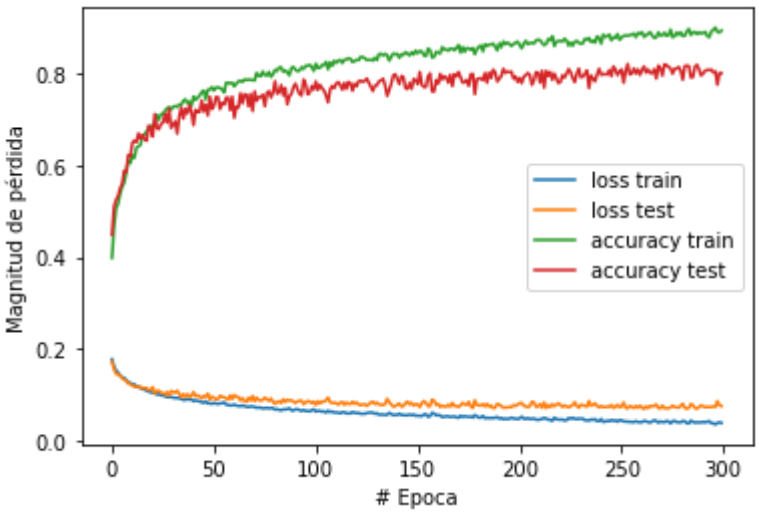
```
datos_modelo=model_ANN.fit(x_train_FEATURES, y_train_FEATURES, validation_data=(x_test_FE
```

```
Epoch 1/300
127/127 [==============================] - 3s 5ms/step - loss: 0.1773 -
accuracy: 0.3986 - val_loss: 0.1726 - val_accuracy: 0.4491
Epoch 2/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1612 -
accuracy: 0.4657 - val_loss: 0.1557 - val_accuracy: 0.5104
Epoch 3/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1529 -
accuracy: 0.5066 - val_loss: 0.1454 - val_accuracy: 0.5251
Epoch 4/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1481 -
accuracy: 0.5169 - val_loss: 0.1440 - val_accuracy: 0.5334
Epoch 5/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1420 -
accuracy: 0.5390 - val_loss: 0.1416 - val_accuracy: 0.5504
Epoch 6/300
127/127 [==============================] - 0s 3ms/step - loss: 0.1379 -
accuracy: 0.5540 - val_loss: 0.1358 - val_accuracy: 0.5606
Epoch 7/300
```

In [52]:

```
pred_ANN = model_ANN.predict(x_test_FEATURES)
_, train_acc=model_ANN.evaluate(x_train_FEATURES, y_train_FEATURES, verbose=0)
_, test_acc=model_ANN.evaluate(x_test_FEATURES, y_test_FEATURES, verbose=0)
plt.xlabel("# Epoca")
plt.ylabel("Magnitud de pérdida")
plt.plot(datos_modelo.history["loss"],label='loss train')
plt.plot(datos_modelo.history["val_loss"],label='loss test')
plt.legend()
plt.plot(datos_modelo.history["accuracy"],label='accuracy train')
plt.plot(datos_modelo.history["val_accuracy"],label='accuracy test')
plt.legend()
plt.show()
```

```
68/68 [==============================] - 0s 2ms/step
```



In [ ]:

In [53]:

```
y_test_FEATURES2=pd.DataFrame(y_test_FEATURES)
pred_ANN1=pd.DataFrame(pred_ANN,columns=[1,2,3,4])
y_test_FEATURES2['cuadrante']=pd.DataFrame(y_test_FEATURES2).idxmax(1)
pred_ANN1['cuadrante']=pred_ANN1.idxmax(1)
print(y_test_FEATURES2)
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, conf
```

```
      1  2  3  4  cuadrante
1664  0  0  0  1          4
5223  0  0  1  0          3
56    0  0  0  1          4
4648  1  0  0  0          1
4753  1  0  0  0          1
...  .. .. .. ..        ...
6609  0  0  0  1          4
812   0  1  0  0          2
434   0  1  0  0          2
1721  0  0  0  1          4
1497  0  0  1  0          3

[2171 rows x 5 columns]
```

In [54]:

```
print(accuracy_score(y_test_FEATURES['cuadrante'], pred_ANN1['cuadrante']))
print(classification_report(y_test_FEATURES['cuadrante'], pred_ANN1['cuadrante']))
```

```
0.8014739751266697
              precision    recall  f1-score   support

           1       0.84      0.88      0.86       532
           2       0.75      0.59      0.66       555
           3       0.84      0.94      0.89       562
           4       0.76      0.80      0.78       522

    accuracy                           0.80      2171
   macro avg       0.80      0.80      0.80      2171
weighted avg       0.80      0.80      0.80      2171
```

In [ ]:

In [ ]:

In [ ]: