|  |  |
| --- | --- |
| Espacio de la Maestra Ma Guadalupe Ibarra – Otro sitio más de Docentes en  Línea :: UAA | **Universidad Autónoma de Aguascalientes** |
| **Centro de Ciencias Básicas** |
| **Departamento de Ciencias de la Computación** |

**Tesina**

**Implementación de una Interfaz Cerebro – Computadora por medio de la diadema Emotiv Epoc+ y la librería de Python Cykit utilizando Modelos de Aprendizaje Supervisado. Análisis comparativo entre una Máquina de Vectores de Soporte, el modelo Random Forest y el modelo de Bayes Ingenuo.**

**Presenta:**

Joel Alejandro Espinoza Sánchez

Para Obtener el Grado de Ingeniería en Computación Inteligente

**Directores:**

Dra. Aurora Torres Soto

Dra. María Dolores Torres Soto

**Comité Evaluador:**

Dr. Francisco Javier Álvarez Rodríguez

Dra. Eunice Esther Ponce de León Senti

Dra. Aurora Torres Soto

Aguascalientes, Ags., Mayo 2023.



Aguascalientes, Ags., XX de Mayo de 2022

A quien corresponda:

Por medio de la presente me permito informar que el alumno(a) **Joel Alejandro Espinoza Sánchez** de la carrera de Ingeniería en Computación Inteligente con **ID: 211800**, ha terminado satisfactoriamente su tesina titulada: **“Implementación de una Interfaz Cerebro – Computadora por Medio de la Diadema Emotiv Epoc+ y la Librería de Python Cykit Utilizando Modelos de Aprendizaje Supervisado. Análisis Comparativo entre una Máquina de Vectores de Soporte, el modelo Random Forest y el modelo de Bayes Ingenuo.”**, correspondiente a la materia de Seminario de Investigación II.

Para los fines que al interesado convengan.

**ATENTAMENTE**

|  |  |
| --- | --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Dra. Aurora Torres Soto**  **Codirector** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Dra. María Dolores Torres Soto**  **Codirector** |



Aguascalientes, Ags., XX de Mayo de 2022

A quien corresponda:

Por medio de la presente me permito informar que el alumno(a) **Joel Alejandro Espinoza Sánchez** de la carrera de Ingeniería en Computación Inteligente con **ID: 211800**, ha terminado satisfactoriamente su tesina titulada: **“Implementación de una Interfaz Cerebro – Computadora por Medio de la Diadema Emotiv Epoc+ y la Librería de Python Cykit Utilizando Modelos de Aprendizaje Supervisado. Análisis Comparativo entre una Máquina de Vectores de Soporte, el modelo Random Forest y el modelo de Bayes Ingenuo.”**, correspondiente a la materia de Seminario de Investigación II.

Para los fines que al interesado convengan.

**ATENTAMENTE**

|  |  |  |
| --- | --- | --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Dra. Aurora**  **Torres Soto**  **Miembro del Comité Tutoral** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Dra. Eunice Esther**  **Ponce de León Senti**  **Miembro del Comité Tutoral** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Dr. Francisco Javier**  **Álvarez Rodríguez**  **Miembro del Comité Tutoral** |

**Resumen**

Resumen de la investigación.

**Abstract**

Abstract of the investigation.

**Agradecimientos**

Sección de agradecimientos.

**Tabla de Contenido**

Tabla de Contenido 1

Índice de Figuras 1

Índice de Gráficas 1

Índice de Tablas 1

Índice de Ecuaciones 1

1. Introducción 1

2. Planteamiento del Problema 1

3. Justificación 1

4. Objetivos 1

5. Hipótesis 1

6. Pregunta de Investigación 1

7. Marco Teórico 1

7.1. Soy un cerebro, Watson. El resto de mí es un mero apéndice 1

Introducción a la neuroanatomía

7.1.1. El cerebro y el sistema nervioso 1

7.1.2. El cerebro a nivel macroestructura 1

7.1.3. El cerebro a nivel microestructura 1

7.1.3.1. Las neuronas 1

7.1.3.2. Morfología de las neuronas 1

7.1.3.3. Funciones de las neuronas 1

7.1.3.4. Interacción entre neuronas 1

7.2. Todo hombre puede ser, si se lo propone, escultor de su propio cerebro 1

Electroencefalografía y estudios del cerebro

7.3. Toda tecnología suficientemente avanzada es indistinguible de la magia 1

Fundamentos computacionales, Inteligencia Artificial y Aprendizaje Automático

7.3.1. Introducción a la Computación 1

7.3.1.1. Computación 1

7.3.1.2. Programación 1

7.3.1.3. El Algoritmo 1

7.3.2. Inteligencia Artificial 1

7.3.3. Aprendizaje Automático 1

7.3.4. Aprendizaje Supervisado 1

7.3.4.1. La técnica Support Vector Machine 1

7.3.4.2. La técnica Random Forest 1

7.3.4.3. La técnica Naive Bayes 1

7.4. El pensamiento humano puede literalmente, transformar el mundo físico 1

El problema de investigación: Interfaces Cerebro – Computadora y el habla imaginada

8. Metodología 1

9. Desarrollo 1

10. Experimentación y pruebas 1

11. Análisis e Interpretación de Resultados 1

12. Conclusiones 1

13. Referencias 1

14. Anexos 1

14.1. Anexo 1: Configuración para recibir datos desde la diadema hacia el equipo de cómputo usando la paquetería de software de Emotiv, OpenViBE y la librería de Python: Cykit 1

14.2. Anexo 2: Código del archivo de Python: functions.py. 1

14.3. Anexo 3: Pseudocódigo de la función RunUpdateStream() encontrada en el archivo de Python: functions.py. 1

14.4. Anexo 4: Pseudocódigo de la función createDataset() encontrada en el archivo de Python: functions.py. 1

14.5. Anexo 5: Acuerdo de privacidad entregado a los usuarios que colaboraron con el experimento. 1

**Índice de Figuras**

Figuras 1

**Índice de Gráficas**

Gráficas 1

**Índice de Tablas**

Tablas 1

**Índice de Ecuaciones**

Ecuaciones 1

**1. Introducción**

Introducción.

**2. Planteamiento del Problema**

Planteamiento del Problema.

**3. Justificación**

Justificación.

**4. Objetivos**

Objetivos.

**5. Hipótesis**

Hipótesis.

**6. Pregunta de Investigación**

Pregunta de Investigación.

**7. Marco Teórico**

**7.1. Soy un cerebro, Watson. El resto de mí es un mero apéndice.**

*Arthur Conan Doyle: La piedra de Mazarino.*

**7.1.1. El cerebro y el sistema nervioso**

El cerebro es un órgano muy interesante, que ha cautivado la atención de muchos investigadores y la curiosidad de las personas, pues como lo señala la corporación Caldaria en su blog HDOSO Magazine, “el cerebro es el órgano más complejo del cuerpo, al menos en los vertebrados, y lo es porque es este órgano el que controla el resto de órganos del cuerpo y el que, en definitiva, nos define como seres humanos” (Caldaria, 2020). Nazareno, estudiante de la Universidad de Harvard mientras escribía un artículo expresando su fascinación por el cerebro declara cómo a su percepción, este órgano es fascinante y los hallazgos sobre éste siguen asombrando a la comunidad científica enfocada en esta parte del cuerpo (Nazareno, 2020).

Hill define al cerebro desde una perspectiva etimológica “del latín cerebrum, con su raíz indoeuropea ker, cabeza, en lo alto de la cabeza y brum, ‘llevar’; teniendo el significado arcaico de «lo que lleva la cabeza». Es un órgano que centraliza la actividad del sistema nervioso y existe en la mayor parte de los animales” (Hill, 2006). Este órgano “es el más complejo del cuerpo. El telencéfalo adquiere su máximo desarrollo y está formado por los hemisferios cerebrales. El cerebro humano contiene en la corteza cerebral, un número estimado de (20 mil millones, ) de neuronas” (von Bartheld, Bahney, & Herculano-Houzel, 2016), (Pelvig, Pakkenberg, Stark, & Pakkenberg, 2008), (Herculano-Houzel, 2009).

Puede deducirse gracias a las definiciones de los autores dadas previamente que el cerebro humano es el órgano central que recibe todo impulso nervioso que el cuerpo detecta y que tanto por su estructura como por su funcionamiento, es necesario dividirlo en algunas clasificaciones, como lo realizan Latarjet y Ruiz Liard, pues ellos señalan que está formado “por las estructuras derivadas del Telencéfalo y el Diencéfalo, los dos sectores anteriores del Prosencéfalo embrionario. Ocupa el sector anterior y superior del cráneo llamados fosa craneal anterior y fosa craneal media” (Latarjet & Ruiz Liard, 2004).

Al ser la piedra angular del sistema nervioso, cabe definir a este grupo como “el conjunto de órganos que regulan, coordinan e integran todas las actividades del organismo. Asimismo, constituye una unidad funcional compleja que se puede dividir, desde el punto de vista didáctico, en dos componentes morfológicos fundamentales: el sistema nervioso central (SNC) y el sistema nervioso periférico (SNP)” (Ojeda Sahagún & Icardo de la Escalera, 2004).

Ojeda Sahagún e Icardo de la Escalera (2004) también señalan en su obra algunas otras consideraciones entre esta división, pues “el SNC agrupa todas las estructuras del sistema nervioso que se encuentran alojadas dentro del estuche osteofibroso formado por la cavidad craneal y el conducto vertebral. Por situarse en la línea media, a veces se denomina neuroeje” (Ojeda Sahagún, 2004). Por otra parte, el SNP, comentan Ojeda Sahagún e Icardo de la Escalera (2004), “comprende el resto de estructuras nerviosas que, aunque en su origen siguen un breve trayecto dentro de la cavidad craneal o del conducto vertebral, se sitúan fuera del estuche osteofibroso”.

El SNC se divide en encéfalo y médula espinal según Ojeda Sahagún (2004). Para efectos de la presente investigación, se centrará la atención en esta parte del sistema nervioso, específicamente en el encéfalo, que a su vez está constituido por el tronco del encéfalo, el cerebelo, el diencéfalo y los hemisferios cerebrales. “El conjunto del diencéfalo y los hemisferios cerebrales se denomina cerebro” (Ojeda Sahagún & Icardo de la Escalera, 2004) y se enfocará la atención en este grupo.

Con lo anterior mencionado, puede realizarse la siguiente definición funcional para efectos de esta investigación, pues el cerebro abarca muchos conceptos gracias a su alta complejidad, pero también los estudios analizados coinciden con que éste es el punto central del sistema nervioso y es nuestro principal procesador de información, el que recibe los estímulos nerviosos del exterior y los convierte en conocimiento a través de la percepción, la razón, las emociones y las otras formas del comprensión de nuestro entorno. Es capaz también de controlar los movimientos voluntarios, el habla y nuestro productor de inteligencia.

Aunque se sabe que el cerebro es capaz de realizar estas acciones, es también importante saber el cómo es capaz de realizar estas actividades. Para ello requerimos de analizar la estructura del órgano tanto a nivel general como a un nivel específico.

**7.1.2. El cerebro a nivel macroestructura**

A un nivel de análisis amplio, Kandel, Schwartz y Jessel (2000) mencionan en su obra que el cerebro se divide en dos hemisferios cerebrales denominados el hemisferio izquierdo y derecho, los cuales “son aproximadamente simétricos, sin embargo el izquierdo es ligeramente mayor. Están separados por la profunda cisura medial. Están cubiertos por una capa cortical sinuosa, la corteza cerebral, formada por sustancia gris” (Kandel, Schwartz, & Jessel, 2000).

Anatomistas como los anteriormente mencionados clasifican cada hemisferio en seis lóbulos: lóbulo frontal, lóbulo parietal, lóbulo occipital, lóbulo temporal, lóbulo insular y lóbulo límbico. En la figura 7.1.1 puede observarse el cerebro y la división de los lóbulos que son visibles desde la corteza cerebral. La división de esta forma del cerebro permite centrarse en las diferentes secciones sabiendo que “cada lóbulo cerebral presente una serie de características propias no significa que cada estructura controle casi en “exclusiva” una determinada tarea. Muchas actividades y procesos se superponen a través de las distintas regiones cerebrales” (Sabater, 2020).



Figura 7.1.1. Los cuatro lóbulos cerebrales que pueden observarse desde la corteza cerebral. (Sabater, 2020).

Comenzando por analizar los lóbulos frontales, éstos posees grandes muestras de la evolución humana, pues “situados en la parte frontal de la cabeza, y justo debajo de los huesos frontales del cráneo y cerca de la frente, conforman la región más afinada de nuestro cerebro, la que más tiempo tardó en evolucionar y aparecer” (Sabater, 2020). Sabater también menciona que las tareas más notorias que este lóbulo realiza son las siguientes:

* La producción de habla y lenguaje.
* Procesos cognitivos que nos permiten planificar, fijar la atención, memorizar datos a largo plazo, comprender lo que vemos, regular nuestras emociones, etc.
* Comprender y reaccionar ante los sentimientos de los demás, como por ejemplo, la empatía.
* Regulación de la motivación y búsqueda de recompensas.

El siguiente lóbulo se trata del lóbulo parietal. Esta parte del cerebro se caracteriza por su importante papel en la percepción de los sentidos, en el razonamiento espacial, el movimiento de nuestro cuerpo y nuestra orientación. Tal como lo dice Sabater (2020) pues es en esta área donde “se capta la información sensorial relativa a la mayoría de nuestros órganos sensoriales. Es aquí donde se procesa y regula la sensación del dolor, la presión física y la temperatura, etc.” (Sabater, 2020). Gracias al área parietal, también es posible comprender la naturaleza de los números y la relación que encontramos con las matemáticas.

El siguiente lóbulo es el lóbulo occipital. Este es el lóbulo más pequeño y aunque se considera un camino de paso de los demás procesos, Sabater (2020) señala que este lóbulo sí realiza algunos procesos importantes como los siguientes:

* Participa en los procesos de percepción y reconocimiento visual.
* Tiene alta importancia en todo lo relativo al sentido de visión, ya que su corteza integra diversas áreas visuales como la detección de patrones, procesamiento de información y envío de esta información a otras áreas del encéfalo.
* Ayuda en la diferenciación de colores.
* Participa también en la elaboración de las emociones y pensamientos.

Dentro de este conjunto tenemos también al lóbulo temporal. Esta parte del cerebro, según Sabater (2020) se encarga especialmente de las siguientes tareas:

* Nos ayuda a reconocer los rostros.
* Tiene estrecha relación con la articulación del lenguaje, la comprensión de sonidos, voces y la música.
* Facilita el equilibrio.
* Participa en la regulación de emociones como la motivación, la rabia, la ansiedad, el placer, entre otras.

Aunque también destaca la autora que es “muy complicado asociar a cada una de estas estructuras a una única función especializada. Todas dependen unas de otras, todas se hallan conectadas y favorecen esa armonía perfecta” (Sabater, 2020) de modo que todos los lóbulos se desempeñen juntos en consonancia.

Sabater menciona que, gracias a los estudios del cerebro realizados, se ha hecho el hallazgo de la quinta región: el lóbulo insular o la ínsula lobular; descrita por esta autora como “un lóbulo oculto justo debajo de los lóbulos temporal, frontal y parietal. Es un área muy recóndita y de complejo acceso localizada entre diversos los vasos venosos y arterias” (Sabater, 2020). Puede observarse una muestra gráfica de su ubicación en la figura 7.1.2.

Sabater menciona que, pese a que no se sabe a ciencia cierta cuáles son sus funciones, gracias a otros procesos y correlaciones, el lóbulo insular participa en el sentido del gusto, el control visceral, la somatopercepción y otros procesos emocionales de los cuales forma parte junto con el sistema límbico. Otros autores comparten este hecho de la quinta región, pues como menciona Laguna, “el lóbulo de la ínsula es el que menos se ha estudiado. Por esta razón, es considerado un lóbulo nuevo para los estudiantes de ciencias de la salud ya que hay muy poca información sobre esta estructura” (Laguna, 2022).

Sin embargo, Laguna menciona otras funciones de esta área del cerebro, encontradas gracias a que “diversos estudios de neuroimagen han relacionado al lóbulo de la ínsula con los deseos, los antojos y las adicciones. Asimismo, se ha evidenciado que este lóbulo juega un papel de suma importancia en cuanto a los trastornos psiquiátricos, tales como la esquizofrenia, los trastornos de pánico, el estrés postraumático y el trastorno obsesivo-compulsivo” (Laguna, 2022).



Figura 7.1.2. Representación gráfica de la ubicación del lóbulo insular, señalada en color verde. (Laguna, 2022).

Finalmente se tiene el sexto lóbulo: el lóbulo límbico. Éste “comprende una serie de estructuras situadas en la zona medial de los hemisferios cerebrales. Interviene principalmente en la expresión de afectos y en la memoria” (Interpsiquis, 2022). Estos mismos autores mencionan las estructuras que forman parte de este lóbulo, así como pueden observarse en la figura 7.1.3, siendo las siguientes:

* Estructuras corticales: cingulado y gyrusparahipocámpico.
* Formaciones hipocámpicas: gyrus dentado, hipocampo y complejo subicular.
* La amígdala.
* El núcleo *accumbens*.
* El hipotálamo.
* El tálamo (núcleo anterior y núcleo dorsomedial).
* Otras estructuras corticales como el orbitofrontal y el polo temporal.



Figura 7.1.3. Estructuras que conforman el lóbulo límbico. (Triglia, 2016).

Con base en las definiciones y proposiciones de los autores previos, es posible enumerar las principales actividades de cada uno de los lóbulos cerebrales; así, serían las siguientes:

* **Lóbulo frontal:** Trabaja principalmente en el habla y el lenguaje, ayuda en fijar la atención y la memoria a largo plazo. Regula la motivación y búsqueda de recompensas, así como la empatía.
* **Lóbulo parietal:** Tiene alta importancia en la percepción de los sentidos, razonamiento espacial, percepción de nuestro cuerpo – también conocida como la percepción somatosensorial – y orientación.
* **Lóbulo occipital:** Participa activamente en la percepción y reconocimiento visual. Detecta patrones y procesa la información para enviarla a otras áreas del encéfalo.
* **Lóbulo temporal:** Actúa en el reconocimiento de rostros, está altamente relacionada con la articulación del lenguaje, comprensión de sonidos, voces y música. Actúa también como soporte en la regulación de emociones como la motivación, placer y ansiedad.
* **Lóbulo insular:** Es parte importante del sentido del gusto, control visceral. También participa en la somatopercepción.
* **Lóbulo límbico:** Actúa en la expresión de afectos y la memoria.

Es claro, entonces, que el cerebro posee ciertas divisiones funcionales que se encargan en mayor medida de alguna tarea. Son, de hecho, éstas descritas por Gray (2002) en su obra donde reconoce como divisiones funcionales al área sensorial primaria, el área motora primaria y las áreas de asociación.

El área sensorial primaria incluye “el área visual del lóbulo occipital, el área auditiva primaria en el lóbulo temporal y la corteza insular, y el área somatosensorial en el lóbulo parietal” (Gray, 2002). Esta área recibe señales de los nervios sensoriales y las envía a través del tálamo. El área de asociación la define Gray (2002) como la receptora de información entrante de áreas sensoriales y está implicada en el proceso de percepción, pensamiento y toma de decisiones.

Finalmente, se encuentra el área motora primaria, la cual, según Gray (2002), ocupa la parte posterior del lóbulo frontal, delante del área somatosensorial. Esta área envía los axones hasta las neuronas del tronco encefálico y la médula espinal para realizar sus tareas. Sin embargo, hablar de axones y neuronas implica entrar en el terreno de la microarquitectura cerebral, pero de esta forma será posible entender qué “señales” envía el cerebro para la intercomunicación corporal e incluso entre el mismo órgano. Comprender este aspecto será de mayor utilidad en capítulos posteriores de la presente investigación.

**7.1.3. El cerebro a nivel microestructura**

Es momento de hablar sobre las neuronas. Componente del sistema nervioso a un nivel de escala menor al de los anteriores constituyentes del sistema nervioso. Las neuronas son parte importante del sistema nervioso, pues incluso “se pueden distinguir dos grupos celulares básicos: células propias del SNC y células comunes con otros sistemas del organismo, como las células endoteliales de los capilares sanguíneos. Las células propias del SNC son de dos tipos diferentes: a) células excitables, denominadas neuronas y b) células no excitables, que incluyen la neuroglia y las células ependimarias” (Ojeda Sahagún & Icardo de la Escalera, 2004).

**7.1.3.1. Las neuronas**

Las neuronas son un componente principal del sistema nervioso, cuya función principal es recibir, procesar y transmitir información a través de señales químicas y eléctricas gracias a la excitabilidad eléctrica de su membrana plasmática. Las neuronas “son las unidades estructurales y funcionales del sistema nervioso. Son células excitables especializadas en la recepción, integración, transformación y transmisión en una sola dirección (conducción ortodrómica) de información codificada por cambios electroquímicos en su membrana plasmática” (Ojeda Sahagún & Icardo de la Escalera, 2004).

Estas células “están especializadas en la recepción de estímulos y conducción del impulso nervioso (en forma de potencial de acción) entre ellas mediante conexiones llamadas sinapsis, o con otros tipos de células como, por ejemplo, las fibras musculares de la placa motora. Altamente diferenciadas, la mayoría de las neuronas no se dividen una vez alcanzada su madurez; no obstante, una minoría sí lo hace” (Cayre, Malaterre, Scotto-Lomassese, Strambi, & Strambi, 2010).

Se han realizado estimaciones que señalan que, “el encéfalo humano contiene (ochenta y seis mil millones. ) de neuronas, de las cuales cerca de (diez mil millones. ) son células piramidales corticales. Estas células transmiten las señales a través de mil billones () de conexiones sinápticas” (Murre & Sturdy, 1995).

**7.1.3.2. Morfología de las neuronas**

Respecto a la morfología de las neuronas, diferentes autores hacen señalizaciones distintas a la composición de estas células. Ojeda Sahagún e Icardo de la Escalera (2004) mencionan que las neuronas “se caracterizan por poseer una gran superficie celular, lo que les permite llevar a cabo sus funciones específicas. El aumento de la superficie celular se consigue mediante la presencia de expansiones ramificadas que parten del cuerpo celular o soma neuronal y que se denominan neuritas (axón y dendritas). Ciertos tipos de neuronas se caracterizan por la presencia en su citoplasma de pigmentos de diferentes tipos, por lo que sus agrupaciones presentan una coloración característica”. Ellos también destacan como los elementos morfológicos más importantes al soma neuronal, las dendritas y al axón o también denominado cilindroeje.

Aunque otros autores destacan de la morfología de las neuronas “un núcleo voluminoso central, situado en el soma; un pericarion que alberga los orgánulos celulares típicos de cualquier célula eucariota; y neuritas (esto es, generalmente un axón y varias dendritas) que emergen del pericarion” (Paniagua, y otros, 2002).

Incluyendo la morfología que propone Merck & Co Inc. (2022) en la cual propone como componentes principales el cuerpo celular o también llamado soma neuronal, las dendritas y axones, así como las vainas de mielina de las neuronas. Estos componentes pueden observarse en la figura 7.1.4 con una observación gráfica de la morfología de las neuronas.



Figura 7.1.4. Morfología de una neurona. (Merck & Co, 2022).

A destacar entre estos componentes morfológicos de la neurona, el primero será el soma neuronal. Esta estructura está localizada en cada una de las neuronas del cerebro, la médula espinal y también en los ganglios espinales del sistema nervioso periférico. La función del soma neuronal es “es mantener al núcleo junto con la información genética (ADN), contener a las proteínas del retículo endoplasmático y además es el que genera la energía necesaria para que la neurona trabaje correctamente” (Fisio, 2022). Gracias a esta definición, dado que otros autores definen al cuerpo celular como el encargado de estas funciones, este parte de la célula recibe ambos nombres, tanto soma neuronal como cuerpo celular.

Es, de hecho, en el soma neuronal donde se asienta el núcleo celular. Tal como lo señalan Ojeda Sahagún e Icardo de la Escalera (2004) pues este elemento “constituye el centro para el crecimiento y mantenimiento de las neuritas y el lugar de la síntesis proteica. El tamaño del soma y del núcleo neuronal se encuentra en relación directa con el número y longitud de neuritas”.

Cabe señalar, para comprender la estructura anterior hay que entender qué es una neurita. Kevin Flynn lo define de la manera más concisa, pero adecuada para los términos necesarios en esta investigación señalando que éstas son “cualquier proyección del soma de una neurona, ya sea una dendrita o un axón. El término se usa con frecuencia al hablar de células nerviosas inmaduras o en desarrollo” (Flynn, 2013). Es por ello que bajo este término, Ojeda Sahagún e Icardo de la Escalera engloban a los axones y las dendritas.

Las neuronas son muy similares a células de otros tejidos en el aspecto de su estructura y organización de su núcleo celular encontrado en el cuerpo de la neurona, pero incluso así, Ojeda Sahagún e Icardo de la Escalera (2004) clasifican en tres tipos de estructuras características al soma neuronal, las cuales son las siguientes:

* Cuerpos de Nissl.
* Neurofilamentos.
* Neurotúbulos.

Otro elemento importante de las neuronas son las dendritas. Las dendritas son “ramificaciones que proceden del soma neuronal que consiste en proyecciones citoplasmáticas envueltas por una membrana plasmática sin envoltura de mielina. En ocasiones, poseen un contorno irregular, desarrollando espinas. Sus orgánulos y componentes característicos son: muchos microtúbulos y pocos neurofilamentos, ambos dispuestos en haces paralelos; además muchas mitocondrias; grumos de Nissl, más abundantes en la zona adyacente al soma; retículo endoplasmático liso, especialmente en forma de vesículas relacionadas con la sinapsis” (Wikipedia, 2022).

Asimismo, éstas también se definen como “prolongaciones celulares que se originan a partir del soma neuronal. Constituyen la porción receptiva de las neuronas y suelen ser muy numerosas. Las dendritas que se originan directamente del soma neuronal se denominan dendritas primarias o principales. Éstas se ramifican repetidamente, por lo general de forma dicotómica, dando origen a las dendritas secundarias. El conjunto de dendritas primarias y sus ramificaciones se conoce como árbol dendrítico” (Ojeda Sahagún & Icardo de la Escalera, 2004).

Estas prolongaciones de las neuronas están “dedicadas principalmente a la recepción de estímulos y, secundariamente, a la alimentación celular” (Roche, 2003). Éstas tienen quimiorreceptores que son capaces de reaccionar con los neurotransmisores que son enviados desde los extremos de los axones y siendo altamente importantes para la correcta transmisión de los impulsos quimioeléctricos a través de la vía nerviosa.

Las dendritas contienen la mayor parte de organelos que están presentes en el soma neuronal, entre ellos el retículo endoplásmico; otra característica es que son más cortas, gruesas y rugosas que el axón. “Las rugosidades se deben a la presencia de numerosas expansiones cortas, las espinas dendríticas que constituyen puntos donde una neurona entra en contacto con otras. Las dendritas no poseen envolturas especializadas, como ocurre en el caso de los axones” (Ojeda Sahagún & Icardo de la Escalera, 2004).

Otra parte importante de la neurona es el axón. El axón es definido en algunas ocasiones como “una delgada y extensa prolongación del soma neuronal, que está rodeado por su membrana, el axolema. El axolema puede estar recubierto por células de Schwann en el sistema nervioso periférico de vertebrados, con producción o no de mielina. Puede dividirse, de forma centrífuga al pericarion, en tres sectores: el cono axónico, el segmento inicial y el resto del axón” (Paniagua, y otros, 2002). Es, de hecho, destacado por Ojeda Sahagún e Icardo de la Escalera (2004) como la prolongación más importante del soma neuronal.

Algunas características notables de los axones, descritas en la obra de Ojeda Sahagún e Icardo de la Escalera (2004), son las siguientes:

1. El axón carece de la maquinaria precisa para realizar la síntesis proteica, de modo que los neurotransmisores y otros materiales deben ser transportados continuamente desde el soma neuronal a las terminaciones axonales, lo que constituye el transporte o flujo axonal anterógrado; también existe un movimiento en sentido contrario de menor intensidad llamado flujo axonal retrógrado.
2. A excepción del segmento inicial, el cual es la zona de origen del soma neuronal, la superficie del axón se encuentra cubierta por una vaina de mielina o por una envoltura celular, recibiendo el nombre de axones mielinizados o amielínicos respectivamente. La envoltura celular (para el caso del sistema nervioso central) es de oligodendroglia.

Cabe resaltar que la vaina de mielina la cual rodea al axón “no es continua, sino que se encuentra interrumpida de manera periódica. En estas zonas desnudas, denominadas nódulos de Ranvier, el axón puede poseer ramas colaterales. Los axones suelen terminar ramificándose en varias prolongaciones, las terminaciones axonales o telodendrias, que finalmente establecen contactos o sinapsis con otras neuronas o con células efectoras (músculo o glándulas)” (Ojeda Sahagún & Icardo de la Escalera, 2004).

Este mismo conjunto de autores, junto con otros especialistas en el área, describen también los tres sectores anteriormente mencionados:

* **Cono axónico.** “Adyacente al pericarion, es muy visible en las neuronas de gran tamaño. En él se observa la progresiva desaparición de los grumos de Nissl y la abundancia de microtúbulos y neurofilamentos que, en esta zona, se organizan en haces paralelos que se proyectarán a lo largo del axón” (Paniagua, y otros, 2002).
* **Segmento inicial del axón.** “En él comienza la mielinización externa. En el citoplasma, a esa altura se detecta una zona rica en material electrodenso en continuidad con la membrana plasmática, constituido por material filamentoso y partículas densas. La membrana se continúa con el axolema y se asume que este sector interviene en la generación del potencial de acción que transmitirá la señal sináptica” (Kole & Stuart, 2012).
* **Resto del axón.** Según Paniagua y los autores que apoyaron su obra (2002), es en este sector donde aparecen los nódulos de Ranvier y las sinapsis.

Con base en los autores destacados previamente es posible obtener una definición propia útil para el objetivo de la presente investigación, pues la importancia de estas células en la presente radica en su composición y comunicación. Para este caso, cabe resaltar que una neurona está compuesta del soma neuronal, las dendritas y los axones, principales componentes de estas células, donde todos estos son importantes para la comunicación entre estas células a través de actividades como la transmisión de potenciales eléctricos y síntesis proteica.

**7.1.3.3. Funciones de las neuronas**

Sin embargo, hablar de la comunicación entre neuronas requiere también profundizar en las funciones que las neuronas cumplen. No sólo es necesaria la comprensión de estos tópicos, sino que, aunque podría parecer de menor importancia mencionar a algunos investigadores en esta área, para esta investigación tendrán alta relevancia, pues no sólo estarán involucrados en la investigación de estos procesos neuronales, también lo están al hablar de modelos matemáticos de redes que formaron para explicar el complejo sistema de neuronas humano; pero previo a los temas de modelado, será primordial entender las funciones neuronales, explicadas a continuación.

La principal función de las neuronas es comunicarse entre sí. Estas células poseen una capacidad de realizar esta tarea con precisión, rapidez y a larga distancia, sin importar si éstas son de carácter nervioso, muscular o glandular. Es a través de estas células por donde se transmiten señales eléctricas a las cuales se les denomina impulsos nerviosos, los cuales “viajan por toda la neurona comenzando por las dendritas hasta llegar a los botones terminales, que se pueden conectar con otra neurona, fibras musculares o glándulas.” (Wikipedia, 2022). Esta conexión entre una neurona y otra es lo que recibe el nombre de sinapsis.

Es de hecho la sinapsis la unidad más simple de funcionamiento nervioso segmentario, la cual requiere dos neuronas: una neurona sensitiva receptora y una neurona motora o efectora; así lo explican Afifi y Bergman (1998) quienes también mencionan que “el acoplamiento anatomofuncional de estas dos neuronas ocurre a través de lo que se llama sinapsis. Las arborizaciones terminales de la neurona sensitiva (axones) se dilatan en pequeños brotes o botones (llamados *boutons terminaux*, un término acuñado por un investigador francés), que yacen en contacto con las dendritas, cuerpos celulares o axones de las neuronas efectoras” (Afifi & Bergman, 1998).

Las neuronas “conforman e interconectan los tres componentes del sistema nervioso: sensitivo, motor e integrador o mixto; de esta manera, un estímulo que es captado en alguna región sensorial entrega cierta información que es conducida a través de las neuronas y es analizada por el componente integrador, el cual puede elaborar una respuesta, cuya señal es conducida a través de las neuronas. Dicha respuesta es ejecutada mediante una acción motora, como la contracción muscular o secreción glandular” (Wikipedia, 2022).

La primera función destacable de las neuronas es el impulso nervioso, en el cual las neuronas transmiten ondas eléctricas causadas por un cambio transitorio en la membrana plasmática. La propagación de estas ondas se debe a “la existencia de una diferencia de potencial o potencial de membrana entre la parte interna y externa de la célula, que surge gracias a las concentraciones distintas de iones a ambos lados de la membrana, según describe el potencial de Nernst” (Cromer, 1996). Usualmente esta diferencia de potencial, según Cromer (1996), es de milivolts (mV).

La carga de una célula inactiva se mantiene en valores negativos, pues se trata de la carga del interior respecto al exterior, y varía de unos estrechos márgenes. “Cuando el potencial de membrana de una célula excitable se despolariza más allá de un cierto umbral (de mV a mV aproximadamente) la célula genera un potencial de acción. Un potencial de acción es un cambio muy rápido en la polaridad de la membrana de negativo a positivo y vuelta a negativo, en un ciclo que dura milisegundos” (Bear, Connors, & Paradiso, 2002).

El proceso del potencial de acción puede verse ilustrado en la gráfica 7.1.1, que muestra el voltaje en función del tiempo transcurrido, observando el cambio que ocurre cuando esta función es realizada.



Gráfica 7.1.1. Gráfica del potencial de acción de una neurona en función del tiempo. (MDurance, 2021).

La siguiente función de las neuronas será en la que se hará mayor énfasis en este documento, pues se trata de las corrientes iónicas que ocurren durante el potencial de acción, siendo de alta importancia para la problemática actual, también se debe a que el primer registro de un potencial de acción fue realizado por los investigadores Alan Lloyd Hodgkin y Andrew Fielding Huxley.

Estos importantes nombres no sólo son notables por el estudio de esta función de las neuronas que realizaron, sino que también realizaron un conjunto de ecuaciones diferenciales ordinarias no lineales las cuales aproximan las características eléctricas de las neuronas (en general, de las células excitables) proponiendo el modelo en 1952 y posteriormente obteniendo el Premio Nobel en fisiología – medicina por esta investigación en 1963 tras investigar y registrar esta exploración experimentando con el axón de un calamar gigante (Hodgkin & Huxley, 1939).

De todas formas, se entrará en amplio detalle a la parte matemática de la investigación de estos dos autores en capítulos posteriores, ya que, para mantenerse dentro de los límites de este capítulo, habrá que tocar la investigación desde el punto de las bases iónicas como función de una neurona.

El potencial de acción comprende tres fases:

1. Potencial en reposo o potencial de membrana, llevándose a cabo una permeabilidad al sodio y al potasio.
2. Despolarización de la membrana celular al sodio y al potasio.
3. Repolarización de la membrana al sodio y al potasio.

Hodgkin y Huxley midieron la conductancia durante la despolarización y repolarización observando cambios de conductancia para el sodio () y el potasio () durante el potencial de acción (Hodgkin & Huxley, 1939).

Al observar los cambios previos puede determinarse la diferencia de potencial por la diferencia absoluta entre las cargas positivas y negativas entre el interior y el exterior con relación a la membrana. Sin embargo, como a continuación se tratarán las cargas eléctricas de las moléculas que atraviesan la membrana, para entender estos conceptos es importante destacar que estas cargas se originan gracias a los iones; átomos o moléculas “cargadas eléctricamente de modo que ya no son eléctricamente neutras” (Ebbing & Gammon, 2010). De este modo, puede haber dos tipos de iones los cuales también describen Ebbing y Gammon, que son los aniones y los cationes. Un anión es un ion con una ganancia de electrones o como se conoce, un ion con carga negativa, así entonces, un catión es un ion con pérdida de electrones, es decir, un ion con carga positiva.

Lo anterior es necesario explicarlo y entenderlo porque de hecho, Wikipedia (2022) en su sitio especializado para la neurona explica que la diferencia durante el potencial de acción se obtiene por la carga aniónica y catiónica – es decir, por la carga negativa y positiva – de los iones involucrados. A destacar, el anión participante en este proceso es el cloro (); por otra parte, los cationes de mayor reconocimiento son el potasio (), magnesio (), calcio () y sodio (). Este procedimiento puede observarse en la gráfica 7.1.2.

Interfaz de usuario gráfica

Descripción generada automáticamente

Gráfica 7.1.2. Gráfica del potencial de acción de una neurona mostrando la interacción iónica. (Olmo, Nave, & Nave, 2022).

Ahora, cuando un canal iónico es abierto, “el tránsito iónico es a favor de su gradiente electroquímico, esto es, pretende equilibrar el número de iones, independientemente del potencial trasmembrana actual. Este mecanismo circunstancial de movimiento iónico permite el tránsito entre estados de polarización y despolarización” (Wikipedia, 2022).

Esto anterior con el objetivo de equilibrar la diferencia electroquímica que existe tanto al exterior como al interior de la membrana, siguiendo, claro, la definición de gradiente en matemáticas, pues se trata de “la variación del valor de una magnitud en dos puntos y la distancia que los separa” (Sáenz, 2013) ya que la definición usada de gradiente electroquímico previamente combina tanto este concepto como su aplicación dentro de la electroquímica, siendo ésta la diferencia iónica en los dos puntos de la membrana anteriormente mencionados.

Este procedimiento fue usado para modelar la propuesta de Hodgkin y Huxley, la cual se retomará más adelante desde un punto de vista más analítico. Finalmente, Wikipedia (2022) menciona otro grupo de funciones importantes de las neuronas, a saber:

* Propiedades electrofisiológicas intrínsecas.
* Neurosecreción.
* Neurodegeneración.

Las anteriores sólo se mencionan de manera superficial dado que no contribuyen en gran medida para solucionar el problema de la presente investigación.

Ha sido un largo camino de neuroanatomía que ha habido que recorrer, sin embargo, es necesario revisar dos tópicos más antes de avanzar al siguiente capítulo y acercarse un poco más al problema que se desea abordar en esta investigación. Estos dos temas son la interacción entre neuronas y, por consiguiente, también las conexiones sinápticas.

Ya es del conocimiento del lector conceptos relacionados al cerebro, como su división a macroescala, así como su funcionamiento a microescala. De esta forma se ha revisado el componente principal de este capítulo: la neurona; observando su morfología y funciones, pero también es importante integrar la neurona con más de este tipo de células y explicar la interacción entre estas células.

**7.1.3.4. Interacción entre neuronas**

Para comenzar con la interacción entre neuronas, es necesario comprender que “el sistema nervioso procesa la información siguiendo un circuito más o menos estándar. La señal se inicia cuando una neurona sensorial recibe un estímulo externo. Su axón se denomina fibra aferente. Esta neurona sensorial transmite una señal a otra aledaña, de modo que acceda un centro de integración del sistema nervioso del animal. Las interneuronas, situadas en dicho sistema, transportan la señal a través de sinapsis. Finalmente, si debe existir respuesta, se excitan neuronas eferentes que controlan músculos, glándulas u otras estructuras anatómicas. Las neuronas aferentes y eferentes, junto con las interneuronas, constituyen el circuito neuronal” (Randall, Burggren, & French, 1998).

La interacción entre neuronas se da gracias a toda la red de neuronas que están interconectadas. Es así que Randall, Burggren y French (1998) realizan definiciones de estos conceptos, como el mencionado circuito neuronal, el cual se trata de “un conjunto de conexiones sinápticas ordenadas que se produce como resultado de la unión de las neuronas a otras en sus regiones correspondientes tras la migración neuronal”.

Mencionando también que “el crecimiento dirigido de los axones y el reconocimiento de las estructuras sinápticas está mediado por el cono de crecimiento, que es una especialización en el extremo de cada axón en crecimiento. El cono de crecimiento detecta y responde a moléculas de señalización que pueden ser de retraimiento, giro o continuación, que identifican las vías correctas, prohíben las incorrectas y facilitan la formación de sinapsis” (Randall, Burggren, & French, 1998).

Cuando se habla de una red neuronal biológica se destaca que “La función de un determinado grupo de neuronas es alcanzar un determinado estado final en función de los estímulos externos. Por ejemplo, en la percepción del color, un grupo de neuronas puede encargarse de acabar en un determinado estado si el estímulo es "rojo" y otro determinado estado si el estímulo es "verde". El número de "estados estables" posibles del circuito neuronal se corresponde con el número de patrones (en este caso colores diferentes) que puede reconocer el circuito neuronal. Los trabajos de Freeman en los años 1990 aclararon que un determinado grupo de neuronas sigue un patrón de evolución temporal caótico hasta alcanzar un determinado estado” (Solé & Manrubia, 1996).

Solé y Manrubia (1996) profundizan mucho en estos tópicos de circuitos neuronales. De hecho también entran en análisis de modelos matemáticos, pues por ejemplo, estudian algunos temas de estas redes como la teoría de Hopfield y la regla de Hebb pues de estos dos argumentos mencionan que “estiman la relación entre el número de neuronas que intervienen en reconocer patrones y la probabilidad de error en el reconocimiento de patrones”.

Sin embargo, la teoría de Hopfield también se trata de un modelo matemático y ciertamente en este capítulo ya se están tocando temas que corresponden a apartados posteriores. Aun así, los conocimientos de neuroanatomía han sido cubiertos en su totalidad y todos y cada uno de los conceptos vistos en este capítulo serán de alta importancia en los posteriores.

Aunque y se están tocando temas de matemáticas, antes es necesario conocer cómo es posible obtener la actividad eléctrica cerebral, de qué forma se puede clasificar la actividad una vez que se capture y cuáles serían los aparatos con los que se obtendrían estos datos. Estos conceptos y más relacionados serán del conocimiento del lector en el próximo capítulo relacionado a la electroencefalografía.

**7.2. Todo hombre puede ser, si se lo propone, escultor de su propio cerebro**

*Santiago Ramón y Cajal*

En este capítulo se hablará centrándose ahora en los estudios de electroencefalografía (hablar aquí de las eeg waves).

**7.3. Toda tecnología suficientemente avanzada es indistinguible de la magia**

*Arthur C. Clarke*

Hemos dejado atrás los conceptos de neurociencia – aunque dejar atrás no significa olvidar – y es momento de revisar lo que, para el autor, son los temas, tanto de mayor interés como los que más le atañen en función de obtener el grado profesional como propósito de esta investigación. Se trata de nada más y nada menos que los conceptos matemáticos y computacionales que rodean a esta investigación.

Nos hemos acercado a revisar el cerebro y su estructura, también su estudio a una gran escala y a una pequeña escala y fue aquí donde nos adentramos en nuestro componente más importante: la neurona. Revisamos la morfología de una neurona y las funciones que realiza así como la manera en la que una de estas células interactúa con otra.

Después de todo esto avanzamos de capítulo para entender cómo podía estudiarse esta actividad cerebral y entender lo que era un electroencefalograma, así como la forma en la que se reciben los datos de la actividad cerebral por medio de este aparato; clasificamos las diferentes ondas cerebrales para entender por separado lo que cada tipo de onda representaba. Pero ahora que hablaremos de conceptos de ciencias computacionales, ¿todo esto qué tiene que ver con la computación?

Bien, pues a lo largo del capítulo se observará la importancia de la computadora para poder tratar los datos recibidos por un electroencefalograma, siendo este aspecto la última pieza del presente rompecabezas al cual le he nombrado tesina de grado licenciatura para poder llegar al problema de investigación, el cual tratara justamente de lo que es posible realizar con los datos de un electroencefalograma al manipularlos, entenderlos y aplicar técnicas matemáticas adaptadas a las máquinas para que estos procedimientos los realicen de forma automática, pero para ello, primero será necesario arrancar desde el principio definiendo lo que una computadora es.

**7.3.1. Introducción a la Computación**

Dado que la presente investigación requiere avanzar a tópicos matemáticos y computacionales considerados algo avanzados, estas primeras definiciones no serán muy discutidas, pero son necesarias de mencionar.

**7.3.1.1. Computación**

Primeramente, hay que definir lo que es una computadora. Una computadora es “es un dispositivo electrónico, utilizado para procesar información y obtener resultados, capaz de ejecutar cálculos y tomar decisiones a velocidades millones o cientos de millones más rápidas que puedan hacerlo los seres humanos. En el sentido más simple una computadora es un dispositivo para realizar cálculos o computar” (Joyanes Aguilar, 2008).

Indiscutiblemente, para los efectos de esta investigación, la definición de Joyanes Aguilar es completa y concisa para abarcar los términos necesarios de lo que es una computadora, pues se trata de una máquina a la que se le ingresan datos y ésta responde procesándolos y regresando un resultado después de su procesamiento. Joyanes Aguilar (2008) también hace la clarificación de dividir una computadora en dos partes distintas: “hardware y software. El hardware es la computadora en sí misma. El software es el conjunto de programas que indican a la computadora las tareas que debe realizar” donde puede interpretarse esta mención al hardware como la parte tangible de la computadora mientras que el software, siendo los programas del equipo, se vuelve la parte intangible de la computadora. Así también se conoce con el término programador a la persona que escribe programas.

Sin embargo, es justo gracias al software donde en su mayoría puede enfocarse el procesamiento de datos, o más bien, las instrucciones para este procesamiento, pues el procesamiento de los datos que la computadora recibe se realiza gracias a “un conjunto de instrucciones denominadas programas de computadora. Estos programas controlan y dirigen a la computadora para que realice un conjunto de acciones (instrucciones) especificadas por personas especializadas” (Joyanes Aguilar, 2008) a los cuales este autor denomina programas de computadora.

Son justamente estos programas de computadora los que reciben datos, realizan el procesamiento y producen un resultado con los datos recibidos. “Los datos y la información se pueden introducir en la computadora por una entrada (input) y a continuación se procesan para producir una salida (output, resultados)” (Joyanes Aguilar, 2008).

**7.3.1.2. Programación**

“La computadora se puede considerar como una unidad en la que se colocan ciertos datos (entrada de datos), se procesan y se produce un resultado (datos de salida o información)” (Joyanes Aguilar, 2008). Este autor menciona que los datos, tanto de entrada como de salida pueden ser de cualquier tipo como texto, dibujos, sonido, imágenes, entre muchos otros tipos. Aunque también menciona que el formato más sencillo de comunicación entre una persona con una computadora es mediante un teclado, una pantalla o monitor y un ratón o mouse.

Dado que en esta investigación lo que se producirá es una solución de software, se centrará específicamente en este aspecto de las ciencias computacionales. Para que el hardware realice las operaciones necesarias se especifican por un conjunto de instrucciones llamados programas. “El software de una computadora es un conjunto de instrucciones de programa detalladas que controlan y coordinan los componentes hardware de una computadora y controlan las operaciones de un sistema informático. El auge de las computadoras el siglo pasado y en el actual siglo XXI, se debe esencialmente al desarrollo de sucesivas generaciones de software potentes y cada vez más amistosas” (Joyanes Aguilar, 2008).

Un programa tiene que escribirse en un lenguaje de programación que será la principal herramienta del programador para indicarle a la computadora qué deberá realizar al escribir el código fuente de un programa. Sin embargo, “la computadora no entiende directamente los lenguajes de programación, sino que se requiere un programa que traduzca el código fuente a otro lenguaje que sí entiende la máquina directamente, pero muy complejo para las personas; este lenguaje se conoce como lenguaje máquina y el código correspondiente código máquina. Los programas que traducen el código fuente escrito en un lenguaje de programación a código máquina se denominan traductores” (Joyanes Aguilar, 2008).

**7.3.1.3. El algoritmo**

Antes de escribir el código fuente de un programa debe de analizarse el problema y abstraer las partes más importantes que componen a la problemática. Esto es diseñar un algoritmo. Joyanes Aguilar (2008) define a un algoritmo como un método para resolver un problema y él también propone que los pasos para la resolución de un problema son los siguientes:

1. Diseño del algoritmo. Describe la secuencia ordenada de pasos —sin ambigüedades— que conducen a la solución de un problema dado. (Análisis del problema y desarrollo del algoritmo).
2. Expresar el algoritmo como un programa en un lenguaje de programación adecuado. (Fase de codificación).
3. Ejecución y validación del programa por la computadora.

Un algoritmo es independiente del lenguaje de programación y también lo es de la computadora que lo ejecuta y, de hecho, el algoritmo posee una alta importancia en este campo de estudio pues “En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es tan sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente” (Joyanes Aguilar, 2008).

Todo algoritmo debe cumplir con unas características muy específicas para clasificarse como tal. Estas características necesarias dentro de todo algoritmo, según Joyanes Aguilar (2008) se describen a continuación:

* Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.
* Un algoritmo debe estar bien definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
* Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

**7.3.2. Inteligencia Artificial**

Sin embargo, el desarrollo de software ha evolucionado a lo largo de los años y esto ha traído avances en este campo notables, entre ellos la aparición de la Inteligencia Artificial o abreviada, IA. Algunos investigadores en el área encuentran que el trabajo en este campo comenzó poco después de la Segunda Guerra Mundial y se le dio su nombre a esta área en 1956. Aun así, la IA ha crecido mucho en los pocos años que ha tenido de desarrollo, con campos que se han creado tales como aprendizaje y percepción, demostración de teoremas matemáticos, diagnóstico de enfermedades, entre muchas otras más. El fundamento de la IA es que “sintetiza y automatiza tareas intelectuales y es, por lo tanto, potencialmente relevante para cualquier ámbito de la actividad intelectual humana. En este sentido, es un campo genuinamente universal” (Russell & Norvig, 2004).

No obstante, la Inteligencia Artificial no tiene una definición sencilla, pues distintos autores dan definiciones muy dispersas. Russell (2004) realiza un compendio de algunas definiciones en un cuadro que remarca cuatro enfoques que combinan la racionalidad y la forma de actuar de los humanos con el pensamiento y el actuar, lo que resulta en el cuadro presentado en la tabla 7.3.1.

Tabla

Descripción generada automáticamente

Tabla 7.3.1. Cuadro de definiciones de la Inteligencia Artificial presentado por Stuart Russell. (Russell & Norvig, 2004).

Afortunadamente, en 1950, Alan Turing apareció en esta ecuación para definir a la Inteligencia Artificial desde un punto de vista operacional. “En vez de proporcionar una lista larga y quizá controvertida de cualidades necesarias para obtener inteligencia artificialmente, él sugirió una prueba basada en la incapacidad de diferenciar entre entidades inteligentes indiscutibles y seres humanos” (Russell & Norvig, 2004). De este modo, una computadora debería poseer las siguientes capacidades para superar la acuñada Prueba de Turing:

* Procesamiento de lenguaje natural que le permita comunicarse satisfactoriamente en algún idioma.
* Representación del conocimiento para almacenar lo que se conoce o siente.
* Razonamiento automático para utilizar la información almacenada para responder a preguntas y extraer nuevas conclusiones.
* Aprendizaje automático para adaptarse a nuevas circunstancias y para detectar y extrapolar patrones.

Los fundamentos de la Inteligencia Artificial se han apoyado en muchas otras ciencias, como la filosofía, las matemáticas, la economía, la neurociencia, la psicología, la ingeniería computacional, la teoría del control y cibernética y la lengua para entender todos estos aspectos que rodean al campo de la IA pero esto no ha detenido el avance de los modelos de Inteligencia Artificial. Entre ellos, Stuart Russell en su obra del 2004 señala una basta cantidad de modelos que se estudian comúnmente y que él expuso en su libro, entre ellos se encuentran modelos de aprendizaje estadístico, como los modelos de Bayes Simples (o también conocidos como Naive Bayes), modelos de aprendizaje con variables ocultas, aprendizaje no supervisado, redes bayesianas con variables ocultas, modelos de Markov, aprendizaje basado en instancias, modelos de vecinos más cercanos, redes neuronales, máquinas núcleo y muchos otros.

**7.3.3. Aprendizaje Automático**

Es tan amplio el campo de la Inteligencia Artificial que dentro de éste se encuentran ramas notables; la más notable y útil para el caso de la investigación actual es la del Aprendizaje Automático. Este campo de estudio tiene como objetivo, según Rusell y Norvig (2004), “desarrollar técnicas que permitan que las computadoras aprendan. Se dice que un agente aprende cuando su desempeño mejora con la experiencia y mediante el uso de datos; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento”. Estos mismos autores, en ediciones más recientes de la obra mencionan cómo ocurre el aprendizaje automático, explicando que la computadora observa datos después construye un modelo basado en estos datos y lo utiliza como hipótesis acerca del mundo, siendo ésta, una pieza de software que puede resolver problemas.

Así como la mera Inteligencia Artificial es tan diversa, el campo del Aprendizaje Automático también mantiene esta tendencia, pues se ha mantenido en constante desarrollo; tan es así que esta rama tiende a clasificar la investigación y aplicación generada en grupos de modelos de aprendizaje, así como algoritmos de aprendizaje. Flach (2012) en su obra menciona algunos modelos destacables por la forma en la que éstos abordan los problemas.

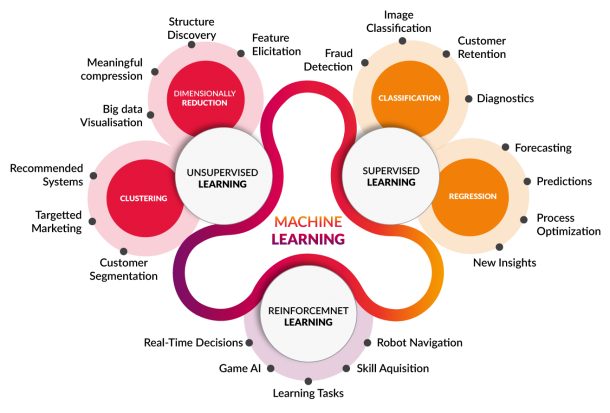


Figura 7.3.1. Clasificación del Machine Learning (o Aprendizaje Automático) en sus tres grandes tipos de aprendizaje (González Barrio, Calleja Ochoa, Gómez-Escudero, Rodríguez Ezquerro, & López de Lacalle Marcaide, 2021).

Sin embargo, la riqueza de este campo recae en los distintos tipos de aprendizaje que se han desarrollado bajo el estudio de esta área siendo tres principales:

1. **Aprendizaje supervisado:** En este tipo de aprendizaje “el operador proporciona al algoritmo de aprendizaje automático un conjunto de datos conocidos que incluye las entradas y salidas deseadas, y el algoritmo debe encontrar un método para determinar cómo llegar a esas entradas y salidas” (APD, 2019). Esto quiere decir que el algoritmo busca cómo establecer una correspondencia entre las entradas y las salidas deseadas del sistema, donde los ejemplos comunes de estos sistemas son el uso de sistemas con bases de conocimiento las cuales están conformadas por ejemplos anteriormente etiquetados.
2. **Aprendizaje no supervisado:** Estos algoritmos están diseñados con el propósito de que el sistema estudie los datos en búsqueda de patrones debido a que “no hay una clave de respuesta o un operador humano para proporcionar instrucción. En cambio, la máquina determina las correlaciones y las relaciones mediante el análisis de los datos disponibles” (APD, 2019). Entre más datos evalúe el sistema, la toma de decisiones del algoritmo mejorará consiguiendo una mejor segmentación de datos en distintos grupos.
3. **Aprendizaje por refuerzo:** El aprendizaje por refuerzo “se centra en los procesos de aprendizajes reglamentados, en los que se proporcionan algoritmos de aprendizaje automáticos con un conjunto de acciones, parámetros y valores finales. Al definir las reglas, el algoritmo de aprendizaje automático intenta explorar diferentes opciones y posibilidades, monitorizando y evaluando cada resultado para determinar cuál es el óptimo" (APD, 2019). Este tipo de sistemas aprenden a través del ensayo y error observando el mundo que les rodea y evaluando la retroalimentación que obtiene al interactuar con el exterior.

Estos tres grandes grupos pueden observarse separados visualmente con mayor claridad en la figura 7.3.1, sin embargo, para efectos de esta investigación se hará énfasis en el Aprendizaje Supervisado, pues posteriormente se observará con claridad el problema que se trata de resolver y las características del mismo, de modo que pueden percibirse rasgos de clasificación y predicción dentro del problema a resolver; es por esto que se eligió este campo para la solución del problema.

**7.3.4. Aprendizaje Supervisado**

El Aprendizaje Supervisado es un método de análisis de datos muy utilizado por la comunidad científica y que trasciende igualmente al ámbito analítico práctico; dentro de esta rama del Machine Learning pueden encontrarse algoritmos que “que aprenden iterativamente de los datos para permitir que los ordenadores encuentren información escondida sin tener que programar de manera explícita dónde buscar” (TIBCO Data Science, 2022).

Otros autores definen esta aproximación como “una técnica de aprendizaje automático en la que se proporciona al sistema un conjunto de datos etiquetados, con el objetivo de que el sistema aprenda a generalizar y hacer predicciones sobre nuevos datos” (Alpaydin, 2010).

Asimismo, este concepto tiene atribuciones como englobar “una clase de algoritmos de aprendizaje automático en los que se proporciona al sistema un conjunto de datos con las respuestas correctas, con el objetivo de que el sistema aprenda a hacer predicciones precisas sobre nuevos datos” (Bishop, 2006).

Para todos estos autores, este enfoque para la resolución de problemas converge en una similitud, siendo ésta la utilización de un conjunto de datos etiquetados – como se mencionó en la sección previa – de modo que se entrene a un algoritmo para una o varias tareas específicas. TIBCO Data Science (2022) usan como ejemplo en su artículo la clasificación y predicción de imágenes en donde se perciban vehículos de dos y cuatro ruedas, donde ellos mencionan que para realizar este procedimiento deberá de entrenarse a un algoritmo con un conjunto de datos que esté correctamente etiquetado, de modo que el algoritmo debe verificar si un vehículo es de dos o de cuatro ruedas.

El aprendizaje supervisado permite “que los algoritmos 'aprendan' de datos históricos/de entrenamiento y los apliquen a entradas desconocidas para obtener la salida correcta. Para funcionar, el aprendizaje supervisado utiliza árboles de decisión, bosques aleatorios y Gradient Boosting Machine” (TIBCO Data Science, 2022).

Un tipo de uso muy común que se le da a nivel práctico al Aprendizaje Supervisado es al estudio del comportamiento en distintas compañías, creando sistemas de negocio capaces de hacer predicciones variadas, generalmente buscando optimizar las ganancias de este negocio por medio de un proceso de obtención y procesamiento de datos con estos algoritmos. Este proceso se puede ver reflejado en la figura 7.3.2.

Calendario

Descripción generada automáticamente

Figura 7.3.2. Procedimiento general del Aprendizaje Supervisado (TIBCO Data Science, 2022).

Es importante aclarar que, aunque los autores previos no mencionan una característica importante del Aprendizaje Supervisado, ésta se encuentra intrínseca en las dos subespecializaciones que existen en este campo, tal como se menciona en la definición siguiente, acotando este proceso como uno en el que “un algoritmo de aprendizaje automático se entrena con un conjunto de datos de entrenamiento etiquetados para poder realizar tareas de clasificación o regresión” (Murphy, 2012).

Murphy en su obra otorga clarificaciones de estos dos tipos de tareas que se encuentran como subcampos de este tipo de Aprendizaje: la clasificación y la regresión; entre lo más destacable que él menciona al respecto es la comparación entre ambas, denotando que “la clasificación se refiere a la tarea de asignar una etiqueta o categoría a una entrada, mientras que la regresión se refiere a la tarea de predecir un valor numérico continuo” (Murphy, 2012).

Aunque como se mencionó anteriormente, los demás autores no se rezagan en hacer notar esta importante distinción, pues Alpaydin (2010) también aporta su propio enunciado, donde define que “la clasificación se refiere a la tarea de asignar una etiqueta o categoría a una entrada, mientras que la regresión se refiere a la tarea de predecir un valor numérico continuo”. Por otra parte, Bishop (2006) menciona que “la clasificación es el proceso de asignar una etiqueta a una instancia, mientras que la regresión es el proceso de predecir un valor numérico continuo”.

Queda claro, entonces, que al tener en cuenta a los autores previos, la diferencia entre clasificación y regresión radica en el tipo de resultado que se desea, lo que causará que se deseé inclinarse más por un tipo de tarea u otro.

A través de las definiciones anteriores, puede construirse un concepto más apropiado para los efectos de esta investigación, pues puede observarse que el uso de clasificación recae en situaciones donde se requiere como resultado una clase de entre un número limitado de éstas, mientras que la regresión se utiliza en problemas donde el resultado será un número dentro de un conjunto infinito de posibles resultados.

Martínez Heras (2020) expone en su artículo algunos ejemplos de clasificación, entre los cuales se encuentran:

1. ¿Comprará el cliente este producto? [sí, no]
2. ¿Tipo de tumor? [maligno, benigno]
3. ¿Subirá el índice bursátil? IBEX mañana [sí, no]
4. ¿Es este comportamiento una anomalía? [sí, no]
5. ¿Nos devolverá este cliente un crédito? [sí, no]
6. ¿Qué deporte estás haciendo? Tal y como lo detectan los relojes inteligentes [caminar, correr, bicicleta, nadar]

Este mismo autor también ofrece algunos ejemplos en donde se observa el uso de regresión:

1. Predecir por cuánto se va a vender una propiedad inmobiliaria
2. Predecir cuánto tiempo va a permanecer un empleado en una empresa
3. Estimar cuánto tiempo va a tardar un vehículo en llegar a su destino
4. Estimar cuántos productos se van a vender

Finalmente, se ha visto el conocimiento suficiente para poder realizar un acercamiento a las herramientas propias de esta investigación: las técnicas de Machine Learning. Martínez Heras (2020) menciona como ejemplos de algunas técnicas de Aprendizaje Automático para la realización de clasificación y regresión son las siguientes:

* Máquinas de vectores de soporte (Support vector machines).
* Árboles de decisión (Decision trees).
* Bosques Aleatorios (Random forests).
* Redes neuronales y Aprendizaje profundo (Neural networks and Deep learning).

Cabe aclarar que el autor menciona que, pese a que hay técnicas sumamente especializadas en clasificación y otras en regresión, la mayoría de las técnicas funcionan para ambos enfoques.

En la figura 7.3.3 se observa la separación entre los tres tipos de aprendizaje del Machine Learning incluyendo algunos ejemplos de técnicas que estos tipos de aprendizaje utilizan para abordar los problemas que se les presentan.

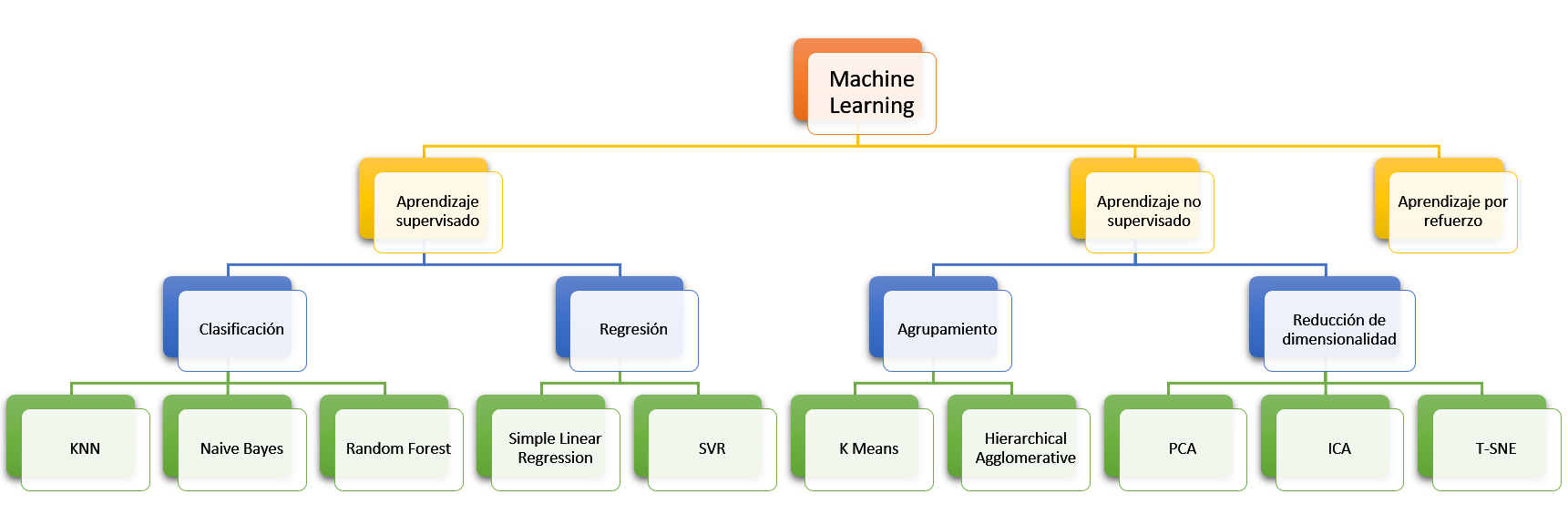


Figura 7.3.3. Principales algoritmos de Machine Learning (Betanzos Gómez, 2020).

Se le pide al lector dar otro salto de fe en este punto, confiando en que, por el momento se menciona que el problema de la presente investigación se trata de un problema de clasificación. Al finalizar este capítulo todos los cabos sueltos coinciden en un mismo rumbo y en el capítulo siguiente todo se habrá esclarecido – aunque con los tópicos ya revisados de Electroencefalografía y Aprendizaje Supervisado, ya puede ser clara la idea del rumbo de esta investigación – pues todo el conocimiento de base necesario estará puesto sobre la mesa.

Dado que el problema de la presente investigación se trata de un problema de clasificación, se tomó la decisión de abordar esta pregunta de investigación con tres distintas técnicas de Machine Learning para clasificación: la técnica de Máquinas de Vectores de Soporte, la técnica de Bosques Aleatorios y la técnica de Bayes Ingenuo. A continuación, se aclara cada técnica y se explica por qué se eligieron estas tres técnicas especialmente.

**7.3.4.1. La técnica Support Vector Machine**

Esta técnica será la primera en uso en esta investigación. La Máquina de Vectores de Soporte, o en inglés “Support Vector Machine” (abreviada de ahora en adelante como SVM) es “un algoritmo de aprendizaje supervisado que se utiliza en muchos problemas de clasificación y regresión, incluidas aplicaciones médicas de procesamiento de señales, procesamiento del lenguaje natural y reconocimiento de imágenes y voz” (The MathWorks Inc., s.f.).

El equipo de MathWorks (s.f.) menciona que este algoritmo trata de “encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. ‘De la mejor forma posible’ implica el hiperplano con el margen más amplio entre las dos clases, representado por los signos más y menos en la siguiente figura”.

El margen que menciona el reconocido equipo dueño de MATLAB es aquel que se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores. El algoritmo solo puede encontrar este hiperplano en problemas que permiten separación lineal; en la mayoría de los problemas prácticos, el algoritmo maximiza el margen flexible permitiendo un pequeño número de clasificaciones erróneas.

Diagrama

Descripción generada automáticamente

Figura 7.3.4. Definición del “margen” entre clases: el criterio que los SVM intentan optimizar (The MathWorks Inc., s.f.).

Estos vectores de soporte hacen referencia a un subconjunto de observaciones de entrenamiento, las cuales identifican posiciones del hiperplano de separación. Puede observarse una ejemplificación en la figura 7.3.4.

Un algoritmo estándar SVM está formulado para problemas de clasificación binaria, en cuanto a los problemas multiclase normalmente se reducen a una serie de problemas binarios. Existen autores que realizan esta reducción usando una técnica denominada “*One vs All*” (o “*One vs Rest*”) en la cual se entrena una SVM para cada clase del problema – como lo menciona Band (2020) – comparando cada clase con las demás. Luego, se utiliza la clase con la mayor probabilidad de pertenencia para clasificar una nueva instancia. Otros autores – como destaca Brownlee (2021) – reducen el problema mediante una técnica “*One vs One*” en la cual se entrena una SVM para cada par de clases, y se elige la clase con la mayor cantidad de victorias.

Cabe destacar que las SVM pertenecen a una clase de algoritmos del Machine Learning que son denominados “Métodos Kernel”. Estos algoritmos pueden utilizar una función de kernel para transformar las características.

Como lo menciona el equipo de MathWorks (s.f.) “las funciones de kernel asignan los datos a un espacio dimensional diferente, que suele ser superior, con la expectativa de que resulte más fácil separar las clases después de esta transformación, simplificando potencialmente los límites de decisión complejos no lineales para hacerlos lineales en el espacio dimensional de características superior asignado”. Los datos no se tienen que transformar explícitamente durante este procedimiento, lo que supondría una alta carga computacional. A esto se le conoce como “*truco de kernel*”.

Existen algunos kernels muy conocidos con los que se pueden trabajar, el equipo desarrollador de MATLAB destaca dentro de las paqueterías de su software kernels como la función de base radial o gaussiana (véase la ecuación 7.3.1), la función lineal (véase la ecuación 7.3.2), la función polinómica (véase la ecuación 7.3.3) o la función sigmoide (véase la ecuación 7.3.4).

Ecuación 7.3.1. Función de base radial o gaussiana (The MathWorks Inc., s.f.).

Ecuación 7.3.2. Función lineal (The MathWorks Inc., s.f.).

Ecuación 7.3.3. Función polinómica (The MathWorks Inc., s.f.).

Ecuación 7.3.4. Función sigmoide (The MathWorks Inc., s.f.).

Un entendimiento visual que es más claro de cómo se realiza la separación del hiperplano puede observarse en la gráfica 7.3.1.

El entrenamiento de una SVM se muy similar a resolver un problema de optimización cuadrática para ajustar un hiperplano que minimice el margen flexible entre las clases. El número de características está determinado por el número de vectores de soporte.

Gráfico

Descripción generada automáticamente con confianza baja

Gráfica 7.3.1. Gráficas de separación del hiperplano de las distintos kernels (Marius, 2020).

El uso de las SVM trae consigo ciertas ventajas, como las menciona Blog Unipython (2018):

* Es eficaz en espacios de grandes dimensiones.
* Mantiene eficacia en casos donde el número de dimensiones es mayor que el número de muestras.
* Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamada vectores de soporte), por lo que también es eficiente en memoria.
* Es versátil dado que se pueden especificar distintas funciones del núcleo para la función de decisión. Se proporcionan kernels comunes pero también es posible especificar kernels personalizados.

Algunos puntos clave de las SVM son los siguientes según MathWorks (s.f.):

* Son muy populares y logran un buen rendimiento en muchas tareas de clasificación y regresión.
* Aunque los algoritmos SVM están formulados para la clasificación binaria, los algoritmos SVM multiclase se construyen combinando varios clasificadores binarios.
* Los kernels hacen que los SVM sean más flexibles y capaces de gestionar problemas no lineales.
* Para construir la superficie de decisión, sólo se requieren los vectores de soporte seleccionados a partir de los datos de entrenamiento. Una vez terminado el entrenamiento, el resto de datos de entrenamiento es irrelevante, produciendo una representación compacta del modelo que es adecuada para generar código de forma automatizada.

Hay que tomar en cuenta dos factores importantes “si el número de características es mucho mayor que el número de muestras evite el exceso de ajuste al elegir las funciones del Kernel y el término de regularización es crucial” (Blog Unipython, 2018). Estos mismos autores también mencionan que los SVMs no proporcionan directamente estimaciones de probabilidad, éstas se calculan utilizando una validación cruzada quíntuple.

**7.3.4.2. La técnica Random Forest**

De la siguiente técnica que se hablará será la técnica de Bosques Aleatorios, o en inglés “Random Forest”. Esta técnica es muy popular en el Machine Learning y el análisis de datos.

La técnica se sustenta en los llamados “Árboles de Decisión”; éstos “ayudan al científico de datos a tomar una decisión gracias a una serie de preguntas (también llamadas tests) cuya respuesta (sí o no) llevará a la decisión final” (Data Scientest, 2022). Pueden encontrarse ejemplos de estos algoritmos basándose en temas de Economía gracias a las facilidades que en este campo han encontrado al utilizar este algoritmo. Un ejemplo es el que brinda Sanabria Castro (2020) el cual se ilustra a continuación en la figura 7.3.5

Diagrama

Descripción generada automáticamente

Figura 7.3.5. Ejemplo de Árbol de Decisión aplicado en la Economía (Sanabria Castro, 2020).

Como breve contextualización de los Árboles de Decisión, Sanabria Castro menciona que este tipo de modelos están formados por “una serie de decisiones lógicas similares a las de un diagrama de flujo. El árbol comienza con un nodo llamado ‘nodo raíz’, en este y en los nodos internos se toman las decisiones basado en diferentes atributos, las ramas indican las decisiones tomadas. Al final del árbol de decisión se encuentran los nodos terminales, que representan el resultado de seguir una combinación de decisiones. Alternativamente, el nodo terminal puede tener asociada una probabilidad de que el valor meta tome cierto valor” (Sanabria Castro, 2020).

Gracias a la construcción del árbol, estos modelos pueden utilizarse para la predicción de variables categóricas y continuas, recibiendo el nombre de árboles clasificación y regresión respectivamente.

La figura 7.3.5 es parte de un ejemplo que Sanabria Castro utiliza para visualizar la idea de un Árbol de Decisión mediante un ejemplo de un banco el cual utiliza un árbol de decisión para decidir si le debería ofrecer un préstamo a una persona.

Sanabria Castro (2020) menciona que “para construir un árbol de decisión es un método heurístico llamado partición recursiva. La idea es que conforme se avance en el árbol de decisión se formen grupos cada vez más uniformes, en donde predomine una clase o un valor. En el ejemplo anterior estos grupos son; personas a las que fue acertado, ofrecerles un préstamo y las que no”. Esta es la forma en la que el algoritmo encuentra patrones que son utilizados para las predicciones que se realizarán con aplicantes nuevos.

La construcción de uno se realiza mediante métodos como el ID3, el C4.5, C5.0 y el CART, sin embargo la explicación en el artículo de Sanabria Castro posee a profundidad los detalles minuciosos de estos algoritmos. La cuestión importante para efectos de esta investigación es conocer que el método está cimentado en encontrar los criterios de separación (o “Splitting Criteria”) necesarios con los que se construye el árbol.

Esta técnica presenta muchas ventajas en comparación con otros algoritmos de datos pues es fácil de interpretar, estable y generalmente regresa buenas coincidencias muy útiles en tareas de regresión o clasificación.

Enlazando la base de los Árboles de Decisión se obtiene el método de Random Forest, pues este método es considerado como un método de conjunto (o Ensemble Method) pues combina resultados con el objetivo de obtener un superresultado final que fue generado de los resultados de los diferentes Árboles de Decisión que componen al algoritmo de Bosque Aleatorio.

Un modelo de Bosque Aleatorio puede “constar de varias decenas, incluso centenas de árboles, el número de árboles es un parámetro que por lo general, se ajusta mediante validación cruzada (o cross-validation, en inglés). Para abreviar, la validación cruzada es una técnica de evaluación de un algoritmo de Machine Learning que consiste en entrenar y probar el modelo en fragmentos de la serie de datos de partida” (Data Scientest, 2022).

El sustento del algoritmo está en entrenar a cada árbol en un subconjunto de la serie de datos y así obtener un resultado. “Posteriormente, se combinan los resultados de todos los árboles de decisión para dar una respuesta final. Cada árbol “vota” (sí o no) y la respuesta final es la que tenga la mayoría de votos” (Data Scientest, 2022).

Esto anteriormente descrito es a lo que en el artículo de Data Scientest (2022) le denominan como “bagging” y otras fuentes como el artículo en El Estadístico Blogspot (2021) coinciden en el uso de este método, el cual es un procedimiento constituido por los siguientes pasos:

1. Se divide la serie de datos en varios subconjuntos compuestos aleatoriamente de muestras, de ahí el “random” de Random Forest.
2. Se entrena un modelo en cada subconjunto: habrá tantos modelos como subconjuntos.
3. Se combinan todos los resultados de los modelos (con un sistema de voto, por ejemplo) lo que proporciona un resultado final.

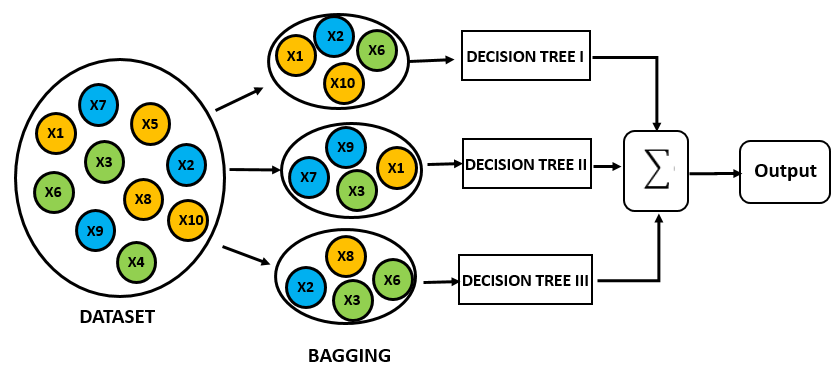


Figura 7.3.6. Visualización del “bagging” como separación, usado en Random Forest (Orellana Alvear, 2018).

La descripción visual del “bagging” puede observarse en la figura 7.3.6. Existe otra técnica llamada “boosting” la cual consiste en una separación y evaluación secuencial, mientras que el “bagging” puede verse como una separación y evaluación en paralelo; esta ejemplificación se observa en la figura 7.3.7 pero no se le profundizará mucho más en este método, pues no es de relevancia para el presente estudio.

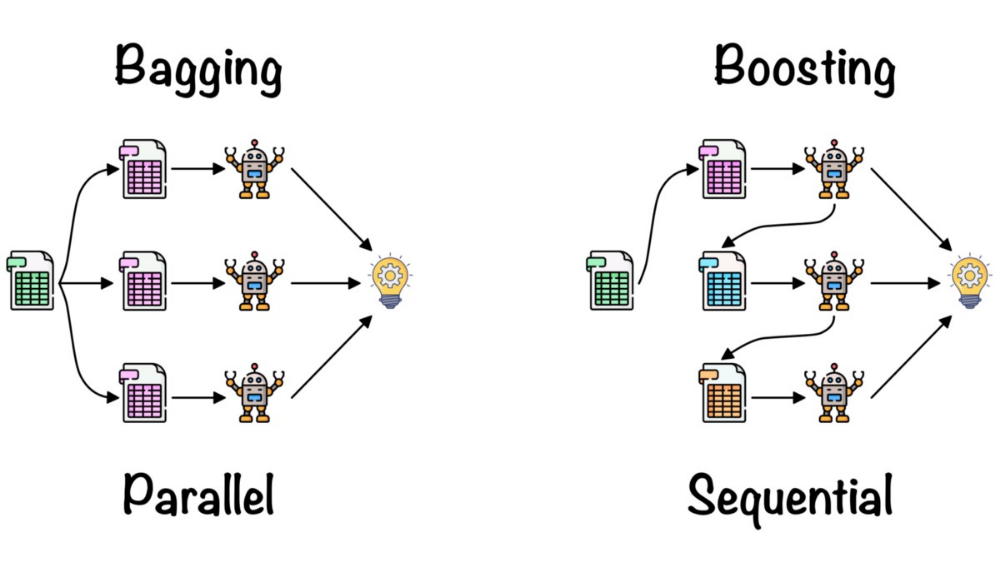


Figura 7.3.7. Comparación visual entre el “bagging” y el “boosting” (Sruthi, 2022).

Matemáticamente, el funcionamiento del Random Forest se puede dividir en cuatro pasos según El Estadístico Blogspot (2021):

1. Seleccionamos *features* o características (columnas o variables) de las totales (siendo ) y se crea un árbol de decisión con esas características.
2. Se crean árboles variando siempre la cantidad de *features* o características. También podría variarse la cantidad de muestras que se pasan a esos árboles (esto es conocido como “*Bootstrap Sample*”).
3. Se toma cada uno de los árboles y se ejecutan realizando una misma clasificación. Se guarda el resultado de cada árbol obteniendo salidas.
4. Se calculan los votos obtenidos para cada clase seleccionada y se considerará a la más votada como la clasificación final de nuestro bosque.

**7.3.4.3. La técnica Naive Bayes**

a

**7.4. El pensamiento humano puede literalmente, transformar el mundo físico**

*Dan Brown: El Símbolo Perdido*

Ha llegado el momento, pues tras todo el marco teórico previo, es el momento de abordar el problema de la presente investigación. Los temas previos funcionaron para poner sobre la mesa todo conocimiento necesario para enfocarse ahora en el habla imaginada.

**8. Metodología**

Metodología.

**9. Desarrollo**

Desarrollo.

**10. Experimentación y Pruebas**

**Esas veces en las que estás muy cansado, no quieres exigirte pero igualmente lo haces… Ese es el sueño. No es el destino, es el trayecto**

*Kobe Bryant*

Todos los conceptos previos revisados han servido únicamente para llegar aquí: la experimentación del documento e investigar de qué manera puede usarse la diadema Emotiv Epoc+ para crear una interfaz cerebro – computadora útil tomando como puntos de comparación distintas redes neuronales en este problema.

Antes, es requerido que la instalación de la diadema y el software se haga de la manera adecuada. Para ello cabe mencionar que la presente investigación se realizó sobre el sistema operativo Windows 11, máquina a la que se le conectaría el modelo de la diadema ya mencionado. También se usará el lenguaje de programación Python en su versión 3.10 y otras instalaciones se cubrirán en el avance de configuración de la diadema con el equipo para recibir y transmitir los datos.

El proceso entero de instalación para tener preparada cada configuración de transmisión y preparación de datos es extenso y no es propio de los objetivos de la investigación, sino sólo es parte de los procedimientos necesarios para poder realizar correctamente la experimentación de esta investigación, por ello el proceso detallado está explicado en el anexo 1.

A grandes rasgos, es necesario tener instalado Python en una versión 3.7.0 o superior y descargar el repositorio de CyKit desde la liga <https://github.com/CymatiCorp/CyKit>. Asimismo debe de seguirse el tutorial detallado por CymatiCorp en la liga <https://github.com/CymatiCorp/CyKit/wiki/How-to-Stream-Data-to-OpenViBE>, en el cual será necesario descargar e instalar OpenViBE en una versión 2.7.2 o superior y crear un escenario como el ilustrado en la figura 10.1.

Diagrama

Descripción generada automáticamente

Figura 10.1. Escenario de OpenViBE Designer para la experimentación de esta investigación.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Figura 10.2. Botón para ciclar la ejecución.

Finalmente, hay que habilitar el ciclado del escenario con el botón señalado en la figura 10.2 y usar el siguiente código de Python para que automáticamente este código cicle el procedimiento y simule la transmisión de datos en tiempo real.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pyautogui  import time  # Aplicando delay para tener tiempo de acomodar el mouse sobre el botón  time.sleep(10)  x, y = pyautogui.position()  # Aplicando delay al código para correr el escenario y automáticamente se mueva a la posición indicada  time.sleep(10)  pyautogui.moveTo(x, y)  # El proceso se mantiene en ejecución mientras el mouse esté en su posición  while(pyautogui.position() == (x, y)):  pyautogui.click()  time.sleep(1) # El CSV se actualiza cada segundo |

Una vez planteada la estructura anterior se procedió a escribir el archivo de Python con el que se hace todo el tratamiento y procedimiento de la actual investigación. Este archivo se encuentra adjunto bajo el nombre functions.py así como puede encontrarse en el anexo 2 en el presente documento. A continuación se explican las cuestiones necesarias para el entendimiento del experimento.

La primera función fue una restructuración al código anteriormente mostrado, formalizándolo en inglés y dándole un formato propio y estándar al resto del código; esto con intenciones de mantener el orden también en el código.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #%% Detect the position of the mouse  def MousePosition(secs = 3):  # Sleeping the script so we have time to set the mouse on its position  time.sleep(secs)  # Getting the mouse position  print(pyautogui.position())  return |

El propósito de este código es parcialmente el mismo que el anterior, pues busca identificar la posición del mouse en la pantalla para posteriormente utilizar la librería pyautogui que controlará los clicks de la máquina basándose en las coordenadas dadas. Para conocer las coordenadas de estos puntos importantes se ejecuta este código pasando únicamente como parámetro un valor “secs” que será un valor entero, simbolizando los segundos de retraso entre la ejecución del comando y la realización del proceso. Esto con el motivo de que el usuario tenga tiempo de establecer el mouse en la posición en la que desea conocer las coordenadas. En una terminal de Python se ejecuta la función sin parámetros y por defecto el usuario dispondrá de tres segundos para posicionar el mouse.

|  |
| --- |
| MousePosition() |

En las posteriores funciones se muestran los resultados personales de la ejecución de esta función.

La función que complementa a ésta para la actualización del script sería RunUpdateStream(), la cual decidió separarse por la forma en la que se planteó (presentada en el anexo 3). Esta función quedó rápidamente depreciada personalmente, pues planteaba una aproximación de evaluación en vivo, sin embargo, se observó que podía ser altamente complicado realizar esto (de modo que en el sucesivo de este documento, se plantea como propuestas de mejora de la investigación).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | #%% Creating the dataset with the input data and the expected output  def createDataset(xMaximize, yMaximize, xPlay, yPlay, xWindow, yWindow, xStop, yStop, profileName, expectedResult, rangeTime):  # Check if profile path exists  if(not(os.path.exists('profiles/' + str(profileName)))):  os.makedirs('profiles/' + str(profileName))  time.sleep(0.2)  # Maximizing OpenViBE Designer  pyautogui.moveTo(xMaximize, yMaximize)  pyautogui.click()  time.sleep(1)    # Playing OpenViBE Designer  pyautogui.moveTo(xPlay, yPlay)  pyautogui.click()  time.sleep(rangeTime)    # Clicking over OpenViBE Designer  pyautogui.moveTo(xWindow, yWindow)  pyautogui.click()  time.sleep(0.5)    # Stopping OpenViBE Designer  pyautogui.moveTo(xStop, yStop)  pyautogui.click()  time.sleep(0.5)  # Loading data.csv  dataframe = pandas.read\_csv('data.csv')    # Reworking the dataframe  dataframe = dataframe.drop(labels='Time:256Hz', axis=1)  dataframe = dataframe.drop(labels='Epoch', axis=1)  dataframe = dataframe.drop(labels='Event Id', axis=1)  dataframe = dataframe.drop(labels='Event Date', axis=1)  dataframe = dataframe.drop(labels='Event Duration', axis=1)    # Add a column with the expected output  dataframe.insert(16, "Expected Output", [expectedResult] \* len(dataframe))    # Append (or create) the dataset of this selected profile  if(not(os.path.isfile('profiles/' + str(profileName) + '/dataset.csv'))):  file = open('profiles/' + str(profileName) + '/dataset.csv', "x")  file.close()  dataframe.to\_csv('profiles/' + str(profileName) + '/dataset.csv', mode='a', index=False, header=True)  else:  dataframe.to\_csv('profiles/' + str(profileName) + '/dataset.csv', mode='a', index=False, header=False)    # Remove the data.csv raw file  os.remove('data.csv')    return |

Así que se optó por un enfoque más estático, donde la transmisión de datos no sería en vivo, sino que, el usuario tendría la orden de mantener concentración en un tema específico mientras la información se registra en un archivo.

Para la realización automática de este proceso, se necesitaban varias posiciones: la posición donde se da click para maximizar OpenViBE, donde se inicia la transmisión de datos con la diadema Emotiv, donde se superpone esta ventana (debido a que, en ocasiones, el electroencefalograma se inicia por encima de los botones importantes) y la posición donde se le da en finalizar a la transmisión (planteamiento presentado en el anexo 4).

Como el algoritmo hace estos clicks automáticamente, también se define el tiempo que sucede entre el inicio y el final de la transmisión con la variable rangeTime la cual es definida por defecto para cada instancia de concentración en cinco segundos.

Como se mencionó anteriormente, al usuario se le pedirá que se concentre en cierto aspecto o tópico. Este se señalará en el algoritmo dentro de la variable expectedResult el cual se incluirá en el conjunto de datos que se genera como producto de la ejecución de esta función.

Los conjuntos de datos están separados en perfiles, para mantener limpia la información y realizar el entrenamiento con los conjuntos independientes por cada usuario; esto para evitar que el experimento falle por la distinta actividad cerebral entre los usuarios. El perfil se define en profileName. Este nombre será el que se le asigne en sistema a una carpeta donde se guardará el conjunto de datos y su respectiva información aislada.

Por último se ejecutarían las funciones de los tres modelos de Aprendizaje Supervisado únicamente indicando el perfil que se evaluará. Con este proceso, cada función arrojará un porcentaje de efectividad después de haber separado a la población en un conjunto de entrenamiento y un conjunto de prueba, produciendo una matriz de confusión en la que sería visible la efectividad del modelo. Estos resultados son presentados a continuación.

**11. Análisis e Interpretación de Resultados**

A continuación se muestran los datos obtenidos; estos se recopilaron de participantes a quienes previamente se les aseguró que la recopilación de información se manejaría con alta seguridad y prudencia (estipulado por escrito bajo un acuerdo de privacidad adjunto en el anexo 5 y entregado a cada uno de los participantes).

**12. Conclusiones**

Conclusión.

# **13. Referencias**

Afifi, A., & Bergman, R. (1998). *Neuroanatomía funcional.* México, D.F.: McGraw Hill.

Alpaydin, E. (2010). *Introduction to Machine Learning.* Cambridge: MA: MIT Press.

APD, R. (04 de 04 de 2019). *¿Cuáles son los tipos de algoritmos del machine learning?* Obtenido de APD: https://www.apd.es/algoritmos-del-machine-learning/

Band, A. (9 de Mayo de 2020). *Multi-class Classification — One-vs-All & One-vs-One*. Obtenido de Towards Data Science: https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

Bear, M., Connors, B., & Paradiso, M. (2002). *Neurociencia: explorando el cerebro.* Barcelona: Masson.

Betanzos Gómez, A. (13 de Marzo de 2020). *Principales algoritmos de Machine Learning*. Obtenido de LinkedIn: https://www.linkedin.com/pulse/principales-algoritmos-de-machine-learning-alejandro-betanzos-g%C3%B3mez/?originalSubdomain=es

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

Blog Unipython. (15 de Enero de 2018). *Support Vector Machines (SVM)*. Obtenido de Unipython: https://unipython.com/support-vector-machines-svm/

Brownlee, J. (2021 de Abril de 27). *One-vs-Rest and One-vs-One for Multi-Class Classification*. Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

Caldaria. (22 de Julio de 2020). *Curiosidades sobre el cerebro*. Obtenido de HDOSO Magazine: https://www.caldaria.es/curiosidades-cerebro/

Cayre, M., Malaterre, J., Scotto-Lomassese, S., Strambi, C., & Strambi, A. (2010). The common properties of neurogenesis in the adult brain: from invertebrates to vertebrates. *Comparative Biochemistry and Physiology Part B: Biochemistry and Molecular Biology*, 1-15.

Cromer, A. (1996). *Física para ciencias de la vida.* New York: Reverté.

Ebbing, D., & Gammon, S. (2010). *Química General.* México, D.F.: Cengage Learning.

Fisio, O. (27 de Septiembre de 2022). *¿Qué es soma o cuerpo neuronal?* Obtenido de Fisioterapia Online: https://www.fisioterapia-online.com/glosario/soma-o-cuerpo-neuronal

Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge: Cambridge University Press.

Flynn, K. (2013). The cytoskeleton and neurite initiation. *Bioarchitecture*, 86-109.

González Barrio, H., Calleja Ochoa, A., Gómez-Escudero, G., Rodríguez Ezquerro, A., & López de Lacalle Marcaide, L. (06 de 04 de 2021). *Los conceptos de Machine Learning y Deep Learning en la industria*. Obtenido de Interempresas: https://www.interempresas.net/MetalMecanica/Articulos/347471-Los-conceptos-de-Machine-Learning-y-Deep-Learning-en-la-industria.html

Gray, P. (2002). *Psychology.* New York: Worth Publishers.

Herculano-Houzel, S. (2009). The human brain in numbers: a linerarly scaled-up primate brain. *Hum Neurosci*.

Hill, R. (2006). *Fisiología Animal.* Bogotá: Médica Panamericana.

Hodgkin, A., & Huxley, A. (1939). Action Potentials Recorded from Inside a Nerve Fibre. *Nature*, 710-711.

Interpsiquis. (25 de Septiembre de 2022). *Lóbulo Límbico.* Obtenido de Congreso Virtual de Psiquiatría: https://psiquiatria.com/glosario/lobulo-limbico

Joyanes Aguilar, L. (2008). *Fundamentos de Programación.* Madrid: McGraw Hill.

Kandel, E., Schwartz, J., & Jessel, T. (2000). *Principles of Neural Science.* New York: McGraw Hill.

Kole, M., & Stuart, G. (2012). Signal processing in the axon initial segment. *Neuron*, 235-247.

Laguna, M. (17 de Marzo de 2022). *Lóbulo de la ínsula.* Obtenido de Kenhub: https://www.kenhub.com/es/library/anatomia-es/lobulo-de-la-insula

Latarjet, M., & Ruiz Liard, A. (2004). Encéfalo, Generalidades y Definición. *Anatomía Humana*, 168-169.

Martínez Heras, J. (29 de Septiembre de 2020). *¿Clasificación o Regresión?* Obtenido de IArtificial: https://www.iartificial.net/clasificacion-o-regresion/

MDurance. (30 de Septiembre de 2021). *Todo lo que debes saber sobre el potencial de acción*. Obtenido de MDurance: https://blog.mdurance.eu/academia/el-potencial-de-accion/

Merck & Co, I. (27 de Septiembre de 2022). *Estructura típica de una neurona*. Obtenido de Manual MSD: https://www.msdmanuals.com/es/hogar/multimedia/figure/estructura-t%C3%ADpica-de-una-neurona

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* Cambridge: MA: MIT Press.

Murre, J., & Sturdy, D. (1995). The connectivity of the brain: multi-level quantitative analysis. *Biological cybernetics*, 73.

Nazareno, J. (8 de Junio de 2020). *Fascinación con el cerebro y lo neuro*. Obtenido de Conexiones. Plataforma de Ciencias del Aprendizaje: https://thelearningsciences.com/fascinacion\_cerebro\_neuro/

Ojeda Sahagún, J. L., & Icardo de la Escalera, J. M. (2004). *Neuroanatomía humana.* Barcelona: Masson.

Olmo, M., Nave, A., & Nave, R. (5 de Octubre de 2022). *Action Potentials*. Obtenido de Hyperphysics: http://hyperphysics.phy-astr.gsu.edu/hbasees/Biology/actpot.html

Paniagua, R., Nistal, M., Sesma, P., Álvarez-Uría, M., Fraile, B., Anadón, R., & Sáez, F. (2002). *Citología e histología vegetal y animal.* España: McGraw Hill.

Pelvig, D., Pakkenberg, H., Stark, A., & Pakkenberg, B. (2008). Neocortical glial cell numbers in human brains. *Neurobiology of aging*, 11.

Randall, D., Burggren, W., & French, K. (1998). *Eckert Fisiología Animal.* Girona: McGraw Hill.

Roche, H. (2003). *Lexikon Medizin.* Alemania: Urban & Schwarzenberg. Obtenido de Urban & Schwarzenberg

Russell, S., & Norvig, P. (2004). *Inteligencia Artificial: Un Enfoque Moderno.* Madrid: Pearson.

Sabater, V. (3 de Agosto de 2020). *Lóbulos cerebrales: características y funciones*. Obtenido de La Mente Es Maravillosa: https://lamenteesmaravillosa.com/lobulos-cerebrales-caracteristicas-y-funciones/

Sáenz, J. (2013). *Cálculo Vectorial.* Venezuela: Hipotenusa.

Solé, R., & Manrubia, S. (1996). *Neurodinámica.* Barcelona: Edicions UPC.

The MathWorks Inc. (s.f.). *Support Vector Machine (SVM)*. Obtenido de MathWorks: https://la.mathworks.com/discovery/support-vector-machine.html

TIBCO Data Science. (12 de Septiembre de 2022). *¿Qué es el aprendizaje supervisado?* Obtenido de TIBCO: https://www.tibco.com/es/reference-center/what-is-supervised-learning

Triglia, A. (23 de Agosto de 2016). *Sistema límbico: la parte emocional del cerebro.* Obtenido de Psicología y Mente: https://psicologiaymente.com/neurociencias/sistema-limbico-cerebro

von Bartheld, C., Bahney, J., & Herculano-Houzel, S. (2016). The search for true numbers of neurons and clial cells in the human brain: A review of 150 years of cell counting. *The Journal of Comparative Neurology*, 18.

Wikipedia. (1 de Octubre de 2022). *Neurona*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Neurona

**14. Anexos**

**14.1. Anexo 1: Configuración para recibir datos desde la diadema hacia el equipo de cómputo usando la paquetería de software de Emotiv, OpenViBE, y la librería de Python: Cykit.**

Primeramente, dado que este proyecto está usando la diadema de la marca Emotiv, será necesario instalar el software de esta marca. La paquetería de programas puede encontrarse – a la fecha de redacción de esta investigación – en la siguiente liga la cual es la página de descargas del software. De la misma forma, en la figura 14.1.1.

<https://www.emotiv.com/emotiv-launcher/>



Figura 14.1.1. Sitio web de Emotiv.

En el sitio web anterior se descargará e instalará el paquete cuyo nombre es “Emotiv Launcher”. Durante el proceso de instalación se preguntará por el conjunto de programas a instalar, entre los cuales están “Emotiv Launcher”, “Emotiv BrainViz”, “EmotivPRO” y “EmotivBCI”. Como elección personal, se decidió instalar todo el conjunto de programas ofrecidos durante la instalación. Para poder usar los programas es necesario registrarse en la página de Emotiv y crear lo que la compañía llama un “Emotiv ID” que es una cuenta propia dentro de su sitio web. Esto puede hacerse desde la siguiente liga.

<https://www.emotiv.com/my-account/>

Asimismo, se muestra en la figura 14.1.2 el ejemplo visual del sitio web en el que hay que realizar el registro de la cuenta Emotiv para poder hacer uso del software de esta empresa.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 14.1.2. Registro para la cuenta Emotiv ID.

Una vez completada la instalación, el programa Emotiv Launcher será el encargado de detectar si una diadema Emotiv está conectada al equipo. Puede observarse en la figura 14.1.3 este programa antes de conectar una diadema. Pero para conectar la diadema, es necesario – al menos en el caso del modelo Epoc+ – encender la diadema y conectar el dispositivo USB Bluetooth que viene con el paquete de la diadema tal como se aprecia en la figura 14.1.4.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 14.1.3. Emotiv Launcher sin la detección de una diadema conectada al equipo.



Figura 14.1.4. Diadema Emotiv encendida y dispositivo USB conectado al equipo.

Una vez que la conexión se haya establecido correctamente, Emotiv Launcher mostrará la diadema que detecta y permitiéndole al usuario conectarse a ella desde la interfaz de su programa, tal y como se muestra en la figura 14.1.5.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura 14.1.5. Emotiv Launcher detectando una diadema conectada al equipo.

A continuación, es necesario que el usuario en cuestión acomode la diadema sobre su cabeza de la manera correcta, realizando un acomodo que permita tanto una gran calidad de contacto y de recepción de las señales EEG. La figura 14.1.6 muestra este procedimiento guiado por la interfaz de Emotiv Launcher.



Figura 14.1.6. Emotiv Launcher guiando al usuario en acomodar correctamente la diadema.

Una vez que esté correctamente configurada la diadema con el equipo puede apagarse para un posterior uso. Es necesario también instalar el repositorio de GitHub creado por CymatiCorp: CyKit el cual puede encontrarse en la siguiente liga:

<https://github.com/CymatiCorp/CyKit>

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 14.1.7. Sitio web de GitHub donde se encuentra el repositorio del proyecto de CyKit por CymatiCorp.

El sitio web puede observarse en la figura 14.1.7. Desde este sitio web puede accederse a la documentación para instalar este paquete y usarlo junto a una herramienta denominada OpenViBE para producir información a partir de la diadema. Las dos siguientes ligas corresponden a cada apartado de la documentación.

<https://github.com/CymatiCorp/CyKit/wiki/How-to-Install-CyKIT>

<https://github.com/CymatiCorp/CyKit/wiki/How-to-Stream-Data-to-OpenViBE>

Se tomará estrictamente lo que dicen ambas guías a continuación para poder avanzar con las instalaciones pertinentes.

Como, inicialmente se debe hacer, habrá que instalar primeramente la librería CyKit; para ello será necesario contar con una versión de Python, ya sea la 3.7.2 o superior. En el caso propio se trabajó con la versión 3.10.

Posteriormente hay que descargar el repositorio. Esto puede hacerse de varias formas, ya sea descargando el archivo ZIP con todo el contenido del repositorio o usando la tecnología Git – que será el caso propio – clonando el repositorio en el directorio del gusto personal usando el siguiente comando:

|  |
| --- |
| git clone https://github.com/CymatiCorp/CyKit.git |

A continuación, puede probarse si la instalación se realizó correctamente probando el programa. Para ello, en una consola que esté ubicada en la raíz del directorio en donde se descargó el repositorio se ejecutarían los siguientes comandos:

|  |
| --- |
| cd CyKit-master  cd Py3  python CyKIT.py |

Texto

Descripción generada automáticamente

Figura 14.1.8. Ejecución correcta de CyKIT.py usando Python.

En la figura 14.1.8 puede observarse la ejecución correcta del archivo CyKIT.py usando Python y por consecuencia, una correcta instalación de la paquetería. Ahora que ya está instalado el paquete de Python, para transmitir datos será necesario ejecutar el archivo similar a como anteriormente se realizó, pero deben de tomarse algunas consideraciones dependiendo de la diadema, frecuencia de transmisión, entre otras cosas, las cuales menciona CymatiCorp en la explicación de instalación y que puede apreciarse en la figura 14.1.9 y en la figura 14.1.10

Texto

Descripción generada automáticamente

Figura 14.1.9. Explicación de algunas variables de configuración para ejecutar CyKit.

Texto

Descripción generada automáticamente

Figura 14.1.10. Explicación de algunas variables de configuración para ejecutar CyKit.

En el caso propio, tomando en cuenta las configuraciones recomendadas por CymatiCorp, se eligió por ejecutar el siguiente comando:

|  |
| --- |
| python CyKIT.py 127.0.0.1 5151 6 openvibe+generic+nocounter+noheader+nobattery+ovdelay:100+float+ovsamples:004 |

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 14.1.11. CyKit ejecutándose bajo las configuraciones indicadas.

En la figura 14.1.11 puede observarse el correcto funcionamiento de CyKit al calibrarse con las opciones especificadas. Ahora, el siguiente paso será instalar OpenViBE, el cual se descarga desde la siguiente liga:

<http://openvibe.inria.fr/downloads/>

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Figura 14.1.12. Sitio web de OpenViBE.

La versión que CymatiCorp recomienda para el trabajo presente es usar de la versión 2.2.0 en adelante. En el caso personal, se usará la versión 3.3.1 de 64 bits. De cumplir con este requisito, lo siguiente es ejecutar los programas “OpenViBE (Acquisition Server)” y “OpenViBE (Designer)”. La instalación no genera un acceso directo a estos programas, por lo que habrá que abrirlos en su mismo directorio o usando atajos que apunten a este directorio, según afirma CymatiCorp como algo necesario.

En OpenViBE (Acquisition Server) deberá seleccionarse “Generic Raw Telnet Reader” como Driver. El puerto de conexión (Connection port) es un puerto local al cual se conectará OpenViBE Designer. Este se deja por facilidad en 1024. Así también, en “Sample count per sent block” se dejará en 4. La configuración anterior puede observarse en la figura 14.1.13.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 14.1.13. Configuración general de OpenViBE Acquisition Server.

Lo siguiente es modificar las opciones de propiedades del driver del botón “Driver Properties”. La configuración aparece en la figura 14.1.14 pero también se enumera a continuación la utilizada:

* **Number of channels:** 14.
* **Sampling frequency:** 256.
* **Telnet host name:** localhost.
* **Telnet host port:** 5151.
* **Endianess:** Big Endian.
* **Sample type:** 32 bits float.
* **Skip at start (bytes):** 0.
* **Skip header (bytes):** 0.
* **Skip footer (bytes):** 0.

Después de haber hecho estas configuraciones, se dará click en el botón de “Apply” y en OpenViBE Acquisition Server podrá dársele click en el botón “Connect” con CyKit.py ya ejecutándose para este punto del avance experimental. La conexión exitosa debe aparecer tanto en CyKit como en OpenViBE Acquisition Server como se muestra en la figura 14.1.15.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 14.1.14. Configuración de las propiedades del driver de OpenViBE Acquisition Server.



Figura 14.1.15. Conexión exitosa entre CyKit y OpenViBE Acquisition Server.

Lo siguiente que debe hacerse es, ahora utilizar OpenViBE Designer, con el que se creará un escenario a través de arrastrar y soltar funciones al lienzo que tiene el diseñador. Para crear el escenario que marca CymatiCorp, será necesario seguir las instrucciones señaladas a continuación:

1. Arrastrar “Acquisition Client” que se puede encontrar dentro del directorio “Acquisition and network IO” en la barra lateral derecha.
2. Arrastrar “Signal Display” y “Matrix Display” que se pueden encontrar dentro del directorio “Visualization” → “Basic” en la barra lateral derecha.
3. Posicionarse sobre la flecha rosa de “Acquisition Client” y arrastrarla a la flecha verde de “Matrix Display”.
4. Posicionarse sobre la flecha rosa de “Acquisition Client” y arrastrarla a la flecha rosa de “Signal Display”.
5. Dar doble click sobre “Acquisition Client” y reemplazar “${AcquisitionServer\_HostName}” con “localhost”. Así también, asegurarse que “Acquisition server port” tenga un valor de “1024” si es que así se calibró en OpenViBE Acquisition Server.
6. Finalmente aplicar los cambios.

Todo este conjunto de instrucciones puede encontrarse en el tutorial dado por CymatiCorp, liga dada previamente. Asimismo, el diagrama que se realizó con las anteriores instrucciones se vería reflejado de una forma similar a la ilustrada en la figura 14.1.16.

Diagrama

Descripción generada automáticamente

Figura 14.1.16. Ilustración del escenario creado en OpenViBE Designer.

Después de lo anterior, sólo queda darle al botón de ejecutar escenario y si todos los pasos se realizaron correctamente, un electroencefalograma y una tabla de valores deberían aparecer en pantalla, actualizando el estado de recepción de datos de la diadema; lo cual demuestra el funcionamiento correcto de la transmisión de datos. Esto se ilustra en las figuras 14.1.17 y 14.1.18.



Figura 14.1.17. Electroencefalograma de OpenViBE Designer en funcionamiento correcto.



Figura 14.1.18. Tabla de datos de OpenViBE Designer en funcionamiento correcto.

Hasta aquí llega la explicación de CymatiCorp para la transmisión de datos. Sin embargo, personalmente todavía no se sentía con mucha facilidad de hacer Machine Learning de esta forma de recopilar datos, por lo que se tuvieron que hacer algunos añadidos personales al aporte hecho por CymatiCorp.

Estos añadidos fueron dos. El primero está en el escenario realizado en OpenViBE Designer, pues al escenario se le agregará un proceso llamado “CSV File Writer” el cual puede encontrarse dentro del directorio “File reading and writing” → “CSV” en la barra lateral derecha. Después habrá que posicionarse sobre la flecha rosa de “Acquisition Client” y arrastrarla a la flecha rosa de “CSV File Writer”. Finalmente dando doble click sobre este último habrá que modificar el campo “Filename” al nombre y directorio de gusto propio. Así también debe marcarse como verdadera la casilla correspondiente a “Append data”. La figura 14.1.19 muestra el escenario modificado y la figura 14.1.20 muestra las configuraciones del proceso de “CSV File Writer”.

Diagrama

Descripción generada automáticamente

Figura 14.1.19. Escenario de OpenViBE Designer modificado personalmente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 14.1.20. Configuraciones para “CSV File Writer”.

Esto finalmente permite una salida de datos mejor tratables para el Machine Learning realizado en esta investigación. Puede observarse en la figura 14.1.21 la tabla en formato CSV que produjo este cambio, con la que ya puede hacer más fácil la lectura de datos para los modelos de redes neuronales que se usarían en la presente investigación.

Imagen que contiene Texto

Descripción generada automáticamente

Figura 14.1.21. Archivo CSV producido tras una ejecución de transmisión de datos desde la diadema.

Sin embargo, aún queda un problema más a resolver y es aquí donde está el segundo cambio notable; pues este archivo CSV se genera sólo hasta el final de la ejecución, es decir, cuando se detiene el proceso. Pero en una investigación y experimentación personal, se encontró una forma de simular una lectura a tiempo real para poder realizar el análisis a tiempo real de los datos transmitidos desde la diadema al equipo, casi en vivo.

Esta forma de simulación se deberá hacer primeramente activando el ciclado del procedimiento. Esto se hace pulsando el botón de ciclado que está cerca del botón de ejecución, como se puede ver en la figura 14.1.22.



Figura 14.1.22. Botón para ciclar la ejecución.

Ahora, al haber activado esta opción, cada vez que se pulse el botón de detener el proceso, sólo se reiniciará el procedimiento, por lo que de esta manera puede simularse la transmisión en vivo de los datos.

Por último, para poder hacer este proceso automático se proponen dos archivos de Python que faciliten este trabajo. Estos archivos de Python requerirán de la librería “PyAutoGUI”, la cual puede instalarse con el siguiente comando:

|  |
| --- |
| pip install pyautogui |

El propósito de estos archivos es que se pulse la detención cíclica del escenario periódicamente, lo cual hará la librería previamente instalada, pues la anterior permite controlar el teclado y el mouse de forma automática con código de Python.

Primeramente hay que encontrar las coordenadas del botón en la pantalla. Para ello se usaría el siguiente archivo Python:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | import pyautogui  import time  # Aplicando delay para tener tiempo de acomodar el mouse sobre el botón  time.sleep(10)  # Obteniendo la posición del mouse  print(pyautogui.position()) |

Ahora, cuando se tengan estos valores pueden guardarse en variables “x”, “y” que se pueden usar en el siguiente código, pues ahora, cuando todo el procedimiento de obtención de datos esté en ejecución: la diadema conectada, Emotiv en ejecución y OpenViBE transmitiendo datos, es momento de ejecutar el código siguiente para que la detención cíclica lo haga automáticamente este código propuesto:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import pyautogui  import time  x = 0  y = 0  # Aplicando delay al código para correr el escenario y automáticamente se mueva a la posición indicada  time.sleep(10)  pyautogui.moveTo(x, y)  # El proceso se mantiene en ejecución mientras el mouse esté en su posición  while(pyautogui.position() == (x, y)):  pyautogui.click()  time.sleep(1) # El CSV se actualiza cada segundo |

Una combinación de ambos en un único código sería el siguiente:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | import pyautogui  import time  # Aplicando delay para tener tiempo de acomodar el mouse sobre el botón  time.sleep(10)  x, y = pyautogui.position()  # Aplicando delay al código para correr el escenario y automáticamente se mueva a la posición indicada  time.sleep(10)  pyautogui.moveTo(x, y)  # El proceso se mantiene en ejecución mientras el mouse esté en su posición  while(pyautogui.position() == (x, y)):  pyautogui.click()  time.sleep(1) # El CSV se actualiza cada segundo |

Y así, al ejecutar este código, se podrá actualizar el archivo CSV cada segundo, o según cuánto quiera personalizarse en la línea 16. Con esto ya podría realizarse Machine Learning con los datos de forma más efectiva para la presente investigación.

Este mismo procedimiento se documentó a su vez de forma audiovisual y se presenta en el siguiente enlace:

<https://youtu.be/MuBZdcLJXbk>

**14.2. Anexo 2: Código del archivo de Python: functions.py.**

El archivo de Python functions.py está constituido de las distintas funciones que a lo largo del procedimiento de investigación se realizaron y se compilaron en este documento. El archivo en su totalidad es el siguiente:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188 | #%% Libraries  import os  import time  import random  import pyautogui  import pandas  import numpy as np  import matplotlib.pyplot as plt  from sklearn.model\_selection import train\_test\_split  from sklearn.svm import SVC  from sklearn.metrics import classification\_report  from sklearn.metrics import precision\_score  from sklearn.metrics import confusion\_matrix  from sklearn.preprocessing import StandardScaler  from sklearn.ensemble import RandomForestClassifier  from sklearn.naive\_bayes import GaussianNB  #%% Detect the position of the mouse  def MousePosition(secs = 3):  # Sleeping the script so we have time to set the mouse on its position  time.sleep(secs)  # Getting the mouse position  print(pyautogui.position())  return  #%% Run a live stream to update the dataset while streaming  def RunUpdateStream(xStop, yStop, xWindow, yWindow, timerRefresher):  # Sleeping the script so we have time to run the schema and then automatically moves to the indicated position  time.sleep(10)  pyautogui.moveTo(xWindow, yWindow)    # The process will be okay while the mouse keeps on the indicated positions  while((pyautogui.position() == (xStop, yStop)) or (pyautogui.position() == (xWindow, yWindow))):  # Clicking to move the EEG  pyautogui.moveTo(xWindow, yWindow)  pyautogui.click()  time.sleep(timerRefresher/2)    # Clicking to reload the CSV  pyautogui.moveTo(xStop, yStop)  pyautogui.click()  time.sleep(timerRefresher/2)    return  #%% Creating the dataset with the input data and the expected output  def createDataset(xMaximize, yMaximize, xPlay, yPlay, xWindow, yWindow, xStop, yStop, profileName, expectedResult, rangeTime):  # Check if profile path exists  if(not(os.path.exists('profiles/' + str(profileName)))):  os.makedirs('profiles/' + str(profileName))  time.sleep(0.2)  # Maximizing OpenViBE Designer  pyautogui.moveTo(xMaximize, yMaximize)  pyautogui.click()  time.sleep(1)    # Playing OpenViBE Designer  pyautogui.moveTo(xPlay, yPlay)  pyautogui.click()  time.sleep(rangeTime)    # Clicking over OpenViBE Designer  pyautogui.moveTo(xWindow, yWindow)  pyautogui.click()  time.sleep(0.5)    # Stopping OpenViBE Designer  pyautogui.moveTo(xStop, yStop)  pyautogui.click()  time.sleep(0.5)  # Loading data.csv  dataframe = pandas.read\_csv('data.csv')    # Reworking the dataframe  dataframe = dataframe.drop(labels='Time:256Hz', axis=1)  dataframe = dataframe.drop(labels='Epoch', axis=1)  dataframe = dataframe.drop(labels='Event Id', axis=1)  dataframe = dataframe.drop(labels='Event Date', axis=1)  dataframe = dataframe.drop(labels='Event Duration', axis=1)    # Add a column with the expected output  dataframe.insert(16, "Expected Output", [expectedResult] \* len(dataframe))    # Append (or create) the dataset of this selected profile  if(not(os.path.isfile('profiles/' + str(profileName) + '/dataset.csv'))):  file = open('profiles/' + str(profileName) + '/dataset.csv', "x")  file.close()  dataframe.to\_csv('profiles/' + str(profileName) + '/dataset.csv', mode='a', index=False, header=True)  else:  dataframe.to\_csv('profiles/' + str(profileName) + '/dataset.csv', mode='a', index=False, header=False)    # Remove the data.csv raw file  os.remove('data.csv')    return  #%% First AI model  def SVM(profileName):  # Condition if profile or dataset doesn't exists  if(not(os.path.isfile('profiles/' + str(profileName) + '/dataset.csv'))):  print('User not found or dataset not created yet')  return    # Support Vector Machine process  df = pandas.read\_csv('profiles/' + str(profileName) + '/dataset.csv')    X = df.drop(['Expected Output'], axis = 1)  Y = df['Expected Output']    xtrain, xtest, ytrain, ytest = train\_test\_split(X, Y, test\_size = 0.20)    classifier = SVC(kernel = 'linear')  classifier.fit(xtrain, ytrain)    ypred = classifier.predict(xtest)  print(ypred)    print(classification\_report(ytest, ypred))    return  #%% Second AI model  def NaiveBayes(profileName):  # Condition if profile or dataset doesn't exists  if(not(os.path.isfile('profiles/' + str(profileName) + '/dataset.csv'))):  print('User not found or dataset not created yet')  return    # Naive Bayes process  df = pandas.read\_csv('profiles/' + str(profileName) + '/dataset.csv')  X = df.drop(['Expected Output'], axis = 1)  Y = df['Expected Output']  xtrain, xtest, ytrain, ytest = train\_test\_split(X, Y, test\_size = 0.20)  classifier = GaussianNB()  classifier.fit(xtrain, ytrain)  ypred = classifier.predict(xtest)  matriz = confusion\_matrix(ytest, ypred)  print('Matriz de Confusión')  print(matriz)  precision = precision\_score(ytest, ypred, pos\_label="a")  print('Precisión del modelo')  print(precision)  return  #%% Third AI model  def RandomForest(profileName):  # Condition if profile or dataset doesn't exists  if(not(os.path.isfile('profiles/' + str(profileName) + '/dataset.csv'))):  print('User not found or dataset not created yet')  return    # Random Forest process  df = pandas.read\_csv('profiles/' + str(profileName) + '/dataset.csv')  X = df.drop(['Expected Output'], axis = 1)  Y = df['Expected Output']  xtrain, xtest, ytrain, ytest = train\_test\_split(X, Y, test\_size = 0.20)  sc = StandardScaler()  xtrain = sc.fit\_transform(xtrain)  xtest = sc.fit\_transform(xtest)  classifier = RandomForestClassifier(n\_estimators = 4)  classifier.fit(xtrain, ytrain)  ypred = classifier.predict(xtest)  matriz = confusion\_matrix(ytest, ypred)  print('Matriz de Confusión')  print(matriz)  precision = precision\_score(ytest, ypred, pos\_label="a")  print('Precisión del modelo')  print(precision)    return |

**14.3. Anexo 3: Pseudocódigo de la función RunUpdateStream() encontrada en el archivo de Python: functions.py.**

El acercamiento para nombrar las variables y revisar el procedimiento a modo de boceto fue el siguiente:

Las variables a utilizar serán las siguientes:

* xStop: La posición en x del botón de Stop.
* yStop: La posición en y del botón de Stop.
* xWindow: La posición en x en otro punto de la pantalla (para evitar que el EEG se interponga).
* yWindow: La posición en y en otro punto de la pantalla (para evitar que el EEG se interponga).
* timerRefresher: El tiempo que reposa el código que ocurre entre click y click para refrescar el data.csv.

El procedimiento será el siguiente:

1. Se asignan xStop y yStop a los valores que corresponden.
2. Se asignan xWindow y yWindow a los valores que corresponden.
3. Mientras la posición del mouse sea (xStop, yStop) o (xWindow, yWindow):
   1. El mouse hace click sobre la posición (xWindow, yWindow).
   2. El sistema hace un sleep de:
   3. El mouse hace click sobre la posición (xStop, yStop).
   4. El sistema hace un sleep de:

**14.4. Anexo 4: Pseudocódigo de la función createDataset() encontrada en el archivo de Python: functions.py.**

El acercamiento para nombrar las variables y revisar el procedimiento a modo de boceto fue el siguiente:

Las variables a usar serán:

* xMaximize: La coordenada en x de donde se dará click para maximizar OpenViBE Designer.
* yMaximize: La coordenada en y de donde se dará click para maximizar OpenViBE Designer.
* xPlay: La coordenada en x de donde se dará click para darle play a OpenViBE Designer.
* yPlay: La coordenada en y de donde se dará click para darle play a OpenViBE Designer.
* xWindow: La coordenada en x de donde se da click para superponer a OpenViBE Designer.
* yWindow: La coordenada en y de donde se da click para superponer a OpenViBE Designer.
* xStop: La coordenada en x de donde se dará click para darle stop a OpenViBE Designer.
* yStop: La coordenada en y de donde se dará click para darle stop a OpenViBE Designer.
* profileName: El nombre de perfil que se usará.
* expectedResult: El resultado que se espera en la ejecución de la construcción del dataset.
* rangeTime: El tiempo en segundos que se empleará en la ejecución de la función.

El procedimiento será el siguiente:

1. Se checa si existe en la carpeta “profiles” una carpeta con nombre “profileName”. Si no existe, se crea esta carpeta y esta se asigna como workspace, si existe directamente se asigna como workspace.
2. La función maximiza la ventana de OpenViBE Designer dándole click a (xMaximize, yMaximize).
3. Posteriormente la función da click en play en la ejecución del Designer en (xPlay, yPlay).
4. La función hará un sleep durante rangeTime recopilando los datos.
5. Al terminar este sleep, por precaución se da click en (xWindow, yWindow) y posteriormente a (xStop, yStop).
6. Python cargará el “data.csv” que se genera en el mismo directorio donde se encuentra el script para añadirle una columna extra “Expected Result” donde se añadiría la variable “expectedResult”.
7. Este DataFrame se añade al archivo “dataset.csv” que se ubique en la carpeta del nombre de perfil dentro de “profiles”. Si el archivo no existe, se crea, si sí existe, sólo se le hace un append al archivo ya existente.
8. Se elimina “data.csv”.

**14.5. Anexo 5: Acuerdo de privacidad entregado a los usuarios que colaboraron con el experimento.**

En la página siguiente se adjunta el acuerdo de privacidad elaborado para asegurar a los usuarios que colaboraron con el experimento, que sus datos serían tratados éticamente según los fines de esta investigación y los permisos cedidos por ellos mismos.

**ACUERDO DE PRIVACIDAD**

**1. Introducción**

Este acuerdo establece los términos y condiciones en los cuales los participantes (en adelante "Usuarios") en el estudio experimental "*Implementación de una Interfaz Cerebro – Computadora por medio de la Diadema Emotiv Epoc+ y la Librería de Python Cykit Utilizando Modelos de Aprendizaje Supervisado. Tesina de Joel Alejandro Espinoza Sánchez para Obtener el Grado de Ingeniero en Computación Inteligente*" (en adelante "Estudio") ceden el derecho de uso de los datos experimentales generados con su participación por los investigadores en el estudio (en adelante "Investigadores"). Al firmar este acuerdo, los Usuarios aceptan las condiciones establecidas en este documento.

**2. Objeto del acuerdo**

Los Usuarios ceden al equipo de investigación el derecho de usar los datos experimentales generados con su participación en el estudio. Los datos que se generarán para el estudio son generados a través de la diadema Emotiv Epoc+. La información utilizada será únicamente el comportamiento cerebral generado por los Usuarios que registre la diadema. Los datos podrán ser utilizados para la realización de investigaciones, publicaciones y presentaciones en eventos científicos.

**3. Tratamiento de los datos**

Los Investigadores se comprometen a proteger la privacidad de los Usuarios y a tratar los datos de manera confidencial. Los datos se almacenarán de manera segura y solo serán accesibles por el equipo de investigación. Los datos serán anonimizados antes de ser compartidos con terceros, y solo se compartirán con instituciones y organizaciones con fines científicos.

**4. Derechos de los Usuarios**

Los Usuarios tienen derecho a acceder, rectificar y suprimir sus datos en cualquier momento, siempre y cuando no interfiera con el desarrollo del Estudio. También tienen derecho a retirar su consentimiento en cualquier momento, siempre y cuando no afecte a la validez de los datos ya recolectados.

**5. Permisos cedidos por el Usuario**

El Usuario cede los permisos señalados a continuación como "Acepto" y restringe aquellos indicados como "No Acepto" señalizados y elegidos por el mismo Usuario:

1. Que se me incluya en el proceso experimental del Estudio proporcionando mis datos relacionados a la actividad cerebral, generados con la diadema Emotiv Epoc+.

Acepto: \_X\_ No Acepto: \_\_\_

1. Que se expongan como únicos datos personales en la redacción, publicación y presentación del Estudio y subsecuentes, mi nombre completo y edad.

Acepto: \_\_\_ No Acepto: \_\_\_

1. Que se puedan realizar grabaciones de audio y video incluyendo mi imagen únicamente como parte de la evidencia del proceso de realización del experimento del Estudio para el control de avances requerido por docentes y sinodales evaluadores del Estudio.

Acepto: \_\_\_ No Acepto: \_\_\_

1. Que el material audiovisual generado con mi imagen sea usado en la redacción, publicación y presentación del Estudio y subsecuentes (No aplica si no se acepta el inciso C).

Acepto: \_\_\_ No Acepto: \_\_\_

**6. Vigencia**

Este acuerdo tendrá vigencia hasta la finalización del estudio, y será rescindido una vez que se hayan cumplido todos los objetivos del mismo.

**7. Aceptación del acuerdo**

La firma del presente acuerdo por parte del Usuario implica la aceptación de las condiciones establecidas en este documento.

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Usuario  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Fecha | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Investigador  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Fecha |

**14.6. Anexo 6: ñe.**

ñe