Universidad Autónoma del Caribe

Facultad de Ingeniería

Programa de Ingeniería Electrónica y Telecomunicaciones

Aplicación Web para la Detección de Patologías en Señales EEG

Sebastian Nevado Meza

Mario Armando Bonilla Brito

Colombia, Barranquilla

2021

Aplicación Web para la Detección de Patologías en Señales EEG

Sebastián Nevado Meza

Mario Armando Bonilla Brito

Trabajo de grado presentado para optar el título de Ingeniero Electrónico y en Telecomunicaciones

Director

Gisella Borja Roncallo

Co- Director

José Ledesma León

Universidad Autónoma del Caribe

Facultad de Ingeniería

Programa de Ingeniería Electrónica y Telecomunicaciones

Colombia, Barranquilla

2021

Resumen

El trastorno depresivo mayor (MDD, por sus siglas en inglés) (también conocido como depresión clínica, depresión mayor, depresión o trastorno unipolares depresivo o como depresión recurrente, en el caso de presentarse episodios repetidos) es una enfermedad mental que se caracteriza por un estado de ánimo invasivo y persistente acompañado de una baja autoestima y una pérdida de interés o de placer (anhedonia) en actividades que normalmente se considerarían entretenidas. Las personas deprimidas tienen una esperanza de vida más corta que aquellos que no tienen depresión, en parte porque los pacientes deprimidos corren el riesgo de morir por suicidio. No obstante, también tienen una tasa bruta de mortalidad más alta por otras causas siendo más susceptibles a afecciones médicas tales como enfermedades del corazón. Hasta un 60% de los suicidios se asocian con algún trastorno del estado de ánimo, como la depresión mayor y el riesgo es especialmente alto si la persona sufre de desesperanza o tiene depresión y trastorno límite de la personalidad. En este proyecto se pretende poder no solo detectar anomalías dentro de los bio-potenciales captados mediante el EEG, tanto anomalías en este como catalogar el trastorno de depresión mayor mediante el uso de la Inteligencia Artificial apoyado en las librerías que ofrece el lenguaje de programación Python para la creación de redes neuronales Autoencoders, crear un servidor local en nuestro PC para peticiones HTTP una API (Interfaz de programación de aplicaciones) para la transferencia de información con el lado del cliente , y a su vez la creación de una interfaz gráfica amigable con el usuario creada en React.js, que pretende utilizarla, se estima que esta aplicación sea un apoyo para las personas que padecen MDD y los especialistas que la tratan.

Abstract

Major depressive disorder (MDD) (also known as clinical depression, major depression, depression or unipolar depressive disorder or as recurrent depression, in the case of repeated episodes) is a mental illness characterized by a invasive and persistent mood accompanied by low self-esteem and a loss of interest or pleasure (anhedonia) in activities that would normally be considered entertaining. Depressed people have a shorter life expectancy than those without depression, in part because depressed patients are at risk of dying by suicide. However, they also have a higher crude death rate from other causes and are more susceptible to medical conditions such as heart disease. Up to 60% of suicides are associated with a mood disorder, such as major depression, and the risk is especially high if the person suffers from hopelessness or has depression and borderline personality disorder. This project aims to be able not only to detect anomalies within the bio-potentials captured by the EEG, both anomalies in this and to catalog the major depression disorder through the use of Artificial Intelligence supported by the libraries offered by the programming language Python for creating neural networks Autoencoders, creating a local server on our PC for HTTP requests an API (Application Programming Interface) for the transfer of information with the client side, and in turn creating a friendly graphical interface with the user created in React.js, who intends to use it, this application is considered to be a support for people suffering from MDD and the specialists who treat it.

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Barranquilla, julio de 2021

Dedicatoria

Dedicamos este proyecto a nuestras familias, amigos y a todas las personas que participaron directa e indirectamente para realizar este sueño. Le pedimos a Dios que nos habrá puertas que nadie pueda cerrar para que el mundo pueda ver la madera que tenemos de ingenieros.

Agradecimientos

Durante el camino hacia el cumplimiento de todo proyecto, nunca es seguro el éxito, que no habrá problemas, inconvenientes o que simplemente todo estar bien… eso no es seguro, ni probable.

Lo que sí es muy probable y seguro es que, durante todo tu camino, encontrarás aspectos y situaciones que tal vez no conocías aun, procesos y actitudes ajenas a las que alguna vez pudiste haber llegado a sentir, pero tan normales como todo lo que sucede en tu cotidianidad.

El darte cuenta que las cosas no eran tan fáciles como al inicio de tu proyecto parecía puede desanimarte, pero debes tener fe y esperanza, creer que siempre hay algo más; que si realizas las cosas con excelencia, siempre tendrás una muy buena recompensa, debes más que creer en ti mismo; darte cuenta de que habrán momentos y situaciones en las que simplemente las cosas se saldrán de tus manos, pero con la confianza puesta en Dios, todo saldrá bien.

Agradecemos a todos los docentes que nos guiaran en el camino.

Contenido

Lista de Tablas

Lista de Figuras

Lista de Anexos

Introducción

Siendo la **psiquiatría** la especialidad médica que estudia las enfermedades mentales, sus tipos, causas, cursos y tratamientos. Las investigaciones muestran que más del 95% de las personas que se suicidan tienen trastorno de depresión que cuestan aproximadamente 800.000 víctimas anualmente (según la OMS). Es importante tratar la depresión cuanto antes para ayudar a prevenir una crisis de salud mental. Considerando lo anterior el trastorno de depresión mayor que se caracteriza por depresión persistente o pérdida de interés en las actividades, lo que puede causar dificultades significativas en la vida cotidiana. El software pretende dar apoyo a los estudios de psiquiatría mediante el análisis de señales EEG, el cual es una prueba que se usa para estudiar el funcionamiento del sistema nervioso central, concretamente de la actividad de la corteza del cerebro. Consiste esencialmente en registrar mediante electrodos especiales las corrientes eléctricas que se forman en las neuronas cerebrales y que son la base del funcionamiento del sistema nervioso (Gil-Nagel, A. (2019)).

. Gracias a él se pueden diagnosticar alteraciones de la actividad eléctrica cerebral que sugiera enfermedades como la epilepsia, la narcolepsia o demencias. En este caso de pacientes diagnosticados con trastorno de depresión mayor y personas “sanas” mediante el uso de las redes neuronales (Blumcke,2019)

Las redes neuronales la mejora de la técnica y la tecnología que se ha empezado a utilizarlas en todos lados reconocimiento de caracteres de imágenes de voz predicción bursátil, generación de texto traducción de idiomas, prevención de fraude, conducción autónoma, análisis genético pronóstico de enfermedades, se trata de una familia de algoritmos muy potentes con los que podemos modelar comportamientos inteligentes funcionan como suele ocurrir con la mayoría de comportamientos y estructuras avanzadas. la

complejidad de estos sistemas emerge de la interacción de muchas partes más simples trabajando conjuntamente. en el caso de una red neuronal que en este caso permiten clasificar de manera correcta una señal MDD mediante el uso de Autoencoders comparándolos con la señal de entrada de la consulta y mediante la detección de anomalías usando Autoencoder una técnica de aprendizaje no supervisada en la que se aprovecha las redes neuronales para la tarea de aprendizaje de representación. Específicamente, se diseña una arquitectura de red neuronal de tal manera que se imponga un cuello de botella en la red que fuerce una representación de conocimiento comprimido de la entrada original. Si las entidades de entrada fueran independientes entre sí, esta compresión y posterior reconstrucción sería una tarea muy difícil. Sin embargo, si existe algún tipo de estructura en los datos (es decir, correlaciones entre las entidades de entrada), esta estructura se puede aprender y, en consecuencia, aprovechar al forzar la entrada a través del cuello de botella de la red (Álvarez-Linera,2019). Para finalizar, una interfaz amigable con el usuario. Dividida en 3 pasos como es el login para registro y logueo de usuario, menú principal para procesar el examen EEG y el informe que se general al cargar el archivo en el menú principal con los datos del logueado y su diagnóstico completo.

Capítulo 1

Descripción del Proyecto

Planteamiento del Problema

El estudio de los bio-potenciales en el ámbito psiquiátrico para evaluación funcional fisiológica: Aunque útil es subestimado muchas veces no se ve reflejada su utilidad por problemas de clasificación con actividades regulares y distintos padecimientos (Guerrero, 2014). La depresión es una causa importante de morbilidad en todo el mundo. Se cree que actualmente afecta a unos millones de personas a partir de 2010 (el 4.3% de la población mundial) **(**Respuesta de la OMS). La prevalencia de vida es muy variable, desde un 3% en Japón hasta un 17% en los EE.UU. En la mayoría de los países el número de personas que van a tener depresión durante su vida está dentro del rango de 8 a 12%. En Norteamérica, la probabilidad de tener un episodio depresivo mayor en el plazo de un año es de 3 a 5% para los hombres y de 8 a 10% para las mujeres. Estudios de población han demostrado consecuentemente que la depresión mayor es aproximadamente dos veces más común en las mujeres que en los hombres, aunque no está claro por qué esto es así y si factores desconocidos están contribuyendo a esta situación. (Respuesta de la OMS)

Formulación del Problema

¿Cómo podemos aprovechar el análisis de los biopotenciales que se reflejan en un electroencefalograma como apoyo para el diagnóstico de Desorden de Depresión Mayor (MDD)?

Impacto Esperado

La detección de anomalías correspondientes al trastorno de depresión mayor catalogado como trastorno de salud mental, que contribuya a los estudios psiquiátricos que buscan la detección temprana de estos problemas de salud mental.

Usuarios Directos e Indirectos

Comunidad Psiquiátrica y personas con MDD en su actividad cerebral.

Objetivos

Objetivo General

Desarrollar una aplicación web capaz de detectar anomalías en un EEG correspondiente al Desorden de Depresión Mayor como complemento a los estudios de psiquiatría.

Objetivos Específicos

- Implementar un algoritmo para la caracterización de señales EEG de acuerdo con sus componentes.
- Crear una red neuronal artificial capaz de detectar anomalías relacionadas con el DDM en una señal EEG.
- Construir una interfaz gráfica amigable con el usuario para una visualización práctica de los resultados.

Metodología

El presente proyecto es considerado de investigación aplicada porque el problema está establecido y es conocido por el investigador, por lo que utiliza la investigación para dar respuesta a preguntas específicas. Considerando que la Investigación Aplicada se basa en una necesidad social práctica por resolver problemas.

El presente proyecto se desarrolló en cuatro fases:

Fase 1: Caracterización de las señales EEG

Para la caracterización de las señales se utilizó una base de datos de acceso libre, construida por Centre for Intelligent Signal and Imaging (CISIR), Universiti Teknologi PETRONAS, Tronoh, Malaysia. Para la construcción de la base de datos EEG-based Diagnosis and Treatment Outcome Prediction for Major Depressive Disorder, según CISIR, la adquisición de datos EEG incluyó 5 minutos de grabaciones de datos EEG durante los ojos cerrados (EC) y 5 minutos de grabaciones EEG durante las condiciones de los ojos abiertas (EO). La adquisición de datos EEG involucró sensores de tapa EEG de 19 canales colocados según el estándar de colocación de electrodos 10-20 con oído vinculado (LE) como referencia. En aras del análisis EEG, los datos EEG se volvieron a hacer referencia a la referencia infinita (IR) como sugiere. Los 19 electrodos que cubren el cuero cabelludo incluyen frontal (Fp1, Fp2, F3, F4, F7, F8, Fpz), temporal (T3, T4, T5, T6), parietal (P3, P4, P7, P8), occipital (O1, O2), Central (C3, C4). La tapa EEG de 19 canales estaba conectada con un amplificador de Brain Máster Systems.

De la base de datos se seleccionan las relacionadas a canales del lóbulo frontal, puesto que investigaciones previas manifiestan que existe cierta evidencia de patología frontal en los cuadros de depresión mayor. La depresión mayor está asociada a una reducción del metabolismo frontal tanto en las presentaciones unipolares como en las bipolares. Además, existe evidencia de una selectiva atrofia cortical y alteraciones en la arquitectura cortical frontal en pacientes con depresión mayor. Los accidentes cerebrovasculares están estrechamente relacionados con síndromes depresivos post-ACV. Depresiones mayores francas pueden ser consecuencias de lesiones en los ganglios basales. Los déficits ejecutivos que acompañan a la depresión mejoran notablemente con la resolución de sus síntomas.

Para la creación de las redes neuronales la etapa de filtrado y demás cantidades de procesos subyacentes dentro de la creación de los algoritmos que son necesarios para la resolución de los distintos planteamientos se eligió Python como el motor para el proyecto, ya que cuenta con una cantidad considerable de librerías que son el apoyo para el análisis, manipulación datos y por supuesto para la creación de los modelos de AI. Aparte de contar con la versatilidad, legibilidad y limpieza de Python hacen que este lenguaje de programación sea excelente para la creación de sistemas completos de inteligencia artificial.

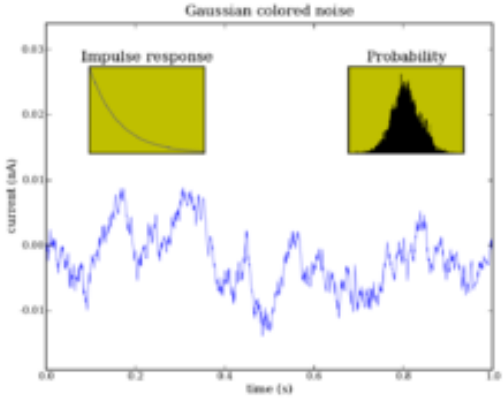
Dentro de la amplia gama de librerías que tiene Python se escoge para la manipulación de datos la librería Pandas que depende de Numpy, librería que añade un potente tipo matricial a Python. Datos tabulares con columnas de tipo heterogéneo con etiquetas en columnas y filas. Permite la lectura, escritura y manipulación de datos de distintos formatos (CSV, Microsoft Excel, bases SQL y formato HDF5) con los que se trabajan a lo largo del proyecto. La librería numpy da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.

En conjunto con la librería Numpy se usa Scipy que compone herramientas y algoritmos matemáticos. SciPy contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs y otras tareas para la ciencia e ingeniería. Entre su alta gama de funcionalidades destacamos la creación y manipulación de filtros que se usan para la extracción de las bandas de frecuencia en sus ritmos alfa, delta, beta, theta y gamma.

Para la visualización de los datos se usó Matplotlib que es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy.

Figura 1.

Ejemplo de una Gráfica en Matplotlib



Nota: Fuente: (https://commons.wikimedia.org/wiki/File:Matplotlib.png)

Fase 2: Creación de la red neuronal

Para la creación de la red neuronal se utiliza KERAS que es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano.

KERAS contiene varias implementaciones de los bloques constructivos de las redes neuronales como por ejemplo los layers, funciones objetivo, funciones de activación, optimizadores matemáticos que se utilizó para la creación de la red neuronal que se creó. Cargando los datos anteriormente filtrados para el proceso de experimentación obteniendo la media y la desviación estándar de ellas, además de crear secuencias que combinen valores de datos contiguos a partir de la señal de entrenamiento. Se construye el modelo Autoencoder de reconstrucción convolucional. El modelo tomará la entrada de una forma y devolverá la salida de la misma forma con sus respectivas capas de entrada, capas ocultas y de salida, creando así un modelo y posteriormente entrenándolo.

Se detectan las anomalías determinando que tan bien el modelo puede reconstruir los datos de entrada y se plantean los siguientes puntos:

- Encontrar la pérdida de MAE en las muestras de entrenamiento.
- Encontrar el valor máximo de pérdida MAE. Esto es lo peor que nuestro modelo ha realizado tratando de reconstruir una muestra. Haremos que esto sea el para la detección de anomalías. threshold
- Si la pérdida de reconstrucción para una muestra es mayor que este valor, entonces podemos inferir que el modelo está viendo un patrón con el que no está familiarizado. Etiquetaremos esta muestra como un archivo. thresholdanomaly

Luego de esto se procede a comparar la reconstrucción de los datos y preparar los datos de prueba para la comparación de los datos de prueba.

Fase 3: Desarrollo de la API

Para la creación y desarrollo de la API se utilizaron 3 tecnologías y con estas se implementó localmente en 3 partes:

Parte 1: Lado del servidor con Python en un script donde se encuentra el proceso de clasificación de señal y entrega de resultados que con esta viene su respuesta en frecuencia apoyados en la transformada de Fourier con Scipy.

Parte 2: Una aplicación Node.js se ejecuta en un único proceso, sin crear un nuevo subproceso para cada solicitud. Cuando Node.js realiza una operación de E/S, como leer desde la red, acceder a una base de datos o al sistema de archivos, en lugar de bloquear el subproceso y desperdiciar ciclos de CPU esperando, Node.js reanudará las operaciones cuando vuelva la respuesta.

Esto permite a Node.js controlar miles de conexiones simultáneas con un único servidor sin introducir la carga de administrar la simultaneidad de subprocesos, que podría ser una fuente importante de errores.

En Node.js los nuevos estándares ECMAScript se pueden usar sin problemas, ya que no tiene que esperar a que todos sus usuarios actualicen sus navegadores: usted es el encargado de decidir qué versión de ECMAScript usar cambiando la versión de Node.js, y también puede habilitar características experimentales específicas ejecutando Node.js con indicadores.

Mediante node.js se realiza la automatización de scripts a través de peticiones http que provienen del lado del cliente hacia el servidor lo cual permite la interacción ambos basados en el lenguaje ECMA6 y a través de la lectura de archivos de las librerías de Python se ejecuta el script que permite la clasificación y con el formato JSON la entrega del resultado dependiendo del umbral preestablecido en las pruebas anteriores

Parte 3: Es la validación de la API local en la interfaz amigable con el usuario donde el usuario desde el front-end envía su examen EEG al back-end y la API hace su función enviándole los resultados de dependiendo de cada examen cargado o procesado.

Fase 4: Validación

Luego de haberse creado la API se hizo una interfaz gráfica amigable con el usuario para una visualización práctica de los resultados este proceso consta de 3 eventos:

Login

La parte del login se hizo con SQLite, es una biblioteca en lenguaje C que implementa un motor de base de datos SQL pequeño, rápido, autónomo, de alta confiabilidad y con todas las

funciones. Y express que es un marco de aplicaciones web de node.js mínimo y flexible que proporciona un sólido conjunto de características para aplicaciones web y móviles.

Con el login se hace el registro para que el usuario agregue sus datos en paralelo a esto se pintó la interfaz de inicio de sesión y registro con reactjs que como librería de JavaScript se pudo enlazar con Nodejs que es un tiempo de ejecución de JavaScript basado en el motor de JavaScript V8 de Chrome para hacer la consulta y los registros de usuario

Menú Principal

Al estar el usuario logueado carga el archivo y procesa su Examen EEG. Básicamente con requerimientos de nodejs el usuario le envía su archivo EEG (la ruta en donde está el archivo con un txt) al Backend (la API creada), procesa el archivo con un clic y envía un resultado en formato Json de la probabilidad de predicción y aparte del formato, la API manda un gráfico de muestra de las anomalías procesada y en frecuencia de este archivo.

Cuando el Usuario procesa el archivo la app le muestra su probabilidad de predicción sobre si tiene el MDD o no

Resultado Obtenido de la API(Informe)

Al presionar y procesar el archivo la interfaz entrega un informe con esos resultados que la API envió, el usuario le da clic a “Mostrar informe” y lo redirige al informe generado donde se puede imprimir con (clic derecho – imprimir).

De igual forma, para validar el correcto funcionamiento de la aplicación web se llevaron a cabo pruebas del error absoluto medio y de la efectividad de la reconstrucción de la señal.

Materiales y Equipos Utilizados

En la Tabla 1 se presentan los materiales y equipos utilizados para el desarrollo del presente proyecto.

Tabla 1.

Materiales y Equipos utilizados

Materiales	Equipos Utilizados
(HTML, CSS, JS, REACTJS, NODEJS) Front-end {Editor de Código VS code}	Acer Aspire3 PC de mesa 20GB de RAM, Procesador amd x6core.
(PYTHON y sus librerías) Back-end {Editor de Código VS code}	Acer Aspire3 procesador i5 8va generación 3.40 GHz, MacBook retina 12, core duo Intel HD graphics 5300 1536 MB PC de mesa 20GB de RAM, Procesador amd x6core.

Nota: Fuente: Elaboración propia.

Capítulo 2

Marco Teórico y Estado del Arte

Electroencefalografía

La señal electroencefalográfica (EEG) se adquiere de la superficie del cráneo, esta registra la actividad eléctrica de la corteza cerebral y refleja el funcionamiento cerebral. A partir del estudio de estas señales se puede acercar a enfermedades cerebrales tales como la epilepsia

Durante muchos años se pensó que los ataques de epilepsia empezaban abruptamente, pero se ha mostrado que se desarrollan durante un tiempo antes de que se desencadene la crisis. Un análisis del EEG permitiría no solo la "predicción" del ataque, sino también una mayor compresión de la enfermedad y así mejorar la calidad de vida de los pacientes *Blumcke, I., Álvarez-Linera, J., Del Pozo, J. M., & Gil-Nagel, A. (2019).*

El problema de prever los ataques está en encontrar particularidades a lo largo de la señal, con este fin se seguirá un análisis en dos etapas, primero extraer características de la señal, descriptores, y segundo utilizar métodos de clasificación.

En este trabajo se obtuvo diferentes descriptores de la señal, tanto temporales como frecuentes, como características de la señal y métodos de clasificación no supervisada como, LAM DA, C-means, K-means y montaña. *Gauthier Umaña, C. (2005). Análisis de señales de electroencefalografía (eeg) para la detección temprana de crisis de epilepsia (Bachelor's thesis, Uniandes).*

Desarrollo y evaluación de un algoritmo para la compresión de señales electrocardiográficas fetales que contribuya a que los profesionales en medicina puedan realizar seguimiento mediante la transmisión, almacenamiento y tratamiento de señales, a la salud fetal evidenciado en las variaciones del ritmo y morfología del electrocardiograma, de manera que puedan tempranamente diagnosticar algún tipo de enfermedad relacionada con la hipoxia y la variación en la frecuencia cardiaca. Con este fin se propone un algoritmo de compresión sin

pérdidas soportado en herramientas tales como: Transformada wavelet discreta y redes neuronales, que permitan detectar los picos R y el segmento ST del complejo QRS de la señal fetal, para umbralizarlos, filtrarlos, cuantificarlos, y codificarlos, etapas que hacen parte de lo que se conoce como compresión, manteniendo un nivel bajo de error, por debajo de la unidad y una rata de compresión por encima de 20. Para llevar a la práctica este desarrollo se presenta primero una revisión bibliográfica respecto a los diferentes estudios y/o modelos que se han publicado, con base en la aplicación de transformada wavelet discreta, separación de señales, codificadores y clasificadores basados en aprendizaje no supervisado, como lo son las redes neuronales, con el fin de enfocar esfuerzos en un método que aporte a los estándares ya empleados. Seguidamente se selecciona mediante un algoritmo la transformada wavelet madre que mejor rendimiento presenta, teniendo como parámetro la dispersión producida por la descomposición multinivel. También se aplica el método de umbralizaciòn seleccionando el valor de umbral de manera automática y teniendo en cuenta cada nivel de descomposición, mediante el uso de una red neuronal. El siguiente paso es la codificación de la señal aplicando codificación de Huffman. Como se ha podido observar la selección de una transformada wavelet madre por cada señal y la aplicación de umbralización multinivel, permite el mejoramiento de la rata de comprensión, manteniendo valores

de porcentaje de error por debajo de la unidad. *Jiménez Pinto, G. Algoritmo de compresión para la detección del segmento st de una señal electrocardiográfica fetal basado en la transformada wavelet y umbralización multinivel.*

El uso de señales de electroencefalografía EEG permite estudiar el registro de la actividad

eléctrica del cerebro bajo condiciones de vigilia, sueño o patológicas como lo es la epilepsia, enfermedades degenerativas, enfermedades cerebrovasculares, entre otras.

En caso de la epilepsia, es un trastorno cerebral caracterizado por excitación y comportamiento eléctrico anormal del cerebro y de las neuronas. La epilepsia puede ser causada por diferentes factores, como lo son lesiones, infecciones, accidentes cerebrovasculares, anomalías congénitas, tumores, etc. Estos factores mencionados dan como resultado comportamientos eléctricos caracterizados por la patología que el paciente presente, es decir que dado a la patología las señales eléctricas captadas por el EEG adquieren patrones únicos que los médicos experimentados logran identificar. Así entonces, al conocer dichos patrones es posible identificar la patología de un paciente a partir del EEG en crisis. Clínicamente el contexto que se presenta es que las cirugías se practican mayormente en adultos, dado a que se puede saber con mayor claridad el origen patológico de la crisis. En el caso de pacientes pediátricos o menores requiere mayor esfuerzo conocer dicho origen, lo cual hace que se practiquen pocas cirugías a este tipo de pacientes. Por lo que encontrar los patrones mencionados y poder conocer el origen patológico de la epilepsia en un paciente pediátrico aumenta la razón de cirugía a temprana edad, lo cual es beneficioso dado a que a edades tempranas hay una mayor reorganización cerebral. *Flórez Torres,* N. (2014). Procesamiento y clasificación de señales de electroencefalografía (EEG) de pacientes epilépticos según patología y expresión eléctrica de crisis usando aprendizaje de máquina (Bachelor's thesis, Bogotá-Uniandes).

Desarrollo de un algoritmo para determinar el riesgo de muertes en pacientes dentro de una unidad de cuidado intensivo utilizando múltiple regresión no lineal *UAC, Cindy García García (2010).*

MDD

El trastorno depresivo mayor (Major depressive disorder, MDD) es un problema de salud grave. El MDD va más allá que solo sentirse “triste” o “desanimado”. Es un problema de salud

grave: un problema que, generalmente, lo trata un médico especialista en salud mental. Si no se trata correctamente, el MDD puede interferir en muchos aspectos de la vida de una persona, como, por ejemplo, trabajar, estudiar, relacionarse, dormir y comer. El MDD se caracteriza por episodios que van más allá de sentirse triste o desanimado temporalmente. Algunas personas pueden tener un solo episodio en toda su vida. Otras pueden tener muchos episodios durante su vida.

Cómo reconocer los signos y síntomas del MDD

Algunos síntomas habituales que pueden presentar las personas que padecen MDD son los siguientes:

- Tristeza, ansiedad o sentimientos de vacío
- Sentimientos de desesperanza o negatividad
- Sentimientos de culpa, inutilidad o desamparo
- Intranquilidad o irritabilidad
- Falta de interés en las cosas
- Cansancio, poca energía
- Dificultad para pensar con claridad, recordar detalles o tomar decisiones
- Dificultad para dormir o dormir demasiado
- Comer en exceso o no estar interesado en la comida
- Pensamientos suicidas o intentos de suicidio
- Dolores, dolor de cabeza, calambres o problemas estomacales

que no desaparecen, incluso si se los trata

Muchas personas que padecen MDD pueden sentirse mejor con la ayuda de un tratamiento. El tratamiento puede incluir medicamentos que ayudan a reducir los síntomas del MDD. También

puede incluir sesiones de terapia con un profesional de la salud mental para ayudar a solucionar problemas personales que pueden agravar los síntomas del MDD.

Tratamiento para el MDD

Si su ser querido recibe tratamiento para el MDD, es importante que este siga el plan de tratamiento indicado por el médico. El plan de tratamiento puede incluir

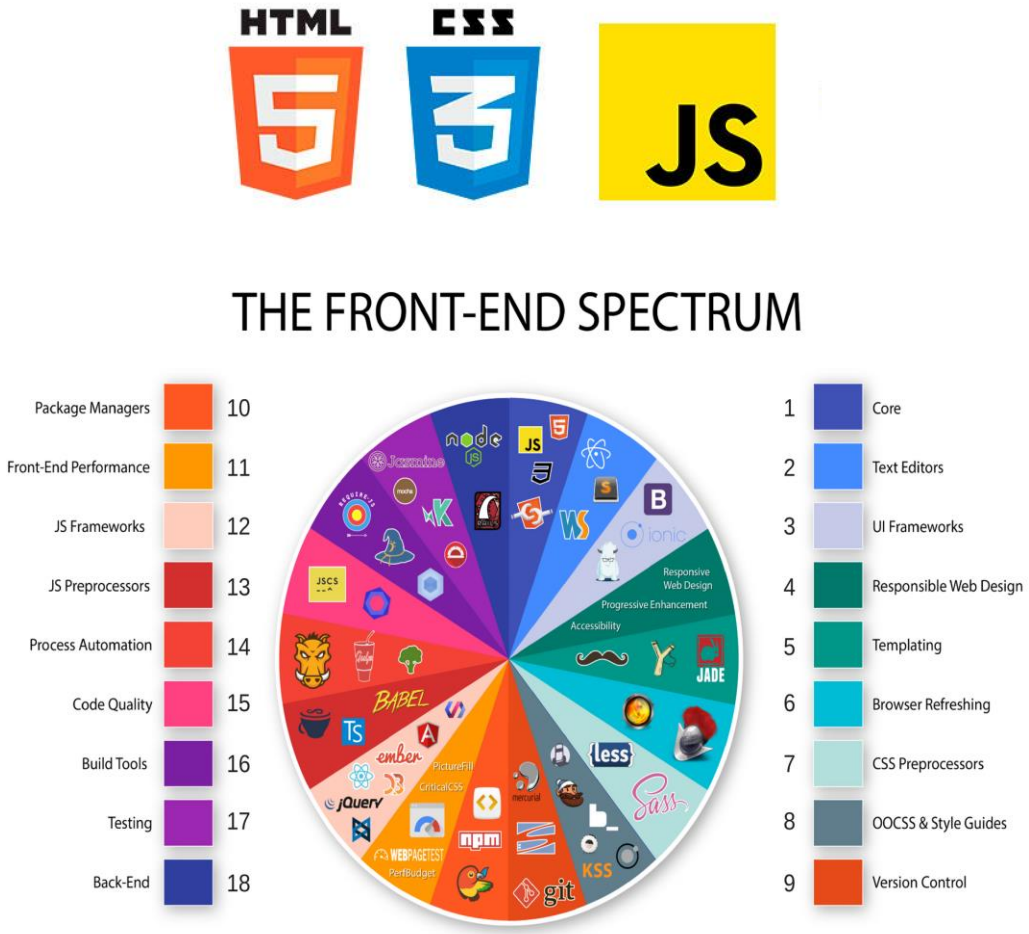
sesiones de psicoterapia y medicamentos para el MDD. Algunos medicamentos para tratar esta afección pueden tener efectos secundarios. El paciente debe comunicarse con su médico si presenta efectos secundarios. Aliente a su ser querido a hablar con su médico sobre qué puede suceder al tomar los medicamentos para tratar el MDD.

Frontend la parte de una aplicación que interactúa con los usuarios, es conocida como el lado del cliente Básicamente es todo lo que se ve en la pantalla cuando se accede a un sitio web o aplicación: tipos de letra, colores, adaptación para distintas pantallas (RWD), los efectos del ratón, teclado, movimientos, desplazamientos, efectos visuales… y otros elementos que permiten navegar dentro de una página web. Este conjunto crea la experiencia del usuario. Como se ha dicho, el desarrollador front-end se encarga de la experiencia del usuario, es decir, en el momento en el que este entra a una página web, debe ser capaz de navegar por ella, por lo que el usuario verá una interfaz sencilla de usar, atractiva y funcional.

Un desarrollador front-end debe conocer los siguientes lenguajes de programación: HTML5, CSS3, JavaScript.

Figura 2.

Front-End



Nota: Fuente: (Pelletier, J. 2015, noviembre 1)

HTML

HyperText Markup Language (‘lenguaje de marcado de hipertexto’), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del *World Wide Web Consortium* (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. HTML se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, *script*, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

HTML es un lenguaje de marcado que nos permite indicar la estructura de nuestro documento mediante etiquetas. Este lenguaje ofrece una gran adaptabilidad, una estructuración lógica y es fácil de interpretar tanto por humanos como por máquinas.

Figura 3.

Estructura HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Example</title>
        <link rel="stylesheet" href="st
    </head>
    <body>
        <h1>
            <a href="/">Header</a>
        </h1>
        <nav>
            <a href="one/">One</a>
            <a href="two/">Two</a>
            <a href="three/">Three</a>
        </nav>
```

Nota: Fuente (Reisio, 2012)

JavaScript

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas y JavaScript del lado del servidor (*Server-side JavaScript* o *SSJS*). Su uso en aplicaciones externas a la web, por

ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Desde 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de JavaScript. Los navegadores más antiguos soportan por lo menos ECMAScript 3. La sexta edición se liberó en julio de 2015.

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

JavaScript en el lado servidor

Netscape introdujo una implementación de script del lado del servidor con Netscape Enterprise Server, lanzada en diciembre de 1994 (poco después del lanzamiento de JavaScript para navegadores web). A partir de mediados de la década de los 2000, ha habido una proliferación de implementaciones de JavaScript para el lado servidor. Node.js es uno de los notables ejemplos de JavaScript en el lado del servidor, siendo usado en proyectos importantes.

Figura 4

JavaScript con HTML

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Ejemplo sencillo</title>
</head>
<body>
    <h1 id="header">Esto es JavaScript</h1>

    <script>
        document.body.appendChild(document.createTextNode('Hola Mundo!'));

        var h1 = document.getElementById('header'); // contiene la referencia al tag <h1>
        h1 = document.getElementsByTagName('h1')[0]; // accediendo al mismo elemento <h1>
    </script>

    <noscript>Tu navegador no admite JavaScript, o JavaScript está deshabilitado.</noscript>
</body>
</html>
```

Nota: Fuente(https://es.wikipedia.org/wiki/JavaScript)

Librería

Chart.js

Es una biblioteca JavaScript gratuita de código abierto para la visualización de datos, que admite 8 tipos de gráficos:

- barra
- línea
- área
- circular
- burbuja
- radar
- polar

- dispersión.

Chart.js se representa en HTML5 Canvas y está ampliamente considerado como una de las mejores bibliotecas de visualización de datos.

Está disponible bajo la licencia MIT.

JSON

JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que resulta mucho más sencillo escribir un analizador sintáctico (pares) para él. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval()

Sintaxis

Los tipos de datos disponibles con JSON son:

Números: Se permiten números negativos y opcionalmente pueden contener parte fraccional separada por puntos. Ejemplo: 123.456

Cadenas: Representan secuencias de cero o más caracteres. Se ponen entre doble comilla y se permiten cadenas de escape. Ejemplo: "Hola"

Booleanos: Representan valores booleanos y pueden tener dos valores: true y false

null: Representan el valor nulo.

Array: Representa una lista ordenada de cero o más valores los cuales pueden ser de cualquier tipo. Los valores se separan por comas y el vector se mete entre corchetes. Ejemplo ["juan","pedro","jacinto"]

Objetos: Son colecciones no ordenadas de pares de la forma **<nombre>:<valor>** separados por comas y puestas entre llaves. El nombre tiene que ser una cadena entre comillas dobles. El valor puede ser de cualquier tipo. Ejemplo:

Figura 5.

Formato Json

Nota: Fuente (Yahoo!.Using JSON with Yahoo! Web services 2010)

React.js

Es una biblioteca JavaScript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre. En el proyecto hay más de mil desarrolladores libres.

React intenta ayudar a los desarrolladores a construir aplicaciones que usan datos que cambian todo el tiempo. Su objetivo es ser sencillo, declarativo y fácil de combinar. React sólo maneja la interfaz de usuario en una aplicación; React es la Vista en un contexto en el que se use el patrón MVC (Modelo-Vista-Controlador) o MVVM (Modelo-vista-modelo de vista). También puede ser utilizado con las extensiones de React-based que se encargan de las partes no-UI (que no forman parte de la interfaz de usuario) de una aplicación web.

Características

Virtual DOM

React mantiene un virtual DOM propio, en lugar de confiar solamente en el DOM del navegador. Esto deja a la biblioteca determinar qué partes del DOM han cambiado comparando contenido entre la versión nueva y la almacenada en el virtual DOM, y utilizando el resultado para determinar cómo actualizar eficientemente el DOM del navegador. Propiedades (props) de react.

Las propiedades

Las propiedades (también conocidas como 'props') pueden definirse como los atributos de configuración para dicho componente. Éstas son recibidas desde un nivel superior, normalmente al realizar la instancia del componente y por definición son inmutables.

El Estado

El estado de un componente se define como una representación del mismo en un momento concreto, es decir, una instantánea del propio componente. Existen dos tipos de componentes con y sin estado, denominados statefull y stateless.

Ciclos de vida

El ciclo de vida es una serie de estados por los cuales pasan los componentes statefull a lo largo de su existencia. Se pueden clasificar en tres etapas de montaje o inicialización, actualización y destrucción. Dichas etapas tienen correspondencia en diversos métodos.

shouldComponentUpdate permite al desarrollador prevenir el re-renderizado innecesario de un componente, devolviendo falso si no es necesario.

componentDidMount es llamado una vez que el componente es "montado" (el componente ha sido creado en la interfaz de usuario, usualmente asociándolo con el nodo del DOM). Esto es comúnmente usado para provocar la carga de datos desde una fuente remota a través de una API.

componentWillUnmount es llamado inmediatamente antes de que el componente es "desmontado". Es comúnmente usado para limpiar la demanda de dependencias del componente que no será simplemente removido con el desmontaje del componente.

render es el método más importante de los ciclos de vida y el único requerido en cualquier componente. Es usualmente llamado cada vez que el estado del componente es actualizado, reflejando los cambios en la interfaz de usuario.

JSX

React utiliza una sintaxis parecida a HTML, llamada JSX. No es necesaria para utilizar React, sin embargo, hace el código más legible, y escribirlo es una experiencia similar a HTML.

Figura 6.

Componente JSX

react.js

```
import React from 'react'; //Se importa React

class Componente extends React.Component{ //Declaración de un componente con el nombre de "Componente".
  render(){
    //Dentro de return se escribe en una sintaxis muy parecida a HTML
    return(
      <div>
        <h1>Hola mundo</h1>
      </div>
    );
  }
}

ReactDOM.render(<Componente />, document.getElementById('MiAppDeReact'))
```

Nota: Ejemplo de un componente escrito con JSX

NODE.js

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009 y su evolución está apadrinada por la empresa Joyent, que además tiene contratado a Dahl en plantilla.

npm es el sistema de gestión de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript, bajo Artistic License 2.0.

RNN (Redes Neuronales Artificiales)

Las redes neuronales artificiales (también conocidas como sistemas conexionistas) son un modelo computacional el que fue evolucionando a partir de diversas aportaciones científicas que están registradas en la historia. Consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitir señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo valores de salida.

Cada neurona está conectada con otras a través de unos enlaces. En estos enlaces el valor de salida de la neurona anterior es multiplicado por un valor de peso. Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Del mismo modo,

a la salida de la neurona, puede existir una función limitadora o umbral, que modifica el valor resultado o impone un límite que no se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

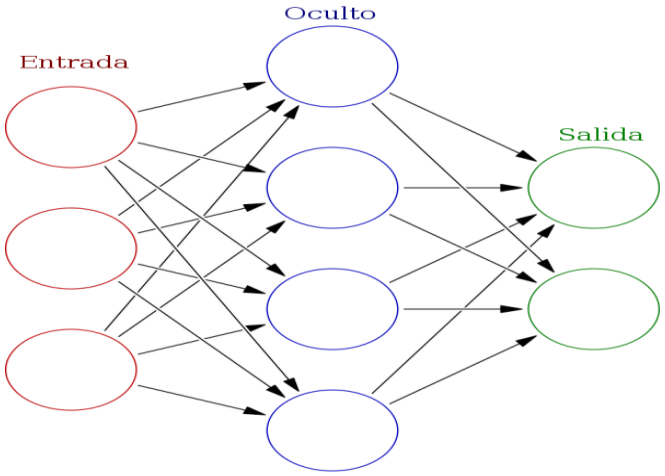
Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional. Para realizar este aprendizaje automático, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su totalidad. Los valores de los pesos de las neuronas se van actualizando, buscando reducir el valor de la función de pérdida. Este proceso se realiza mediante la propagación hacia atrás.

El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque las redes neuronales son más abstractas. Las redes neuronales actuales suelen contener desde unos miles a unos pocos millones de unidades neuronales.

Nuevas investigaciones sobre el cerebro a menudo estimulan la creación de nuevos patrones en las redes neuronales. Un nuevo enfoque está utilizando conexiones que se extienden mucho más allá y capas de procesamiento de enlace en lugar de estar siempre localizado en las neuronas adyacentes. Otra investigación está estudiando los diferentes tipos de señal en el tiempo que los axones se propagan, como el aprendizaje profundo, interpola una mayor complejidad que un conjunto de variables booleanas que son simplemente encendido o apagado.

Figura 7.

Red neuronal



Nota: Fuente: (Introduction to autoencoders Jeremy Jordan 2018.)

Las redes neuronales se han utilizado para resolver una amplia variedad de tareas, como la visión por computador y el reconocimiento de voz, que son difíciles de resolver usando la ordinaria programación basada en reglas. Históricamente, el uso de modelos de redes neuronales marcó un cambio de dirección a finales de los años ochenta de alto nivel, que se caracteriza por sistemas expertos con conocimiento incorporado en si-entonces las reglas, a bajo nivel de aprendizaje automático, caracterizado por el conocimiento incorporado en los parámetros de un modelo cognitivo con algún sistema dinámico.

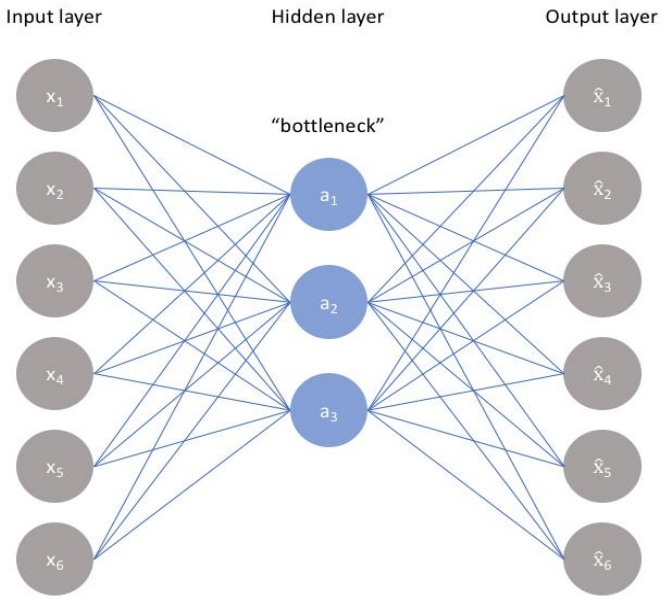
Autoencoder

Los autocodificadores (Autoencoders) son una técnica de aprendizaje no supervisada en la que aprovechamos las redes neuronales para la tarea de aprendizaje de la representación. Específicamente, diseñaremos una arquitectura de red neuronal de manera que impongamos un

cuello de botella en la red que fuerce una representación de conocimiento comprimido de la entrada original. Si las características de entrada fueran independientes entre sí, esta compresión y reconstrucción posterior serían una tarea muy difícil. Sin embargo, si existe algún tipo de estructura en los datos (es decir, correlaciones entre las características de entrada), esta estructura se puede aprender y, en consecuencia, aprovechar al forzar la entrada a través del cuello de botella de la red.

Figura8.

Autoencoders



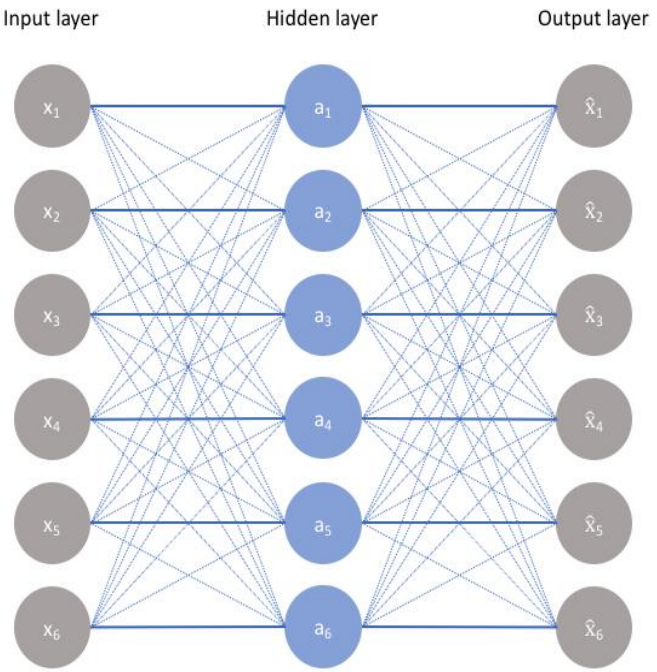
Nota: Fuente: (Jeremy Jordan Introduction to autoencoders. 2018)

Como se visualizó anteriormente, podemos tomar un conjunto de datos sin etiquetar y enmarcarlo como un problema de aprendizaje supervisado con salida $\hat{x}$, una reconstrucción de la entrada X, Esta red se puede entrenar minimizando el error de reconstrucción $L(x,\hat{x})$,que mide las

diferencias entre nuestra aportación original y la consiguiente reconstrucción. El cuello de botella es un atributo clave de nuestro diseño de red; sin la presencia de un cuello de botella de información, nuestra red podría aprender fácilmente a memorizar fácilmente los valores de entrada pasando estos valores a través de la red.

Figura 9.

Bottleneck



Nota: Fuente: (Introduction to autoencoders. Jeremy Jordan 2018)

Un cuello de botella restringe el montaje de información que puede atravesar toda la red, forzando una compresión aprendida de los datos de entrada.

El modelo de autoencoder ideal equilibra lo siguiente:

Sensible a las entradas lo suficiente como para construir con precisión una reconstrucción.

Lo suficientemente insensible a las entradas que el modelo no memoriza simplemente ni sobre ajusta los datos de entrenamiento.

Esta compensación obliga al modelo a mantener solo las variaciones en los datos necesarios para reconstruir la entrada sin aferrarse a las redundancias dentro de la entrada. Para la mayoría de los casos, esto implica la construcción de una función de pérdida donde un término anima a nuestro modelo a ser sensible a las entradas (es decir, pérdida de reconstrucción ($L(x,\hat{x})$) y un segundo término desalienta la memorización/sobreajuste (es decir, un regularizador añadido).

Normalmente añadiremos un parámetro de escalado delante del término de regularización para que podamos ajustar la compensación entre los dos objetivos

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Librerías

Pandas

Pandas es un paquete de Python que proporciona estructuras de datos similares a los dataframes de R. Pandas depende de Numpy, la librería que añade un potente tipo matricial a Python.

NumPy

Es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.

SciPy

Es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos. SciPy contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs y otras tareas para la ciencia e ingeniería.

Matplotlib

Es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy.

KERAS

Es una biblioteca de Redes Neuronales de Código Abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano.

Capítulo 3

Análisis de Resultados y Propuesta Ingenieril

Algoritmo para la caracterización de señales EEG de acuerdo con sus componentes.

Se encuentra que el estándar europeo de los archivos que contienen la información de los EEG son. edf y para una mejor manipulación de datos en Python se procede a realizar una conversión de estos archivos a un formato clásico del manejo de bases de datos .csv esto se logra apoyados en la librería ‘mne’ que brinda Python y se obtiene de manera satisfactoria el resultado esperado sin pérdidas en el cambio de formato.

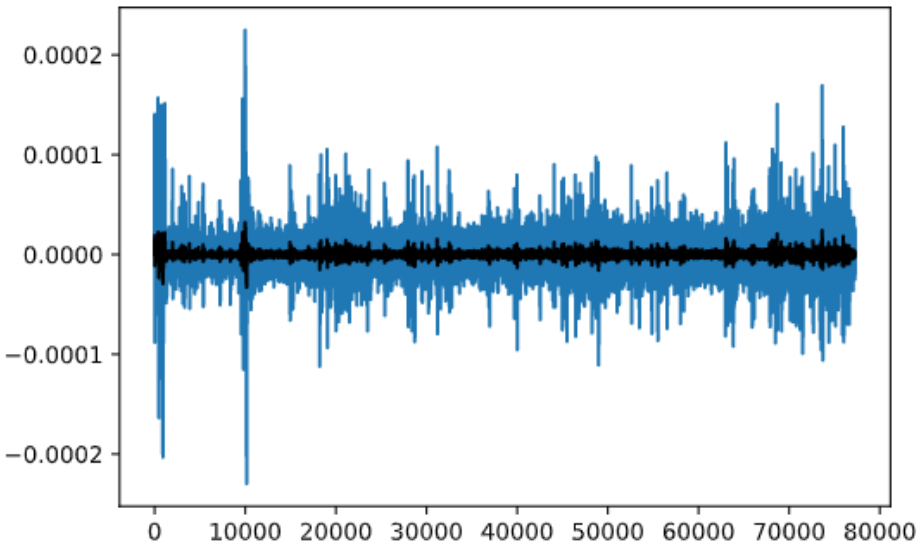
Como principal problemática dentro del análisis de señales de EEG se observa la cantidad de ruido que tienen estas señales, el EEG tiene diferentes “bandas”, definidas por la frecuencia de las ondas; ondas delta (lentas) de menos de 4 Hz; bandas theta de 4-8 Hz, las Alpha de 8 a 12 Hz, la beta de aproximadamente 14-30 Hz y la gamma de 30-80 Hz. Para obtener las características relevantes de la señal se procede a crear un filtro para poder extraer esas bandas de frecuencia que nos interesan.

Una comparación correcta de los resultados obtenidos en señales tanto de una persona sana, así como de sujeto con MDD serían las siguientes figuras:

Una

Figura 10.

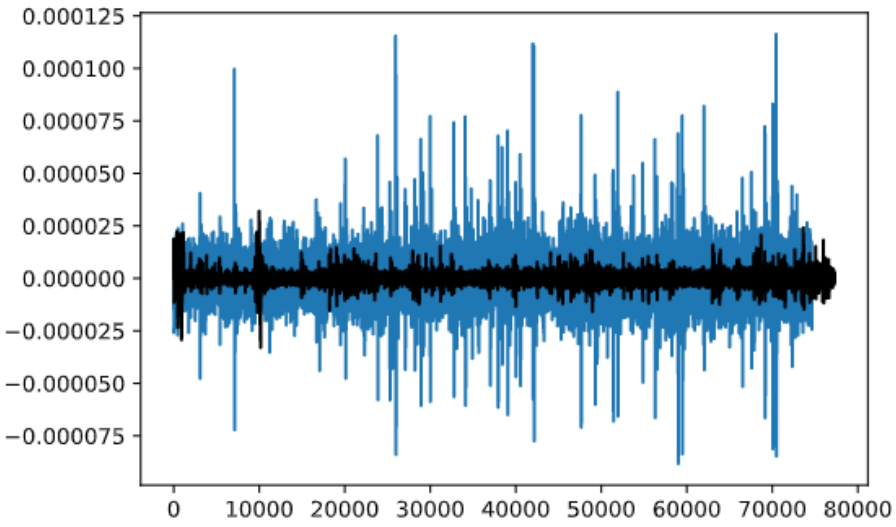
Señal filtrada sana 80hz



Nota (Generada por Matplotlib 2021)

Figura 11.

Señal filtrada sick 80hz



Nota (Señal generada con MatPlotlib,2021)

Logrado ya el filtrado de los datos teniendo como soporte la librería de keras para el análisis de la señal nos apoyamos en el electrodo FP1 esto teniendo como referencia lo establecido en la sección de metodología donde se explica brevemente el porque enfocarse en el lóbulo frontal. y descartar el método PCA usado en otros proyectos ya que la aplicación de la reducción de dimensiones a los datos de EEG mediante el análisis de componentes principales reduce la calidad de su posterior descomposición de componentes independientes sumado a una pérdida de información que pudiese ser significativa dentro de un esquema tan complejo como lo es las señales EEG.

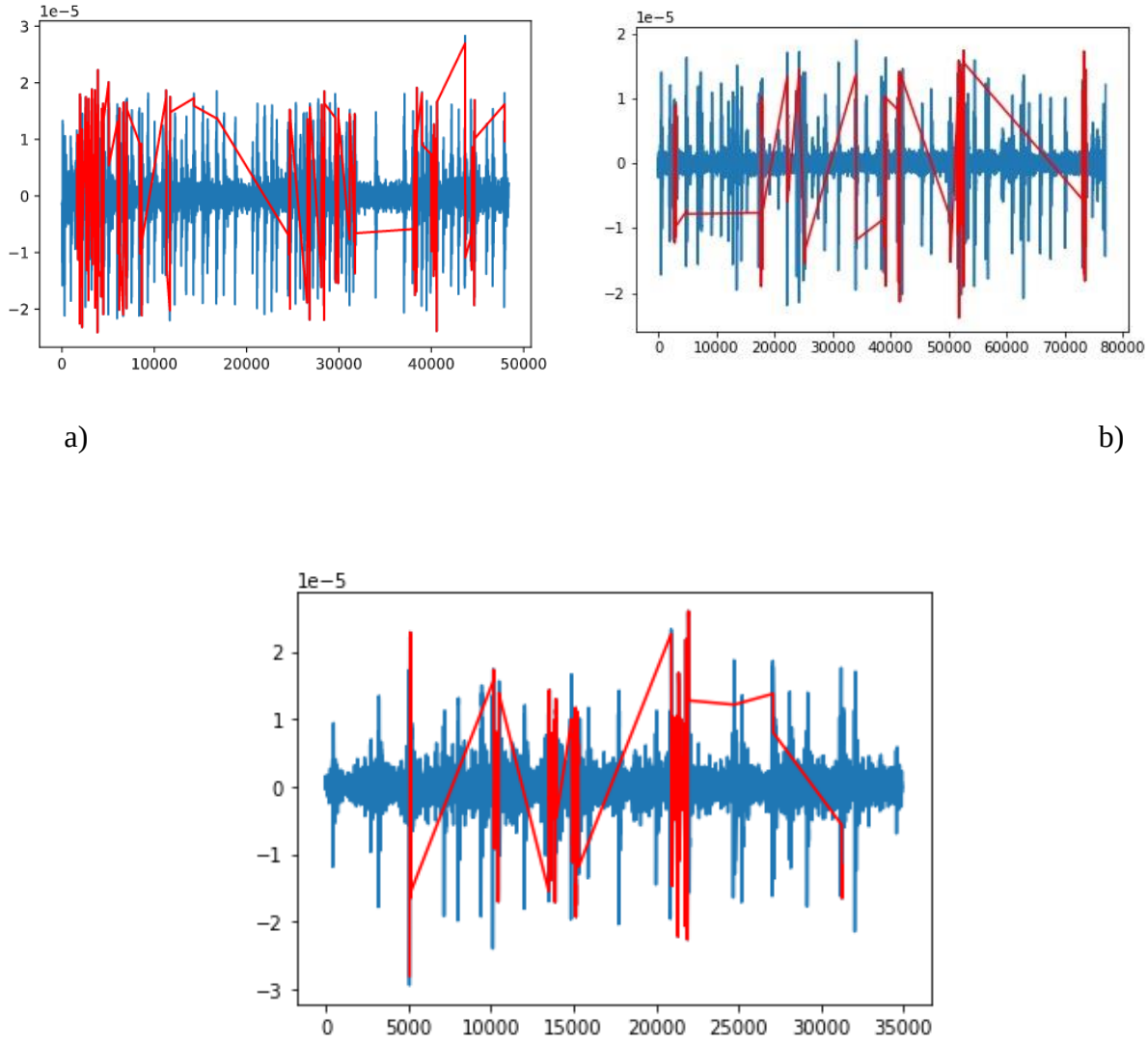
Si bien se logra apreciar en las figuras anteriores la diferencia entre las señales no es posible observar todas las diferencias que existen entre las señales y por supuesto también las similitudes entre estas. por eso apoyado en la librería keras se usó para entrenar redes neuronales autoencoders para la detección de anomalías en los, por supuesto partiendo de la idea de que ningún EEG es igual al otro se procede a probar un tanto los siguientes escenarios:

- Sano vs Sano
- Sano vs MDD
- MDD vs MDD

Con resultados satisfactorios ahora es más fácil poder categorizar a los pacientes y detectar anomalías redundantes dentro de un EEG

Figura 12.

Anomalía MDD vs MDD



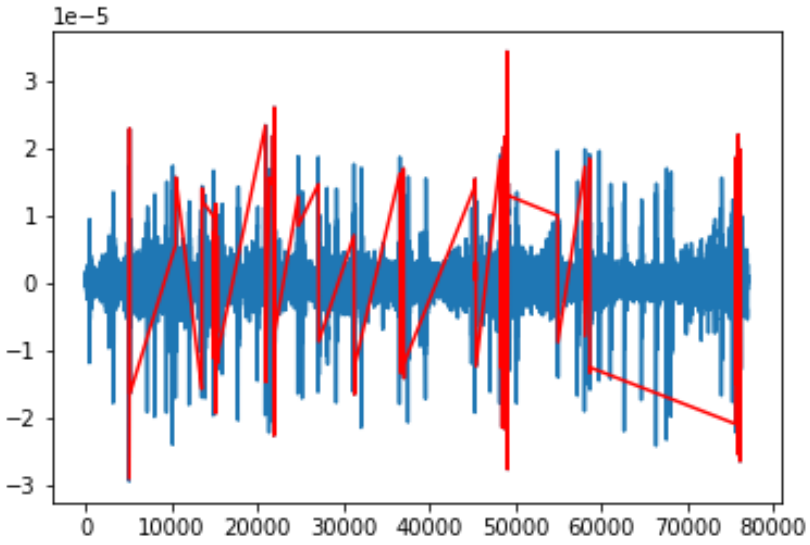
a) b)

c)

Nota: a) Anomalías de unas señales típicas de pacientes enfermo y paciente A b) Anomalías de unas señales típicas de pacientes enfermo y paciente B. c) Anomalías de unas señales típicas de pacientes enfermo y paciente C, Señal generada con MatPlotlib,2021

Figura13.

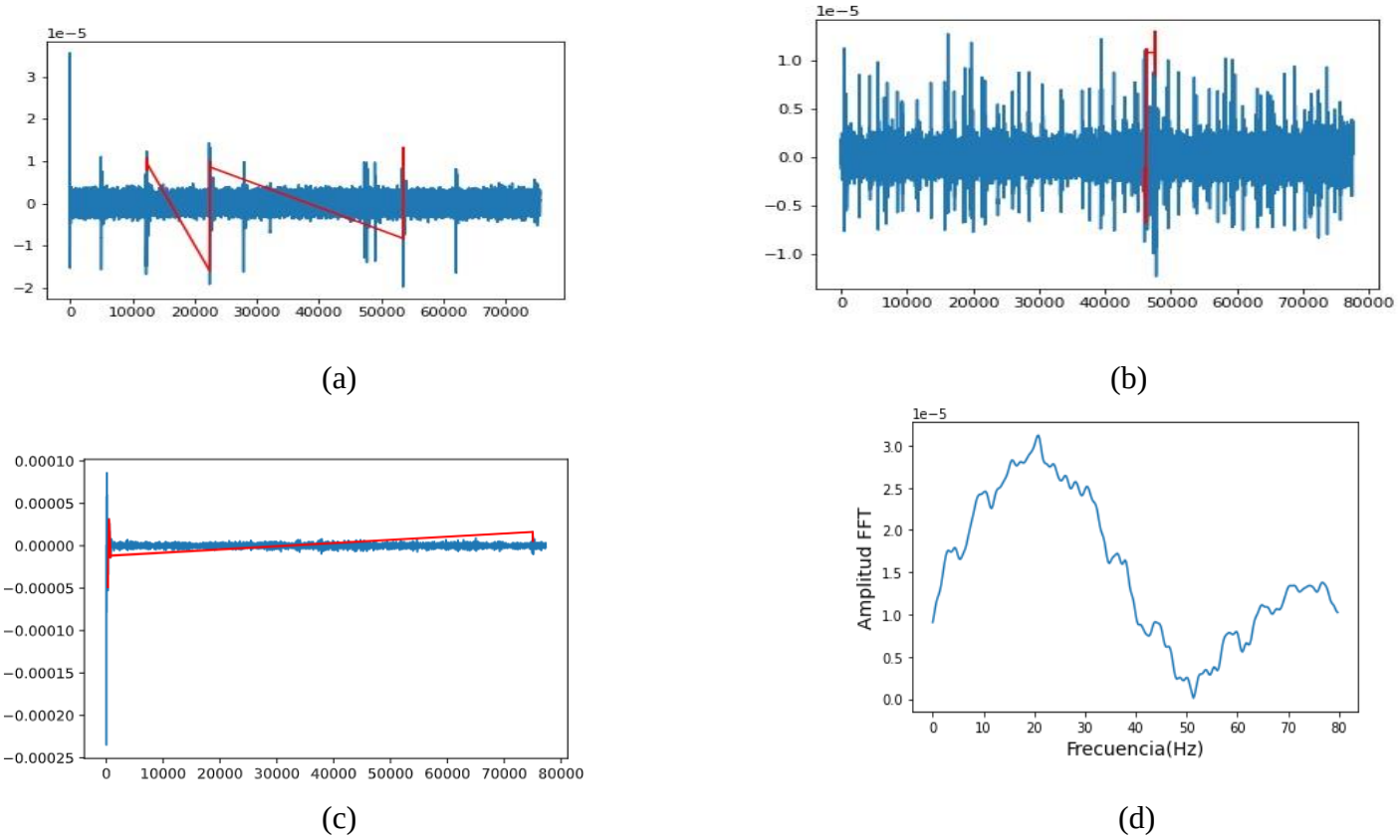
Anomalías SanosvsMDD



Nota (Señal generada con MatPlotlib, 2021)

Figura 14.

Anomalías: SanovsSano



Nota: a) Anomalías de unas señales típicas de pacientes sano y paciente A b) Anomalías de unas señales típicas de pacientes sano y paciente B. c) Anomalías de unas señales típicas de pacientes sano y paciente C d) Respuesta en frecuencia, Señal generada con MatPlotlib,2021

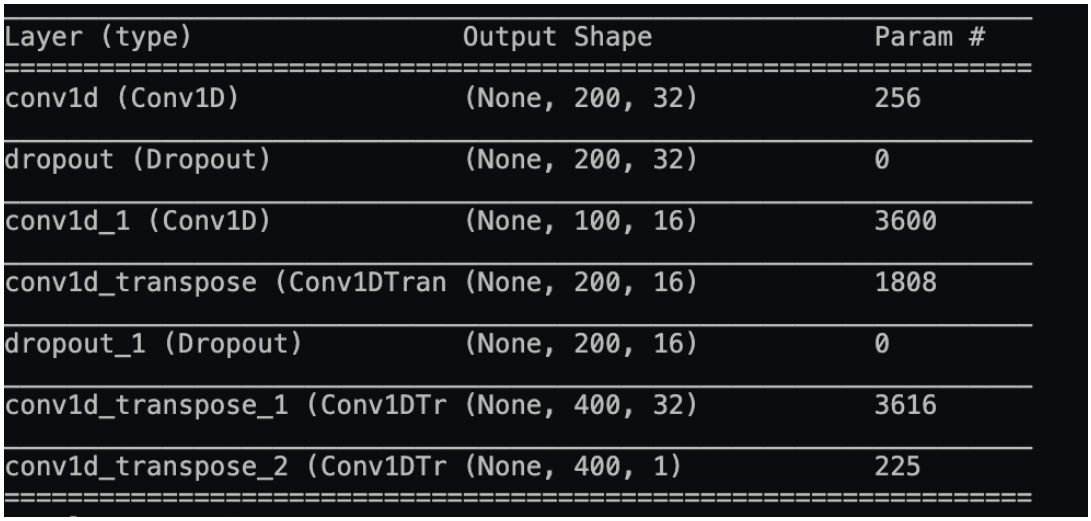
Basado en estos Resultados pudiendo categorizar las señales mediante la detección de anomalías se procede a crear una neurona que establezca un umbral basado en la longitud de las pruebas presentadas anteriormente.

Red neuronal artificial para la detección del MDD en una señal EEG.

Se crea un modelo autoencoder de reconstrucción convolucional con sus respectivas capas de entrada y capas ocultas y de la misma forma a la salida

Figura 15.

ModeloKeras



Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 200, 32)	256
dropout (Dropout)	(None, 200, 32)	0
conv1d_1 (Conv1D)	(None, 100, 16)	3600
conv1d_transpose (Conv1DTran	(None, 200, 16)	1808
dropout_1 (Dropout)	(None, 200, 16)	0
conv1d_transpose_1 (Conv1DTr	(None, 400, 32)	3616
conv1d_transpose_2 (Conv1DTr	(None, 400, 1)	225

Con una pérdida de reconstrucción real del 0.381432635415466

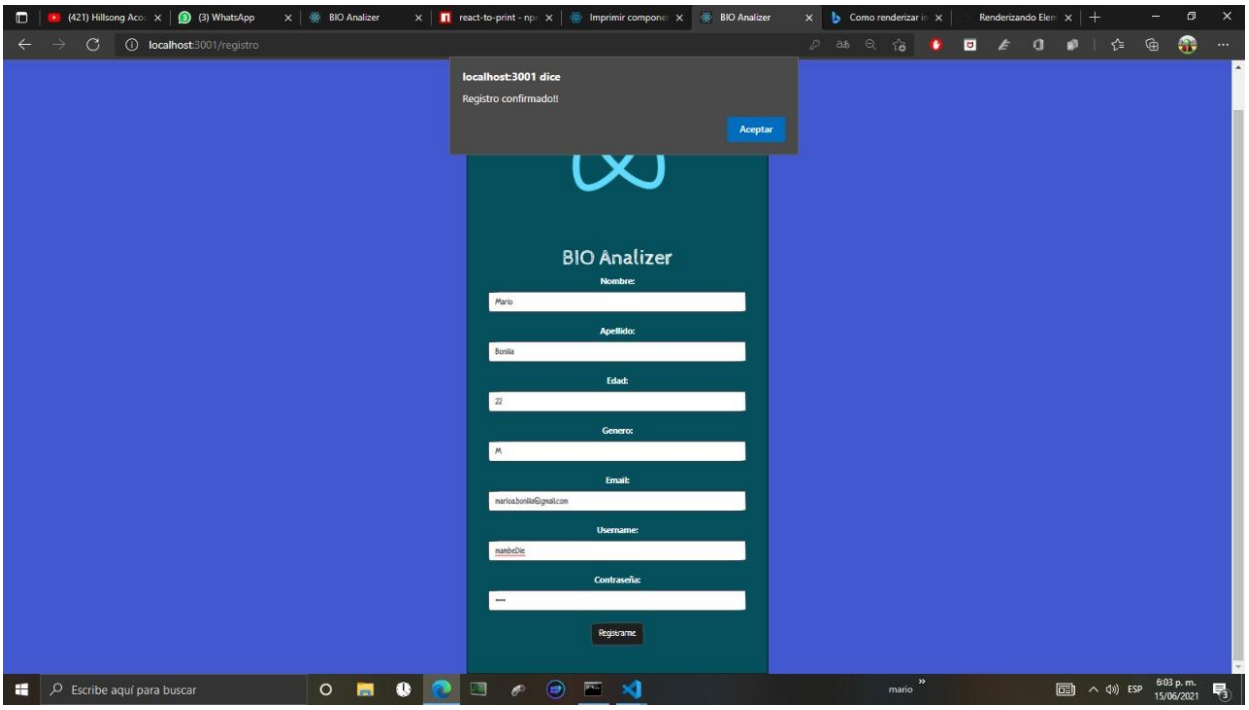
Interfaz gráfica.

Del lado del navegador se usa una librería del lenguaje JavaScript, Reactjs creada y soportada por la compañía Facebook. Crea UIs interactivas. Se puede diseñar vistas simples para cada estado de una aplicación.

Login

Figura 16.

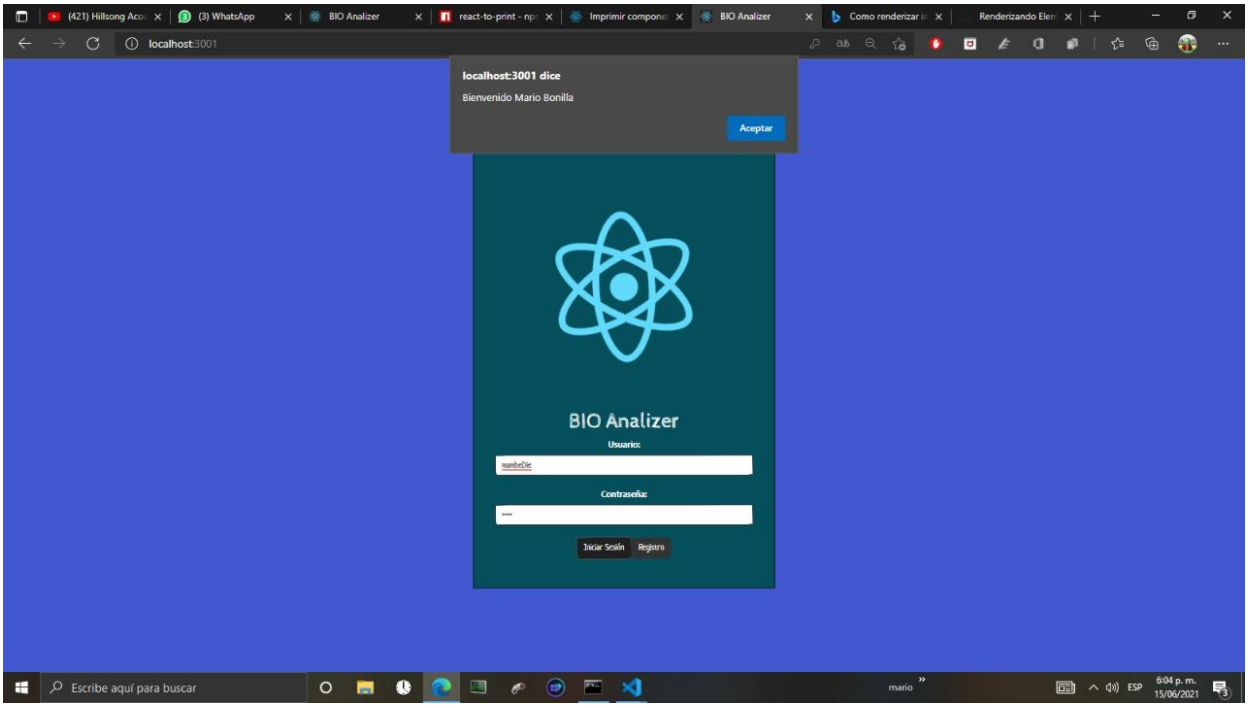
Login Registro



Nota: Fuente: (Mario Bonilla, Registro)

Figura 17.

Login inicio de sesión

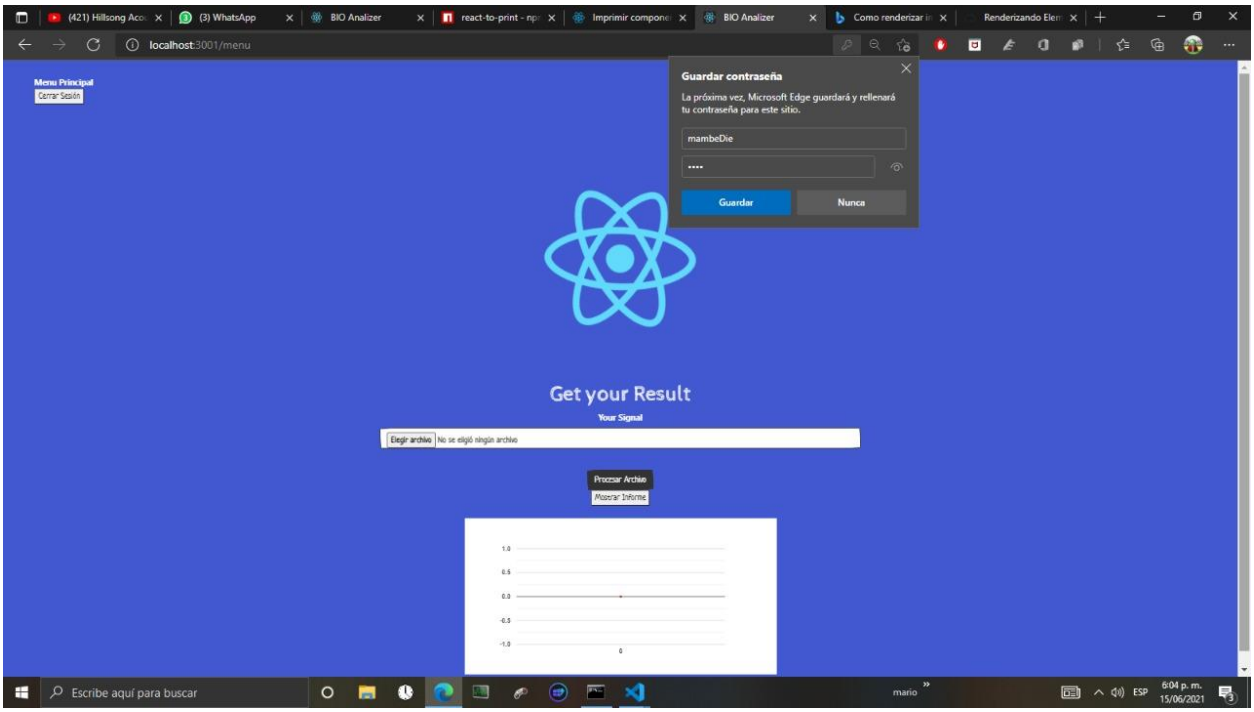


Nota: Fuente: (Mario Bonilla, Inicio de sesión)

Menú principal

Figura 18.

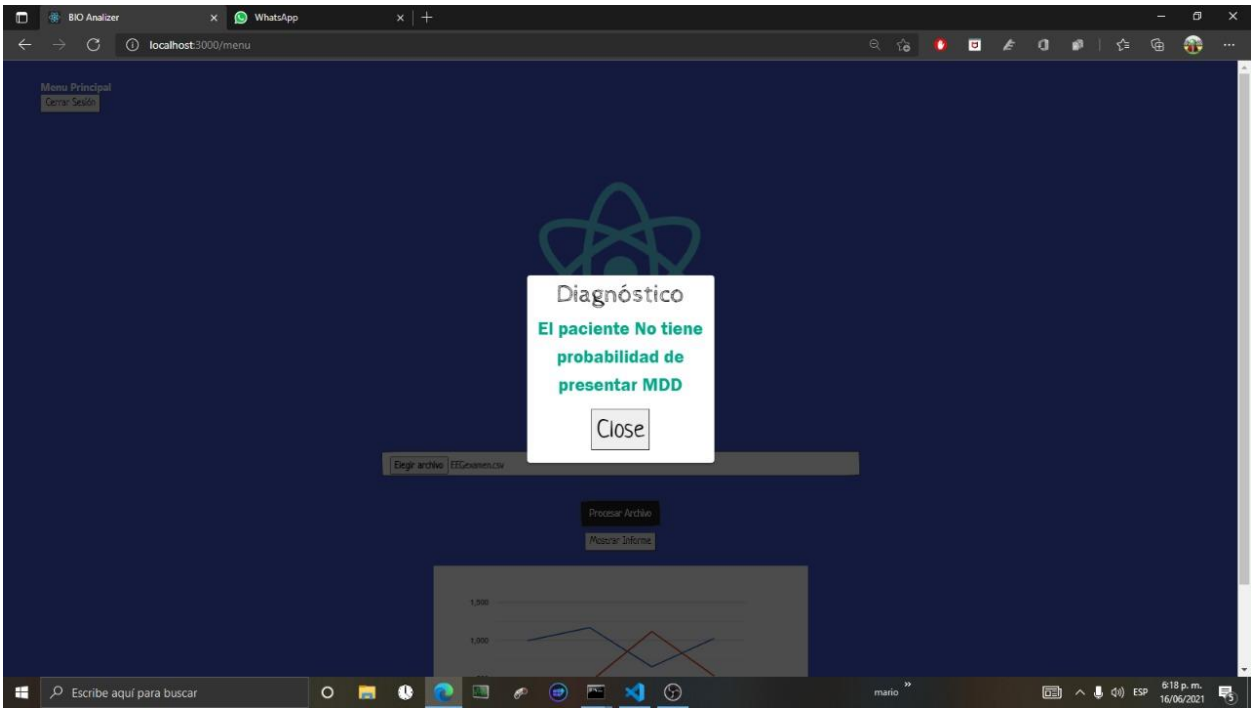
Menu principal menú



Nota: Fuente: (Mario Bonilla, Menu principal)

Figura 19.

Archivo procesado y cargado

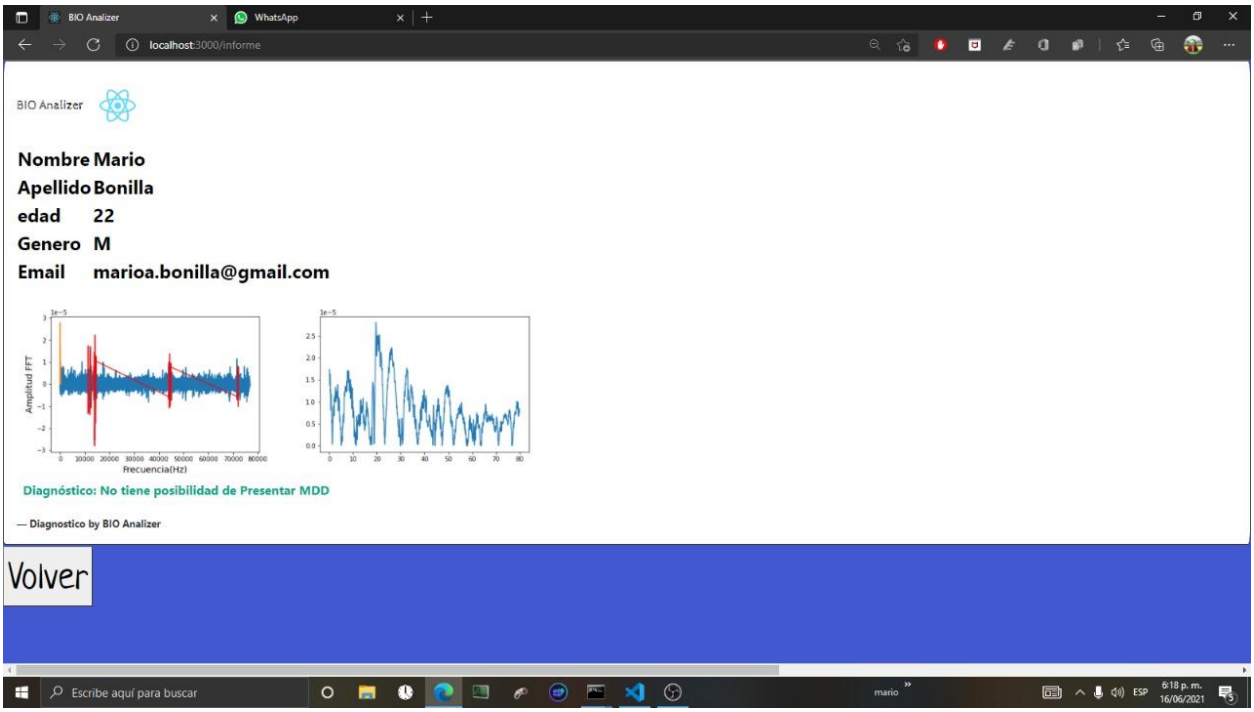


Nota: Fuente: (Mario Bonilla, Archivo cargado y procesado)

Informe

Figura 20.

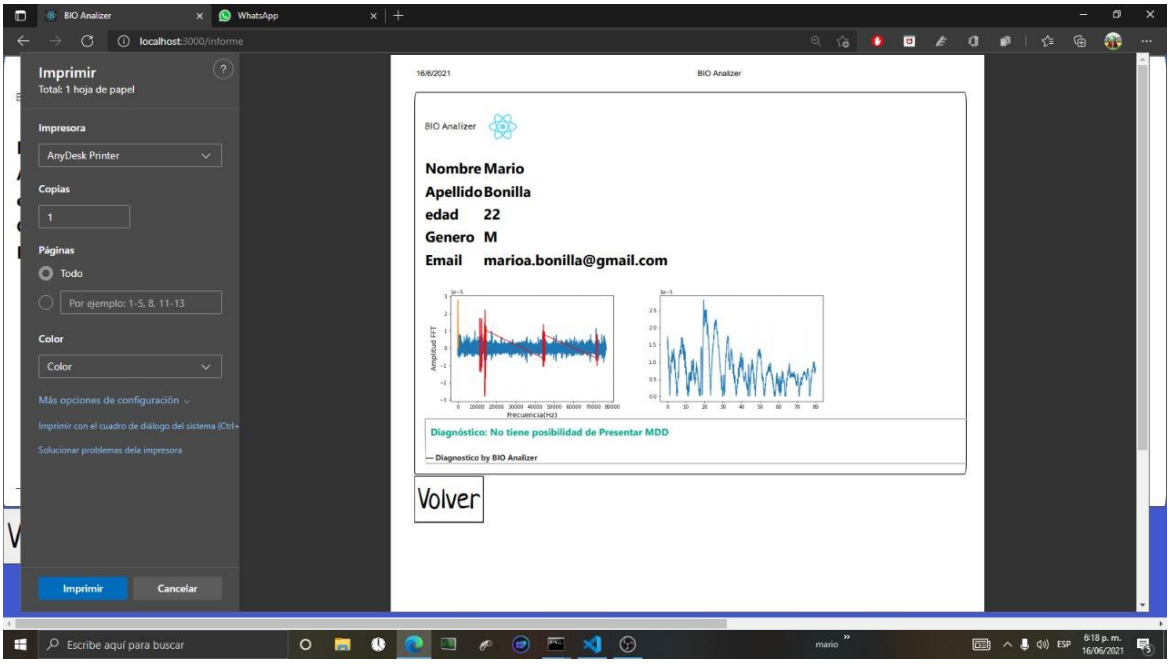
Informe generado



Nota: Fuente: (Mario Bonilla, Informe generado)

Figura 21.

Impresión informe



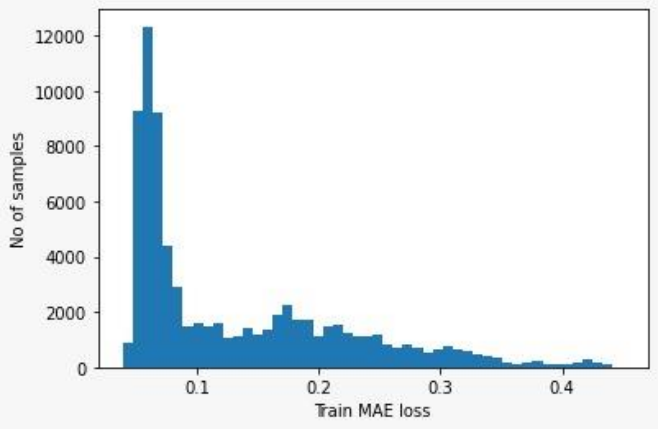
Nota: Fuente: (Mario Bonilla, Impresión de informe)

Validación de la Herramienta

En estadística, el error absoluto medio es una medida de la diferencia entre dos variables continuas. Considerando dos series de datos (unos calculados y otros observados) relativos a un mismo fenómeno, el error absoluto medio sirve para cuantificar la precisión de una técnica de predicción comparando por ejemplo los valores predichos frente a los observados, el tiempo real frente al tiempo previsto, o una técnica de medición frente a otra técnica alternativa de medición.

Figura 22.

Perdida vs Entrenamiento

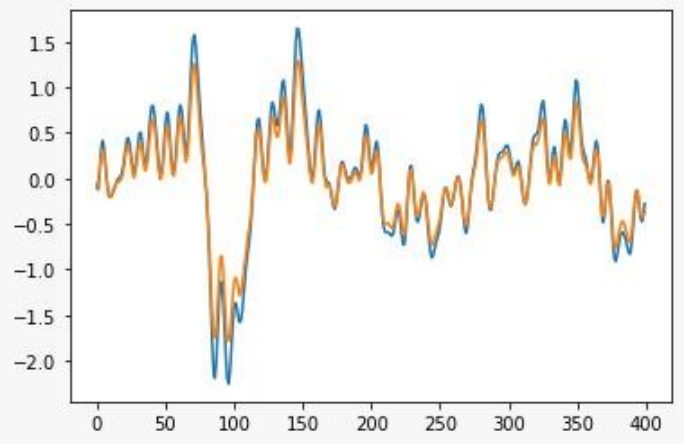


Nota: Elaboración propia

Se detectan las anomalías calculando si la pérdida de reconstrucción es mayor que un umbral fijo. Se calcula el error medio promedio para ejemplos MDD del conjunto de entrenamiento, luego clasificará los ejemplos futuros como anómalos si el error de reconstrucción es mayor que una desviación estándar del conjunto de entrenamiento.

Figura 23.

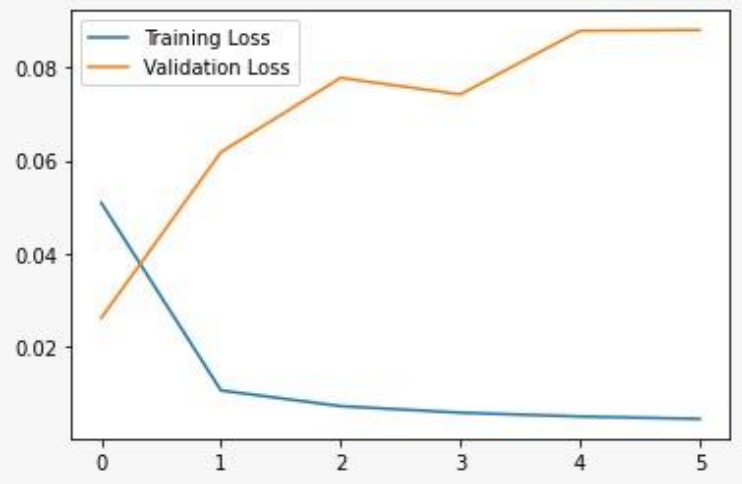
Ejemplo de cómo el algoritmo reconstruye la señal



Nota: Elaboración propia

Figura 24.

Pérdida de entrenamiento vs pérdida de validación.



Nota: Elaboración propia generada por matplotlib (Sebastian Nevado)

Capítulo 4

Conclusiones

De manera óptima se logró caracterizar las señales correspondientes y crear un dataset que sirvió como pibot para la creación del sistema de inteligencia artificial que funciona como eje del software clasificador. La red neuronal autoencoder muestra resultados satisfactorios desde la creación de su modelo hasta su uso práctico en la detección de anomalías para señales MDD por ende se puede afirmar que este tipo de redes neuronales son más que satisfactorias a la hora de tratar con los distintos de problemas que se le plantean y una gran utilidad al momento de usarse con señales y La interfaz gráfica es sencilla y escalable a través de reactjs. Y una visualización de resultados en formato pdf para que pueda ser agregado al registro clínico del paciente.

Capítulo 5

Recomendaciones

Backend

Implementar Django rest framework para una arquitectura más completa permite el uso de más herramientas para el desarrollo de software del lado del servidor, así como una singularidad de distintas funciones una orm para integración de bases de datos SQL y NoSQL.

Aplicar LSTM a las señales clasificadas para predicción de uso de inhibidores de serotonina en los pacientes, para predecir el comportamiento de los pacientes a estos fármacos.

Utilizar el lenguaje Golang para la programación asíncrona y correr varios procesos de manera simultánea, además de que el lenguaje es compilado directamente por ende es más rápido y también puede usar TensorFlow.

Frontend

Para la ventana emergente se pudo hacer más bordeado con otros colores y con esto podríamos utilizar SweetAlert2.

Se pudo hacer un login para organizar los usuarios que entran y que ellos guardaran sus resultados, pero preferimos que no, ya que la aplicación se construyó para detectar esas patologías y no tener un Google drive local por así decirlo.

Bibliografía

Guerrero, F. N., Haberman, M. A., & Spinelli, E. M. (2014). Sistema multicanal para adquisición de biopotenciales.

Es.wikipedia.org. 2021. Visual Studio Code - Wikipedia, la enciclopedia libre. [online] Recuperado de: <https://es.wikipedia.org/wiki/Visual_Studio_Code> [Accessed 17 May 2021].

Chart.js. (n.d.). Retrieved June 14, 2021, from Chartjs.org website: https://www.chartjs.org/

Node.js. (n.d.). Retrieved June 14, 2021, from Nodejs.org website: https://nodejs.org/en/

Welcome to python.Org. (n.d.). Retrieved June 14, 2021, from Python.org website: https://www.python.org/

JavaScript.com. (n.d.). Retrieved June 14, 2021, from Javascript.com website: https://www.javascript.com/

React. (n.d.). Retrieved June 14, 2021, from Reactjs.org website: https://reactjs.org/

Welcome to flask — flask documentation (2.0.X). (n.d.). Retrieved June 14, 2021, from Palletsprojects.com website: https://flask.palletsprojects.com/en/2.0.x/

Pelletier, J. (2015, November 1). The front-end spectrum. Retrieved June 30, 2021, from Medium website: https://medium.com/@withinsight1/the-front-end-spectrum-c0f30998c9f0

Es.wikipedia.org. 2021. npm - Wikipedia, la enciclopedia libre. [online] Recuperado de: <https://es.wikipedia.org/wiki/Npm> [Accessed 17 May 2021].

Es.wikipedia.org. 2021. HTML - Wikipedia, la enciclopedia libre. [online] Recuperado de: <https://es.wikipedia.org/wiki/HTML> [Accessed 17 May 2021].

Depresión. (n.d.). Retrieved June 29, 2021, from Who.int website: https://www.who.int/es/news-room/fact-sheets/detail/depression

Es.wikipedia.org. 2021. Hoja de estilos en cascada - Wikipedia, la enciclopedia libre. [online] Recuperado de: <https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada> [Accessed 17 May 2021].

Express - Node.js web application framework. (n.d.). Retrieved June 17, 2021, from Expressjs.com website: http://expressjs.com/

SQLite Home Page. (n.d.). Retrieved June 17, 2021, from Sqlite.org website: https://sqlite.org/index.html

Máquinas de Vector Soporte (SVM) con Python Cienciadedatos.net. Recuperado de: https://www.cienciadedatos.net/documentos/py24-svm-python.html (Accessed: May 17, 2021).

"La Transformada Rápida de Fourier (FFT)", [internet], Recuperado de http://www.ehu.es/Procesadodesenales/tema7/ty3.html,

Wikipedia, *La enciclopedia libre*, [internet], Recuperado de http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm,

Wikipedia, *La enciclopedia libre*, [internet], Recuperado de http://en.wikipedia.org/wiki/Fast_Fourier_transform#Algorithms,

http://mathworld.wolfram.com/FastFourierTransform.html,

Facundo, M. and Teresa, T. (no date) Funciones ejecutivas y trastornos del lóbulo frontal, Core.ac.uk. Recuperado de : https://core.ac.uk/download/pdf/95627971.pdf (Accessed: May 17, 2021).

Mumtaz, W., Ali, S. S. A., Yasin, M. A. M., & Malik, A. S. (2018). A machine learning framework involving EEG-based functional connectivity to diagnose major depressive disorder (MDD). Medical & biological engineering & computing, 56(2), 233-246.

Wikipedia contributors (no date) *Keras*, *Wikipedia, The Free Encyclopedia*. Recuperado de: https://es.wikipedia.org/w/index.php?title=Keras&oldid=128778152 (Accessed: May 27, 2021).

Understanding LSTM Networks (no date) *Github.io*. Recuperado de: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Accessed: May 27, 2021).

Los Cuidadores Pueden Cumplir Una Función Importante a la Hora, de A. a. su S. Q. Q. P. T. D. M. (no date) *Comprender qué es el trastorno depresivo mayor*, *Psychu.org*. Recuperado de: https://psychu.org/wp-content/uploads/2016/08/MRC2.UNB_.X.00081_MDD_SPA_patientguideforcaregivers_june-2016.pdf (Accessed: May 27, 2021).

American Medical Network, Inc. Consultado el 15 de enero de 2011.

Jordan, J. (2018, marzo 19). Introduction to autoencoders. Jeremyjordan.me; Jeremy Jordan. Recuperado de: https://www.jeremyjordan.me/autoencoders/

Scientific computing tools for Python — SciPy.org. (s/f). Scipy.org. Recuperado el 28 de julio de 2021, Recuperado de: https://www.scipy.org/about.html

Cookbook/Matplotlib - SciPy.org. (s/f). Archive.org. Recuperado el 28 de julio de 2021, Recuperado de: https://web.archive.org/web/20061202072022/http://www.scipy.org/Cookbook/Matplotlib

Using JSON with Yahoo! Web Services - YDN. (s/f). Archive.org. Recuperado el 28 de julio de 2021, Recuperado de: https://web.archive.org/web/20100106010113/http://developer.yahoo.com/common/json.html

Anexos

Estos incluyen el instrumento mediante el cual se hará la recolección de información con su respectiva codificación para guiar la tabulación de datos y el instructivo para diligenciarlo. También se incluye cualquier otra información necesaria para complementar el proyecto, incluyendo el cronograma, el presupuesto y los recursos necesarios (humanos, financiero sy técnicos). Los apéndices y anexos deben estar citados en el cuerpo del trabajo.

Anexo 1

Manual de Usuario

DESCRIPCION DE LA APLICACIÓN WEB

Esta aplicación fue realizada en dos partes, bien llamadas back-end y front-end donde en el back se realizó el software y en el front la interfaz amigable con el usuario. Esta aplicación en su totalidad detecta patologías en señales EEG.

Considerando lo anterior el trastorno de depresión mayor que se caracteriza por depresión persistente o pérdida de interés en las actividades, lo que puede causar dificultades significativas en la vida cotidiana. nuestro software pretende dar apoyo a los estudios de psiquiatría mediante el análisis de señales EEG.

¿Cómo podemos aprovechar el análisis de los biopotenciales que se reflejan en un electroencefalograma como apoyo para el diagnóstico de Desorden de Depresión Mayor (MDD)? La detección de anomalías correspondientes al trastorno de depresión mayor catalogado como trastorno de salud mental, que contribuya a los estudios psiquiátricos desarrollada para la comunidad psiquiátrica y personas con MDD en su actividad cerebral.

FUNCIONES BÁSICAS

Importante

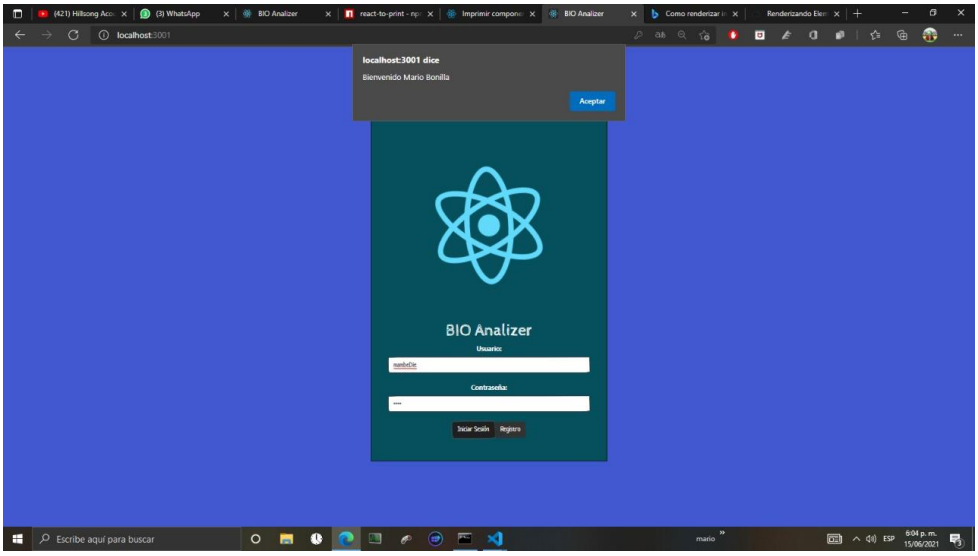
Esta aplicación ha sido desarrollada para trabajar en cualquier ambiente ya que es web, por lo cual los usuarios del sistema deben estar familiarizados con este ambiente de trabajo y conocer aspectos básicos como:

· Uso del Mouse
· Manejo de ventanas (abrir, cerrar, minimizar, maximizar, moverlas con el mouse, etc.)
· Uso de botones
. Tener un examen EEG para procesar y dar su diagnóstico.

A continuación, se detallan los diferentes pasos para poder interactuar con BIO Analizer y obtener los datos requeridos:

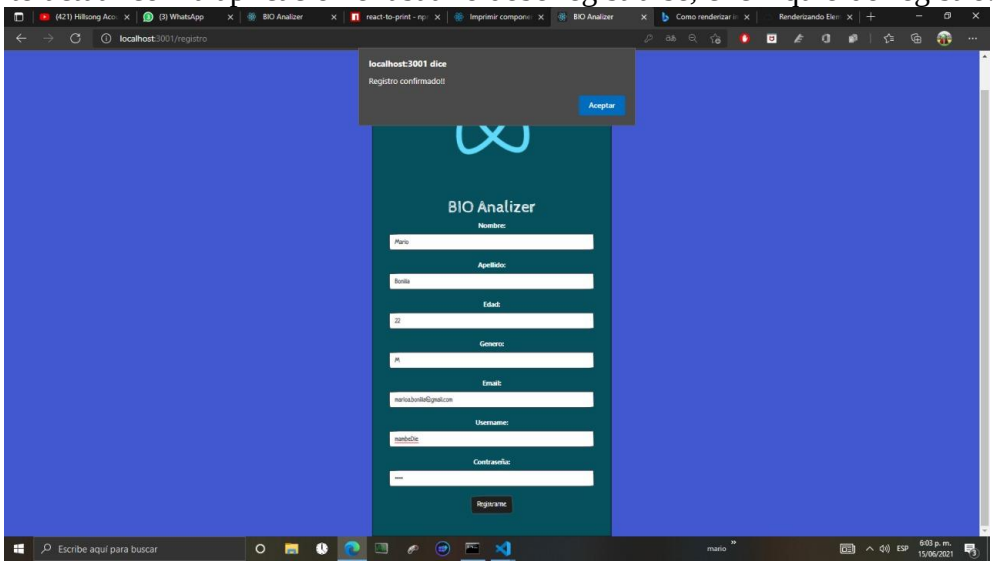
Paso 1:
INGRESO A LA APLICACIÓN

Como primera instancia para ejecutar la aplicación, se muestra el icono con el login para registrar e iniciar sesión.

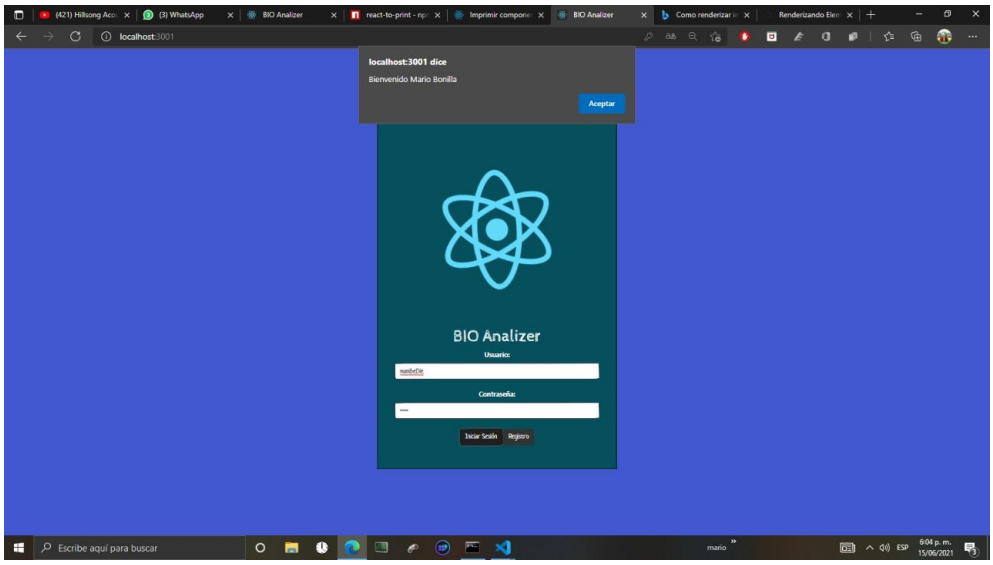


Paso 2:
LOGIN

Para interactuar con la aplicación el usuario debe registrarse, clic izquierdo registro.



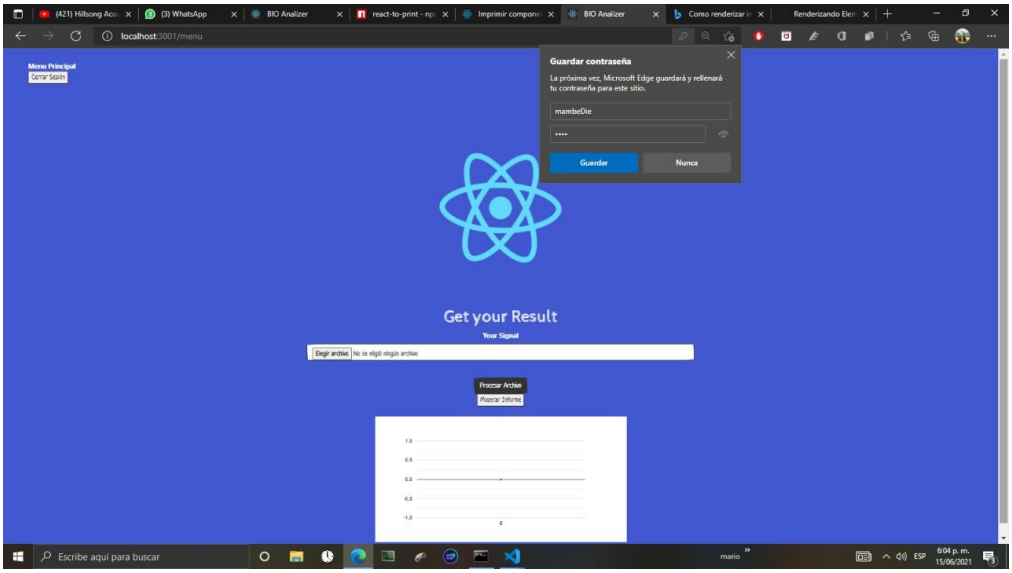
Al momento de finalizar su registro la app le confirma su registro y lo redirige a inicio de sesión, si ya está registrado puede dar clic izquierdo en una flecha que esta más abajito.



- USUARIO (El que usted Registró)
- CONTRASEÑA (La que usted Registró)

En esta ventana se deben ingresar los datos, Luego de llenar los campos requeridos haga un clic con el botón izquierdo del mouse sobre el botón iniciar sesión automáticamente la app le da la bienvenida para confirmar la operación y proceder a ingresar a la aplicación BIO Analizer.

Paso 3:
MENU PRINCIPAL

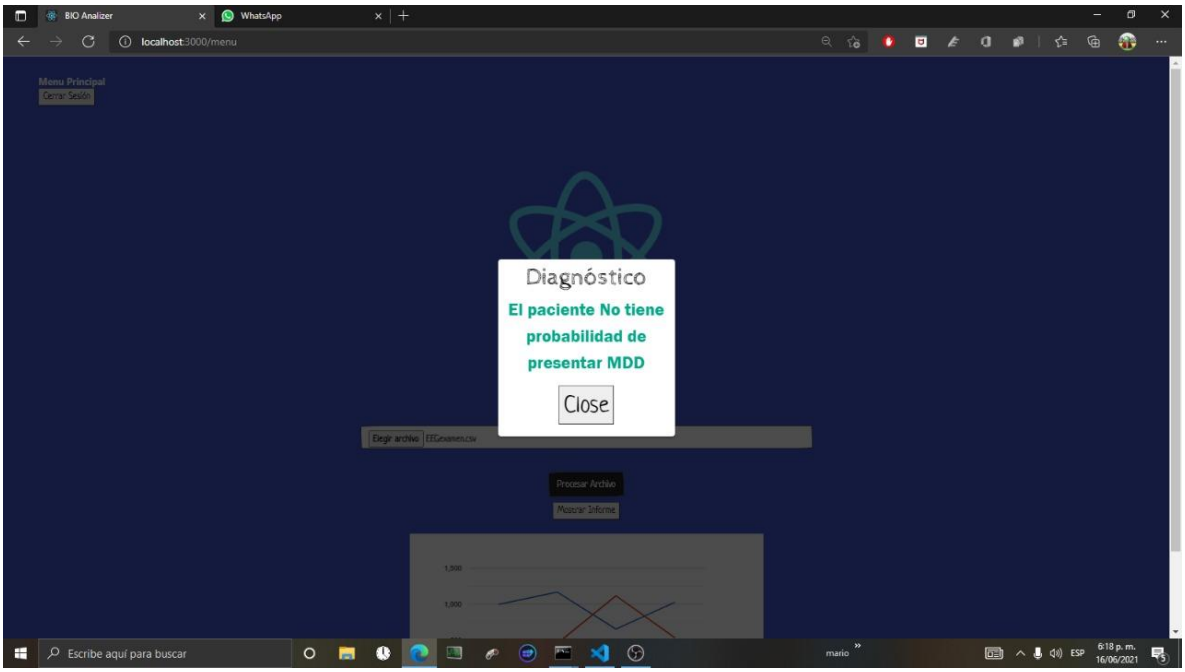


Esta es la ventana principal de la aplicación BIO Analizer el cual tiene como función procesar y dar diagnostico al usuario logueado según su examen cargado EEG.

Se puede ver el Icono de la app con un plano grafico para poder observar parte de la señal que se procesó.

Debajo de donde se carga el examen están dos botones, el negro para procesar y el blanco para mostrar el informe luego de que el usuario logueado procese su examen EEG.

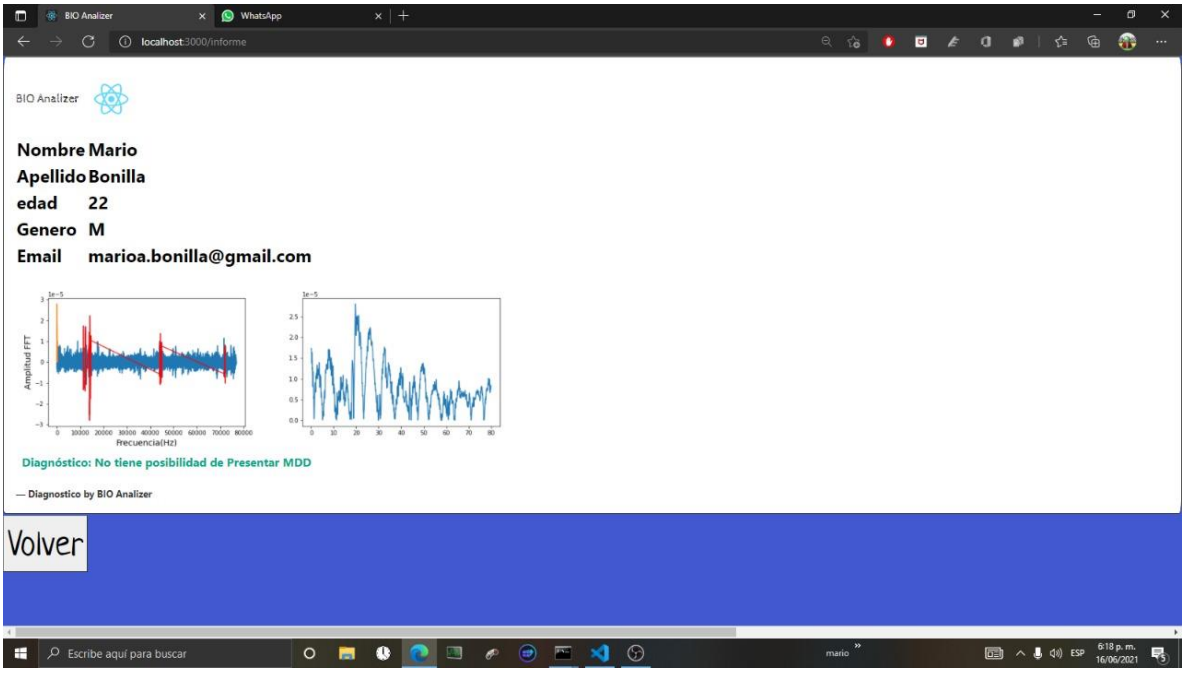
Paso 4:
ARCHIVO PROCESADO Y CARGADO



Al procesar el archivo dando clic izquierdo al botón ‘procesar archivo’ se genera un diagnostico con la probabilidad del paciente de tener la patología MDD. Y se puede observar una parte de la señal procesada en el plano grafico dicho antes.

Para dejar de ver la ventana dar clic izquierdo en ‘Close’.

Paso 5:
INFORME GENERADO

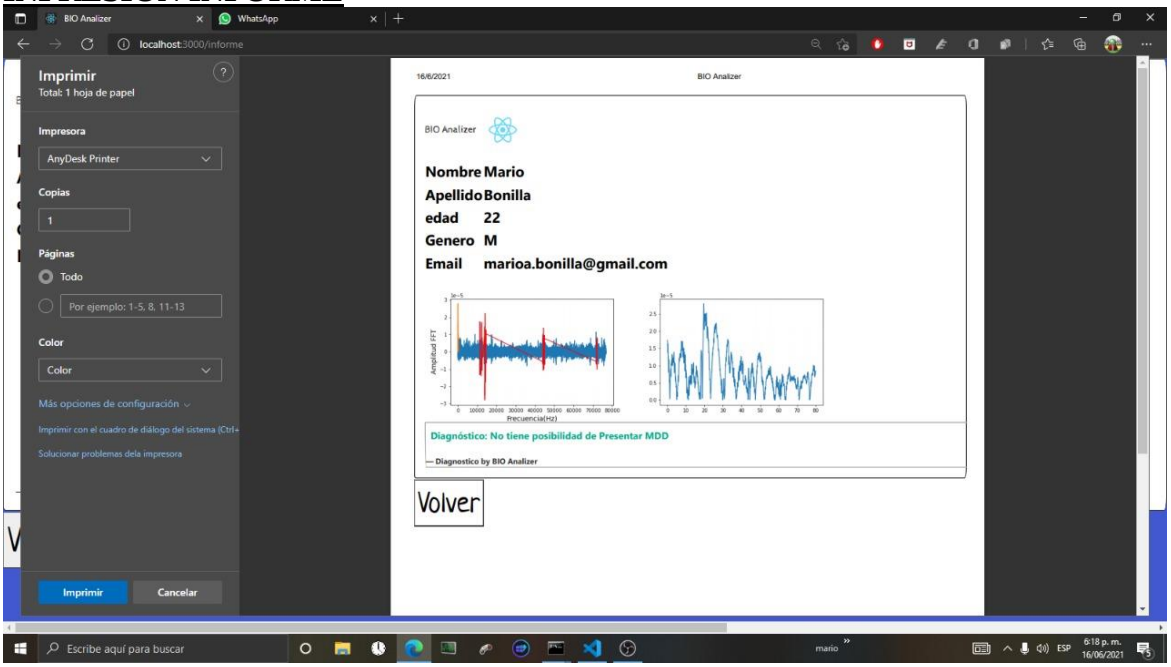


Luego de haberse generado la ventana emergente con el diagnostico, si el usuario quiere ver de manera detallada su diagnóstico se genera un informe con los datos del usuario logueado graficas en frecuencia (plano grafico en azul clarito) y las anomalías que se presentaron (plano grafico con línea rojas) además el diagnostico que había salido antes en la ventana emergente debajo de las gráficas.

Para volver y generar otro informe con otro examen dar clic izquierdo en volver.

Paso 5:

INPRESION INFORME



Si el usuario quiere imprimir el informe el navegador tiene su función de impresión donde puede generar su informe diagnosticado limpio y con el tamaño que quiera. Además, se puede observar el día que genera la impresión del informe.

Para esto al tener el informe debe presionar clic derecho – imprimir.

Frontend

Anexo 2

Algoritmo de Login

```
import React, { Component } from 'react';
import logo from '../logo.svg';
import '../App.css';
import axios from 'axios';
import md5 from 'md5';
import Cookies from 'universal-cookie';

const baseUrl="http://localhost:3002/user";
const cookies = new Cookies();

class Login extends Component {
    state={
        form:{
            username: '',
            password: ''
        }
    }

    handleChange=async e=>{
        await this.setState({
            form:{
                ...this.state.form,
                [e.target.name]: e.target.value
            }
        });
    }

    iniciarSesion=async()=>{
        await axios.get(baseUrl +'/'+this.state.form.username+'/'+md5(this.state.
form.password, { headers: {'x-apikey': 'API_KEY'} }))
        .then(response=>{
            console.log(response)
            return response.data;
```

```
        })
        .then (response=>{ if(response.length>0){
            console.log(response)
            var respuesta=response[0];
            cookies.set('id', respuesta.id, {path: "/"});
            cookies.set('apellido', respuesta.apellido, {path: "/"});
            cookies.set('edad', respuesta.edad, {path: "/"});
            cookies.set('email', respuesta.email, {path: "/"});
            cookies.set('genero', respuesta.genero, {path: "/"});
            cookies.set('nombre', respuesta.nombre, {path: "/"});
            cookies.set('username', respuesta.username, {path: "/"});
            cookies.set('password', respuesta.password, {path: "/"});

            alert(`Bienvenido ${respuesta.nombre} ${respuesta.apellido}`);
            window.location.href="./menu";
        }else{
            alert('El usuario o la contraseña no son correctos');
        }
    })
    .catch(error=>{
        console.log(error);
    })

}



    componentDidMount() {
        if(cookies.get('username')){
            window.location.href="./menu";
        }
    }

    goRegistro(){
        window.location.href = "./registro";
    }
    render() {
        return (
    <div className="containerPrincipal">
        <div className="containerSecundario">
        <img src={logo} className="App-
logo" alt="logo" style={{ width: '500px' }} />
        <h1>BIO Analizer</h1>
          <div className="form-group">
            <label color='black'>Usuario: </label>
```

```
            <br />
            <input
              type="text"
              className="form-control"
              name="username"
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Contraseña: </label>
            <br />
            <input
              type="password"
              className="form-control"
              name="password"
              onChange={this.handleChange}
            />
            <br />
            <button className="btn btn-
primary" onClick={()=> this.iniciarSesion()}>Iniciar Sesión</button>
            <button className="btn btn-
primary" onClick={()=> this.goRegistro()}>Registro</button>

          </div>
        </div>
      </div>
        );
    }
}

export default Login;
```

Anexo 3

Algoritmo de Registro

```
import React, { Component } from 'react';
import logo from '../logo.svg';
import '../App.css';
import axios from 'axios';
import md5 from 'md5';
import Cookies from 'universal-cookie';
```

```
const baseUrl="http://localhost:3002/newuser";
const cookies = new Cookies();

class Registro extends Component {
    state={
        form:{
            nombre: '',
            apellido: '',
            edad: '',
            genero: '',
            email: '',
            username: '',
            password: ''
        }
    }

    handleChange=async e=>{
        await this.setState({
            form:{
                ...this.state.form,
                [e.target.name]: e.target.value
            }
        });
    }

    registrame=async()=>{
        await axios.get(baseUrl +'/'+this.state.form.nombre+'/'+this.state.form.a
pellido+'/'+this.state.form.edad+'/'+this.state.form.genero+'/'+this.state.form.e
mail+'/'+this.state.form.username+'/'+md5(this.state.form.password), { headers: {
'x-apikey': 'API_KEY'} })
        .then(response=>{
            console.log(response)
            return response.data;
        })
        .then (response=>{ if(!response[0].error){
            console.log(response[0])
            alert('Registro confirmado!!');
            window.location.href="./";
        }else{
            console.log(response[0])
            alert('Error al registrar usuario, validar datos');
        }
    })
    .catch(error=>{
        console.log(error);
```

```
    })

}

componentDidMount() {
    if(cookies.get('username')){
        //window.location.href="./menu";
    }
}
 volverLogin=()=>{
  window.location.href="./";
 }
    render() {
        return (
    <div className="containerPrincipal">
        <div className="containerSecundario">
        <img src={logo} className="App-
logo" alt="logo" style={{ width: '500px' }} />
        <h1>BIO Analizer</h1>
          <div className="form-group">

            <label color='black'>Nombre: </label>
            <br />
            <input
              type="text"
              className="form-control"
              name="nombre"
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Apellido: </label>
            <br />
            <input
              type="text"
              className="form-control"
              name="apellido"
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Edad: </label>
            <br />
            <input
              type="text"
              className="form-control"
              name="edad"
```

```
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Genero: </label>
            <br />
            <input
              type="text"
              className="form-control"
              name="genero"
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Email: </label>
            <br />
            <input
              type="text"
              className="form-control"
              name="email"
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Username: </label>
            <br />
            <input
              type="text"
              className="form-control"
              name="username"
              onChange={this.handleChange}
            />
            <br />
            <label color='black'>Contraseña: </label>
            <br />
            <input
              type="password"
              className="form-control"
              name="password"
              onChange={this.handleChange}
            />
            <br/>
            <button className="btn btn-
primary" onClick={()=> this.registrame()}>Registrarme</button>

          </div>

        </div>
```

```
        <div>
        <button onClick={()=>this.volverLogin()}>Volver</button></div>
      </div>


        );
    }
}

export default Registro;
```

Anexo 4

Algoritmo de Menu Principal donde traemos el componente de FileUpload que contiene los requerimientos del archivo cargado, procesado y diagnosticado con su informe

```
import React, { Component, useRef} from 'react';
import Cookies from 'universal-cookie';
import logo from '../logo.svg';
import  '../App.css';
import { Chart }from "react-google-charts";
import  FileUpload from '../FileUpload'
import  Informe from '../Informe'

import ReactToPrint from 'react-to-print';




const cookies = new Cookies();

class Menu extends Component {
    cerrarSesion=()=>{
        cookies.remove('id', {path: "/"});
        cookies.remove('apellido', {path: "/"});
        cookies.remove('nombre', {path: "/"});
        cookies.remove('username', {path: "/"});
        cookies.remove('edad', {path: "/"});
        cookies.remove('genero', {path: "/"});
        cookies.remove('email', {path: "/"});

        cookies.remove('password', {path: "/"});
```

```
        window.location.href='./';
    }
    informe=()=>{
      window.location.href="./informe";
    }
    componentDidMount() {
        if(!cookies.get('username')){
            window.location.href="./";
        }
    }
    constructor() {
        super()
        this.state = {
          mydata: [['class', 'AF3 delta std', 'AF3 delta m','AF3 theta std','AF3
theta m','AF3 alpha std','AF3 alpha m','AF3 beta std','AF3 beta m','F7 delta std'
],
          ['0', 0, 0, 0, 0, 0, 0, 0, 0, 0],]
        }
      }

      handleData(data) {
        this.setState({
          mydata: data
        })
      }


    render() {
        console.log('id: '+ cookies.get('id'));
        console.log('apellido: '+cookies.get('apellido'));
        console.log('nombre: '+cookies.get('nombre'));
        console.log('username: '+cookies.get('username'));
        console.log('edad: '+cookies.get('edad'));
        console.log('genero: '+cookies.get('genero'));
        console.log('email: '+cookies.get('email'));
        console.log('password: '+cookies.get('password'));



        return (
            <div className="Menu">
            <div className="continer">

            <div className='menuprincipal'>
                Menu Principal
```

```
                <br />
                <button onClick={()=>this.cerrarSesion()}>Cerrar Sesión</button>

            </div>
            <div className= 'body'>
          <img src={logo} className="App-
logo" alt="logo" style={{ width: '500px' }} />
          <h1>Get your Result</h1>
          <FileUpload onSetdata={this.handleData.bind(this)} />
          <button onClick={()=>this.informe()}>Mostrar Informe</button>


          <div style={{ display: 'flex' }}>
            <div className="mt-4" style={{ margin: '0 auto' }}>
              <Chart
                width={600}
                height={300}
                chartType="LineChart"
                loader={<div>Loading Chart</div>}
                data={this.state.mydata}
                options={{
                  intervals: { style: 'sticks' },
                  legend: 'none',
                }}
              />
            </div>
          </div>
        </div>
      </div>

        );
    }
}

export default Menu;
```

Anexo 5

Algoritmo al consumirse la API

```
import pandas as pd
import numpy as np
```

```
import scipy.signal as signal
import matplotlib.pyplot as plt

import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
import scipy.fftpack as fourier
from keras.models import load_model

import json

with open("try.txt", "r") as f:
    lines = f.readlines()

f = lines[0]

df = pd.read_csv("MDD1.csv")

fc = 80  # Cut-off frequency of the butterworth filter
w = fc / (360 / 2) # Normalize the frequency
sos1, b, a = signal.butter(5, w, 'low', output='sos')
filtered_data_butterworth = signal.filtfilt(b, a, df["# EEG Fp1-LE"])
filtered = signal.sosfilt(sos1, df["# EEG Fp1-LE"])
#MDD filtrada
sick = filtered

df2 = pd.read_csv(f)

fc = 80  # Cut-off frequency of the butterworth filter
w = fc / (360 / 2) # Normalize the frequency
sos2, b, a = signal.butter(5, w, 'low', output='sos')
filtered_data_butterworth = signal.filtfilt(b, a, df2["# EEG Fp1-LE"])
#señal de consulta filtrada
filtered2 = signal.sosfilt(sos2, df2["# EEG Fp1-LE"])
patient = filtered2

df_small_noise = pd.DataFrame(sick)

df_daily_jumpsup = pd.DataFrame(patient)

training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
TIME_STEPS = 400

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)

x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)

model = load_model('anomally_sick.h5')

df_test_value = (df_daily_jumpsup - training_mean) / training_std
fig, ax = plt.subplots()
df_test_value.plot(legend=False, ax=ax)
plt.show()

# Create sequences from test values.
x_test = create_sequences(df_test_value.values)
print("Test input shape: ", x_test.shape)

# Get test MAE loss.
x_test_pred = model.predict(x_test)
test_mae_loss = np.mean(np.abs(x_test_pred - x_test), axis=1)
test_mae_loss = test_mae_loss.reshape((-1))

plt.hist(test_mae_loss, bins=50)
plt.xlabel("test MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
```

```
print("Reconstruction error threshold: ", threshold)

# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))
# data i is an anomaly if samples [(i - timesteps + 1) to (i)] are anomalies
anomalous_data_indices = []
for data_idx in range(TIME_STEPS - 1, len(df_test_value) - TIME_STEPS + 1):
    if np.all(anomalies[data_idx - TIME_STEPS + 1 : data_idx]):
        anomalous_data_indices.append(data_idx)

df_subset = df_daily_jumpsup.iloc[anomalous_data_indices]
fig, ax = plt.subplots()
df_daily_jumpsup.plot(legend=False, ax=ax)
df_subset.plot(legend=False, ax=ax, color="r")
plt.savefig("anomalies.jpg")#Aqui lo ubicas en la ruta que quieras

#Respuesta en frecuencia de las anomalias
Ts = 0.0125 #Definimos un tiempo y frecuencia de muestreo
Fs = 1/Ts
gk = fourier.fft(df_subset) #Calculamos la FFT
M_gk = abs(gk)      #Calculamos la magnitud de la FFT

F= Fs*np.arange(0, len(df_subset))/len(df_subset)

plt.plot(F, M_gk)
plt.xlabel('Frecuencia(Hz)', fontsize='14')
plt.ylabel('Amplitud FFT', fontsize='14')
plt.savefig('./apiconpython')#Aqui agregas laruta que quieras

if len(df_subset) != 0 or len(df_subset)< 2500:
    respuesta = {
        'Diagnostico':'No Tiene probabilidad de presentar MDD'
    }
else:
    respuesta = {
        'Diagnostico':'Tiene probabilidad de presentar MDD'
    }


cadena_json = json.dumps(respuesta)

#Escritura
```

```
with open('datos.json','w') as f:
    json.dump(respuesta, f)
```

Nodejs para consumirla

```
// Python
app.get('/execpy/:ruta', function(req, res){
    console.log('1')
    const fs = require('fs');
    fs.appendFile('try.txt','C:\apinodejs\python\H1.csv',(error)=>{
        if(error){
            throw error;
        }
            console.log('Ruta Enviada exitosamente');
    });
    const exec = require('child_process').exec;
    const scriptExecution = exec("python", ["./python/nop.ipynb"]);
    console.log(datos.json)

    // Manejo normal output
    scriptExecution.stdout.on('data', (data) => {
        console.log(data.toString());
        console.log('3')
    });
    res.send(req.params.ruta)
})

server.listen(3002,function(){
    console.log("Server listening on port: 3000");
})
```

```
C:\apinodejs>node index.js
Server listening on port: 3000
1
No Tiene probabilidad de presentar MDD
Ruta Enviada exitosamente
```

Anexo 6

Funciona tanto para la señal con MDD como la sana solo hay que cambiar el archivo de lectura

Backend

main.py > ...

```python
"""
Esta es la parte encargada de filtrar las señales:
        Nos apoyamos en pandas para la lectura del archivo(funciona para archivos csv,xlsx,txt...)
        Scipy para la creacion de filtros pasa baja para extraer las bandas de frecuencia
        que vamos a utilizar apoyados en el modulo scipy.signal para la creacion del filtro
        Matplotlib para la visualizacion de los datos obtenidos
"""

import pywt
import numpy as np

import scipy.signal as signal
import matplotlib.pyplot as plt

import pandas as pd

df = pd.read_csv("MDD1.csv")


fc = 80  # Cut-off frequency of the butterworth filter
w = fc / (360 / 2) # Normalize the frequency
sos1, b, a = signal.butter(5, w, 'low', output='sos')#Signal filtred
filtered_data_butterworth = signal.filtfilt(b, a, df["# EEG Fp1-LE"])
filtered = signal.sosfilt(sos1, df["# EEG Fp1-LE"])#Add to filtred variable the signal
plt.plot(df["# EEG Fp1-LE"])#Comparison of filtered signal vs original signal
plt.plot(filtered, label='Señal filtrada',color='black')
```

En esta parte creamos el modelo preparamos las señales de prueba y procedemos a crear unos parámetros que nos servirán para identificar la precisión de nuestro modelo y capturar las anomalias y la visualización

```python
df_small_noise = pd.DataFrame(sick)

df_daily_jumpsup = pd.DataFrame(health)

# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
TIME_STEPS = 400

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
#create model
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001), loss="mse")
model.summary()

#Train model
history = model.fit(
    x_train,
    x_train,
    epochs=7,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.legend()


# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)

# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()


df_test_value = (df_daily_jumpsup - training_mean) / training_std
fig, ax = plt.subplots()
df_test_value.plot(legend=False, ax=ax)
plt.show()

# Create sequences from test values.
x_test = create_sequences(df_test_value.values)
print("Test input shape: ", x_test.shape)

# Get test MAE loss.
x_test_pred = model.predict(x_test)
test_mae_loss = np.mean(np.abs(x_test_pred - x_test), axis=1)
test_mae_loss = test_mae_loss.reshape((-1))

plt.hist(test_mae_loss, bins=50)
plt.xlabel("test MAE loss")
plt.ylabel("No of samples")
plt.show()

# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))
```

```
# data i is an anomaly if samples [(i - timesteps + 1) to (i)] are anomalies
anomalous_data_indices = []
for data_idx in range(TIME_STEPS - 1, len(df_test_value) - TIME_STEPS + 1):
    if np.all(anomalies[data_idx - TIME_STEPS + 1 : data_idx]):
        anomalous_data_indices.append(data_idx)


df_subset = df_daily_jumpsup.iloc[anomalous_data_indices]
fig, ax = plt.subplots()
df_daily_jumpsup.plot(legend=False, ax=ax)
df_subset.plot(legend=False, ax=ax, color="r")

plt.plot(df_subset)#plot the anomalies

import scipy.fftpack as fourier

#observe its response frequently with fourier

Ts = 0.0125 #Definimos un tiempo y frecuencia de muestreo
Fs = 1/Ts
gk = fourier.fft(df_subset) #Calculamos la FFT
M_gk = abs(gk)      #Calculamos la magnitud de la FFT

F= Fs*np.arange(0, len(df_subset))/len(df_subset)

plt.plot(F, M_gk)
plt.xlabel('Frecuencia(Hz)', fontsize='14')
plt.ylabel('Amplitud FFT', fontsize='14')
plt.show()
```

Anexo 7

Resultados de pruebas para la clasificación de señales de las anomalías

https://www.mediafire.com/file/6cdq5c6onzbmifw/Anomalias-1.zip/file

Nota: Son Muchas imágenes por eso se enviaron en un enlace.