TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

Estudio de la activación de señales cerebrales en distintas actividades con dispositivos wearables

Autor
Ximo Sanz Tornero

Directores
Francisco Manuel García Moreno, Ana Álvarez Muelas

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Julio de 2024

Estudio de la activación de señales cerebrales en distintas actividades con dispositivos wearables

Ximo Sanz Tornero

Palabras clave: EEG, Muse, ML, Human Activity Recognition

Resumen

El estudio realizado en este trabajo de fin de grado, (TFG), se centra en el uso de tecnologías de bajo coste para la medición y análisis de señales electroencefalográficas, (EEG), mediante técnicas de aprendizaje automático, (ML). El objetivo principal es evaluar la viabilidad de utilizar dispositivos económicos para distinguir entre diferentes tipos de actividades cerebrales, ampliando así las aplicaciones prácticas del ML en el análisis de datos biomédicos.

Para realizar el estudio, se ha desarrollado una base matemática con los principios teóricos necesarios para el análisis de resultados y la creación de modelos. Además, se han establecido las bases necesarias para el procesamiento y análisis de señales EEG, a través de un estudio exhaustivo del estado del arte. En consecuencia, se ha creado un dataset con datos recolectados de pacientes mientras realizan una variedad de actividades, y se ha desarrollado código capaz de preprocesar estos datos, extraer características, seleccionarlas, y evaluar distintos modelos de ML, mejorando sus parámetros en cada ejecución.

Los resultados obtenidos han sido especialmente prometedores en la distinción de la actividad de meditación en contraposición al resto. Es más, en clasificaciones binarias de actividades los modelos estudiados han ofrecido un rendimiento destacable. Sin embargo, en la distinción global de actividades los resultados son menos esperanzadores. Es necesario continuar la investigación en varias direcciones para esclarecer los hallazgos.

Este TFG pretende ofrecer una nueva perspectiva sobre cómo las tecnologías de bajo costo pueden ser utilizadas para llevar a cabo estudios innovadores. Además, proporciona un conjunto de datos utilizable para futuras investigaciones y el código necesario para el tratamiento de los mismos, complementado con un análisis exhaustivo en el ámbito del ML.

Study of brain signal activation in various activities using wearable devices

Ximo Sanz Tornero

Keywords: EEG, Muse, ML, Human Activity Recognition

Abstract

The conducted study focuses on the use of low-cost technologies for measuring and analyzing electroencephalographic, (EEG), signals using machine learning techniques, (ML). The primary objective is to evaluate the feasibility of using economical devices to distinguish between different types of brain activities, thereby expanding the practical applications of ML in biomedical data analysis.

In order to carry out the study, a mathematical foundation with the necessary theoretical principles for analysis and model creation has been developed. Additionally, the groundwork for processing and analyzing EEG signals has been established through an exhaustive review of the state of the art. Consequently, a dataset was created with data collected from patients performing various activities, and code was created to process this data, extract features, select them, and evaluate different ML models, improving their parameters.

The results obtained have been particularly promising in distinguishing the activity of meditation from the rest. Moreover, in binary classifications of activities, the models studied have shown incredible performance. However, in the overall distinction of activities, the results are less encouraging. It is necessary to continue research in several directions to clarify the findings.

This study aims to offer a new perspective on how low-cost technologies can be used to conduct innovative studies. Furthermore, it provides a dataset for future research and the necessary code for data processing, complemented by a comprehensive analysis in the field of ML.

Yo, **Ximo Sanz Tornero**, alumno de la titulación Doble Grado en Ingeniería Informática y Matemáticas en la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación** y en la **Facultad de Ciencias de la Universidad de Granada**, con DNI 20857490D, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Ximo Sanz Tornero

Granada a 15 de Julio de 2024.

D. **Francisco Manuel García Moreno**, Profesor del Área de Lenguajes Sistemas Informáticos del Departamento Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Dña. **Ana Álvarez Muelas**, Profesora del Departamento de Psicología Evolutiva y de la Educación de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Estudio de la activación de señales cerebrales en distintas actividades con dispositivos wearables***, ha sido realizado bajo su supervisión por **Ximo Sanz Tornero**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 15 de Julio de 2024.

Los directores:

Francisco Manuel García Moreno **Ana Álvarez Muelas**

Agradecimientos

A mis padres, por todo el apoyo proporcionado, no lo podría haber logrado de otra forma.
A mis amigos, por estar siempre ahí y por ofrecer su tiempo para que este sea un proceso más ameno y divertido.
En especial, con cariño, a mi madre, que siempre ha salido adelante incluso en los momentos más duros.

Índice general

Capítulo 1

Introducción

El avance incesante de la tecnología moderna ha jugado un papel determinante en la transformación de numerosos campos científicos y técnicos. En particular, el aprendizaje automático (ML, por sus siglas en inglés), ha revolucionado la capacidad de análisis y procesamiento al posibilitar el trabajo con conjuntos de datos de magnitudes gigantes. Este enfoque ha abierto las puertas a explorar y desarrollar aplicaciones prácticas que antes se consideraban inalcanzables, marcando un hito en la intersección de la informática y la investigación aplicada.

En este contexto, el presente trabajo de fin de grado, (TFG), se centra en el empleo de tecnologías de bajo coste para la medición de señales electroencefalográficas, (EEG), en varios sujetos. Mediante el uso de un dispositivo económico, se busca recolectar y analizar datos de EEG para intentar distinguir entre diferentes tipos de actividades cerebrales. Con esto se busca ampliar las aplicaciones prácticas del ML en el análisis de datos biomédicos. Este enfoque, espera aportar una nueva perspectiva sobre cómo las tecnologías emergentes pueden ser implementadas en la investigación científica a un coste reducido, democratizando así el acceso a herramientas avanzadas de análisis y estudio.

1.1. Objetivos

Este TFG tiene como finalidad el desarrollo y la aplicación de técnicas de aprendizaje automático para el análisis de señales EEG recogidas mediante un dispositivo de bajo coste. A continuación, se detallan los objetivos específicos que guiarán la investigación:

1. **Desarrollo de la teoría matemática necesaria:** Elaborar y profundizar en los fundamentos matemáticos y estadísticos que sustentan los métodos de procesamiento y análisis de señales EEG, asegurando una base sólida para la interpretación y manejo de los datos.

2. **Realización de un estudio del estado del arte:** Investigar y docu-

mentar los trabajos previos relacionados con el análisis de EEG mediante técnicas de aprendizaje automático, identificando las metodologías, herramientas y resultados más relevantes para fundamentar y orientar el desarrollo del presente estudio.

3. **Creación de un dataset comprensivo:** Generar un conjunto de datos a partir de 30 sujetos, a los cuales se les aplicarán distintos experimentos que serán repetidos para asegurar la consistencia y la robustez de los datos.

4. **Implementación de un código para el preprocesamiento de datos:** Desarrollar un software que permita la adecuada preparación de los datos, incluyendo la extracción y selección de características relevantes que faciliten posteriormente el entrenamiento de modelos de aprendizaje automático.

5. **Elección de modelos de ML adecuados al estudio:** Seleccionar y configurar modelos de aprendizaje automático que se ajusten óptimamente a las características del dataset y a los objetivos específicos del estudio, garantizando así la mayor eficacia en la distinción de actividades cerebrales.

6. **Análisis de resultados utilizando métricas específicas:** Evaluar el rendimiento de los modelos implementados utilizando una metodología de validación *leave-one-out*, (LOO), por sujeto, lo que permitirá una estimación realista de la capacidad del modelo para generalizar a nuevos individuos.

7. **Extracción de conclusiones relevantes:** Analizar los resultados obtenidos para responder a la pregunta central de la investigación sobre la viabilidad de distinguir entre diferentes tipos de actividad cerebral utilizando dispositivos EEG de bajo costo y técnicas de ML.

8. **Orientación para futuras investigaciones:** Proporcionar recomendaciones y directrices basadas en los hallazgos del estudio para guiar a futuros investigadores en el campo, destacando áreas potenciales para una exploración más profunda y el desarrollo de nuevas aplicaciones y mejoras tecnológicas.

1.2. Justificación

Esta investigación tiene una importancia notable, no solo por su contribución al avance tecnológico y científico, sino también por su impacto potencial en la vida cotidiana de muchas personas. Al utilizar dispositivos de bajo costo para la recolección de señales EEG, esta investigación propone hacer más accesible una tecnología que tradicionalmente ha estado limitada a contextos clínicos o laboratorios bien financiados. Esto podría abrir nuevas oportunidades para investigadores y clínicas en comunidades con recursos limitados, permitiéndoles explorar y contribuir al campo de la neurociencia.

Más allá de la accesibilidad, la aplicación de técnicas avanzadas de ML para analizar estos datos EEG puede proporcionar nuevas comprensiones sobre cómo funciona nuestro cerebro en diferentes estados cognitivos y emocionales. Este conocimiento tiene el potencial de transformar áreas como la educación, donde las técnicas adaptativas podrían personalizarse según las respuestas neuronales de los estudiantes, o la salud mental, donde diagnósticos y tratamientos podrían ser más precisos y personalizados. En concreto, se podría monitorizar en tiempo real a los estudiantes para saber qué asignaturas les suscitan mayor concentración o en cuales tienden a dispersarse con mayor facilidad. Además de poder estimar con certeza el tiempo máximo que un alumno puede estar concentrado.

El enfoque metodológico de este estudio, con su rigor en la creación de un dataset amplio y el uso de validación exhaustiva de modelos ML, busca garantizar que los resultados sean tanto confiables como aplicables. Este compromiso con la rigurosidad proporciona una base sólida para futuras investigaciones que deseen replicar o expandir los hallazgos.

En esencia, este proyecto no solo avanza el conocimiento científico, sino que también tiene el potencial de hacer la ciencia más inclusiva, extendiendo sus beneficios a una mayor variedad de personas y contextos. Al final, lo que se busca es que estos avances no se queden solo en el laboratorio, sino que lleguen a las comunidades y tengan un efecto real y positivo en la sociedad.

1.3. Estructura del documento

Este documento se organiza en varios capítulos que abordan de manera sistemática los distintos aspectos del trabajo realizado. A continuación, se presenta una breve descripción de cada uno de ellos para facilitar la comprensión de la estructura y el contenido del proyecto:

El **Capítulo 1 : Introducción** se establecen los fundamentos del trabajo, comenzando con los objetivos principales y específicos del proyecto. Se justifica la relevancia del estudio en el contexto actual.

En el **Capítulo 2: Metodología de desarrollo**, se discuten las metodologías disponibles, justificando la elección de la utilizada. Además, se detalla la planificación temporal y el presupuesto estimado para el proyecto, proporcionando una visión clara del cronograma y los costos asociados.

El **Capítulo 3: Estado del arte** ofrece un contexto teórico y un marco de referencia mediante la revisión de estudios previos relevantes. Se hace especial énfasis en áreas clave como el aprendizaje automático y la selección de características, proporcionando una base sólida para el desarrollo del proyecto.

En el **Capítulo 4: Fundamentos matemáticos**, se contextualizan los conceptos matemáticos necesarios. Se explican conceptos básicos y se describen fundamentos esenciales para comprender los aspectos técnicos del trabajo.

El **Capítulo 5: Fundamentos teóricos** se centra en la electroencefalografía y el aprendizaje automático. Se abordan los principios de la EEG, la generación de señales y los métodos de extracción y preprocesamiento. Además, se describen los métodos y modelos de *machine learning* utilizados, así como la

partición de conjuntos de datos.

En el **Capítulo 6: Análisis**, se enumeran y describen los requisitos funcionales y no funcionales que debe cumplir el sistema. Esto incluye aspectos de rendimiento y otras características clave del sistema.

El **Capítulo 7: Diseño e Implementación** detalla el procedimiento de experimentación y los casos de uso. Se describen las etapas de preprocesamiento de datos, normalización, extracción de características y entrenamiento de modelos. También se incluye la selección de hiperparámetros y la implementación de meta-modelos para optimizar los resultados.

El **Capítulo 8: Resultados** presenta los hallazgos del estudio. Se discuten los mejores modelos y resultados obtenidos en la clasificación multiclase, así como la optimización de estos resultados. Además, se exponen los resultados de la clasificación binaria en diferentes contextos, como la meditación y la música.

Finalmente, el **Capítulo 9: Conclusiones** sugiere posibles líneas de investigación y mejoras para futuros trabajos en este campo, proporcionando una perspectiva sobre el trabajo futuro y las oportunidades de desarrollo en esta área.

Capítulo 2

Metodología de desarrollo

2.1. Metodología

Una metodología de desarrollo se refiere al conjunto de prácticas, procesos y herramientas utilizadas para gestionar y llevar a cabo un proyecto de manera eficiente y efectiva. Define el enfoque general para planificar, ejecutar, controlar y entregar el trabajo en el contexto del desarrollo de software.

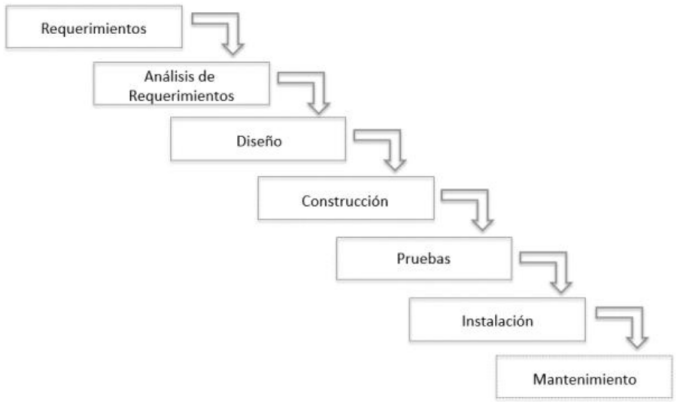


Figura 2.1: Etapas de un proceso de desarrollo [1]

La elección de una metodología adecuada determina cómo se abordarán los desafíos del proyecto, cómo se asignarán los recursos y cómo se gestionará el proceso de desarrollo en su totalidad [2] . Una metodología inapropiada puede llevar a retrasos, costos adicionales y, en última instancia, a la insatisfacción del cliente o a la no consecución de los objetivos del proyecto.

Por lo tanto, es esencial seleccionar una metodología que se ajuste a las características del proyecto, las necesidades del equipo y las expectativas del

cliente. Una metodología bien elegida proporciona un marco de trabajo claro, promueve la colaboración y la comunicación efectiva, y facilita la adaptación a los cambios en los requisitos del proyecto.

2.1.1. Metodologías disponibles

Podemos diferenciar entre dos vertientes de metodologías de programción.

Figura 2.2: Comparativa entre ágiles y tradicionales [1]

La tradicional, que suele seguir un desarrollo linear con fases claramente diferenciadas y preestablecidas. Esta es óptima cuando el cliente tiene una idea clara del producto final y no va a necesitar cambios. La ágil, que permite a los programadores adaptarse a los cambios difuminándose las líneas entre las fases de desarrollo, las cuales estaban tan marcadas en la tradicional [3].Podemos encontrar las siguientes metodologías, vease también la tabla 2.3:

- Tradicional
  - Cascada
  - Prototipado
  - Espiral
  - Incremental
  - Diseño rápido de aplicaciones
- Ágil
  - Kanban
  - Scrum
  - Lean

- Programación extrema (XP)

Se abordarán a continuación las características distintivas de las metodologías de desarrollo, proporcionando una visión detallada de sus enfoques y prácticas. Se sugiere hacer referencia a las tablas 2.1 y 2.2 para obtener una rápida comparación entre ellas. En primer lugar, se encuentran las metodologías tradicionales, que establecen una serie de pasos secuenciales para guiar el proceso de desarrollo.

Tradicionales

La metodología de programación en cascada, por ejemplo, requiere una planificación exhaustiva para dividir el proyecto en etapas bien definidas, donde cada fase se completa antes de pasar a la siguiente. Este enfoque proporciona una estructura clara y permite una gestión eficiente del proyecto, aunque puede ser menos adaptable a cambios en los requisitos.

En contraste, el enfoque de prototipado se orienta hacia la iteración rápida y continua con el cliente. Se desarrollan prototipos funcionales que se presentan al cliente para obtener retroalimentación temprana, lo que facilita la adaptación a las necesidades cambiantes y garantiza que el producto final satisfazca las expectativas del cliente. Sin embargo, este método puede requerir una inversión adicional en el desarrollo de múltiples prototipos.

Continuando con las metodologías tradicionales, la metodología en espiral combina elementos de la cascada con un enfoque iterativo. Se lleva a cabo un proyecto inicial utilizando el modelo en cascada, pero después de cada fase se realiza una evaluación y se decide qué funcionalidades agregar en la siguiente iteración, repitiendo este ciclo hasta alcanzar el producto final. Este enfoque permite una mayor flexibilidad y adaptación a medida que se obtiene más información sobre el proyecto.

La metodología incremental, por otro lado, se centra en la entrega de un producto funcional en etapas sucesivas. Se desarrolla una versión inicial del producto con las funcionalidades básicas y se van añadiendo nuevas características en pequeños incrementos, lo que permite una rápida implementación y una mayor satisfacción del cliente al obtener beneficios tangibles de manera temprana.

Por último, la metodología de diseño rápido de aplicaciones, (RAD), prioriza la rapidez y la usabilidad. Se enfoca en el desarrollo rápido de prototipos funcionales sin una planificación detallada, permitiendo una rápida iteración y adaptación a medida que se recibe feedback del usuario. Aunque este enfoque puede resultar en entregas más rápidas, puede haber un compromiso en la calidad y la documentación del proyecto.

Cuadro 2.1: Resumen y características clave de las metodologías de desarrollo tradicionales

Metodología	Fortalezas	Debilidades
Cascada	Fácil de entender y gestionar. Claramente definido en términos de objetivos. Documentación detallada. Funciona bien para proyectos con requerimientos estables.	Poca flexibilidad para cambios. Detección tardía de problemas. Riesgo de incompatibilidad con cambios. Entrega tardía y sobrecostos.
Prototipado	Retroalimentación temprana del usuario. Mejora en la precisión de los requisitos. Detección temprana de errores. Aumenta la satisfacción del usuario.	Riesgo de desviarse del objetivo. Incremento en la complejidad y el costo. Dependencia del prototipo. Expectativas poco realistas del producto.
Espiral	Combina ventajas de cascada y prototipado. Enfatiza la evaluación de riesgos. Permite cambios iterativos. Adecuado para proyectos grandes.	Más costoso que otros métodos. Requiere experiencia en riesgos. Proceso complejo y difícil de seguir. No ideal para proyectos pequeños.
Incremental	Entregas parciales desde etapas tempranas. Facilita retroalimentación y ajustes. Reduce tiempo de espera para el cliente. Identificación temprana de problemas.	Necesita buena planificación inicial. Carga de mantenimiento alta. Riesgo de desviación en incrementos. Dependencia de fases iniciales claras.
Diseño rápido de aplicaciones	Desarrollo y entrega acelerados. Flexibilidad para cambios de requisitos. Fomenta la colaboración. Ideal para tiempos de entrega cortos.	No adecuado para proyectos a gran escala. Requiere compromiso constante del cliente. Puede comprometer la calidad. Falta de documentación adecuada.

Ágiles

Por otro lado se hallan las metodologías ágiles [4]. Kanban se distingue por su enfoque en la gestión visual de la carga de trabajo a través de un tablero, donde las tareas se mueven de una etapa a otra, lo que garantiza un flujo constante y una mayor transparencia en el proceso de desarrollo. Este método promueve la eficiencia al identificar y resolver cuellos de botella de manera proactiva.

Por otro lado, scrum se asemeja a kanban en su enfoque en la entrega continua, pero se distingue por su estructura de sprints. Estos sprints, que generalmente tienen una duración de 2 a 4 semanas, permiten que el equipo se concentre intensamente en características específicas del producto, lo que resulta en entregas incrementales y regulares de productos funcionales al final de cada sprint.

La metodología lean, por su parte, se enfoca en la eliminación de desperdicios

y la optimización de procesos para maximizar la productividad. Al minimizar las fases y los tiempos de espera, lean busca entregar valor de manera rápida y eficiente, adaptándose continuamente a los cambios en los requisitos del cliente y basándose en el feedback para realizar ajustes.

Finalmente, la programación extrema, (XP), se caracteriza por su enfoque en la flexibilidad y la adaptabilidad. XP prioriza la comunicación efectiva y la colaboración entre los miembros del equipo, así como la retroalimentación constante de los usuarios finales. Esta metodología promueve prácticas de desarrollo de software como la integración continua, las pruebas automatizadas y la programación en parejas, lo que resulta en una mayor calidad del producto y una mayor satisfacción del cliente en entornos donde los requisitos son cambiantes o poco definidos.

Cuadro 2.2: Resumen y características clave de las metodologías de desarrollo ágil

Metodología	**Fortalezas**	**Debilidades**
Kanban	Manejo flexible de cambios. Transparencia y visibilidad del trabajo. Entrega continua. Reduce tiempo de ciclo.	Requiere disciplina del equipo. Riesgo de sobrecarga de trabajo. Menos predictivo a largo plazo. Depende de la colaboración del equipo.
Scrum	Fomenta colaboración y comunicación. Entrega rápida y regular de productos. Adaptación a cambios rápidos. Mejora transparencia del proyecto.	Desafíos en equipos no comprometidos. Riesgo de desviación sin Scrum Master eficaz. No ideal para requisitos poco claros. Requiere reuniones frecuentes.
Lean	Elimina desperdicio y optimiza recursos. Enfatiza eficiencia y mejora continua. Fomenta responsabilidad del equipo. Aumenta velocidad y calidad.	Necesita cultura organizacional fuerte. Difícil en equipos grandes. Riesgo de sobrecarga de trabajo. Enfoque menos estructurado desafiante.
Programación extrema	Mejora calidad del software y satisfacción. Comunicación y trabajo en equipo. Adaptabilidad a cambios de requisitos. Pruebas continuas.	Compromiso total del equipo necesario. Difícil de escalar en equipos grandes. Dependencia de comunicación constante. Alto nivel de habilidades técnicas.

Cuadro 2.3: Resumen y características clave de las principales metodologías de desarrollo

Tipo	**Metodología**	**Característica Clave**
Tradicional	Cascada	Secuencial
	Prototipado	Prototipos iterativos
	Espiral	Iterativo + Cascada
	Incremental	Funcionalidades añadidas
	Diseño rápido, (RAD)	Rapidez sin planificación concreta
Ágil	Kanban	Gestión visual de tareas
	Scrum	Sprints de 2 a 4 semanas
	Lean	Optimización y ajustes rápidos
	Programación extrema, (XP)	Flexibilidad y prioridad a feedback

2.1.2. Elección de la metodología

La elección de la metodología de desarrollo juega un papel importante en el éxito de cualquier proyecto de software. Después de una evaluación exhaustiva de las distintas metodologías disponibles, se decidió utilizar la programación extrema [5] para este TFG. La elección de XP se basó en las siguientes razones:

1. Flexibilidad: XP es conocida por su capacidad de adaptarse a cambios imprevistos en los requisitos del proyecto. Dado que este TFG implica una interacción constante con el tutor y posibles cambios en los objetivos, esta flexibilidad es esencial.

2. Colaboración constante: XP promueve la comunicación continua entre los desarrolladores y otras partes interesadas. Esta característica es esencial en un TFG, donde el feedback del tutor es fundamental para el avance del proyecto.

3. Entregas frecuentes: XP favorece las entregas pequeñas y frecuentes de software funcional, lo que permite una revisión constante y adaptaciones según sea necesario.

4. Aptitud para proyectos de tamaño medio: Dado el alcance y la duración de un TFG, XP es una metodología adecuada que permite dividir el trabajo en iteraciones manejables y proporciona herramientas para enfrentar desafíos específicos de un proyecto de esta magnitud.

Figura 2.3: XP

Si bien otras metodologías ágiles también ofrecen características similares, XP fue elegida específicamente por la combinación de los factores mencionados y su énfasis en la calidad del código y las buenas prácticas de programación.

2.2. Planificación realista

En esta sección se describen las fases del proyecto. La metodología seguida para este proyecto ha sido ágil, permitiendo una planificación flexible y adaptada a las circunstancias y avances del proyecto. En la siguiente imágen 2.4 puede encontrar la información resumida de los párrafos siguientes.

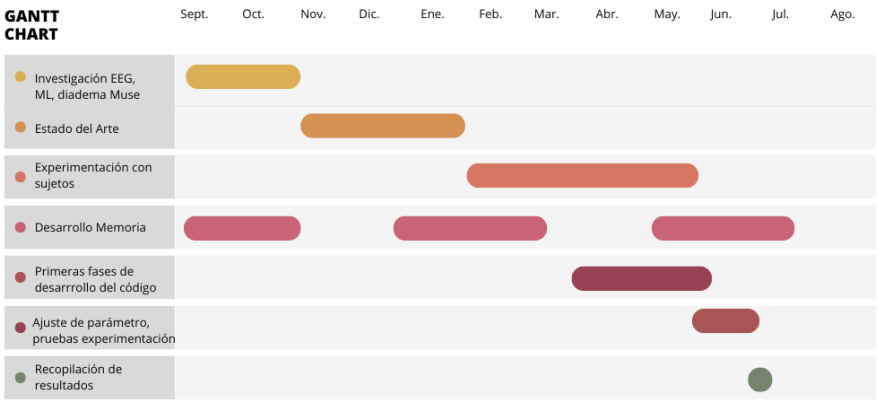


Figura 2.4: Fases en el desarrollo del proyecto

Fase 1: Investigación y preparación La primera fase del proyecto se centró en la investigación y preparación. Desde septiembre hasta mediados de noviembre, el enfoque principal fue investigar sobre la diadema MUSE, teoría sobre el EEG, los modelos de aprendizaje automático y las bibliotecas de python necesarias para el desarrollo del proyecto.

Fase 2: Revisión bibliográfica y estado del arte De noviembre a finales de enero, el trabajo se centró en la búsqueda y revisión de artículos de investigación relacionados con la distinción de actividades humanas. Durante este periodo, se desarrolló un estado del arte que permitió conocer los procedimientos más comunes, las prácticas estándar, las características de las señales a extraer, así como los modelos más utilizados y con mejor rendimiento en la literatura. Esta revisión bibliográfica fue esencial para orientar las decisiones metodológicas y técnicas del proyecto.

Fase 3: Experimentación y recolección de datos Desde febrero hasta principios de junio, se llevó a cabo la fase de experimentación. Se realizaron pruebas con 30 participantes, asegurando que se cumpliera el procedimiento de experimentación de manera rigurosa y que todas las pruebas se ejecutaran correctamente. Más información la puede encontrar en 7.1.1.

Fase 4: Desarrollo inicial del código y optimización En paralelo a la fase de experimentación, desde principios de abril hasta principios de junio, se inició el desarrollo del código en sus primeras fases. Durante este periodo, se realizaron pruebas preliminares con los pocos sujetos disponibles para optimizar el código y preparar el desarrollo final. Este enfoque iterativo permitió identificar y resolver problemas tempranos en el desarrollo del software, todo esto con el fin de tener un código funcional al realizar la totalidad de los experimentos.

Fase 5: Refinamiento del código y análisis de resultados Con todos los experimentos completados en junio, la atención se centró en refinar el código, seleccionando correctamente los modelos y métricas necesarias. Se realizó un análisis exhaustivo para obtener los mejores resultados posibles para cada uno de los modelos. A finales de julio, se recopilaron y compararon los resultados de todos los modelos probados, proporcionando una visión clara de su rendimiento y efectividad.

Fase 6: Desarrollo de la memoria El desarrollo de la memoria del proyecto fue un proceso continuo a lo largo de los diez meses, con una atención especial en los periodos de octubre, de enero a mediados de marzo y desde mayo hasta la entrega final en julio. Este enfoque constante en la documentación garantizó que cada etapa del proyecto estuviera bien documentada y que los resultados y conclusiones estuvieran claramente expuestos.

En resumen, la planificación ágil permitió una adaptación continua a los desafíos y descubrimientos a lo largo del proyecto.

2.3. Presupuesto

Durante el desarrollo de este proyecto, se ha distribuido el tiempo de forma eficiente para intentar abarcar en la mejor medidad de lo posible el problema en cuestión. No solo había que insvestigar en un nuevo ámbito y conseguir información actualizada y contrastada sino que había que experimentar y redactar una memoria a la altura. La recolección de muestras de los distintos sujetos, ha requerido de un tiempo considerable, alrededor de 50 horas. Esta etapa no solo ha sido esencial para asegurar la calidad de los datos, sino que también ha permitido establecer una base sólida para los análisis posteriores.

En los últimos 10 meses, he desempeñado un rol científico de datos. He dedicado aproximadamente 12 horas semanales a formarme y perfeccionar mis habilidades en el campo. Entre las actividades destacadas, la optimización de modelos y parámetros ha sido una prioridad, permitiéndome mejorar tanto la precisión como la eficacia de los modelos desarrollados. Esta dedicación ha sido complementada con un aprendizaje continuo en nuevas técnicas y herramientas, lo cual ha sido fundamental para enriquecer los resultados y la calidad del análisis en el proyecto.

2.3.1. Estimación del coste

Coste de personal

Tomando como referencia un salario medio de 35 euros por hora para un científico de datos, el coste directo de las horas de trabajo es el siguiente:

- 50 horas en recolección de muestras: 1.750 euros.
- Formación y tareas de optimización durante 10 meses:
  35 euros/hora $\times$ 12 horas/semana $\times$ 4 semanas/mes $\times$ 10 meses = 16.800 euros.

Coste de recursos y equipamiento

El uso de recursos incluye gastos por conexión a internet, consumo eléctrico y la utilización de dispositivos como ordenadores portátiles y software necesario para el desarrollo del proyecto. En particular para mi proyecto he podido desarrollarlo gracias a mi procesador multinúcleo, concretamente de 8 núcleos, con menos capacidad el tiempo de entrenamiento de los modelos habría aumentado considerablemente. Se estima el siguiente desglose de costes:

- Conexión a internet: 30 euros/mes $\times$ 10 meses = 300 euros.
- Consumo eléctrico: 20 euros/mes $\times$ 10 meses = 200 euros.
- Portátil: 1000 euros.
- Diadema MUSE: 300 euros.
- Dispositivo conectado a la diadema para almacenar y guardar datos, (móvil): 200 euros

Presupuesto total

El presupuesto total estimado para el proyecto se calcula como sigue:

- Coste de personal: 18.550 euros.
- Coste de recursos y equipamiento: 1.500 euros.
- **Presupuesto total estimado**: 20.050 euros.

Entonces como hemos podido desarrollar el presupuesto de este TFG sería de aproximadamente 20000 euros.

Capítulo 3

Estado del arte

3.1. Introducción

En el contexto actual de la investigación interdisciplinaria, la sinergia establecida entre el aprendizaje automático y la electroencefalografía emerge como un campo de estudio notablemente prometedor. Esta integración de tecnologías avanzadas ha inaugurado nuevas vías para la exploración profunda de los procesos cerebrales humanos, facilitando avances significativos en áreas tan variadas como la medicina, la neurociencia y la ciencia computacional. Destaca especialmente la relevancia de esta convergencia en el desarrollo y aplicación de dispositivos portátiles, como la diadema MUSE, que democratizan el acceso a la captura y análisis de datos EEG de forma accesible y no invasiva.

3.2. Revisión de la literatura

En esta sección, se aborda una revisión exhaustiva de la literatura relacionada con el reconocimiento de actividades cognitivas mediante el uso de EEG, con énfasis en la extracción de características relevantes y su aplicación en el reconocimiento de actividades tales como la lectura, la solución de problemas y la escucha de música. Este análisis sienta las bases para el desarrollo de metodologías eficaces en la detección de dichas actividades, a través de señales EEG procesadas mediante dispositivos wearables como la diadema MUSE.

3.2.1. Estudios previos

El estudio de Skutt et al. (2023) [6] explora la viabilidad de diferenciar entre géneros musicales utilizando cinco canciones distintas, logrando una precisión del 56 % con el dispositivo MUSE. En contraste, el Dr. Ian Daly de la Universidad de Essex, en 2023, combinó EEG y fMRI con técnicas de aprendizaje automático para identificar qué canción estaba siendo escuchada por el sujeto, alcanzando una precisión del 72 % con un conjunto de 18 participantes válidos y 36 canciones

diferentes a lo largo de varias sesiones [7]. Tsekoura et al. (2020) avanzaron en esta línea de investigación, identificando señales EEG vinculadas a la percepción de notas específicas en una escala musical, y lograron una precisión del 70 % en la clasificación de estas notas [8], todo esto con el dispositivo Emotiv EPOC+ con una frecuencia de entrada de 128Hz. Estos estudios reflejan una precisión general en la identificación musical que varía entre el 56 % y el 72 %. Como vemos, todos estos estudios se centran en la música y no optan por diferenciarla con alguna otra actividad, sobre todo, ponen su foco en diferenciar el género musical.

En el ámbito de la concentración, gracias a la diadema MUSE, el uso de máquinas de soporte vectorial, (SVM), puede alcanzar una precisión del 94 % en la detección de la concentración en estudiantes, según se reporta en [9]. De manera similar, Alirezaei et al. (2017) lograron una precisión del 92.8 % utilizando SVM en un estudio con 12 participantes [10], subrayando la eficacia de las técnicas de ML en la evaluación de la concentración. En este caso el dispositivo utilizado fue el EMOTIV. Un gran problema de estos estudios es la falta de muestras y diversidad de los pacientes. Además, en el segundo estudio ni siquiera se usa una política *subject wise* a la hora de tomar los conjuntos de entrenamiento y test.

Respecto a la diferenciación entre estados de lectura y no lectura, se ha obtenido una precisión del 78 % en comparaciones entre diferentes pruebas realizadas a tres pacientes, determinando cuándo estaban leyendo y cuándo no [11] mediante el dispositivo EMOTIV. Al igual que en el estudio anterior, la cantidad de datos para el entrenamiento es insuficiente.

Finalmente, en relación con el reconocimiento de actividades similares a nuestro estudio, el uso del dispositivo MUSE ha permitido diferenciar en 8 participantes entre leer, hablar y ver televisión mediante una red neuronal, alcanzando una precisión del 94.6 % [12]. En contraste, Hussain et al. (2023) distinguieron entre leer, caminar, estar de pie y realizar actividades físicas utilizando electrodos en el cuero cabelludo, logrando una precisión del 80 % con el método de *Random Forest* [13]. Estos experimentos, aunque parecidos al que desempañaremos, presentan diferencias notables con el proyecto. Por un lado, el segundo estudio fue llevado por un equipo con capacidad para experimentar con 75 personas, incluso disponían de un equipo de alta gama para recoger las señales EEG en vez de un dispositivo económico portable. Por otro lado, el primer experimento se asemeja mucho más, pero el número de participantes, 8, es bastante reducido. Otro problema surge a la hora del entrenamiento, donde los conjuntos de entrenamiento, validación y test no se rigen por una política *subject wise* lo que contamina el experimento. Esto hace que con tan pocos datos no podamos afirmar si la red neuronal realmente está aprendiendo a diferenciar entre las distintas actividades o cualquier otro patrón.

El estudio que desarrollaremos en este TFG se caracteriza por su rigor metodológico. En el proceso de entrenamiento, aplicaremos consistentemente una política *subject wise* para garantizar la fiabilidad de los resultados. Además, nuestro conjunto de datos de entrenamiento será significativamente mayor en comparación con la mayoría de los estudios revisados, proporcionando una ma-

yor riqueza y diversidad en los datos. Un factor distintivo de nuestro estudio es el uso de un dispositivo económico, lo que permitirá que el experimento sea fácilmente replicable sin incurrir en altos costos de inversión en materiales. La mayor innovación respecto a los estudios analizados es nuestra capacidad para distinguir entre cuatro actividades diferentes, todas realizadas en posición sentada. Esto implica que la diferenciación entre actividades dependerá exclusivamente de la actividad cerebral, sin otros estímulos externos. En resumen, nuestro estudio no solo busca ser más general y riguroso que los analizados previamente, sino también más accesible y replicable, con una metodología sólida y un enfoque innovador en la diferenciación de actividades mediante señales cerebrales.

3.2.2. Aprendizaje automático

Como se ha expuesto en la sección precedente, el ML demuestra una capacidad notable para clasificar ondas EEG dentro de las categorías establecidas de manera eficiente. A continuación, se procederá a compilar los modelos y características destacados en los estudios analizados, proporcionando una visión integral de las metodologías y herramientas empleadas en el ámbito del ML para el tratamiento y análisis de señales EEG. En la tabla 3.1, se presenta una comparativa de los estudios analizados.

Modelos ML

La exploración de diferentes modelos de aprendizaje automático ha sido fundamental en la investigación sobre el procesamiento de señales EEG para el reconocimiento de actividades cognitivas. A través de diversos estudios, se han aplicado modelos variados con el fin de alcanzar mayor precisión y eficacia en la clasificación de estas actividades. Los modelos destacados incluyen:

- XGBoost: Utilizado por Skutt et al. (2023) para clasificar géneros musicales [6].
- C-SVC con kernel RBF: Aplicado por Tsekoura et al. (2020) en la clasificación de señales EEG provocadas por estímulos de notas musicales [8].
- BiLSTM: Empleado por Daly (2023) para decodificar música a partir de análisis EEG [7].
- Máquinas de soporte vectorial, (SVM): Usadas para detectar estados de concentración con una alta precisión [9], [10].
- *Random Forest*: Utilizado para distinguir entre diversas actividades cognitivas, como leer, caminar, estar de pie, y realizar ejercicios físicos [13].
- KNN: Aunque usado en la mayoría de los estudios anteriores, en general nunca destaca por encima del resto a la hora de clasificar.

Estos modelos reflejan la diversidad de enfoques metodológicos adoptados en el campo del ML aplicado al análisis de señales EEG, cada uno con sus ventajas específicas en el contexto de las diferentes investigaciones realizadas.

Extracción de características

La extracción de características robustas y relevantes es fundamental en el procesamiento de señales para el reconocimiento de actividades humanas, (HAR). Se consideran diversas métricas que capturan la esencia de las señales en cuestión como vemos en [10]:

- **Entropía basada en características:** La entropía de Shannon para cada canal de señal en cada banda de frecuencia se calcula utilizando la siguiente fórmula:

$$H(x_{fb}) = -\sum_i P(x_{fb}(i)) \log_2(P(x_{fb}(i)))$$

  donde $x_{fb}(t)$ representa la señal filtrada en una banda de frecuencia específica y $P(x_{fb})$ es la distribución de probabilidad asociada.

- **Potencia máxima de cada banda de frecuencia:** Para cada banda de frecuencia, se realiza la transformada de Fourier y se identifica la potencia máxima mediante:

$$P_{fb} = \max_f \left(\frac{|X_{fb}(f)|^2}{N} \right)$$

  donde $X_{fb}(f)$ es la transformada de Fourier de la banda y N es el número de puntos de dicha transformada.

- **Energía de cada banda de frecuencia:** La energía de cada banda se obtiene integrando el cuadrado de la magnitud de la transformada de Fourier:

$$E_{fb} = \int_f |X_{fb}(f)|^2 \, df$$

- **Valor pico a pico de las bandas de frecuencia:** El valor pico a pico en el análisis de señales es un indicador clave de la variabilidad de la señal dentro de una ventana temporal específica y se utiliza ampliamente para caracterizar la amplitud de las fluctuaciones en la señal. Se calcula como:

$$\text{Pico a pico}(x) = \max(x) - \min(x)$$

  donde x representa los valores de la señal en una banda de frecuencia específica durante una ventana de tiempo determinada. Esta medida es útil para identificar la amplitud total de las oscilaciones en la señal.

- **Valor medio de las bandas de frecuencia en el dominio temporal:** El valor medio de una señal en el dominio temporal proporciona una medida de la tendencia central de la señal. Esto es fundamental para entender el

comportamiento general de la señal sin las fluctuaciones de menor escala. Se calcula utilizando la siguiente fórmula:

$$\text{Valor medio}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

donde x_i son los valores de la señal en una ventana temporal y n es el número total de muestras en esa ventana. Esta métrica determina la posición promedio o el nivel de línea base de la señal, que puede ser determinante en estudios que dependen de la comparación de características normalizadas a lo largo del tiempo o entre diferentes condiciones experimentales.

- **Entropía de muestra, (sample entropy en inglés):** Es una métrica que evalúa la complejidad de una serie temporal mediante la estimación de la nueva información contenida en cada punto de datos adicional. A diferencia de la entropía aproximada, la entropía de muestra no cuenta secuencias auto-comparativas, ofreciendo una perspectiva más realista de la complejidad. La entropía de muestra se define matemáticamente como:

$$SE(m, r) = -\log\left(\frac{A}{B}\right)$$

donde A representa el número de pares de secuencias similares de longitud $m+1$ y B el número de pares de secuencias similares de longitud m. En este contexto, m es la longitud de la secuencia de datos considerada, que determina cuántos puntos consecutivos en la serie se comparan entre sí. r es el radio de tolerancia, una fracción del desvío estándar de la serie temporal, que define qué tan cercanos deben estar dos puntos para considerarse similares.

En este proyecto se ha decidido utilizar $m = 2$ y $r = 0{,}2 \times \text{std(signal)}$. Estos valores son seleccionados porque proporcionan un equilibrio entre la sensibilidad y la robustez del cálculo. Un valor de m pequeño reduce la susceptibilidad a la variación aleatoria de los datos, mientras que un r moderado garantiza que solo las fluctuaciones significativas en la serie temporal alteren la medida de entropía. Esto es particularmente útil en señales biomédicas, donde es importante detectar regularidades subyacentes en presencia de potencial ruido.

- **Entropía diferencial ,(differential entropy en inglés):** Se utiliza para medir la incertidumbre o la complejidad de una señal continua. La entropía diferencial se calcula a partir de la distribución de probabilidad continua de la señal, lo que proporciona una medida de la cantidad de información necesaria para describir el estado de la señal. Para una variable aleatoria X con una densidad de probabilidad $p(x)$, la entropía diferencial se define como:

$$h(X) = -\int p(x)\log p(x)\,dx$$

En el contexto de señales digitales, y asumiendo que la distribución se puede aproximar por una distribución gaussiana, la entropía diferencial se puede estimar como:

$$h(X) \approx \frac{1}{2} \log(2\pi e \sigma^2)$$

donde σ^2 es la varianza de la señal. Esta aproximación supone normalidad y es frecuentemente utilizada en el análisis de señales debido a su simplicidad.

El reconocimiento de patrones y la precisión de los modelos analíticos necesita de una selección de características capaces de caracterizar la señal EEG. Es por eso que las características mencionadas anteriormente aportarán valor a nuestro estudio ayudando a los modelos ML.

Cuadro 3.1: Comparativa de modelos estudiados

Ref	**Clasificador**	**Estudio**	**Nº**	**Repeticion**	**Aparato EEG**	**Tiempo Señal EEG**	**Tiempo Reposo**	**Precisión**	**Partición Test**
[6]	XGBoost	Clasificación géneros musicales	12	3	MUSE 2	Duración canción	No pausa	56 %	50 %
[7]	BiLSTM	Decodificar música a partir de EEG	18	3	EEG y fMRI	40 s	0.5 s / 1 min	72 %	3 × 3 cross-fold train and test
[8]	C-SVC (RBF)	Clasificación señales EEG (12 notas musicales)	5	Aprox 6	Emotiv EPOC+ sampling rate of 128 Hz	70 notas producidas cada una 3 s	3 s	70 %	20 %
[9]	SVM	Detección concentración estudiantes	12	-	MUSE	-	-	92 %	-
[10]	KNN-SVM-Bayesian-LDA	Evaluación concentración	12	0	EMOTIV	180 s	-	92.8 %	5 fold cross validation
[11]	KNN	Estados de lectura/no lectura	3	3	EMOTIV	5 min / 3 min	-	78 %	3 fold cross validation
[12]	Red Neuronal	Diferenciar leer/hablar/ver TV	8	1	MUSE	5 min / 3 min / 3 páginas	-	94.6 %	20 % validation 20 % test
[13]	Random Forest-XGBoost	Leer/caminar/ estar de pie/actividades físicas	75	3	Electrodos cuero cabelludo	Duración ejercicio	5 min	80 %	20 %

3.2.3. Selección de características

La selección de características es un proceso crítico en el análisis de datos, que busca reducir la dimensionalidad del conjunto de datos y mejorar la eficiencia y efectividad de los clasificadores. Se destacan dos métodos esenciales basados en investigaciones previas, [10], [13] respectivamente:

1. **Forward feature selection:** Método iterativo que inicia con un conjunto vacío de características y añade secuencialmente aquellas que optimizan un criterio predefinido. Utiliza el criterio de Fisher, que evalúa la capacidad discriminativa de las características entre clases, definido como:

$$J(\mathcal{F}) = \frac{\text{tr}(S_B(\mathcal{F}))}{\text{tr}(S_W(\mathcal{F}))}$$

   donde $S_B(\mathcal{F})$ es la matriz de dispersión entre-clases y $S_W(\mathcal{F})$ es la matriz de dispersión intra-clase para el conjunto de características $\mathcal{F}$. Este proceso continúa hasta que se alcanza un criterio de parada.

2. **Selección con selectKBest de scikit-learn:** Procedimiento efectuado mediante la biblioteca Scikit-Learn, que selecciona las k características más significativas según una prueba estadística. La prueba de chi-cuadrado es una métrica común en este contexto, y se calcula para un conjunto de datos con n características y m clases como:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

   donde O_i representa el número observado y E_i el número esperado de ocurrencias de la i-ésima característica, bajo la suposición de independencia. Esta técnica es valiosa para enfocar el entrenamiento de modelos en las características más informativas. En nuestro estudio, utilizaremos la función *f_classif* para la selección de características. Esta función realiza una prueba ANOVA, (análisis de varianza), entre cada característica y la variable de destino. La prueba ANOVA evalúa si las medias de diferentes grupos son significativamente diferentes entre sí, y se calcula como:

$$F = \frac{\text{MSB}}{\text{MSW}}$$

   donde MSB, (Mean Square Between), es la variabilidad entre las medias de los grupos y MSW, (Mean Square Within), es la variabilidad dentro de cada grupo. f_classif nos permite identificar las características que tienen una mayor capacidad para discriminar entre las clases, es decir, mejora la precisión de nuestros modelos predictivos.

Estos enfoques son herramientas poderosas que facilitan el entrenamiento y optimización de clasificadores en aprendizaje automático, proporcionando una manera eficiente de manejar la alta dimensionalidad de los datos y seleccionar características que tienen mayor relevancia para la tarea en cuestión.

Capítulo 4

Fundamentos matemáticos

4.1. Introducción

Antes de adentrarnos en la descripción detallada del experimento y discutir las herramientas tecnológicas y los principios de bioinformática implicados en nuestro estudio, es imprescindible establecer primero las bases matemáticas que fundamentan nuestra investigación. Esta necesidad surge de la importancia de entender profundamente los resultados obtenidos y de poder interpretar adecuadamente las implicaciones de estos en el contexto de nuestro trabajo.

Los fundamentos matemáticos no solo proporcionan el lenguaje formal a través del cual se expresan los modelos teóricos y las hipótesis de nuestra área de estudio, sino que también ofrecen las herramientas analíticas esenciales para procesar y analizar los datos de manera efectiva. Por ello, este capítulo se dedica a revisar y explicar las teorías y los métodos matemáticos que serán utilizados posteriormente en las aplicaciones prácticas de nuestra investigación.

4.2. Probabilidad

La probabilidad constituye un pilar fundamental en el campo del aprendizaje automático y desempeña un papel crucial en la creación y análisis de modelos predictivos y explicativos. Esta rama de las matemáticas permite a los investigadores cuantificar la incertidumbre, modelar fenómenos complejos y tomar decisiones basadas en datos incompletos o sujetos a aleatoriedad.

En el contexto de nuestro trabajo, la probabilidad no solo facilita el diseño de algoritmos robustos y eficientes para el aprendizaje automático, sino que también es esencial para interpretar los resultados de los experimentos y validar hipótesis de manera científica. A través de la teoría de la probabilidad, podemos establecer modelos que reflejen la realidad con mayor precisión y realizar inferencias que son críticas para la bioinformática y otras áreas aplicadas.

4.2.1. Conceptos básicos

En esta sección, se introducen los conceptos fundamentales de la teoría de la probabilidad descritos en [14], que son esenciales para el desarrollo de modelos estadísticos. A partir de ahora con el fin de ahorrar notación consideraremos el conjunto $\Omega \neq \emptyset$. Además siempre que encontremos n, este $n \in \mathbb{N}$.

Un **experimento aleatorio** es aquel cuyo resultado no puede predecirse con certeza, aunque se conozcan todas las condiciones iniciales posibles. Estos experimentos son la base para el estudio de la probabilidad, ya que permiten modelar fenómenos sujetos a incertidumbre. El espacio muestral, denotado como Ω, es el conjunto de todos los resultados posibles de un experimento aleatorio. Cada elemento de Ω representa un posible resultado del experimento.

Por ejemplo, considere el lanzamiento de un dado justo de seis caras. Aquí, el espacio muestral Ω es $\{1, 2, 3, 4, 5, 6\}$, representando cada número la cara del dado que podría aparecer hacia arriba al final del lanzamiento. Este ejemplo ilustra cómo cada posible resultado del experimento (cada lanzamiento del dado) está claramente definido y es parte de un conjunto finito de posibles resultados.

Definición 4.1. Un **suceso** es cualquier subconjunto de Ω. Se considera que un suceso ha ocurrido si el resultado del experimento es un elemento de este subconjunto.

Podemos encontrar estos distintos tipos de sucesos:

- **Suceso cierto o seguro:** El suceso cierto o seguro es el propio espacio muestral Ω. Dado que todos los resultados posibles del experimento están contenidos en Ω, este suceso ocurre siempre y se denota también como Ω.
- **Suceso imposible:** El suceso imposible es el conjunto vacío, denotado como $\emptyset$. Este suceso no contiene ningún resultado posible del experimento y, por lo tanto, nunca ocurre.
- **Unión de sucesos:** La unión de dos sucesos, A y B, denotada como $A \cup B$, es el suceso que contiene todos los elementos que pertenecen a A, a B, o a ambos. El suceso $A \cup B$ ocurre si el resultado del experimento está en A, en B, o en ambos.
- **Intersección de sucesos:** La intersección de dos sucesos, A y B, denotada como $A \cap B$, es el suceso que contiene todos los elementos que pertenecen tanto a A como a B. El suceso $A \cap B$ ocurre solo si el resultado del experimento está en ambos sucesos simultáneamente.
- **Sucesos complementarios:** Dados un suceso $A \subseteq \Omega$, el complemento de A, denotado como A^c, es el conjunto de todos los elementos de Ω que no pertenecen a A. Los sucesos A y A^c son complementarios, ya que $A \cup A^c = \Omega$ y $A \cap A^c = \emptyset$.
- **Resta de sucesos:** El suceso $A \setminus B$ es el conjunto de todos los elementos que pertenecen a A pero no a B, denotado a su vez como $A \cap B^c$. Este suceso ocurre si el resultado del experimento está en A pero no en B.

- **Sucesos incompatibles:** Se dice que dos sucesos son incompatibles si no comparten ningún resultado, es decir, su intersección es el suceso imposible, $A \cap B = \emptyset$. Un ejemplo de sucesos incompatibles son un suceso A y su complementario A^c, ya que no tienen elementos en común.

Definimos ahora un concepto fundamental que nos ayudará enormemente a lo largo de esta sección.

Definición 4.2. Una familia de conjuntos $\mathcal{A}$ definida sobre Ω se dice que es una **σ-álgebra** si:

1. $\Omega \in \mathcal{A}$.
2. $A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A}$.
3. $\{A_n\}_{n\geq 1} \subset \mathcal{A} \Rightarrow \bigcup_{n\geq 1} A_n \in \mathcal{A}$.

Hemos dado una definición de lo que es un suceso pero aún no tenemos una forma de cuantificar cual será la probabilidad de que dicho suceso ocurra. De ahí surge la siguiente función.

Definición 4.3. Una función, P, definida sobre la σ-álgebra $\mathcal{A}$ es una **medida de probabilidad** si:

1. $P(A) \geq 0, \forall A \in \mathcal{A}$.
2. $P(\Omega) = 1$.
3. P es numerablemente aditiva (o σ-aditiva), es decir, si $\{A_n\}_{n\geq 1}$ es una sucesión de sucesos disjuntos de $\mathcal{A}$, entonces

$$P\left(\bigcup_{n\geq 1} A_n\right) = \sum_{n\geq 1} P(A_n).$$

Gracias a las dos definiciones anteriores, podemos dar una defínción del espacio de probabilidad. Esta será necesaria para modelar los experimentos aleatorios.

Definición 4.4. Un **espacio de probabilidad** se define como una tripleta $(\Omega, \mathcal{A}, P)$ donde:

1. Ω es el conjunto de los sucesos elementales.
2. $\mathcal{A}$ es una colección de sucesos aleatorios que forma una σ-álgebra sobre Ω.
3. P es una función de probabilidad que asigna una probabilidad a cada uno de los sucesos.

De las anteriores definiciones se extraen fácilmente las siguientes propiedades de la probabilidad.

Proposición 4.1. *1. La probabilidad del vacío es cero: $P(\emptyset) = 0$*

2. Aditividad finita: Si $A_1, \ldots, A_n$ son elementos disjuntos de A, entonces

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i).$$

3. Probabilidad del suceso complementario: $P(A^c) = 1 - P(A)$.

4. Monotonía: Si $A, B \in \mathcal{A}$ y $A \subset B$, entonces $P(A) \leq P(B)$.

5. Probabilidad de una unión cualquiera de sucesos: Si $A_1, \ldots, A_n \in \mathcal{A}$, entonces

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i) - \sum_{i<j} P(A_i \cap A_j) + \cdots + (-1)^{n+1} P(A_1 \cap \cdots \cap A_n).$$

6. Subaditividad: Dados los sucesos $A_1, \ldots, A_n$, tenemos

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i).$$

Si tenemos una sucesión de sucesos $\{A_n\}_{n\geq 1}$, entonces

$$P\left(\bigcup_{n\geq 1} A_n\right) \leq \sum_{n\geq 1} P(A_n).$$

Demostración. 1. **La probabilidad del vacío es cero:** Considerando que $\Omega = \Omega \cup \emptyset \cup \emptyset \ldots$ y por la aditividad numerable, tenemos que $P(\Omega) = P(\Omega) + \sum_{k\geq 1} P(\emptyset)$, lo que implica que $P(\emptyset) = 0$.

2. **Aditividad finita:** Si $A_1, \ldots, A_n$ son elementos disjuntos de $\mathcal{A}$, aplicando la σ-aditividad, la propiedad anterior y asignando $A_i = \emptyset$ para $i > n$, obtenemos:

$$P\left(\bigcup_{i=1}^{n} A_i\right) = P\left(\bigcup_{i\geq 1} A_i\right) = \sum_{i=1}^{n} P(A_i).$$

3. **Complementos:** Se deduce de

$$1 = P(\Omega) = P(A \cup A^c) = P(A) + P(A^c) \implies P(A^c) = 1 - P(A)$$

4. **Monotonía:** Si $A, B \in \mathcal{A}$ y $A \subset B$, entonces de $P(B) = P(A) + P(B \setminus A)$ como P es mayor o igual que 0 se deduce que $P(A) \leq P(B)$.

5. **Probabilidad de una unión cualquiera de sucesos (fórmula de inclusión-exclusión):** Para su obtención observemos que si $n = 2$, entonces:

$$P(A_1 \cup A_2) = P(A_1) + P(A_2 \setminus A_1) = P(A_1) + P(A_2) - P(A_1 \cap A_2).$$

El resto de la demostración se sigue por inducción.

6. **Subaditividad:** Dados los sucesos $A_1, \ldots, A_n$, la relación existente entre la probabilidad de la unión de los A_i y la probabilidad de cada uno de ellos es la siguiente:

$$P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i).$$

En efecto, sean $B_1 = A_1$ y $B_i = A_i \setminus \bigcup_{j=1}^{i-1} A_j$ para $i = 2, \ldots, n$. Los B_i son disjuntos y $\bigcup_{i=1}^{n} B_i = \bigcup_{i=1}^{n} A_i$. Por la aditividad finita y la monotonía de P, tenemos:

$$P\left(\bigcup_{i=1}^{n} A_i\right) = P\left(\bigcup_{i=1}^{n} B_i\right) = \sum_{i=1}^{n} P(B_i) \leq \sum_{i=1}^{n} P(A_i).$$

□

Vamos a intentar definir coloquialmente la probabilidad de que ocurra un suceso. Sea el experimento a realizar el lanzamiento de un dado justo de 6 caras. Al realizar el lanzamiento, los casos posibles que pueden darse son que aparezca un número del siguiente conjunto {1, 2, 3, 4, 5, 6}. Si denotamos A como el suceso en el que el resultado del lanzamiento es par, A se identifica con los resultados {2, 4, 6}. Podemos entonces definir la probabilidad de A como:

$$P(A) = \frac{\text{número de casos favorables}}{\text{número de casos totales}} = \frac{3}{6} = 50\,\%$$

De esta puntualización podemos extraer la siguiente definición.

Definición 4.5. Sea $(\Omega, \mathcal{A}, P)$ un espacio de probabilidad y sean A y B dos sucesos, con $P(B) > 0$, se define la **probabilidad de A condicionada a B** mediante la expresión,

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Se deduce que la probabilidad de la intersección de dos sucesos puede expresarse de la forma $P(A \cap B) = P(A|B)P(B)$.

Teorema 4.1 (Teorema de la probabilidad total)**.** *Supongamos que $A_1, A_2, \ldots, A_n$ constituyen una partición de Ω, esto es, $\bigcup_{i=1}^{n} A_i = \Omega$ y $A_i \cap A_j = \emptyset$ para $i \neq j$. Supongamos también que tienen probabilidad no nula, $P(A_i) > 0$ para todo i. Entonces cualquier suceso B puede expresarse como $B = \bigcup_{i=1}^{n}(B \cap A_i)$ donde*

los conjuntos $B \cap A_i$ son disjuntos. Por tratarse de una unión disjunta podemos escribir

$$P(B) = \sum_{i=1}^{n} P(B \cap A_i) = \sum_{i=1}^{n} P(B|A_i)P(A_i).$$

Demostración. Por hipótesis tenemos una partición $A_1, A_2, \ldots, A_n$ del espacio muestral Ω. Por lo tanto, el suceso B se puede escribir como

$$B = B \cap \Omega = B \cap (A_1 \cup A_2 \cup \cdots \cup A_n).$$

$$B = (B \cap A_1) \cup (B \cap A_2) \cup \cdots \cup (B \cap A_n).$$

Ahora bien, los conjuntos $B \cap A_i$ son disjuntos dos a dos, ya que en caso contrario los A_i tampoco lo serían. Por consiguiente,

$$P(B) = P(B \cap A_1) + P(B \cap A_2) + \cdots + P(B \cap A_n).$$

Por último, se sabe que $P(C \cap D) = P(C|D)P(D)$ para cualesquiera sucesos C y D con $P(D) > 0$. Luego,

$$P(B) = P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + \ldots + P(B|A_n)P(A_n) = \sum_{i=1}^{n} P(B|A_i)P(A_i),$$

como queríamos.

□

Con base en los resultados discutidos anteriormente, nos aproximamos al teorema de Bayes, un pilar fundamental en el estudio de la probabilidad. Este teorema no solo es útil y versátil en una vasta gama de aplicaciones prácticas, sino que también proporciona una estructura robusta para el razonamiento bajo incertidumbre. Ofrece información actualizada de probabilidades ante la adquisición de nueva evidencia.

Teorema 4.2 (Teorema de Bayes)**.** *Supongamos que los sucesos $\{A_i\}_{i=1}^{n}$ forman una partición del espacio muestral Ω y consideramos un suceso arbitrario B tal que $P(B) > 0$. Entonces se verifica*

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{n} P(B|A_j)P(A_j)}.$$

Demostración. Trivialmente

$$P(A_i \cap B) = P(B \cap A_i) = P(B|A_i)P(A_i)$$

Además el denominador es el resultado del Teorema de la probabilidad total. □

Para terminar esta introducción, daremos una definición que será bastante útil a la hora de diferenciar las actividades en nuestro experimento o saber si alguna de ellas esta correlacionada.

Definición 4.6. Sean A y B dos sucesos. Decimos que A y B son **sucesos independientes** si

$$P(A \cap B) = P(A)P(B).$$

De aquí podemos obtener la propiedad,

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A),$$

y su simétrica

$$P(B|A) = P(B).$$

4.2.2. Variables y vectores aleatorios

Las variables aleatorias son fundamentales en el estudio de la probabilidad y la estadística. Estas proporcionan un mecanismo formal para describir y analizar los resultados de experimentos aleatorios y fenómenos estocásticos. Al modelar fenómenos del mundo real como variables aleatorias, podemos aplicar herramientas matemáticas para predecir comportamientos y calcular diversas características de interés.

Definición 4.7. Consideramos los espacios medibles $(\Omega, \mathcal{A})$ y $(\mathbb{R}, \mathcal{B})$. Una **variable aleatoria** es una aplicación, $X : \Omega \to \mathbb{R}$, que verifica

$$X^{-1}(B) \in \mathcal{A}, \forall B \in \mathcal{B}.$$

Cuando hacemos intervenir una variable aleatoria en nuestro proceso es porque ya estamos en presencia de un espacio de probabilidad, $(\Omega, \mathcal{A}, P)$. La variable aleatoria traslada la información probabilística relevante de Ω a $\mathbb{R}$. Una variable aleatoria no es más que una aplicación medible entre $(\Omega, \mathcal{A})$ y $(\mathbb{R}, \mathcal{B})$.

Lema 4.1. *La **familia de sucesos***

$$\sigma(X) = \{X^{-1}(B) : B \in \mathcal{B}\}$$

es una σ-álgebra que verifica $\sigma(X) \subset \mathcal{A}$.

Definición 4.8. Dada una variable aleatoria X, se denomina **σ-álgebra inducida** por X a

$$\sigma(X) = \{X^{-1}(B) : B \in \mathcal{B}\}.$$

X induce sobre $(\mathbb{R}, \mathcal{B})$ una probabilidad, P_X, de la siguiente forma,

$$P_X(A) = P(X^{-1}(A)), \forall A \in \mathcal{B}.$$

La probabilidad P_X también recibe el nombre de distribución de la variable aleatoria X. Observemos que P_X hereda las características que tenía P, pero lo hace a través de X.

Ahora que ya hemos conseguido expresar el concepto de variable aleatoria, pasamos a estudiar funciones que nos permitan manejar la distribución de la probabilidad.

Definición 4.9. Dada una variable aleatoria X definimos su **función de distribución** o **función de densidad** como

$$F_X(x) = P_X((-\infty, x]) = P(X^{-1}((-\infty, x])) = P(X \leq x), \quad \text{con } x \in \mathbb{R}.$$

Proposición 4.2. *Dada una variable aleatoria X, su **función de distribución** asociada **verifica** una serie de **propiedades** que son: no negativa, monótona creciente, continua por la derecha,* $\lim_{x\to+\infty} F_X(x) = 1$ *y* $\lim_{x\to-\infty} F_X(x) = 0$. *Además, es continua en un punto x si y solo si $P(X = x) = 0$.*

Demostración. **No negatividad.** Consecuencia inmediata de su definición, P es no negativa.

Monótona creciente. De la monotonía de la probabilidad se deduce fácilmente que $F_X(x_1) \leq F_X(x_2)$ si $x_1 \leq x_2$.

Continuidad por la derecha. Consideremos una sucesión decreciente de números reales $x_n \downarrow x$. La correspondiente sucesión de intervalos verifica $\bigcap_n (-\infty, x_n] = (-\infty, x]$, y por la continuidad desde arriba de la probabilidad respecto del paso al límite tendremos $\lim_{x_n\downarrow x} F_X(x_n) = F_X(x)$.

Observemos por otra parte que

$$(-\infty, x] = \{x\} \cup \lim_{n\to+\infty} (-\infty, x - \frac{1}{n}],$$

lo que al tomar probabilidades conduce a

$$F_X(x) = P(X = x) + \lim_{n\to+\infty} F_X\left(x - \frac{1}{n}\right) = P(X = x) + F(x-).$$

A partir de esta relación se sigue que $F_X(x)$ es continua en x sí y solo sí $P(X = x) = 0$.

Valores límites. Si $x_n \uparrow +\infty$ o $x_n \downarrow -\infty$ entonces $(-\infty, x_n] \uparrow \mathbb{R}$ y $(-\infty, x_n] \downarrow \emptyset$ y por tanto

$$F(+\infty) = \lim_{x_n\uparrow+\infty} F(x_n) = 1, \quad F(-\infty) = \lim_{x_n\downarrow-\infty} F(x_n) = 0.$$

□

Definición 4.10. Se dice que la función F es una **función de distribución** de probabilidad si es no negativa, monótona creciente, continua por la derecha y verifica: $\lim_{x\to+\infty} F(x) = 1$, $\lim_{x\to-\infty} F(x) = 0$.

Hasta este momento, nuestro enfoque se ha centrado en el análisis continuo. No obstante, en nuestro experimento resulta esencial modelar datos discretos, ya que los dispositivos tecnológicos recopilan información en intervalos regulares.

Definición 4.11. Una **variable aleatoria discreta** es aquella que verifica que existe un conjunto numerable D tal que la variable toma valores en este conjunto o, de otro modo,

$$P(X \in D) = P_X(D) = 1.$$

En este caso también podemos expresar más rigurosamente y extender el ámbito en el que trabajábamos con nuestra función P.

Definición 4.12. Definimos la **función de probabilidad** o cuantía de una variable aleatoria discreta como

$$f_X(x) = \begin{cases} P(X = x), & \text{si } x \in D \\ 0, & \text{en el resto} \end{cases}$$

Proposición 4.3. *Si f_X es la función de probabilidad asociada a una variable aleatoria discreta entonces se verifican las dos propiedades siguientes:*

1. *f_X es no negativa.*
2. *$\sum_{x_i \in D} f_X(x_i) = 1$.*

Demostración. 1. Al tratarse de una probabilidad, $f_X(x) \geq 0, \forall x \in \mathbb{R}$,

2. Como $P(X \in D) = 1$ se tiene por la σ-aditividad de la medida de probabilidad que
$$\sum_{x_i \in D} f_X(x_i) = 1.$$

□

Hemos desarrollado un método para modelar nuestro experimento. Sin embargo, para comparar múltiples variables simultáneamente, es necesario profundizar en el estudio de los vectores aleatorios, lo cual será nuestro próximo paso.

Definición 4.13. La **σ-álgebra de Borel** en el espacio $\mathbb{R}^k$ es la mínima σ-álgebra que contiene a los rectángulos $(a, b]$ con $a, b \in \mathbb{R}^k$.

Definición 4.14. Un **vector aleatorio**, $X = (X_1, X_2, \ldots, X_k)$, es una aplicación de un espacio de probabilidad $(\Omega, \mathcal{A}, P)$ en el espacio medible $(\mathbb{R}^k, \beta_k)$, que verifica
$$X^{-1}(B) \in \mathcal{A}, \quad \forall B \in \beta_k.$$

Disponemos de una medida de probabilidad en el espacio medible $(\Omega, \mathcal{A})$. Utilizando esta medida, es posible definir una probabilidad en el espacio $(\mathbb{R}^k, \beta_k)$, donde β_k denota la σ-álgebra de Borel en $\mathbb{R}^k$. En consecuencia, podremos definir también una función de distribución de la que poder extraer propiedades de nuestro vector aleatorio.

Definición 4.15. Dado un vector aleatorio definido de un espacio de probabilidad $(\Omega, \mathcal{A}, P)$ en $(\mathbb{R}^k, \beta_k)$, se define la **probabilidad inducida** a partir de P mediante X como la medida de probabilidad definida como

$$P_X(B) = P(X^{-1}(B)), \quad \forall B \in \beta_k.$$

Definición 4.16. Dado un vector aleatorio $X = (X_1, X_2, \ldots, X_k)$, definimos su **función de distribución** asociada (o también la función de distribución asociada a la distribución P_X) en el punto $x = (x_1, \ldots, x_k)$ de $\mathbb{R}^k$ mediante

$$F_X(x) = F_X(x_1, \ldots, x_k) = P_X(S_x) = P(X \in S_x) = P\left(\bigcap_{i=1}^{k}\{X_i \leq x_i\}\right),$$

siendo $S_x = \prod_{i=1}^{k}(-\infty, x_i]$, a los que denominaremos región suroeste de x.

Proposición 4.4. *La función de distribución definida verifica las siguientes propiedades:*

1. *Es no negativa.*
2. *Es monótona creciente en cada componente.*
3. *Es continua desde arriba: si $x_n \downarrow x$ entonces $F_X(x_n) \downarrow F_X(x)$.*
4. *$\lim_{x_i \to \infty} F_X(x_1, \ldots, x_k) = 1$ y $\lim_{x_i \to -\infty} F(x_1, \ldots, x_k) = 0$.*

Demostración. **No negatividad** es consecuencia inmediata de ser una probabilidad.
Monotonía en cada componente: Si $x \leq y$, es decir, $x_i \leq y_i$ para $i = 1, \ldots, k$, $S_x \subset S_y$ y $F_X(x) \leq F_X(y)$.
Continuidad desde la derecha: Si $x^{(n)} \downarrow x$, entonces $F_X(x^{(n)}) \downarrow F_X(x)$.
Valores límites: Al tender a $\pm\infty$ las componentes del punto, se tiene el comportamiento límite especificado. □

Podemos asimismo dar una noción más natural del concepto de vector aleatorio.

Definición 4.17. $X = (X_1, \ldots, X_k)$ es un **vector aleatorio** sí y solo sí cada una de sus componentes es una variable aleatoria.

Demostración. La condición de variable aleatoria le viene a una aplicación por el hecho de que las anti-imágenes de conjuntos de Borel son sucesos, están en la σ-álgebra de nuestro espacio de probabilidad original, $X^{-1}(B) \in \mathcal{A}, B \in \beta$. Supongamos que las componentes de $X = (X_1, X_2, \ldots, X_k)$ son variables aleatorias. Entonces, para $S_x = \prod_{i=1}^{k}(-\infty, x_i]$ podemos escribir

$$\{X \in S_x\} = \bigcap_{i=1}^{k}\{X_i \leq x_i\}$$

y cada componente siendo aleatoria, $\{X_i \leq x_i\} \in \mathcal{A}$, por lo tanto X será un vector aleatorio. Para demostrar el inverso, considérese que

$$\{X_j \leq x_j\} = \lim_{x_i \to +\infty, i \neq j}\{X \leq S_x\},$$

lo que se puede conseguir haciendo $x_i = n, i \neq j$. X_j es medible pues al tratarse de una secuencia monótona creciente de conjuntos, su límite es la unión de todos ellos que es una operación estable en $\mathcal{A}$. □

Definición 4.18. Dado un vector aleatorio que toma un número finito o infinito numerable de posibles valores, el **soporte de la distribución del vector**, se define como un conjunto $D \subset \mathbb{R}$ numerable tal que $P(X \in D) = 1$, se dice que el vector aleatorio es discreto.

Notemos que si X es discreto también lo es cada variable que lo compone o cualquier subvector de X que consideremos. En este tipo de vectores aleatorios tenemos que $P(X \in D^c) = 0$ y si conocemos las probabilidades $P(X = x)$ para $x \in D$ entonces podemos conocer la probabilidad de cualquier otro suceso ya que

$$P(X \in B) = \sum_{x \in B} P(X = x).$$

Definición 4.19. Se define la **función de cuantía o probabilidad conjunta** en $x = (x_1, \ldots, x_k) \in \mathbb{R}^k$ como

$$f_X(x_1, \ldots, x_k) = \begin{cases} P(X_i = x_i, i = 1, \ldots, k), & \text{si } x = (x_1, \ldots, x_k) \in D \\ 0, & \text{en el resto} \end{cases}$$

Proposición 4.5. *Una función de probabilidad verifica que es no negativa y la suma de probabilidades sobre D es igual a 1.*

Demostración. Al tratarse de una probabilidad, $f_X(x) \geq 0, \forall x \in \mathbb{R}^k$, y como $P(X \in D) = 1$, $\sum_{x \in D} f_X(x) = P(X \in D) = 1$. □

Definición 4.20. La **función de probabilidad marginal** de X_i se obtiene sumando las probabilidades conjuntas sobre todas las configuraciones posibles de los otros componentes, expresada como:

$$f_{X_i}(x_i) = \sum_{x_j \in D_j, j \neq i} f_X(x_1, \ldots, x_k),$$

lo que permite obtener la función de cuantía marginal de la X_i a partir de la conjunta.

Al igual que en nuestro caso continuo con sucesos también hallamos formas de diferenciar la independencia entre variables.

Definición 4.21. Dos variables aleatorias discretas X e Y definidas sobre un mismo espacio de probabilidad $(\Omega, \mathcal{A}, P)$ se dicen independientes cuando:

$$P(X = x, Y = y) = P(X = x)P(Y = y), \forall x, y \in \mathbb{R}.$$

Definición 4.22. Una colección de variables aleatorias discretas $X_1, \ldots, X_n$ se dicen independientes cuando:

$$P(X_1 = x_1, \ldots, X_n = x_n) = P(X_1 = x_1) \cdots P(X_n = x_n),$$

para cualesquiera $x_1, \ldots, x_n \in \mathbb{R}$.

Además podemos caracterizar la independencia cuando tenemos conjuntos de variables definiéndola sobre vectores aleatorios.

Definición 4.23. Supongamos un vector aleatorio continuo $X = (X_1, \ldots, X_k)$ con densidad conjunta f y marginales f_i con $i = 1, \ldots, k$. Las variables $X_1, \ldots, X_k$ se dicen independientes cuando:

$$f(x_1, \ldots, x_k) = f_1(x_1) \cdots f_k(x_k),$$

para cualesquiera $x_1, \ldots, x_k \in \mathbb{R}$.

Hasta ahora, hemos sentado las bases teóricas con la definición y caracterización de variables y vectores aleatorios, junto con la exploración de la independencia y la función de distribución. Estos preparativos nos han llevado a destacar una propiedad fundamental para nuestro estudio: la esperanza matemática.

Definición 4.24. Sea X una variable discreta no negativa. Definimos la **esperanza** de esta variable como:

$$E[X] = \sum_{i \geq 1} x_i P(X = x_i),$$

y decimos que existe la esperanza cuando el sumatorio anterior es finito.

Proposición 4.6. *Existe la media de la variable aleatoria X si y solo si existe la media del valor absoluto de X, $|X|$.*

Demostración. La prueba es inmediata si tenemos en cuenta que:

$$|X| = X^+ + X^-.$$

□

Como habíamos entro anteriormente con el resto de propiedades extendemos la esperanza a los vectores aleatorios.

Definición 4.25. Sea $X = (X_1, \ldots, X_k)$ un vector aleatorio y sea g una función medible de $\mathbb{R}^k$ en $\mathbb{R}$. Definimos la **esperanza del vector** en los casos en que el vector sea discreto como:

$$E[g(X)] = \sum_{(x_1, \ldots, x_k) \in D_X} g(x_1, \ldots, x_k) f_X(x_1, \ldots, x_k).$$

Otra prodiedad fundamental de la que obtendremos información acerca de nuestro estudio es la siguiente:

Definición 4.26. Supongamos un vector bidimensional (X_1, X_2). Definimos la **covarianza** entre X_1 y X_2 como:

$$\operatorname{cov}(X_1, X_2) = E[(X_1 - E[X_1])(X_2 - E[X_2])].$$

Definición 4.27. Dado el vector aleatorio $X = (X_1, \ldots, X_k)$, definimos la matriz de covarianzas como:

$$\Sigma = [\sigma_{ij}]_{i,j=1,\ldots,k},$$

Esta matriz es fundamental en el estudio de las dependencias en un vector aleatorio y en estadística multivariante.

Otra medida muy útil para el estudio de variables aleatorias es la varianza. Esta nos proporciona una forma de medir la dispersión dentro de un conjunto de datos.

Definición 4.28. Si tenemos variables $X_1, \ldots, X_k$ y números reales $a_1, \ldots, a_k$ entonces definimos la varianza de una combinación lineal de estas variables como:

$$\operatorname{var}(a_1X_1 + \ldots + a_kX_k) = \sum_{i=1}^{k} a_i^2 \operatorname{var}(X_i) + 2\sum_{i<j} a_i a_j \operatorname{cov}(X_i, X_j).$$

Demostración. Denotemos $S = a_1X_1 + \ldots + a_kX_k$. Entonces:

$$\operatorname{var}(S) = E[(S - E[S])^2] = E\left[\left(\sum_{i=1}^{k} a_i(X_i - E[X_i])\right)^2\right] = \sum_{i=1}^{k} a_i^2 \operatorname{var}(X_i) + 2\sum_{i<j} a_i a_j \operatorname{cov}(X_i, X_j).$$

□

Finalizamos esta sección sobre probabilidad extendiendo el concepto de esperanza en relación con la ocurrencia de un suceso.

Definición 4.29. Sea (X, Y) un vector aleatorio definido sobre el espacio de probabilidad $(\Omega, \mathcal{A}, P)$ y denotemos por $P_{X|Y=y}$ la **distribución de probabilidad de X condicionada a $Y = y$.**

En el caso de nuestro estudio, discreto, el soporte de la distribución, D, es numerable y

$$E[g(X) \mid y] = \sum_{x \in D_y} g(x) P(X = x \mid Y = y) = \sum_{x \in D_y} g(x) f_{X|Y}(x \mid y),$$

donde $D_y = \{x; (x, y) \in D\}$ es la sección de D mediante y.

4.3. Álgebra lineal

El álgebra lineal es una rama esencial de las matemáticas que subyace a numerosos aspectos del análisis y la modelación en diversas disciplinas científicas y de ingeniería, incluyendo el ML y análisis de datos. Este campo matemático proporciona las herramientas necesarias para describir y manipular espacios y

transformaciones lineales, lo cual es necesario para entender y resolver sistemas de ecuaciones lineales, trabajar con vectores y matrices, y explorar propiedades fundamentales como valores y vectores propios.

En el marco de este trabajo, el álgebra lineal no solo apoya el desarrollo y la optimización de algoritmos avanzados para el procesamiento de datos y la modelación matemática, sino que también juega un papel determinante en la estructuración de datos complejos y en la extracción de características significativas. Empezaremos con los conceptos necesarios para desarrollar esta rama matemática tal como se expone en [15].

4.3.1. Espacio vectorial

Definición 4.30. Un conjunto X es un **espacio vectorial**, (EV), sobre los números reales si se definen dos operaciones en X: la adición y la multiplicación por escalar, tales que, para $x, y \in X$ y $\forall a \in \mathbb{R}$ se tiene,

- $x + y \in X$,
- $ax \in X$,
- $1x = x$,
- $0x = 0$,

además, X constituye un grupo conmutativo con elemento neutro 0 bajo la operación de adición y satisface las leyes distributivas para la multiplicación escalar:

$$a(x + y) = ax + ay,$$

Los elementos de X también se denominan **vectores**, mientras que los números reales se refieren como **escalares**.

Definición 4.31. Un subconjunto no vacío M de un espacio vectorial X es un **subespacio** de X, si la restricción de las operaciones a M lo convierte en un espacio vectorial.

Definición 4.32. Una **combinación lineal** de los vectores $x_1, \ldots, x_n$ en un espacio vectorial es una suma de la forma $\sum_{i=1}^{n} a_i x_i$ donde $a_i \in \mathbb{R}$. Si los a_i son positivos y $\sum_{i=1}^{n} a_i = 1$, la suma también se llama **combinación convexa**.

Es fácil ver que una combinación lineal de vectores de un subespacio aún pertenece al subespacio, y que las combinaciones lineales pueden usarse de hecho para construir subespacios a partir de un subconjunto arbitrario de un espacio vectorial X. Si S es un subconjunto de X, denotamos por $\text{span}(S)$ el subespacio de todas las posibles combinaciones lineales de vectores en S.

Definición 4.33. Un conjunto finito de vectores $S = \{x_1, \ldots, x_n\}$ es **linealmente dependiente** si es posible encontrar constantes $a_1, \ldots, a_n$, no todas cero, tales que

$$\sum_{i=1}^{n} a_i x_i = 0.$$

Si esto no es posible, los vectores se denominan **linealmente independientes**.

Esto implica que un vector y en span(S), donde S es un conjunto de vectores linealmente independientes, tiene una expresión única de la forma:

$$y = \sum_{i=1}^{n} a_i x_i,$$

para algún n y donde cada $x_i \in S$ y $i = 1, \ldots, n$.

Definición 4.34. Un conjunto de vectores $S = \{x_1, \ldots, x_n\}$ se dice que forma una **base** para X si S es linealmente independiente y si cada elemento $x \in X$ puede expresarse de manera única como una combinación lineal de los vectores en S. Aunque siempre hay muchas bases diferentes, sus tamaños son siempre iguales. El tamaño de una base de un espacio vectorial se conoce como su **dimensión**.

Consideramos introducir una medida que nos proporcione una distancia en un espacio vectorial.

Definición 4.35. Un **espacio lineal normado** es un espacio vectorial X junto con una función de valor real que asigna a cada elemento $x \in X$ un número real $\|x\|$ llamado la **norma** de x, satisfaciendo las siguientes tres propiedades:

1. **Positividad**: $\|x\| > 0$ para todo $x \in X$, con igualdad si y solo si $x = 0$;
2. **Desigualdad triangular**: $\|x + y\| < \|x\| + \|y\|$ para todos $x, y \in X$;
3. **Homogeneidad**: $\|ax\| = |a|\|x\|$ para todo escalar $a \in \mathbb{R}$ y todo $x \in X$.

De lo anterior podemos extraer que la distancia entre dos vectores x y y puede definirse como la norma de su diferencia, $d(x, y) = \|x - y\|$.

Definición 4.36. En un espacio lineal normado, se dice que una secuencia infinita de vectores x_n converge a un vector x si la secuencia de números reales $\|x - x_n\|$ converge a cero.

Definición 4.37. Una secuencia x_n en un espacio lineal normado se dice que es una secuencia de Cauchy si $\|x_n - x_m\| \to 0$ conforme $n, m \to \infty$. Más precisamente, dado $\epsilon > 0$, existe un entero N tal que $\|x_n - x_m\| < \epsilon$ para todos $n, m > N$. Un espacio se dice completo cuando toda secuencia de Cauchy converge a un elemento del espacio.

En un espacio normado, toda secuencia convergente es una secuencia de Cauchy, pero lo contrario no siempre es cierto. Los espacios en los que toda secuencia de Cauchy tiene un límite se dicen completos. Los espacios lineales normados completos se denominan espacios de Banach.

4.3.2. Espacios con producto interno

La teoría de espacios con producto interno es una herramienta utilizada en geometría, álgebra, cálculo, análisis funcional y teoría de aproximación.

Definición 4.38. Una función f de un EV X a un EV Y se dice que es lineal si $\forall \alpha, \beta \in \mathbb{R}$ y $x, y \in X$, cumple que

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y).$$

Nótese que podemos considerar los números reales como un espacio vectorial de dimensión 1. Por lo tanto, una función de valores reales es lineal si satisface la misma definición.

Sea $X = \mathbb{R}^n$ y $Y = \mathbb{R}^m$. Una función lineal de X a Y puede denotarse mediante una matriz $m \times n$ con entradas A_{ij}, de modo que el vector $x = (x_1, \ldots, x_n)^T$ es mapeado al vector $y = (y_1, \ldots, y_m)^T$ donde

$$y_i = \sum_{j=1}^{n} A_{ij} x_j, \quad i = 1, \ldots, m.$$

Definición 4.39. Una matriz con entradas $A_{ij} = 0$ para $i \neq j$ se llama **matriz diagonal**.

Definición 4.40. Un espacio vectorial X se llama **espacio con producto interno** si existe un mapa bilineal (lineal en cada argumento) que, para cada dos elementos $x, y \in X$, otorga un número real denotado por (x, y) satisfaciendo las siguientes propiedades:

- $\langle x, y \rangle = \langle y, x \rangle$,
- $\langle x, x \rangle > 0$, y $\langle x, x \rangle = 0$ si y solo si $x = 0$.

La cantidad $\langle x, y \rangle$ se llama el **producto interno** de x y y, aunque también es conocido como el producto punto o producto escalar.

Definición 4.41. Dos elementos x y y de X se llaman **ortogonales** si

$$\langle x, y \rangle = 0$$

.

Definición 4.42. Un conjunto $S = \{x_1, \ldots, x_n\}$ de vectores de X se llama **ortonormal** si $(x_i, x_j) = \delta_{ij}$, donde $\delta_{ij} = 1$ si $i = j$, y 0 en caso contrario. Para un conjunto ortonormal S, y un vector $y \in X$, la expresión

$$\sum_{i=1}^{n} \langle x_i, y \rangle x_i$$

se dice que es una serie de Fourier para y. Si S forma una base ortonormal, cada vector y es igual a su serie de Fourier.

Teorema 4.3 (Desigualdad de Schwarz)**.** *En un espacio con producto interno, se cumple que*

$$|\langle x, y\rangle|^2 \leq \langle x, x\rangle\langle y, y\rangle$$

y el signo de igualdad se sostiene si y solo si x e y son dependientes.

Teorema 4.4 (Ley del paralelogramo)**.** *Para x e y vectores de un espacio con producto interno X:*

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2 + 2\langle x, y\rangle,$$

$$\|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2\langle x, y\rangle.$$

Definición 4.43. El **ángulo θ entre dos vectores** x e y de un espacio con producto interno se define por

$$\cos(\theta) = \frac{\langle x, y\rangle}{\|x\|\|y\|}$$

Si $|\langle x, y\rangle| = \|x\|\|y\|$, el coseno es 1, $\theta = 0$, y se dice que x y y son paralelos. Si $\langle x, y\rangle = 0$, el coseno es 0, $\theta = \frac{\pi}{2}$ y los vectores se dicen ortogonales.

Definición 4.44. Dado un conjunto $S = \{x_1, \ldots, x_n\}$ de vectores de un espacio con producto interno X, la matriz $n \times n$ G con entradas $G_{ij} = \langle x_i, x_j\rangle$ se llama la **matriz de Gram** de S.

4.3.3. Espacios de Hilbert

Definición 4.45. Un espacio H se dice separable si existe un subconjunto contable $D \subseteq H$, tal que cada elemento de H es el límite de una secuencia de elementos de D. Un **espacio de Hilbert** es un espacio con producto interno completo y separable. (Los espacios vectoriales de dimensión finita, como $\mathbb{R}^n$, son espacios de Hilbert).

Teorema 4.5 (Teorema de la proyección)**.** *Sea H un espacio de Hilbert, M un subespacio cerrado de H y $x \in H$. Existe un único vector $m_0 \in M$, conocido como la proyección de x sobre M, tal que*

$$\|x - m_0\| < \text{ínf}\{\|x - m\| : m \in M\}.$$

Una condición necesaria y suficiente para que $m_0 \in M$ sea la proyección de x sobre M es que el vector $x - m_0$ sea ortogonal a los vectores en M.

Una consecuencia de este teorema es que la mejor aproximación a x en el subespacio M generado por los vectores ortonormales $\{e_1, \ldots, e_n\}$ se da por su serie de Fourier

$$\sum_{i=1}^{n} \langle x, e_i\rangle e_i.$$

Esto lleva naturalmente al estudio de las propiedades de series como esta en el caso de bases infinitas.

Definición 4.46. Si S es un conjunto ortonormal en un espacio de Hilbert H y ningún otro conjunto ortonormal contiene a S como un subconjunto propio, es decir, S es maximal, entonces S se llama **base ortonormal** (o sistema ortonormal completo) para H.

Teorema 4.6 (Teorema sobre la existencia de base ortonormal)**.** *Todo espacio de Hilbert H posee una base ortonormal. Supongamos que $S = \{x_\alpha\}_{\alpha \in A}$ es una base ortonormal para un espacio de Hilbert H. Entonces, para todo $y \in H$,*

$$y = \sum_{\alpha \in A} \langle y, x_\alpha \rangle x_\alpha \quad y \quad \|y\|^2 = \sum_{\alpha \in A} |\langle y, x_\alpha \rangle|^2.$$

Este teorema establece que, como en el caso finito, cada elemento de un espacio de Hilbert puede ser expresado como una combinación lineal de elementos de base (posiblemente infinita). Los coeficientes $\langle y, x_\alpha \rangle$ son a menudo llamados los coeficientes de Fourier de y con respecto a la base $S = \{x_\alpha\}_{\alpha \in A}$.

4.3.4. Operadores, vectores y valores propios

Definición 4.47. Una función lineal de un espacio de Hilbert H a sí mismo se conoce como un **operador lineal**. Un operador lineal A es **acotado** si existe un número $\|A\|$ tal que

$$\|Ax\| \leq \|A\|\|x\|, \quad \text{para todo } x \in H.$$

Definición 4.48. Sea A un operador lineal en un espacio de Hilbert H. Si existe un vector $x \neq 0 \in H$ tal que $Ax = \lambda x$ para algún escalar λ, entonces λ es un **valor propio** de A con **vector propio** correspondiente x.

Definición 4.49. Un operador lineal acotado en un espacio de Hilbert H es **autoadjunto** si

$$\langle Ax, z \rangle = \langle x, Az \rangle, \quad \forall x, z \in H.$$

En el caso de dimensiones finitas $\mathbb{R}^n$, esto implica que la matriz $n \times n$ correspondiente A cumple $A = A^T$, es decir, $A_{ij} = A_{ji}$. Dichas matrices se conocen como **simétricas**.

Teorema 4.7 (Teorema de Hilbert-Schmidt)**.** *Sea A un operador lineal autoadjunto y compacto en un espacio de Hilbert H. Entonces existe una base ortonormal completa $\{\xi_i\}_{i=1}^{\infty}$ para H tal que*

$$A\xi_i = \lambda_i \xi_i$$

y $\lambda_i \to 0$ conforme $i \to \infty$. Obsérvese que en el caso finito, el teorema establece que las matrices simétricas tienen un conjunto ortonormal de vectores propios.

Definición 4.50. Se dice que una matriz cuadrada simétrica es **positiva (semi-) definida** si sus valores propios son todos positivos (no negativos).

Proposición 4.7. *Sea A una matriz simétrica. Entonces A es positiva (semi-) definida si y solo si para cualquier vector $x \neq 0$*

$$x^T Ax > 0 \quad (\geq 0).$$

Sea M una matriz (posiblemente no cuadrada) y definamos $A = M^T M$. Entonces, A es una matriz **semidefinida positiva** ya que podemos escribir

$$x^T Ax = x^T M^T Mx = (Mx)^T Mx = \|Mx\|^2 \geq 0,$$

para cualquier vector x. Si tomamos M como la matriz cuyas columnas son los vectores x_i, $i = 1, \ldots, n$, entonces A es la matriz de Gram del conjunto $S = \{x_1, \ldots, x_n\}$, mostrando que las matrices de Gram son siempre semidefinidas positivas, y definidas positivas si el conjunto S es linealmente independiente.

4.4. Máquinas de aprendizaje lineales

Depués de establecer las bases matemáticas de nuestro estudio, desarrollaremos a lo largo de las siguientes secciones, la teoría necesaria para desarrollar las SVM. En el aprendizaje supervisado, la máquina de aprendizaje recibe un conjunto de entrenamiento de ejemplos o entradas con etiquetas asociadas o valores de salida. Los ejemplos generalmente están en forma de vectores de atributos, por lo que el espacio de entrada es un subconjunto de $\mathbb{R}^n$. Una de las hipótesis más simples y mejor entendidas para resolver problemas en este contexto son las funciones lineales. Estas técnicas, desarrolladas en la estadística tradicional y la literatura de redes neuronales clásicas, son fundamentales para la discriminación entre dos clases y la interpolación. Primeramente, plantearemos el ejemplo más sencillo de resolver, con una clasificación binaria, mientras seguiremos dando definiciones fundamentales para esta teoría.

4.4.1. Clasificación lineal

La clasificación binaria se realiza frecuentemente utilizando una función de valor real $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$. Para un vector de entrada $x = (x_1, \ldots, x_n)$, la asignación a la clase positiva se hace si $f(x) > 0$, y a la clase negativa si $f(x) \leq 0$. Cuando $f(x)$ es una función lineal, puede escribirse como

$$f(x) = \langle w, x \rangle + b,$$

donde $w \in \mathbb{R}^n$ y $b \in \mathbb{R}$ son los parámetros de la función. La regla de decisión está dada por signo($f(x)$).

Definición 4.51. Un **hiperplano** es una subvariedad afín de dimensión $n - 1$ que divide el espacio en dos semiespacios, definidos por la ecuación $\langle w, x \rangle + b = 0$.

Definición 4.52. El **margen funcional** de un ejemplo (x_i, y_i) respecto a un hiperplano (w, b) se define como $y_i(\langle w, x_i \rangle + b)$.

El hiperplano divide el espacio en dos regiones correspondientes a las dos clases. Los vectores w y b se refieren respectivamente como el vector de pesos y el sesgo.

Para resolver el problema binario, uno de los primeros algoritmos propuestos fue el **perceptrón de Rosenblatt**. Es un algoritmo que actualiza el vector de pesos cada vez que un punto de entrenamiento es clasificado incorrectamente. El algoritmo está garantizado para converger si existe un hiperplano que clasifica correctamente los datos de entrenamiento.

Teorema 4.8 (Teorema de Novikoff)**.** *Sea S un conjunto de entrenamiento no trivial, y sea $R = \max_i \|x_i\|$. Supongamos que existe un vector w_{opt} tal que $\|w_{opt}\| = 1$ y $y_i(\langle w_{opt}, x_i\rangle + b_{opt}) \geq \gamma$ para todos los i. Entonces, el número de errores cometidos por el algoritmo del perceptrón es como máximo $\left(\frac{R}{\gamma}\right)^2$.*

Demostración. Para el análisis, aumentamos los vectores de entrada con una coordenada extra con valor R. Denotamos el nuevo vector por $x' = (x, R)^T$. De manera similar, añadimos una coordenada extra al vector de pesos w incorporando el sesgo b, para formar el vector de pesos aumentado $w' = (w, b/R)^T$. El algoritmo comienza con un vector de pesos aumentado $w_0 = 0$ y lo actualiza en cada error.

Sea w_{t-1} el vector de pesos aumentado antes del t-ésimo error. La t-ésima actualización se realiza cuando

$$y_i(\langle w_{t-1}, x_i'\rangle) < 0,$$

donde $(x_i, y_i) \in S$ es el punto incorrectamente clasificado por w_{t-1}. La actualización es la siguiente:

$$w_t = w_{t-1} + \eta y_i x_i',$$

donde hemos usado el hecho de que $b_t/R = b_{t-1}/R$ ya que $b_t = b_{t-1} + \eta y_i R^2$.

La derivación

$$\langle w_t, w_{\text{opt}}\rangle = \langle w_{t-1}, w_{\text{opt}}\rangle + \eta y_i \langle x_i', w_{\text{opt}}\rangle \geq \langle w_{t-1}, w_{\text{opt}}\rangle + \eta\gamma$$

implica que

$$\langle w_t, w_{\text{opt}}\rangle \geq t\eta\gamma.$$

Tenemos a su vez

$$\|w_t\|^2 = \|w_{t-1}\|^2 + 2\eta y_i \langle w_{t-1}, x_i'\rangle + \eta^2 \|x_i'\|^2 \leq \|w_{t-1}\|^2 + \eta^2 R^2,$$

lo que nos proporciona la siguiente desigualdad

$$\|w_t\|^2 \leq t\eta^2 R^2.$$

Al combinar las desigualdades anteriores

$$\|w_{\text{opt}}\|\sqrt{t}R \geq \|w_{\text{opt}}\|\|w_t\| \geq t\eta\gamma,$$

dandonos la cota que bucabamos

$$t \leq \frac{R^2}{\gamma^2},$$

pues $\|w_{\text{opt}}\| = 1$. □

El problema de aprender un hiperplano que separe dos conjuntos no está correctamente definido. Ahora mismo, el algoritmo del perceptrón puede dar una solución diferente dependiendo del orden en que se procesan los ejemplos. Una forma de corregir problema es optimizar una función de costo diferente, asegurando que en caso de existencia de solución, esta sea única.

Definición 4.53. El **hiperplano de margen máximo** es aquel que maximiza el margen geométrico sobre todos los hiperplanos posibles que separan los datos.

4.4.2. Regresión lineal

El problema de la regresión lineal consiste en encontrar una función lineal $f(x) = \langle w, x\rangle + b$ que mejor interpole un conjunto dado de puntos de entrenamiento. La técnica más conocida para resolver este problema es la de **mínimos cuadrados**, que minimiza la suma de los cuadrados de las distancias desde los puntos de entrenamiento al hiperplano.

Definición 4.54. Dado un conjunto de entrenamiento S, con $x_i \in X \subseteq \mathbb{R}^n$ y $y_i \in Y \subseteq \mathbb{R}$, el **problema de la regresión lineal** es encontrar una función lineal f que modele los datos minimizando la función de pérdida cuadrática

$$L(w, b) = \sum_{i=1}^{t} (y_i - (\langle w, x_i\rangle + b))^2.$$

La solución de mínimos cuadrados puede expresarse en términos de la pseudoinversa de la matriz de diseño X.

Teorema 4.9. *Si la matriz $X^T X$ es invertible, la solución del problema de mínimos cuadrados está dada por*

$$w = (X^T X)^{-1} X^T y.$$

4.4.3. Máquinas de aprendizaje lineal en representación dual

Muchas máquinas de aprendizaje lineales tienen una representación dual, donde los datos aparecen solo a través de entradas en la matriz de Gram. Esto tiene importantes consecuencias, especialmente en el desarrollo de SVM.

Teorema 4.10 (Freund y Schapire)**.** *Sea S un conjunto de entrenamiento con $\|x_i\| \leq R$. El número de errores en la primera ejecución del algoritmo del perceptrón está acotado por $\left(\frac{R+\gamma}{\gamma}\right)^2$.*

Demostración. La prueba define un espacio de entrada extendido parametrizado por λ en el cual existe un hiperplano con margen γ que tiene la misma funcionalidad que, (w, b), en datos no vistos. Optimizar la elección de λ dará el resultado deseado. El espacio de entrada extendido tiene una coordenada extra para cada ejemplo de entrenamiento. Las nuevas entradas para el ejemplo x_i son todas cero excepto el valor λ en la i-ésima coordenada adicional. Denotamos este vector extendido por x_i y S el conjunto de entrenamiento correspondiente. Ahora extendemos w con el valor $\gamma d_i/\lambda$ en la i-ésima entrada adicional para dar el vector w'. Observamos que

$$y_i(\langle w', x_i\rangle + b) = y_i(\langle w, x_i\rangle + b) + \xi_i \geq \gamma,$$

mostrando que (w', b) tiene margen γ en S. Sin embargo, $\|w'\| = \sqrt{1 + D^2/\lambda^2}$, por lo que el margen geométrico γ se reduce por este factor. Como los ejemplos de entrenamiento extendidos tienen entradas no nulas en coordenadas diferentes, ejecutar el algoritmo del perceptrón en el primer bucle en S tiene el mismo efecto que ejecutarlo en S, y podemos acotar el número de errores por el Teorema de Novikoff con

$$t \leq \frac{2(R^2 + \lambda^2)(1 + D^2/\lambda^2)}{\gamma^2}.$$

La cota se optimiza eligiendo $\lambda = \gamma/RD$, dando el resultado presentado. □

Definición 4.55. Para un valor fijado de $\gamma > 0$, definimos la **variable de holgura de margen** de un ejemplo (x_i, y_i) con respecto al hiperplano (w, b) y el margen objetivo γ como

$$\xi_i = \text{máx}(0, \gamma - y_i(\langle w, x_i\rangle + b)).$$

Definición 4.56. Un **hiperplano de margen máximo** es aquel que maximiza el margen geométrico sobre todos los hiperplanos posibles que separan los datos.

4.5. Espacios de características y núcleos

En general, las aplicaciones complejas del mundo real requieren espacios de hipótesis más expresivos que las funciones lineales. Con el fin de solventar esta carencia, se propuso la proyección de los datos en un espacio de características de alta dimensión para aumentar la potencia computacional de las máquinas de aprendizaje lineales. Esta técnica se realiza de manera implícita utilizando la representación dual de las máquinas lineales. La ventaja radica en que el número de parámetros ajustables no depende del número de atributos utilizados. Al reemplazar el producto interno con una función núcleo elgida adecuadamente, se puede realizar implícitamente un mapeo no lineal a un espacio de características de alta dimensión sin aumentar el número de parámetros ajustables.

4.5.1. Aprendizaje en el espacio de características

La complejidad de la función objetivo a aprender depende de cómo se representa, y la dificultad de la tarea de aprendizaje puede variar en consecuencia. Idealmente, se debe elegir una representación que coincida con el problema específico de aprendizaje. Un enfoque común en el aprendizaje automático implica cambiar la representación de los datos, mapeando el espacio de entrada X en un nuevo espacio $F = \{\varphi(x)|x \in X\}$.

Demos un ejemplo, consideremos la función objetivo

$$f(m_1, m_2, r) = \frac{Gm_1m_2}{r^2}$$

que expresa la ley de gravitación de Newton en términos de las masas m_1, m_2 y la distancia r. Una máquina lineal no podría representar esta función tal como está, pero un simple cambio de coordenadas

$$(m_1, m_2, r) \to (x, y, z) = (\ln m_1, \ln m_2, \ln r)$$

da la representación

$$g(x, y, z) = \ln f(m_1, m_2, r) = \ln G + \ln m_1 + \ln m_2 - 2 \ln r = c + x + y - 2z,$$

que podría ser aprendida por una máquina lineal.

La selección de características debe ser vista como parte del proceso de aprendizaje mismo, y debe ser automatizada tanto como sea posible. Es un paso algo arbitrario que refleja expectativas previas sobre la función objetivo subyacente. Sin embargo, comprender profundamente la generalización nos permite usar espacios de características de dimensión infinita.

4.5.2. El mapeo implícito en el espacio de características

Para aprender relaciones no lineales con una máquina lineal, se necesita seleccionar un conjunto de características no lineales y reescribir los datos en la nueva representación. Esto equivale a aplicar un mapeo no lineal fijo de los datos a un espacio de características F, en el cual se puede usar la máquina lineal. Los modelos que consideramos serán funciones del tipo

$$f(x) = \sum_{i=1}^{n} \alpha_i \varphi(x_i) \cdot \varphi(x)$$

donde $\varphi : X \to F$ es un mapeo no lineal desde el espacio de entrada a algún espacio de características. Esto permite construir máquinas no lineales en dos pasos: primero, un mapeo no lineal fijo transforma los datos en un espacio de características F, luego, se usa una máquina lineal para clasificarlos en el espacio de características.

Definición 4.57. Un **núcleo** es una función K tal que para todos $x, z \in X$,

$$K(x, z) = \langle \varphi(x), \varphi(z) \rangle,$$

donde φ es un mapeo desde X a un espacio de características con producto interno F.

El uso de núcleos permite mapear los datos implícitamente a un espacio de características y entrenar una máquina lineal en dicho espacio, evitando problemas computacionales inherentes a la evaluación del mapeo de características.

4.5.3. Construcción de núcleos

Proporcionaremos métodos para la creación de funciones de núcleo. Un resultado fundamental que utilizamos es el siguiente:

Teorema 4.11 (Teorema de Mercer)**.** *Sea X un subconjunto compacto de $\mathbb{R}^n$. Supongamos que K es una función continua y simétrica tal que el operador integral $T_K : L^2(X) \to L^2(X)$ definido por*

$$(T_K f)(x) = \int_X K(x, z) f(z)\, dz$$

es positivo, es decir,

$$\int_X \int_X K(x, z) f(x) f(z)\, dx\, dz \geq 0, \forall f \in L^2(X).$$

Entonces, K se puede expandir en una serie convergente uniformemente en $X \times X$ en términos de las funciones propias φ_i de T_K y los valores propios positivos asociados λ_i:

$$K(x, z) = \sum_{i=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(z).$$

Demostración. Sea K una función continua y simétrica en $X \times X$ que define un operador integral positivo T_K en $L^2(X)$. Consideramos la ecuación integral asociada

$$\lambda \varphi(x) = \int_X K(x, z) \varphi(z)\, dz.$$

Dado que T_K es compacto y auto-adjunto, el espectro de T_K consiste en un conjunto discreto de valores propios $\{\lambda_i\}$ con funciones propias ortonormales $\{\varphi_i\}$ que forman una base de $L^2(X)$. Podemos escribir

$$K(x, z) = \sum_{i=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(z).$$

La positividad de T_K garantiza que $\lambda_i \geq 0$ para todo i, y la continuidad de K asegura la convergencia uniforme de la serie. Así, hemos demostrado que K puede representarse como una suma infinita de productos de funciones propias con sus correspondientes valores propios positivos. □

Este teorema sugiere una representación de las características en el espacio de características de Hilbert reproducible, (RKHS). Los núcleos que cumplen con las condiciones del Teorema de Mercer se denominan núcleos de Mercer.

Podemos crear nuevos núcleos combinando núcleos existentes utilizando propiedades de cierre. La siguiente proposición resume algunas de estas propiedades.

Proposición 4.8. *Sean K_1 y K_2 núcleos sobre $X \times X$. Entonces, las siguientes funciones también son núcleos:*

1. $K(x,z) = K_1(x,z) + K_2(x,z)$,
2. $K(x,z) = aK_1(x,z)$ *para* $a \geq 0$,
3. $K(x,z) = K_1(x,z)K_2(x,z)$,
4. $K(x,z) = f(x)f(z)$ *para cualquier función real valorada* f,
5. $K(x,z) = K_3(\varphi(x), \varphi(z))$ *para cualquier función* φ *y núcleo* K_3,
6. $K(x,z) = x^T Bz$ *para cualquier matriz simétrica semidefinida positiva* B.

Demostración. Consideremos un conjunto finito de puntos $\{x_1, \ldots, x_n\}$ en X y las matrices de Gram correspondientes K_1 y K_2 para los núcleos K_1 y K_2. Sea a un vector en $\mathbb{R}^n$.

1. Para $K = K_1 + K_2$:

$$a^T(K_1 + K_2)a = a^T K_1 a + a^T K_2 a \geq 0,$$

ya que K_1 y K_2 son núcleos, por lo tanto, semidefinidos positivos.

2. Para $K = aK_1$ con $a \geq 0$:

$$a^T(aK_1)a = a(a^T K_1 a) \geq 0.$$

3. Para $K = K_1 K_2$:

$$a^T(K_1 K_2)a = (K_1 a)^T K_2 a \geq 0.$$

4. Para $K = f(x)f(z)$:

$$a^T K a = \sum_{i,j} a_i a_j f(x_i) f(x_j) = \left(\sum_i a_i f(x_i) \right)^2 \geq 0.$$

5. Para $K = K_3(\varphi(x), \varphi(z))$:

$$a^T K a = \sum_{i,j} a_i a_j K_3(\varphi(x_i), \varphi(x_j)) \geq 0,$$

ya que K_3 es un núcleo en el espacio de características.

6. Para $K = x^T Bz$ con B simétrica semidefinida positiva:

$$a^T K a = a^T (X^T B X) a = (Xa)^T B (Xa) \geq 0.$$

En cada caso, hemos demostrado que la función combinada es un núcleo. □

4.5.4. Núcleos y procesos gaussianos

Si consideramos la salida de una función $f(x)$, para x fijo, como una variable aleatoria, la colección de estas variables puede ser vista como un proceso estocástico. Un proceso gaussiano es un proceso estocástico para el cual la distribución marginal para cualquier conjunto finito de variables es una distribución gaussiana de media cero. Existe una función de covarianza $K(x, z)$ tal que la matriz de covarianza es positiva definida para todos los conjuntos finitos de puntos de entrada. Esto implica que definir un proceso gaussiano sobre un conjunto de variables indexadas por un espacio X es equivalente a definir un núcleo de Mercer sobre $X \times X$.

Teorema 4.12. *Para cada núcleo de Mercer $K(x, z)$ definido sobre el dominio $X \subseteq \mathbb{R}^d$, existe un espacio de Hilbert de funciones definido sobre X para el cual K es el núcleo reproducible.*

Demostración. Sea $K(x, z)$ un núcleo de Mercer definido sobre $X \subseteq \mathbb{R}^d$. Consideremos el espacio de Hilbert de funciones H_K generado por K con el producto interno definido por

$$\langle f, g \rangle_{H_K} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j K(x_i, z_j),$$

donde $f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$ y $g(x) = \sum_{j=1}^{m} \beta_j K(x, z_j)$. La positividad definida de K garantiza que este producto interno es válido y que H_K es completo. Además, para cualquier $x \in X$, la función $K(\cdot, x)$ pertenece a H_K y satisface

$$\langle f, K(\cdot, x) \rangle_{H_K} = f(x),$$

lo que demuestra que K es el núcleo reproducible de H_K. □

Otro enfoque para obtener un núcleo es a partir de las características, trabajando su producto interno. No es necesario verificar la semidefinitud positiva, ya que esto sigue automáticamente de la definición como un producto interno.

4.6. Teoría de generalización

La introducción de núcleos incrementa considerablemente el poder expresivo de las máquinas de aprendizaje mientras se mantiene la linealidad subyacente, lo que garantiza que el aprendizaje sea manejable. Sin embargo, esta flexibilidad aumentada incrementa el riesgo de sobreajuste, ya que la elección del hiperplano separador se vuelve cada vez más imprecisa, en parte debido al número de grados de libertad.

4.6.1. Aprendizaje correcto probablemente aproximado, (PAC)

El modelo PAC se basa en la suposición de que los datos utilizados en el entrenamiento y la prueba son generados de manera independiente e idénticamente

distribuidos según una distribución fija pero desconocida. Se asume que esta es una distribución sobre emparejamientos entrada/salida $(x, y) \in X \times \{-1, 1\}$.

Definición 4.58. Definimos el **error** de una función de clasificación h en una distribución $\mathcal{D}$ como

$$\text{err}_{\mathcal{D}}(h) = P_{(x,y)\sim\mathcal{D}}[h(x) \neq y].$$

El objetivo del análisis PAC es establecer cotas sobre este error en términos de varios factores, siendo el más determinante el número de ejemplos de entrenamiento utilizados. Estas cotas son útiles para seleccionar entre diferentes clases de modelos, resolviendo así el problema de la selección del modelo.

4.6.2. Teoría de Vapnik-Chervonenkis, (VC)

Para un conjunto finito de hipótesis, no es difícil obtener una cota de la forma de la desigualdad PAC. Supongamos que usamos una regla de inferencia que selecciona cualquier hipótesis h que sea consistente con los ejemplos de entrenamiento en S. La probabilidad de que todas las m muestras independientes sean consistentes con una hipótesis h para la cual $\text{err}_{\mathcal{D}}(h) > \epsilon$, está acotada por

$$P\left(S : h \text{ consistente y } \text{err}_{\mathcal{D}}(h) > \epsilon\right) < (1-\epsilon)^m < e^{-\epsilon m}.$$

Ahora, incluso si asumimos que todas las $|H|$ hipótesis tienen un error grande, la probabilidad de que una de ellas sea consistente con S es a lo sumo

$$|H|e^{-\epsilon m}.$$

Para garantizar que el lado derecho sea menor que δ, establecemos

$$|H|e^{-\epsilon m} < \delta.$$

Esto muestra cómo la complejidad de la clase de funciones H, medida aquí por su cardinalidad, tiene un efecto directo sobre la cota del error.

Teorema 4.13 (Vapnik y Chervonenkis)**.** *Sea H un espacio de hipótesis con dimensión VC d. Para cualquier distribución de probabilidad $\mathcal{D}$ sobre $X \times \{-1, 1\}$, con probabilidad al menos $1-\delta$ sobre m ejemplos aleatorios S, cualquier hipótesis $h \in H$ que sea consistente con S tiene un error no mayor que*

$$err(h) \leq \epsilon(m, H, \delta) = \frac{2(d\log(2em/d) + \log(2/\delta))}{m}.$$

siempre que $d \leq m$ y $m \geq 2/\epsilon$

4.6.3. Dimensión VC y máquinas de aprendizaje lineales

Para aplicar la teoría de VC a las máquinas de aprendizaje lineales, debemos calcular la dimensión VC de una clase de funciones lineales en $\mathbb{R}^n$.

Proposición 4.9. *Sea $\mathcal{F}$ la clase de máquinas de aprendizaje lineales en $\mathbb{R}^n$.*

1. *Dado cualquier conjunto S de $n+1$ ejemplos de entrenamiento en posición general, no colineales, existe una función en $\mathcal{F}$ que clasifica consistentemente S, independientemente de la etiquetación de los puntos en S.*

2. *Para cualquier conjunto de $m > n+1$ entradas, existe al menos una clasificación que no puede ser realizada por ninguna función en $\mathcal{F}$.*

Esto implica que la dimensión VC de la clase de máquinas de aprendizaje lineales en $\mathbb{R}^n$ es $n+1$.

Definición 4.59. La **dimensión VC** de una clase de funciones $\mathcal{H}$ es el tamaño máximo de un conjunto de puntos que puede ser clasificado en todas las formas posibles por $\mathcal{H}$.

La teoría de VC proporciona una cota libre de distribución sobre la generalización de una hipótesis consistente, pero más que eso, puede mostrarse que la cota es de hecho ajustada hasta factores logarítmicos.

Teorema 4.14. *Sea $\mathcal{H}$ un espacio de hipótesis con dimensión VC $d > 1$. Entonces, para cualquier algoritmo de aprendizaje, existen distribuciones tales que con probabilidad al menos δ sobre m ejemplos aleatorios, el error de la hipótesis h devuelta por el algoritmo es al menos*

$$\text{máx}\left(\frac{d-1}{32m}, \frac{1}{16}\left(\frac{\log(1/\delta)}{m}\right)\right).$$

La prueba se basa en construir una distribución de probabilidad que es particularmente difícil de aprender para cualquier algoritmo, demostrando así que la cota inferior es necesaria. En resumen, la teoría de VC proporciona herramientas poderosas para entender y controlar la capacidad de generalización de las máquinas de aprendizaje, permitiendo establecer cotas precisas sobre el error de generalización basado en la complejidad del espacio de hipótesis.

4.6.4. Aplicación de la teoría de VC a SVM

Las SVM son diseñadas para tomar ventaja de distribuciones benignas que permiten buenos resultados de generalización. Las cotas de error para SVM se derivan de la observación de márgenes grandes y su influencia en la dimensión VC efectiva.

Teorema 4.15. *Considerando funciones lineales con vectores de peso unitario en un espacio de producto interno X y fijando $y > 0$, para cualquier distribución de probabilidad $\mathcal{D}$ sobre $X \times \{-1, 1\}$ con soporte en una bola de radio R alrededor del origen, con probabilidad $1-\delta$ sobre m ejemplos aleatorios S, cualquier hipótesis f con margen $m_s(f) > y$ en S tiene un error no mayor que*

$$err(f) \leq \frac{2}{my^2}\left(\log \mathcal{N}(\mathcal{F}, 2m, y/2) + \log\frac{4}{\delta}\right).$$

Se demuestra utilizando la noción de número de cobertura para acotar el espacio de funciones reales que pueden aproximar el comportamiento de toda la clase en una muestra de $2m$ puntos. Este número de cobertura se relaciona con la dimensión VC y proporciona una cota efectiva del error de generalización.

Estas cotas muestran cómo el margen observado durante el entrenamiento puede influir en la capacidad de generalización, destacando la importancia de los márgenes grandes en las SVM para obtener buenos resultados en la práctica.

4.6.5. Cotas Basadas en el Margen

Las cotas basadas en el margen proporcionan una medida de cómo el margen afecta la generalización. Consideramos el margen $m_s(f)$ de una función f en un conjunto de entrenamiento S.

Definición 4.60. El **margen** de un clasificador f respecto a un ejemplo $(x_i, y_i) \in X \times \{-1, 1\}$ se define como

$$\gamma_i = y_i \langle w, x_i \rangle.$$

El **margen mínimo** de f en el conjunto de entrenamiento S es

$$m_s(f) = \min_{i=1,\ldots,m} \gamma_i.$$

Teorema 4.16. *Considerando funciones lineales con vectores de peso unitario en un espacio de producto interno X y fijando $y > 0$, para cualquier distribución de probabilidad $\mathcal{D}$ sobre $X \times \{-1, 1\}$ con soporte en una bola de radio R alrededor del origen, con probabilidad $1 - \delta$ sobre m ejemplos aleatorios S, cualquier hipótesis f con margen $m_s(f) > y$ en S tiene un error no mayor que*

$$err(f) \leq \frac{2}{my^2} \left(\log \mathcal{N}(\mathcal{F}, 2m, y/2) + \log \frac{4}{\delta} \right).$$

La demostración sigue un argumento similar al de las cotas basadas en la dimensión VC, utilizando el número de cobertura para aproximar el comportamiento de la clase de funciones en una muestra de puntos. Al restringir el análisis a funciones con un margen mínimo y, se reduce efectivamente el tamaño del espacio de hipótesis, lo que permite obtener una cota más ajustada.

Teorema 4.17 (Teorema del margen de SVM)**.** *Considerando funciones lineales con vectores de peso unitario en un espacio de producto interno X y fijando $y > 0$, para cualquier distribución de probabilidad $\mathcal{D}$ sobre $X \times \{-1, 1\}$ con soporte en una bola de radio R alrededor del origen, con probabilidad $1 - \delta$ sobre m ejemplos aleatorios S, cualquier hipótesis f con margen $m_s(f) > y$ en S tiene un error no mayor que*

$$err(f) \leq \frac{2}{my^2} \left(\log \mathcal{N}(\mathcal{F}, 2m, y/2) + \log \frac{4}{\delta} \right).$$

Demostración. Primero, consideramos la definición de margen de una función f en un conjunto de entrenamiento S:

$$m_s(f) = \min_{(x_i,y_i)\in S} y_i f(x_i).$$

Si $m_s(f) > y$, esto implica que todos los ejemplos en S están correctamente clasificados con un margen de al menos y.

A continuación, consideremos una cubierta $(y/2)$-cobertura B de $\mathcal{F}$ respecto a la secuencia S. Sea $g \in B$ tal que está dentro de $y/2$ de f. Esto implica que g tiene $\text{err}_S(g) = 0$ y $m_s(g) > y/2$. Si f comete un error en algún punto $x \in S$, entonces g debe tener un margen menor que $y/2$ en x.

Denotemos $(y/2)$-$\text{err}_S(g)$ como el número de puntos en S para los cuales g tiene un margen menor que $y/2$. Podemos acotar el lado derecho de la desigualdad (4.4) por:

$$\mathbb{P}\left(\exists f \in \mathcal{F} : \text{err}(f) = 0, m_s(f) > y, \text{err}(f) > \epsilon\right) \leq \mathbb{P}\left(\exists g \in B : \text{err}(g) = 0, m_s(g) > y/2, (y/2)\text{-err}_S(g) > \frac{m\epsilon}{2}\right).$$

Utilizando un argumento de permutación similar y la unión de límites, obtenemos:

$$\mathbb{P}\left(\exists g \in B : \text{err}(g) = 0, m_s(g) > y/2, (y/2)\text{-err}_S(g) > \frac{m\epsilon}{2}\right) \leq |\mathcal{N}(\mathcal{F}, 2m, y/2)| \exp\left(-\frac{m\epsilon}{2}\right).$$

Para que esta probabilidad sea menor o igual a δ, necesitamos:

$$|\mathcal{N}(\mathcal{F}, 2m, y/2)| \exp\left(-\frac{m\epsilon}{2}\right) \leq \delta.$$

Despejando ϵ, tenemos:

$$\epsilon \geq \frac{2}{m}\left(\log \mathcal{N}(\mathcal{F}, 2m, y/2) + \log\frac{1}{\delta}\right).$$

Por lo tanto, con probabilidad $1-\delta$ sobre m ejemplos aleatorios S, cualquier hipótesis f con $m_s(f) > y$ en S tiene un error no mayor que:

$$\text{err}(f) \leq \frac{2}{my^2}\left(\log \mathcal{N}(\mathcal{F}, 2m, y/2) + \log\frac{4}{\delta}\right).$$

Esto concluye la demostración del teorema. □

4.6.6. Cotas basadas en la dimensión VC

Las cotas basadas en la dimensión VC proporcionan una medida de la complejidad de una clase de funciones y cómo esta afecta la capacidad de generalización.

Teorema 4.18 (Cota basada en la dimensión VC)**.** *Sea H un espacio de hipótesis con dimensión VC $d > 1$. Entonces, para cualquier algoritmo de aprendizaje, existen distribuciones tales que con probabilidad al menos δ sobre m ejemplos aleatorios, el error de la hipótesis h devuelta por el algoritmo es al menos*

$$\text{máx}\left(\frac{d-1}{32m}, \frac{1}{16}\left(\frac{\log(1/\delta)}{m}\right)\right).$$

La prueba se basa en construir una distribución de probabilidad que es particularmente difícil de aprender para cualquier algoritmo, demostrando así que la cota inferior es necesaria.

4.7. Teoría de la optimización

Todas las estrategias inductivas presentadas en la sección anterior tienen un formato similar. La función hipótesis debe elegirse para minimizar o maximizar un funcional. En el caso de las máquinas de aprendizaje lineal, esto equivale a encontrar un vector de parámetros que minimice o maximice, una determinada función de costo, típicamente sujeta a algunas restricciones. La teoría de la optimización es la rama de las matemáticas que se ocupa de caracterizar las soluciones de clases y de desarrollar algoritmos efectivos para encontrarlas. El aprendizaje automático se ha convertido, por tanto, en algo que podemos comprender y abordar utilizando los principios de la teoría de la optimización

Dependiendo de la función de costo específica y de la naturaleza de las restricciones, podemos distinguir varias clases de problemas de optimización para los cuales existen estrategias de solución eficientes. En esta sección describiremos algunos de los resultados que se aplican a casos en los que la función de costo es una función cuadrática convexa, mientras que las restricciones son lineales. Esta clase de problemas de optimización se llama programación cuadrática convexa, y es esta clase la que resulta adecuada para la tarea de entrenar SVM.

La teoría de la optimización no solo nos proporcionará técnicas algorítmicas, sino que también definirá las condiciones necesarias y suficientes para que una función dada sea una solución. Un ejemplo de esto es proporcionado por la teoría de la dualidad, que nos proporcionará una interpretación natural de la representación dual de las máquinas de apredizaje lineal presentadas en las secciones anteriores.

4.7.1. Formulación del problema

La forma general del problema considerado es la de encontrar el máximo o mínimo de una función sujeta a algunas restricciones. El problema general de optimización se puede expresar de la siguiente manera:

Definición 4.61 (Problema de optimización primal)**.** Dadas funciones f, $g_i, i = 1, \ldots, k$, y $h_i, i = 1, \ldots, m$, definidas en un dominio $\Omega \subseteq \mathbb{R}^n$,

$$\min_{w \in \Omega} f(w),$$

sujeto a

$$g_i(w) \leq 0, \quad i = 1, \ldots, k,$$
$$h_i(w) = 0, \quad i = 1, \ldots, m,$$

donde $f(w)$ se llama la función objetivo, y las relaciones restantes se llaman, respectivamente, las restricciones de desigualdad y de igualdad. El valor óptimo de la función objetivo se llama el valor del problema de optimización.

Para simplificar la notación, escribiremos $g(w) \leq 0$ para indicar $g_i(w) \leq 0, i = 1, \ldots, k$. La expresión $h(w) = 0$ lo indica de la misma manera para las restricciones de igualdad. Dado que los problemas de maximización se pueden convertir en problemas de minimización invirtiendo el signo de $f(w)$, la elección de la minimización no representa una restricción. Del mismo modo, cualquier restricción se puede reescribir en la forma anterior.

La región del dominio donde la función objetivo está definida y donde se satisfacen todas las restricciones se llama la región factible, y la denotaremos por

$$R = \{w \in \Omega : g(w) \leq 0, h(w) = 0\}.$$

Una solución del problema de optimización es un punto $w^* \in R$ tal que no existe otro punto $w \in R$ para el cual $f(w) < f(w^*)$. Tal punto también se conoce como un mínimo global. Un punto $w^* \in \Omega$ se llama un mínimo local de $f(w)$ si existe $\epsilon > 0$ tal que $f(w) \geq f(w^*)$, $\forall w \in \Omega$ tal que $\|w - w^*\| < \epsilon$.

Diferentes suposiciones sobre la naturaleza de la función objetivo y las restricciones crean diferentes problemas de optimización.

Definición 4.62. Un problema de optimización en el que la función objetivo, las restricciones de desigualdad y de igualdad son todas funciones lineales se llama **programación lineal**. Si la función objetivo es cuadrática mientras que las restricciones son todas lineales, el problema de optimización se llama **programación cuadrática**.

Una restricción de desigualdad $g_i(w) \leq 0$ se dice que está activa si la solución w^* satisface $g_i(w^*) = 0$; de lo contrario, se dice que está inactiva. En este sentido, las restricciones de igualdad siempre están activas. A veces se introducen cantidades llamadas variables de holgura, denotadas por ξ, para transformar una restricción de desigualdad en una de igualdad, de la siguiente manera:

$$g_i(w) \leq 0 \Leftrightarrow g_i(w) + \xi_i = 0, \quad \text{con } \xi_i \geq 0.$$

Las variables de holgura asociadas con restricciones activas son iguales a cero, mientras que aquellas para restricciones inactivas indican la cantidad de holgura en la restricción.

Consideraremos clases restringidas de problemas de optimización. Primero definimos lo que se entiende por una función convexa y un conjunto convexo.

Definición 4.63. Una **función** $f(w)$ con valores reales se llama **convexa** para $w \in \Omega$ si, $\forall w, u \in \mathbb{R}^n$, y para cualquier $\theta \in (0, 1)$,

$$f(\theta w + (1 - \theta)u) \leq \theta f(w) + (1 - \theta) f(u).$$

Si se cumple una desigualdad estricta, la función se dice estrictamente convexa. Una función que es dos veces diferenciable será convexa si su matriz Hessiana es semidefinida positiva. Una función afín es aquella que puede expresarse en la forma

$$f(w) = Aw + b,$$

para alguna matriz A y vector b. Nótese que las funciones afines son convexas ya que tienen Hessiana cero. Un conjunto $\Omega \subseteq \mathbb{R}^n$ se llama convexo si, $\forall w, u \in \Omega$, y para cualquier $\theta \in (0, 1)$, el punto $\theta w + (1 - \theta)u \in \Omega$.

Si una función f es convexa, cualquier mínimo local w^* del problema de optimización sin restricciones con función objetivo f también es un mínimo global, ya que para cualquier $u \neq w^*$, por la definición de mínimo local, existe θ suficientemente cercano a 1 tal que

$$f(w^*) \leq f(\theta w + (1 - \theta)u).$$

Esto implica que $f(w^*) \leq f(u)$. Es esta propiedad de las funciones convexas la que hace que los problemas de optimización sean tratables cuando las funciones y los conjuntos involucrados son convexos.

Definición 4.64. Un **problema de optimización** en el que el conjunto Ω, la función objetivo y todas las restricciones son convexos se dice que es **convexo**.

Para los propósitos de entrenar SVMs, podemos restringirnos al caso donde las restricciones son lineales, la función objetivo es convexa y cuadrática, y $\Omega = \mathbb{R}^n$; por lo tanto, consideramos programas cuadráticos convexos.

4.7.2. Teoría de lagrange

El propósito de la teoría de Lagrange es caracterizar la solución de un problema de optimización inicialmente cuando no hay restricciones de desigualdad. Los conceptos principales de esta teoría son los multiplicadores de Lagrange y la función de Lagrange. Este método fue desarrollado por Lagrange en 1797 para problemas mecánicos, generalizando un resultado de Fermat de 1629. En 1951, Kuhn y Tucker ampliaron el método para permitir restricciones de desigualdad en lo que se conoce como teoría de Kuhn-Tucker. Estos tres resultados cada vez más generales proporcionarán todo lo que necesitamos para desarrollar soluciones eficientes para la tarea de optimizar SVM. Para facilitar la comprensión, primero introducimos el caso más simple y luego consideramos el tipo de problemas más complejos.

Proposición 4.10 (Fermat)**.** *Una condición necesaria para que w^* sea un mínimo de $f(w)$, $f \in C^1$, es que $\nabla f(w^*) = 0$. Esta condición, junto con la convexidad de f, también es una condición suficiente.*

Demos un ejemplo de este tipo de optimización. Supongamos que deseamos realizar regresión a partir de un conjunto de entrenamiento $S = \{(x_i, y_i)\}_{i=1}^n \subseteq$

$(X \times Y) \subseteq (\mathbb{R}^n \times \mathbb{R})$, generado a partir de la función objetivo $t(x)$. Si asumimos una representación dual de la forma

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x),$$

previamente mostramos que para minimizar la norma del error en el RKHS, debemos minimizar

$$\sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j K(x_i, x_j))^2.$$

La semidefinitud positiva del núcleo K asegura que la función objetivo es convexa. Usando la condición de Fermat, calculamos las derivadas con respecto a α_i y las igualamos a cero, obteniendo

$$-2y_i + 2\sum_{j=1}^{n} \alpha_j K(x_i, x_j) = 0, \quad i = 1, \ldots, n,$$

o

$$y_i = \sum_{j=1}^{n} \alpha_j K(x_i, x_j),$$

donde hemos denotado por G la matriz de Gram con entradas $G_{ij} = K(x_i, x_j)$. Por ende, los parámetros α^* para la solución se pueden obtener como

$$\alpha^* = G^{-1} y.$$

En problemas con restricciones, se necesita definir una función, conocida como lagrangiano, que incorpora información sobre la función objetivo y las restricciones. Además, su estacionariedad puede usarse para detectar soluciones. Precisamente, el lagrangiano se define como la función objetivo más una combinación lineal de las restricciones, donde los coeficientes de la combinación se llaman multiplicadores de Lagrange.

Definición 4.65. Dado un problema de optimización con función objetivo $f(w)$, y restricciones de igualdad $h_i(w) = 0, i = 1, \ldots, m$, definimos la función lagrangiana como

$$L(w, \lambda) = f(w) + \sum_{i=1}^{m} \lambda_i h_i(w),$$

donde los coeficientes λ_i se llaman multiplicadores de Lagrange.

Si un punto w^* es un mínimo local para un problema de optimización con solo restricciones de igualdad, es posible que $\nabla f(w^*) \neq 0$, pero que las direcciones en las que podríamos movernos para reducir f nos hagan violar una o más de las restricciones. Para respetar la restricción de igualdad h_i, debemos movernos perpendicularmente a $\nabla h_i(w^*)$, y para respetar todas las restricciones debemos movernos perpendicularmente al subespacio V generado por $\nabla h_i(w^*)$. Si los

$\nabla h_i(w^*)$ son linealmente independientes, ningún movimiento legal puede cambiar el valor de la función objetivo, siempre que $\nabla f(w^*)$ esté en el subespacio V, o en otras palabras, cuando existan λ_i tales que

$$\nabla f(w^*) = \sum_{i=1}^{m} \lambda_i \nabla h_i(w^*).$$

Esta observación forma la base del segundo resultado de optimización que trata problemas de optimización con restricciones de igualdad.

Teorema 4.19 (Lagrange)**.** *Una condición necesaria para que un punto normal w^* sea un mínimo de $f(w)$ sujeto a $h_i(w) = 0, i = 1, \ldots, m$, con $f, h_i \in C^1$ y f convexa, es que*

$$\nabla_w L(w, \lambda) = 0,$$

para algunos valores λ_i. Las condiciones anteriores también son suficientes, siempre y cuando $L(w, \lambda)$ sea una función convexa de w.

La primera de las dos condiciones da un nuevo sistema de ecuaciones, mientras que la segunda devuelve las restricciones de igualdad. Al imponer las condiciones,resolviendo los dos sistemas, se obtiene la solución.

Daremos un ejemplo para este tipo de problema. Consideremos el problema de encontrar las dimensiones de los lados de una caja w, u, v cuyo volumen sea máximo y cuya superficie sea igual a un valor dado c. El problema se puede reescribir como

$$\text{máx} -wuv,$$

sujeto a

$$wu + uv + vw = c/2.$$

El lagrangiano de este problema es $L = -wuv + \lambda(wu + uv + vw - c/2)$ y las condiciones necesarias para la optimalidad están dadas por las restricciones y por las condiciones de estacionariedad

$$\frac{\partial L}{\partial w} = -uv + \lambda(u + v) = 0,$$

$$\frac{\partial L}{\partial u} = -vw + \lambda(v + w) = 0,$$

$$\frac{\partial L}{\partial v} = -wu + \lambda(w + u) = 0.$$

Estas condiciones implican que $\lambda v(w - u) = 0$ y $\lambda w(u - v) = 0$, cuya única solución no trivial es $w = u = v = \sqrt{c/6}$. Por lo tanto, dado que las condiciones son necesarias para un mínimo y las soluciones triviales tienen volumen cero, la caja de volumen máximo es un cubo.

Definición 4.66 (Distribución de máxima entropía)**.** La entropía de una distribución de probabilidad $p = (p_1, \ldots, p_n)$ sobre un conjunto finito $\{1, 2, \ldots, n\}$ se define como $H(p) = -\sum_{i=1}^{n} p_i \log p_i$, donde naturalmente $\sum_{i=1}^{n} p_i = 1$.

La distribución con máxima entropía se puede encontrar resolviendo un problema de optimización con el lagrangiano

$$L = -\sum_{i=1}^{n} p_i \log p_i + \lambda \left(\sum_{i=1}^{n} p_i - 1 \right),$$

sobre el dominio $\Omega = \{p : p_i \geq 0, i = 1, \ldots, n\}$. Las condiciones de estacionariedad implican que $\log e^{p_i} + \lambda = 0$ para todos i, lo que indica que todos los p_i deben ser iguales a $1/n$. Esto, junto con la restricción, da $p = (1/n, \ldots, 1/n)$. Dado que Ω es convexa, la restricción es afín y la función objetivo es convexa, teniendo una Hessiana diagonal con entradas $(e^{p_i} \ln 2)^{-1}$, esto muestra que la distribución uniforme tiene la máxima entropía.

Dado que las restricciones son iguales a cero, el valor del Lagrangiano en el punto óptimo es igual al valor de la función objetivo. Continuamos considerando el caso más general, en el que el problema de optimización contiene tanto restricciones de igualdad como de desigualdad. Primero damos la definición del lagrangiano generalizado.

Definición 4.67. Dado un problema de optimización con dominio $\Omega \subseteq \mathbb{R}^n$,

$$\min_{w \in \Omega} f(w),$$

sujeto a

$$g_i(w) \leq 0, i = 1, \ldots, k,$$
$$h_i(w) = 0, i = 1, \ldots, m,$$

definimos la **función lagrangiana generalizada** como

$$L(w, \alpha, \lambda) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{m} \lambda_i h_i(w),$$

donde $\alpha_i \geq 0$ y λ_i son los multiplicadores de Lagrange.

Definición 4.68. El **problema dual de Lagrange** del problema primal de la definición 4.61 es el siguiente problema:

$$\max_{\alpha \geq 0} \min_{w \in \Omega} L(w, \alpha, \lambda).$$

El valor de la función objetivo en la solución óptima se llama el valor del problema.

Comenzamos demostrando un teorema conocido como el teorema de dualidad débil, que proporciona una de las relaciones fundamentales entre los problemas primal y dual y tiene dos corolarios útiles.

Teorema 4.20 (Dualidad débil)**.** *Sea $w \in \Omega$ una solución factible del problema primal de 4.61 y (α, λ) una solución factible de su problema dual. Entonces*

$$f(w) \geq \theta(\alpha, \lambda),$$

donde $\theta(\alpha, \lambda) = \inf_{w \in \Omega} L(w, \alpha, \lambda)$.

Demostración. De la definición de $\theta(\alpha, \lambda)$ para $w \in \Omega$ tenemos

$$\theta(\alpha, \lambda) = \inf_{w \in \Omega} \left(f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{m} \lambda_i h_i(w) \right).$$

Dado que w es factible, tenemos $g_i(w) \leq 0$ y $h_i(w) = 0$, mientras que la factibilidad de (α, λ) implica $\alpha_i \geq 0$. Por lo tanto,

$$\theta(\alpha, \lambda) \leq f(w).$$

□

El valor del dual está acotado superiormente por el valor del primal,

$$\sup_{\alpha \geq 0} \theta(\alpha, \lambda) \leq \inf_{w \in \Omega} \{ f(w) : g(w) \leq 0, h(w) = 0 \}.$$

Si $f(w^*) = \theta(\alpha^*, \lambda^*)$, donde $\alpha^* \geq 0$, y $g(w^*) \leq 0, h(w^*) = 0$, entonces w^* y (α^*, λ^*) resuelven los problemas primal y dual respectivamente. En este caso

$$\alpha_i g_i(w^*) = 0, \quad i = 1, \ldots, k.$$

En consecuencia, si intentamos resolver los problemas primal y dual en conjunto, podemos detectar que hemos alcanzado la solución comparando la diferencia entre los valores de los problemas primal y dual. Si esto se reduce a cero, hemos alcanzado el óptimo. Este enfoque se basa en que las soluciones de los problemas primal y dual tienen el mismo valor, algo que no está garantizado en general. La diferencia entre los valores de los problemas primal y dual se conoce como la brecha de dualidad.

Otra forma de detectar la ausencia de una brecha de dualidad es la presencia de un punto de silla.

Definición 4.69. Un **punto de silla** de la función Lagrangiana para el problema primal es un trío

$$(w^*, \alpha^*, \lambda^*), \quad \text{con } w^* \in \Omega, \alpha^* \geq 0,$$

que satisface la propiedad adicional de que

$$L(w^*, \alpha, \lambda^*) \leq L(w^*, \alpha^*, \lambda^*) \leq L(w, \alpha^*, \lambda^*),$$

para todo $w \in \Omega, \alpha \geq 0$.

Nótese que w aquí no está obligado a satisfacer las restricciones de igualdad o desigualdad.

Teorema 4.21. *El trío $(w^*, \alpha^*, \lambda^*)$ es un punto de silla de la función lagrangiana para el problema primal, si y solo si sus componentes son soluciones óptimas de los problemas primal y dual y no existe brecha de dualidad, teniendo los problemas primal y dual el valor*

$$f(w^*) = \theta(\alpha^*, \lambda^*).$$

Ahora citaremos el teorema de dualidad fuerte, que garantiza que los problemas dual y primal tienen el mismo valor para los problemas de optimización que consideraremos.

Teorema 4.22 (Dualidad fuerte)**.** *Dado un problema de optimización con dominio convexo* $\Omega \subseteq \mathbb{R}^n$*,*

$$\min_{w \in \Omega} f(w),$$

sujeto a

$$g_i(w) \leq 0, i = 1, \ldots, k,$$

$$h_i(w) = 0, i = 1, \ldots, m,$$

donde g_i *y* h_i *son funciones afines, es decir,*

$$h(w) = Aw - b,$$

para alguna matriz A *y vector* b*, la brecha de dualidad es cero.*

Estamos ahora en posición de dar el teorema de Kuhn-Tucker que da las condiciones para una solución óptima a un problema de optimización general.

Teorema 4.23 (Kuhn-Tucker)**.** *Dado un problema de optimización con dominio convexo* $\Omega \subseteq \mathbb{R}^n$*,*

$$\min_{w \in \Omega} f(w),$$

sujeto a

$$g_i(w) \leq 0, i = 1, \ldots, k,$$

$$h_i(w) = 0, i = 1, \ldots, m,$$

con $f \in C^1$ *convexa y* g_i, h_i *afines, las condiciones necesarias y suficientes para que un punto normal* w^* *sea un óptimo son la existencia de* α^*, λ^* *tales que*

$$\nabla_w L(w, \alpha^*, \lambda^*) = 0,$$

$$\alpha_i^* g_i(w^*) = 0, \quad i = 1, \ldots, k,$$

$$g_i(w^*) \leq 0, \quad i = 1, \ldots, k,$$

$$h_i(w^*) = 0, \quad i = 1, \ldots, m.$$

La tercera relación se conoce como condición de complementariedad de Karush-Kuhn-Tucker. Implica que para restricciones activas, $\alpha_i^* > 0$, mientras que para restricciones inactivas, $\alpha_i^* = 0$. Además, es posible demostrar que para restricciones activas, al cambiar nuevamente la restricción a $h_i(w) = b_i$, $\alpha_i^* = \left| \frac{\partial f^*}{\partial b_i} \right|$, por lo que el multiplicador de Lagrange representa la sensibilidad del valor óptimo a la restricción. Perturbar las restricciones inactivas no tiene efecto en la solución del problema de optimización.

Una forma de interpretar los resultados anteriores es que un punto de solución puede estar en una de dos posiciones con respecto a una restricción de

desigualdad, ya sea en el interior de la región factible, con la restricción inactiva, o en el límite definido por esa restricción, con la restricción activa. En el primer caso, las condiciones de optimalidad para esa restricción están dadas por el la condición de Fermat, por lo que α_i debe ser cero. En el segundo caso, se puede usar el teorema de Lagrange con un α_i distinto de cero. Por lo tanto, las condiciones dicen que o una restricción es activa, lo que significa $g_i(w^*) = 0$, o el multiplicador correspondiente satisface $\alpha_i^* = 0$. Esto se resume en la ecuación $g_i(w^*)\alpha_i^* = 0$.

4.7.3. Dualidad

El tratamiento lagrangiano de los problemas de optimización convexa lleva a una descripción dual alternativa, que a menudo resulta ser más fácil de resolver que el problema primal, ya que manejar restricciones de desigualdad directamente es difícil. El problema dual se obtiene introduciendo multiplicadores de Lagrange, también llamados variables duales. Los métodos duales se basan en la idea de que las variables duales son las incógnitas fundamentales del problema.

Podemos transformar el primal en un dual simplemente igualando a cero las derivadas del lagrangiano con respecto a las variables primales, y sustituyendo las relaciones obtenidas de vuelta en el Lagrangiano, eliminando así la dependencia de las variables primales. Esto corresponde a calcular explícitamente la función

$$\theta(\alpha, \lambda) = \inf_{w \in \Omega} L(w, \alpha, \lambda).$$

La función resultante contiene solo variables duales y debe maximizarse bajo restricciones más simples. La característica de la expresión resultante para las variables primales coincide exactamente con la representación dual introducida en la sección de máquinas de aprendizaje lineal y, por lo tanto, parecerá muy natural en el contexto en el que la utilizaremos. Por ende, el uso de representaciones duales en las SVM no solo nos permite trabajar en espacios de alta dimensión como se indicó en la sección de espacios de características, sino que también allana el camino para técnicas algorítmicas derivadas de la teoría de optimización.

Demostramos el uso práctico de la dualidad aplicándola al caso especial importante de una función objetivo cuadrática.

$$\min \frac{1}{2} w^T Q w - k^T w,$$

sujeto a

$$Xw \leq c,$$

donde Q es una matriz $n \times n$ definida positiva, k es un vector de n; c es un vector de m, w es la incógnita, y X es una matriz $m \times n$. Suponiendo que la región factible no está vacía, este problema se puede reescribir como

$$\max_{\alpha \geq 0} \min_{w} \left(\frac{1}{2} w^T Q w - k^T w + \alpha^T (Xw - c) \right).$$

El mínimo sobre w no está restringido, y se alcanza en

$$w = Q^{-1}(k - X^T\alpha).$$

Resustituyendo esto de vuelta en el problema original, se obtiene el dual:

$$\text{máx} -\frac{1}{2}\alpha^T P\alpha + \alpha^T d,$$

sujeto a $\alpha \geq 0$, donde $P = XQ^{-1}X^T$, y $d = c - XQ^{-1}k$. Se concluye que el dual de un programa cuadrático es otro programa cuadrático pero con restricciones más simples.

4.8. SVM

Concluiremos este capítulo con las SVM, que son un enfoque de aprendizaje desarrollado originalmente por Vapnik. Es más, son un sistema para entrenar eficientemente las máquinas de aprendizaje lineal en los espacios de características inducidos por kernel. Respetan las teorías de generalización y explotan la teoría de optimización. Una característica importante de estas máquinas es que producen representaciones duales de las hipótesis, resultando en algoritmos extremadamente eficientes. Otra característica es que, debido a las condiciones de Mercer en los kernels, los problemas de optimización correspondientes son convexos y, por lo tanto, no tienen mínimos locales.

4.8.1. Clasificación con SVM

El objetivo de la clasificación con SVM es idear una forma computacionalmente eficiente de aprender hiperplanos separadores en un espacio de características de alta dimensión, donde los hiperplanos escogidos son aquellos que optimizan los límites de generalización.

Definición 4.70. El **clasificador de margen máximo** separa los datos con el hiperplano de margen máximo, optimizando el límite de generalización al minimizar la norma del vector de pesos bajo restricciones de desigualdad lineal.

Proposición 4.11. *Dada una muestra de entrenamiento linealmente separable* $S = \{(x_1, y_1), \ldots, (x_l, y_l)\}$*, el hiperplano* (w, b) *que resuelve el problema de optimización*

$$\textit{minimizar } \frac{1}{2}(w \cdot w),$$

sujeto a $y_i((w \cdot x_i) + b) \geq 1, \quad i = 1, \ldots, l$*, realiza el hiperplano de margen máximo con margen geométrico* $\gamma = \frac{1}{||w||}$.

Demostración. Primero, definimos el margen funcional como la salida de la función $y_i((w \cdot x_i) + b)$. Para un hiperplano con margen funcional igual a 1, se le llama hiperplano canónico. El margen geométrico es el margen funcional del vector de pesos normalizado. Entonces, el problema se reduce a minimizar la norma

del vector de pesos w bajo la restricción de que el margen funcional sea 1. Esto se puede formular como un problema de optimización cuadrática convexa. □

Proposición 4.12. *Considerando una muestra de entrenamiento*

$$S = \{(x_1, y_1), \ldots, (x_l, y_l)\},$$

y suponiendo que los parámetros α_i resuelven el siguiente problema de optimización cuadrática:

$$\text{maximizar } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j),$$

sujeto a $\sum_{i=1}^{l} \alpha_i y_i = 0$ y $\alpha_i \geq 0, \quad i = 1, \ldots, l$, entonces el vector de pesos $w^* = \sum_{i=1}^{l} \alpha_i^* y_i x_i$ realiza el hiperplano de margen máximo con margen geométrico $\gamma = \frac{1}{\|w\|}$.

Demostración. El lagrangiano primal se define como:

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i [y_i((w \cdot x_i) + b) - 1],$$

donde $\alpha_i \geq 0$ son los multiplicadores de Lagrange. Diferenciando con respecto a w y b y estableciendo las derivadas igual a cero, obtenemos:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i,$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0.$$

Sustituyendo estas relaciones en el lagrangiano primal, obtenemos el problema dual de optimización. □

El valor de b no aparece en el problema dual y debe ser encontrado usando las restricciones primales. Esto se logra tomando la media de los valores de b obtenidos para los vectores de soporte positivos y negativos.

Teorema 4.24 (Condiciones de Karush-Kuhn-Tucker)**.** *Las condiciones de complementariedad de Karush-Kuhn-Tucker, (KKT), establecen que las soluciones óptimas α^*, w^*, b^* deben satisfacer:*

$$\alpha_i^* [y_i((w^* \cdot x_i) + b^*) - 1] = 0, \quad i = 1, \ldots, l.$$

Esto implica que solo para los puntos x_i para los cuales el margen funcional es uno, los correspondientes α_i^* son diferentes de cero.

La demostración se sigue de que dadas las soluciones óptimas, estas deben satisfacer las condiciones KKT, solo los vectores que se encuentran en el margen tienen multiplicadores de Lagrange no nulos. Esto se deriva de la propiedad de complementariedad de las condiciones KKT.

Proposición 4.13. *Considerando una muestra de entrenamiento linealmente separable S y suponiendo que los parámetros α_i resuelven el problema dual de optimización:*

$$\textit{maximizar } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j),$$

sujeto a $\sum_{i=1}^{l} \alpha_i y_i = 0$ y $\alpha_i \geq 0, \quad i = 1, \ldots, l$, entonces el vector de pesos $w = \sum_{i=1}^{l} \alpha_i y_i x_i$ realiza el hiperplano de margen máximo con margen geométrico $\gamma = \frac{1}{\|w\|}$.

La demostración sigue de la solución del problema dual y la relación entre el problema primal y dual, utilizando las condiciones KKT para garantizar que las soluciones satisfacen las restricciones primales.

4.8.2. Optimización de margen suave

El clasificador de margen máximo no puede ser utilizado en muchos problemas del mundo real, ya que los datos pueden no ser linealmente separables debido al ruido. Para abordar esto, se introducen las variables de holgura que permiten que las restricciones de margen sean violadas.

Definición 4.71. Las **SVM de margen suave** permiten que las restricciones de margen se transgredan introduciendo variables de holgura ξ_i. El problema de optimización se formula como:

$$\text{minimizar } \frac{1}{2}(w \cdot w) + C \sum_{i=1}^{l} \xi_i,$$

sujeto a $y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, l.$

Proposición 4.14. *Considerando una muestra de entrenamiento S y usando el espacio de características definido implícitamente por el kernel $K(x, z)$, los parámetros α_i^* resuelven el siguiente problema de optimización cuadrática:*

$$\textit{maximizar } W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j),$$

sujeto a $\sum_{i=1}^{l} \alpha_i y_i = 0$ y $0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l$. La regla de decisión dada por $sgn(f(x))$, donde $f(x) = \sum_{i=1}^{l} \alpha_i^* y_i K(x_i, x) + b^*$, es equivalente al hiperplano en el espacio de características definido por el kernel que resuelve el problema de margen suave.

Demostración. El lagrangiano primal para el problema de margen suave se define como:

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2}(w \cdot w) + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i[y_i((w \cdot x_i) + b) - 1 + \xi_i] - \sum_{i=1}^{l} \beta_i \xi_i,$$

donde $\alpha_i \geq 0$ y $\beta_i \geq 0$ son los multiplicadores de Lagrange. Diferenciando con respecto a w, b y ξ_i y estableciendo las derivadas igual a cero, obtenemos:

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i,$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0,$$

$$\alpha_i = C - \beta_i.$$

Sustituyendo estas relaciones en el lagrangiano primal, obtenemos el problema dual de optimización. □

En el caso de margen suave, las variables de holgura ξ_i permiten violaciones en las restricciones de margen, lo que permite manejar datos no separables linealmente y ruidosos. La elección del parámetro C controla el equilibrio entre maximizar el margen y minimizar el error de clasificación.

4.8.3. Regresión con SVM

La regresión con SVM mantiene las características principales del algoritmo de margen máximo: una función no lineal es aprendida por una máquina de aprendizaje lineal en un espacio de características inducido por kernel mientras que la capacidad del sistema es controlada por un parámetro que no depende de la dimensionalidad del espacio.

Definición 4.72. La **función de pérdida insensible a** ϵ se define como:

$$L_\epsilon(x, y, f) = \text{máx}(0, |y - f(x)| - \epsilon).$$

Proposición 4.15. *Para realizar regresión en un conjunto de entrenamiento S usando el espacio de características definido por el kernel $K(x, z)$, los parámetros α_i^* resuelven el siguiente problema de optimización cuadrática:*

$$\textit{maximizar}\, W(\alpha) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)y_i - \epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) - \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j),$$

sujeto a $\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0$ *y* $0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \ldots, l.$

Demostración. El lagrangiano primal para el problema de regresión se define como:

$$L(w,b,\xi,\alpha,\beta) = \frac{1}{2}(w\cdot w)+C\sum_{i=1}^{l}(\xi_i+\xi_i^*)-\sum_{i=1}^{l}\alpha_i[y_i-((w\cdot x_i)+b)-\epsilon-\xi_i]-\sum_{i=1}^{l}\beta_i[\epsilon-(y_i-((w\cdot x_i)+b))-\xi_i^*],$$

donde $\alpha_i, \beta_i \geq 0$ son los multiplicadores de Lagrange. Diferenciando con respecto a w, b y ξ_i, ξ_i^* y estableciendo las derivadas igual a cero, obtenemos:

$$w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)x_i,$$

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0,$$

$$\alpha_i = C - \beta_i, \quad \alpha_i^* = C - \beta_i^*.$$

Sustituyendo estas relaciones en el lagrangiano primal, obtenemos el problema dual de optimización. □

La función de pérdida insensible a ϵ ignora errores dentro de una cierta distancia del valor verdadero, permitiendo obtener soluciones escasas en el espacio de características. Esto asegura una representación eficiente y generalización confiable.

Teorema 4.25. *El valor del margen geométrico y la norma del vector de pesos w en el caso de regresión con pérdida insensible a ϵ se define por:*

$$\|w\| = \sqrt{\sum_{i=1}^{l}(\alpha_i - \alpha_i^*)x_i},$$

y el margen geométrico es $\gamma = \frac{1}{\|w\|}$.

La demostración sigue del problema dual de optimización y la relación entre el problema primal y dual, utilizando las condiciones KKT para garantizar que las soluciones satisfacen las restricciones primales.

Capítulo 5

Fundamentos teóricos

En este capítulo profundizaremos en la teoría necesaria para abordar este TFG. Nos centraremos en las bases de los modelos que utilizamos para crear nuestro análisis, así como en un explicación de donde se obtienen los datos para el proyecto.

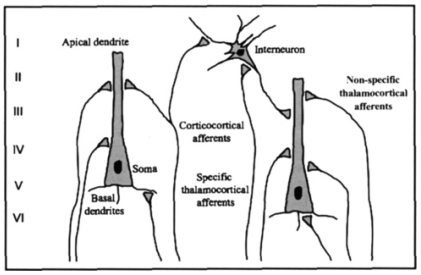
5.1. Electroencefalografía

5.1.1. Introducción

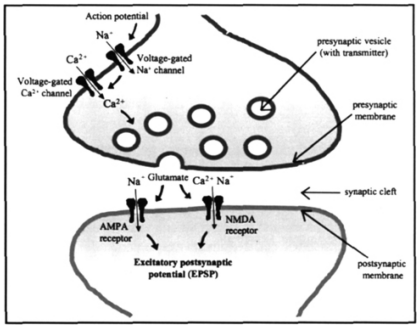
La electroencefalografía, (EEG), es una técnica neurofisiológica que permite el registro de la actividad eléctrica del cerebro. Esta actividad, generada por la actividad sináptica de las neuronas corticales [16], proporciona información sobre los procesos neurales subyacentes a diversas funciones cognitivas y comportamentales. Desde su descubrimiento por Hans Berger en 1929 [17], la EEG ha sido ampliamente utilizada en la investigación científica y aplicaciones clínicas debido a su facilidad de uso, no invasividad y seguridad.

5.1.2. Generación de la señal EEG

La actividad electroencefalográfica surge de los cambios en el potencial de reposo de las células cerebrales, principalmente las neuronas piramidales corticales. Este potencial, generado por la actividad de bombas de membrana que mantienen un equilibrio iónico entre el interior y el exterior celular, establece las bases para la actividad eléctrica cerebral. Aunque los potenciales de acción son fundamentales en la comunicación neuronal, su amplitud intrínsecamente baja no es suficiente para producir la señal EEG detectable en el cuero cabelludo.



(a) Neuronas piramidales corticales [16]



(b) Sinapsis excitatoria [16]

Figura 5.1: Comparación de neuronas piramidales corticales y sinapsis excitatoria.

La señal EEG se origina en las sinapsis de las neuronas piramidales corticales, donde se generan potenciales postsinápticos excitatorios e inhibitorios. Estos potenciales, mediados por la entrada de iones positivos o negativos a través de receptores específicos, se propagan electrotrónicamente a lo largo de las dendritas neuronales. La sincronización de estos eventos neuronales determina la amplitud de la señal EEG, mientras que la desincronización modula su frecuencia, reflejando así los diferentes estados de actividad cerebral, tales como el sueño profundo o la vigilia.

5.1.3. Bandas de frecuencia en EEG

La actividad electroencefalográfica no solo ofrece un registro de la actividad cerebral en tiempo real, sino que también permite el análisis de diferentes bandas de frecuencia [18] que reflejan distintos estados y procesos mentales, un ejemplo lo podemos observar en la figura 5.2. Estas bandas incluyen Delta, Theta, Alpha, Beta y Gamma.

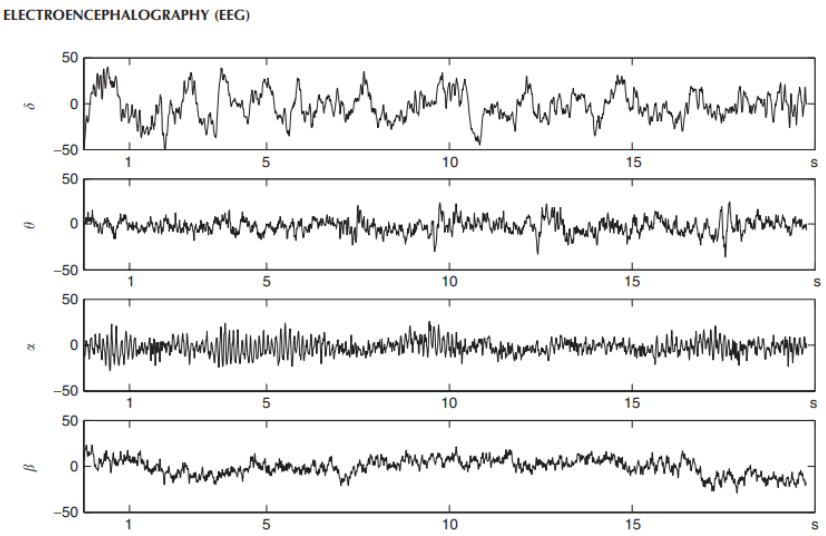


Figura 5.2: Se muestra un ejemplo de las bandas Alpha, Beta, Theta y Delta. Fuente: [19].

La banda Delta, (0-4 Hz), está estrechamente vinculada a tareas cognitivas que implican la detección de un objetivo entre distracciones o estímulos no pertinentes. Las fuentes de estas oscilaciones suelen encontrarse en la corteza frontal y cingulada, y se cree que tienen un papel inhibitorio en procesos cognitivos como la atención.

Las oscilaciones Theta, (4-7 Hz), están estrechamente asociadas con procesos de memoria, reflejando la comunicación con el hipocampo, una región clave para las funciones mnemónicas. Además, la actividad Theta se observa durante la inhibición funcional que sirve a funciones ejecutivas, principalmente en la corteza frontal, regulando otras estructuras cerebrales a través de la inhibición.

Las oscilaciones Alpha, (8-13 Hz), están moduladas durante la estimulación sensorial y reflejan procesos de memoria y atención. Existe una correlación inversa entre las oscilaciones Alpha y el rendimiento cognitivo, sugiriendo una inhibición de las estructuras corticales no relevantes para la tarea en curso.

Las oscilaciones Beta, (14-30 Hz), se observan principalmente durante la realización de tareas motoras y cognitivas que implican interacción sensoriomotora. Se ha propuesto que reflejan si el estado sensoriomotor actual se espera que permanezca estable o cambie en el futuro cercano.

Finalmente, las oscilaciones Gamma, (superiores a 30 Hz), están estrechamente relacionadas con la activación cortical y procesos cognitivos como el procesamiento atento de la información, el mantenimiento activo de los contenidos de la memoria y la percepción consciente. Mientras que las oscilaciones de baja frecuencia se asocian más con la inhibición funcional, las oscilaciones gamma más rápidas se consideran indicativas de activación cortical.

5.1.4. Métodos de extracción

La electroencefalografía ha surgido como una herramienta central en la neurociencia debido a su versatilidad y bajo costo. Su capacidad para capturar incluso los fenómenos más sutiles, como el parpadeo de un ojo, la convierte en una herramienta invaluable para estudiar una amplia gama de procesos cerebrales, desde la actividad básica hasta las complejas interacciones neuronales que subyacen a la cognición y el comportamiento humano. El EEG ofrece la posibilidad de emplear diversos métodos de extracción de señales, que varían en su grado de invasividad y precisión. Se pueden utilizar métodos más invasivos, como la colocación de electrodos intracerebrales, o métodos menos invasivos, como la colocación de electrodos en el cuero cabelludo. Además, una vez adquiridas las señales EEG, se pueden aplicar técnicas de procesamiento de señales para eliminar artefactos no deseados o resaltar patrones de interés, lo que permite un análisis más preciso y específico de la actividad cerebral.

Los métodos invasivos implican la inserción de electrodos directamente en el cerebro para registrar la actividad eléctrica. Esta técnica proporciona una alta resolución espacial al permitir la colocación precisa de electrodos cerca de las regiones de interés cerebral. Los electrodos intracerebrales, por ejemplo, pueden colocarse en áreas específicas del cerebro para estudiar la actividad neuronal con gran precisión. Aunque estos métodos ofrecen una excelente resolución espacial y la capacidad de capturar señales eléctricas directamente de la fuente, su aplicación se limita a entornos clínicos y de investigación debido a la invasión quirúrgica necesaria y los riesgos asociados.

Por otro lado, los métodos no invasivos son menos intrusivos y más ampliamente utilizados en diversas aplicaciones, como se menciona en [20]. Estos métodos registran la actividad cerebral sin requerir la inserción de electrodos en el tejido cerebral. Por ejemplo, se podrían utilizar electrodos colocados en el cuero cabelludo para detectar la actividad eléctrica cortical. Aunque estos métodos ofrecen una resolución espacial ligeramente inferior en comparación con los métodos invasivos, son más seguros, menos costosos y más adecuados para aplicaciones que requieren un monitoreo continuo y no invasivo de la actividad cerebral en entornos clínicos y de investigación.

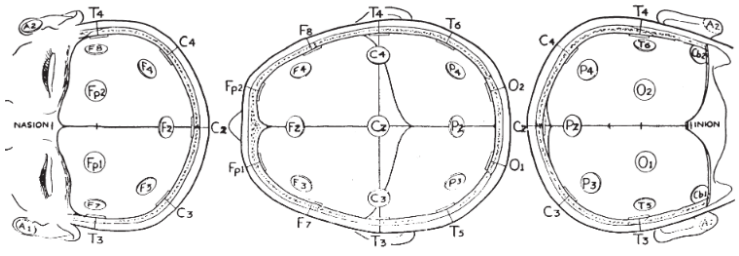


Figura 5.3: Método más popular de colocación de electrodos, sistema 10-20. [19]

En la actualidad, está surgiendo una tecnología innovadora en el ámbito del

EEG que está ganando reconocimiento gracias a su conveniencia, asequibilidad y precisión. Esta tecnología se basa en dispositivos portátiles y de fácil manipulación, como las diademas EEG, que permiten a los individuos llevar a cabo mediciones de su actividad cerebral en cualquier momento y lugar. Ejemplos notables de estos dispositivos incluyen el dispositivo MUSE.

Figura 5.4: Imagen de la diadema MUSE S. [21]

Al proporcionar una alternativa no invasiva y accesible para la captura de señales EEG, estos dispositivos se destacan por su idoneidad en aplicaciones que abarcan desde el biofeedback hasta la investigación clínica y la vigilancia del estado mental. Su creciente aceptación dentro de la comunidad científica ha motivado la realización de numerosos estudios que exploran su utilidad en diversos campos, como se observa en los estudios [22] o [23].

5.1.5. Preprocesamiento de señales EEG

El preprocesamiento de señales EEG es un componente crítico en la investigación neurocientífica, ayuda en la normalización y adecuación de los datos para la eliminación de interferencias indeseadas [24]. La adopción de un esquema de preprocesamiento riguroso, siguiendo estructuras bien establecidas como la propuesta por Marshall et al. (2023) [25], es imperativa para asegurar la integridad de los datos y la confiabilidad de los análisis resultantes.

El proceso inicia con la corrección de la desviación de la media, un paso esencial para anular el desplazamiento de corriente continua en la señal, a través de la sustracción de la media de los datos en cada canal EEG. Esta operación estandariza la señal y eleva la precisión de las mediciones. Acto seguido, se aplica un filtrado mediante un filtro pasa banda Butterworth de segundo orden, cuyo objetivo es depurar la señal de ruido y concentrar el análisis en las frecuencias de interés, potenciando así la detección de los componentes cerebrales relevantes.

Posterior a este filtrado, se procede a la segmentación de la señal en fragmentos de tamaño uniforme, vinculados a eventos específicos o intervalos de tiempo relevantes para el estudio, lo que permite una inspección más refinada

de la actividad cerebral. A continuación, se efectúa un recorte para descartar datos redundantes al final de cada segmento, minimizando así la contaminación por artefactos.

La fase siguiente implica la segmentación de línea base, durante la cual se ajusta y elimina la línea base para cada segmento. Idealmente, se preferiría un período de línea base previo al estímulo; sin embargo, debido a limitaciones como la falta de datos pre-estímulo en dispositivos como MUSE, se opta por usar el inicio de la señal. La etapa de detección de artefactos se centra en la identificación y eliminación de segmentos con alta varianza, los cuales son indicativos de artefactos, siendo este un paso crítico para preservar la validez de los análisis realizados con los datos EEG.

El proceso culmina con el promedio de los segmentos válidos, generando así el potencial evocado, (ERP), por cada condición experimental y canal EEG. El ERP, reflejo de la respuesta cerebral media a un estímulo específico, facilita la identificación de patrones de actividad neuronal asociados al experimento. Este meticuloso esquema de preprocesamiento es esencial no solo para la calidad de los datos sino también para la interpretación precisa de los fenómenos estudiados en la neurociencia cognitiva.

5.2. Machine learning

5.2.1. Introducción

En un mundo cada vez más impulsado por los avances tecnológicos, el ML emerge como una rama de la inteligencia artificial, (IA), prometiendo revolucionar numerosos aspectos de nuestras vidas cotidianas. La introducción al campo del ML marca el inicio de un viaje hacia la automatización inteligente, donde las máquinas no solo procesan datos, sino que también aprenden de ellos para resolver problemas de manera autónoma.

La esencia misma del ML [26] reside en su capacidad para aprender de ejemplos y experiencias, sin necesidad de ser programado explícitamente. En lugar de escribir código de manera convencional, se alimenta a los algoritmos con datos, permitiendo que construyan lógica basada en la información proporcionada [27]. Este enfoque está comenzando a ser ampliamente utilizado en la investigación de encuestas, donde se aplica en el diseño adaptable de encuestas, procesamiento de datos, ajustes por falta de respuesta y ponderación.

El resurgimiento del interés en el ML se atribuye a varios factores, entre ellos el crecimiento exponencial en la disponibilidad de datos, el procesamiento computacional más económico y potente, así como el almacenamiento de datos asequible. Estos avances tecnológicos han allanado el camino para la creación rápida y automática de modelos que pueden analizar conjuntos de datos más grandes y complejos, proporcionando resultados más precisos en menos tiempo, incluso a una escala masiva, como se estudia en [28].

El ML ha encontrado aplicación en una amplia gama de industrias que manejan grandes volúmenes de datos. En el sector financiero, por ejemplo, se utiliza

para identificar oportunidades de inversión y prevenir fraudes, incluso en tecnologías tan novedosas como es blockchain [29]. En el campo de la salud, el ML se integra con dispositivos portátiles y sensores para evaluar la salud del paciente en tiempo real y mejorar los diagnósticos y tratamientos [24]. En la industria petrolera y de gas, se emplea para optimizar la distribución y predecir fallas en los sensores de las refinerías, entre otros usos.

Los gobiernos también están adoptando el ML para analizar datos de múltiples fuentes y obtener información valiosa para aumentar la eficiencia y reducir costos, así como para detectar fraudes y minimizar el robo de identidad. En el comercio minorista, las recomendaciones personalizadas y la optimización de precios son posibles gracias al análisis de datos generado por algoritmos de ML. Del mismo modo, en la industria del transporte, el ML se emplea para identificar patrones y tendencias que optimizan rutas y aumentan la rentabilidad.

En resumen, el ML representa una evolución significativa en la capacidad de las máquinas para aprender y adaptarse, abriendo nuevas oportunidades y desafíos en diversos ámbitos. En este trabajo, profundizaremos en el área de la salud, utilizando este herramienta para clasificar un conjunto de datos obtenidos de señales EEG.

5.2.2. Métodos de aprendizaje

En el vasto y dinámico campo del aprendizaje automático, se vislumbran múltiples enfoques fundamentales que no solo guían, sino que también moldean el proceso de instrucción de las máquinas como se recoge en [30]. Este panorama diverso y en constante evolución se despliega en una serie de metodologías intrincadas, cada una diseñada para abordar una amplia gama de datos y desafíos específicos. Desde la identificación de patrones hasta la toma de decisiones complejas, estos enfoques se entrelazan en un tapiz complejo que define la frontera entre la capacidad humana y la inteligencia artificial. A continuación concretaremos en dichas técnicas.

El aprendizaje supervisado [31] se emplea cuando los datos consisten en variables de entrada y valores objetivo de salida. Aquí, el algoritmo aprende la función de mapeo de las entradas a las salidas. Este enfoque se divide principalmente en dos categorías: clasificación y regresión. La clasificación se aplica cuando la variable de salida pertenece a un conjunto conocido de categorías, como 'positivo' y 'negativo'. En la regresión, la variable de salida es un valor real o continuo, como 'precio' o 'ubicación geográfica'.

El aprendizaje no supervisado, por otro lado, se aplica cuando solo se dispone de datos de entrada y no hay una variable de salida correspondiente. Estos algoritmos modelan los patrones subyacentes en los datos para comprender mejor sus características. Uno de los tipos principales de algoritmos no supervisados es la agrupación [32], donde se descubren grupos inherentes en los datos y se utilizan para predecir resultados para entradas no vistas.

El aprendizaje por refuerzo [33] se aplica cuando la tarea consiste en tomar una secuencia de decisiones hacia una recompensa final. Durante el proceso de aprendizaje, un agente artificial recibe recompensas o penalizaciones por las

acciones que realiza, con el objetivo de maximizar la recompensa total. Ejemplos incluyen agentes de aprendizaje para jugar videojuegos o realizar tareas de robótica con un objetivo final.

Los enfoques híbridos [34] combinan elementos de los enfoques supervisados y no supervisados para abordar desafíos específicos en el aprendizaje automático. El aprendizaje semi-supervisado se sitúa entre los enfoques supervisados y no supervisados y se entrena utilizando una combinación de datos etiquetados y no etiquetados. El aprendizaje auto-supervisado etiqueta automáticamente los datos de entrenamiento encontrando y explotando las relaciones entre diferentes características de entrada de manera no supervisada. El aprendizaje auto-didacta se aplica para resolver tareas de aprendizaje supervisado con datos etiquetados y no etiquetados, utilizando aprendizaje transferido desde datos no etiquetados.

5.2.3. Modelos ML, estudios previos

En la diversidad de métodos que presenta el campo del aprendizaje automático, varios modelos destacan por su eficacia y popularidad. En esta subsección, discutiremos cuatro de estos modelos: *k-Nearest Neighbors*, *Support Vector Machines*, *XGBoost*, y *Random Forest*. Cada uno de estos modelos será posteriormente utilizado en la clasificación de los datos del experimento conducido.

k-Nearest Neighbors

El algoritmo *k-Nearest Neighbors*, (kNN), es una técnica de clasificación que asume que las muestras más cercanas en un espacio de características tienden a pertenecer a la misma categoría [35]. Utiliza la votación de los k vecinos más cercanos para clasificar un dato, donde k es un número entero seleccionado que generalmente es pequeño. En la práctica, el valor óptimo de k es crítico, ya que afecta directamente la precisión y generalización del modelo.

Para determinar la proximidad entre muestras, se emplea un conjunto de métricas de distancia, siendo la más común en la implementación de kNN en scikit-learn la distancia Minkowski con $p = 2$, equivalente a la distancia euclidiana. Esta se define como:

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

donde x y y son vectores en un espacio n-dimensional, representando dos puntos de datos. Esta métrica es preferida para datos continuos, y en el contexto de scikit-learn, es la predeterminada para kNN.

Además de la distancia euclidiana, existen otras métricas de distancia relevantes como la distancia Manhattan (o de bloques de ciudad), donde la distancia se mide como la suma de las diferencias absolutas de sus coordenadas cartesianas:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Esta métrica es útil para datos categóricos donde la rectitud de los caminos es más representativa que la distancia directa.

La distancia Minkowski, más general, engloba ambas métricas (Euclidiana y Manhattan) variando el parámetro p, con $p = 1$ para Manhattan y $p = 2$ para Euclidiana. Se calcula como:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

En la operativa del kNN, el proceso se inicia con el cálculo de la distancia entre el punto de consulta y todos los puntos en el conjunto de entrenamiento para luego identificar los k vecinos más cercanos. Posteriormente, se agregan los valores de la clase o etiqueta de estos vecinos para predecir la clase del dato de consulta, seleccionando la clase que más aparece entre los vecinos [36].

La elección de k es crítica y se puede optimizar a través de la validación cruzada, probando diferentes valores de k y seleccionando aquel que maximice la precisión del modelo. En este sentido, se suele hacer una búsqueda con los $\mathbb{N}$ impares hasta $\sqrt{n}$ donde n es el tamaño de la entrada. Este proceso evita el sobreajuste y provee un rendimiento que posiblemente refleje más fielmente la realidad del conjunto de datos en análisis.

Support Vector Machines

Support Vector Machines, (SVM), [37] es un algoritmo poderoso y versátil para clasificación binaria. Busca el mejor hiperplano que divide el conjunto de datos en dos clases. Este hiperplano se define de manera que maximice el margen entre las dos clases. Matemáticamente, el hiperplano puede describirse como:

$$w \cdot x + b = 0$$

donde w es el vector normal al hiperplano y b es el término de sesgo. El modelo resuelve el siguiente problema de optimización:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

aquí, y_i son las etiquetas de clase de los puntos de entrenamiento x_i, tomando valores de -1 o 1. La solución busca maximizar el margen $\frac{2}{\|w\|}$ entre las clases.

XGBoost

XGBoost, (*extreme Gradient Boosting*), representa una de las implementaciones más populares y eficientes del algoritmo de *Gradient Boosted Trees*, (GBT), un método de aprendizaje supervisado que se basa en la aproximación

de funciones mediante la optimización de funciones de pérdida específicas y la aplicación de diversas técnicas de regularización [38].

El núcleo del modelo *XGBoost* es su función objetivo compuesta, que se define de forma aditiva pero se optimiza de manera iterativa utilizando árboles de decisión de tipo CART, (*Classification and Regression Trees*), como funciones base:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

donde $\mathcal{F}$ es el conjunto de árboles, K el número de árboles y x_i las características de la instancia i. En cada iteración, se añade un nuevo árbol f_t para mejorar las predicciones del modelo. La función objetivo que XGBoost intenta minimizar en la iteración t se compone de dos partes: una función de pérdida $l(y_i, \hat{y}_i)$ y un término de regularización $\Omega(f_k)$, que ayuda a controlar la complejidad del modelo y a prevenir el sobreajuste:

$$\text{Obj}_t = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{K} \Omega(f_k)$$

donde $\hat{y}_i^{(t)}$ es la predicción del modelo en la iteración t.

Para optimizar esta función objetivo, XGBoost utiliza la expansión de Taylor de segundo orden de la función de pérdida l alrededor del valor predicho anterior $\hat{y}_i^{(t-1)}$:

$$l(y_i, \hat{y}_i^{(t)}) \approx l(y_i, \hat{y}_i^{(t-1)}) + g_i \Delta y_i + \frac{1}{2} h_i (\Delta y_i)^2$$

donde g_i y h_i son el gradiente y el hessiano de la función de pérdida, respectivamente, evaluados en $\hat{y}_i^{(t-1)}$, y $\Delta y_i = \hat{y}_i^{(t)} - \hat{y}_i^{(t-1)}$ es el cambio debido al nuevo árbol f_t.

En cada iteración t, XGBoost busca un árbol f_t que minimice esta aproximación de Taylor de la función objetivo. El componente de regularización $\Omega(f)$ es usualmente definido como:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

donde T es el número de hojas del árbol, w_j los valores de las hojas, γ y λ son parámetros que penalizan la cantidad de hojas y la magnitud de los valores en las hojas, respectivamente.

XGBoost implementa un algoritmo *Exact Greedy* para construir cada árbol f_t. Este proceso involucra iterar sobre todas las características y todos los posibles puntos de corte, calculando la "ganancia" de cada división como la reducción en la función objetivo que resultaría de hacer esa división. La división que produce la máxima ganancia es elegida para dividir el nodo.

Con estas herramientas, XGBoost ajusta el conjunto de árboles de decisión para minimizar la función de pérdida global, ajustando su complejidad mediante términos de regularización y utilizando avanzadas técnicas de optimización numérica, lo que le permite ser uno de los algoritmos de aprendizaje automático más poderosos y versátiles disponibles hoy en día.

Random Forest

Es un enfoque de aprendizaje automático basado en el ensamblaje de múltiples árboles de decisión para construir un modelo predictivo más estable y preciso [39]. Este método combina tanto la *bagging* como la selección aleatoria de características para aumentar la precisión y controlar el sobreajuste, siendo efectivo tanto en clasificación como en regresión.

El predictor de *Random Forest* se define como el promedio de las predicciones de B árboles de decisión independientes, cada uno entrenado con un conjunto de datos elegido al azar:

$$\hat{y}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x),$$

donde B es el número total de árboles en el bosque y T_b representa el b-ésimo árbol de decisión.

En la implementación de *scikit-learn*, cada árbol se construye siguiendo un procedimiento detallado:

1. **Selección de subconjuntos:** Se seleccionan muestras de entrenamiento de manera aleatoria con reemplazo para cada árbol.

2. **División de nodos:** Los nodos de cada árbol se dividen utilizando la mejor división entre un subconjunto de características elegidas al azar, optimizando un criterio de pureza.

3. **Crecimiento del árbol:** Los árboles crecen hasta el máximo tamaño permitido sin poda, salvo restricciones explícitas como la profundidad máxima o el número mínimo de muestras por hoja.

La impureza de un nodo se reduce mediante el mejor *split* y se calcula usando:

- **Impureza de Gini para clasificación:**

$$G(p) = 1 - \sum_{k=1}^{K} p_k^2,$$

  donde p_k es la proporción de muestras de la clase k en el nodo.

- **Reducción de varianza para regresión:**

$$\mathrm{Var}(Y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \bar{y})^2,$$

donde y_i son los valores objetivo, $\bar{y}$ es el promedio de estos valores, y N es el número de muestras en el nodo.

La importancia de las características se basa en la suma de la reducción de la impureza de Gini, ponderada por el número de muestras que alcanzan el nodo, promediada a través de todos los árboles.

En Scikit-learn, la importancia de las características se calcula a partir de la reducción de la impureza de Gini de cada árbol y se promedia sobre todos los árboles del bosque:

$$\mathrm{FI}_i = \frac{1}{B} \sum_{b=1}^{B} \mathrm{FI}_{i,b},$$

donde $\mathrm{FI}_{i,b}$ es la importancia de la característica i en el árbol b.

Este enfoque hace que los *Random Forest* sean altamente efectivos para manejar conjuntos de datos grandes con numerosas características y proporciona un rendimiento computacional eficiente, adaptado para resolver problemas complejos de aprendizaje automático.

5.2.4. Modelos ML de contraste

En esta subsección, analizaremos varios modelos de aprendizaje automático que no se han discutido previamente. Estos modelos serán utilizados para contrastar los resultados obtenidos con los modelos previamente mencionados. Cada uno de estos modelos ofrece diferentes enfoques y capacidades en la clasificación de datos, y su implementación y eficacia pueden variar significativamente dependiendo del conjunto de datos y del problema específico.

MLPClassifier

El Perceptrón Multicapa, (*MLPClassifier* [40]), es una red neuronal artificial compuesta por al menos tres capas de nodos: una capa de entrada, una capa oculta y una capa de salida. Cada nodo, o neurona, en una capa se conecta a cada nodo en la siguiente capa con un peso que se ajusta durante el entrenamiento para minimizar el error de predicción.

La salida de una neurona se calcula como:

$$a_i = \sigma \left(\sum_{j=1}^{n} w_{ij} x_j + b_i \right)$$

donde σ es la función de activación (típicamente la tangente hiperbólica o ReLU), w_{ij} son los pesos, x_j son las entradas y b_i es el sesgo. La función de activación σ en el caso de la tangente hiperbólica se define como:

$$\sigma(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

El modelo se entrena utilizando algoritmos de optimización como Adam, que ajustan los pesos para minimizar una función de pérdida, típicamente la entropía cruzada:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

donde y_i es la etiqueta verdadera y $\hat{y}_i$ es la predicción del modelo. Es un modelo de clasificación de ML en el que es difícil obtener los hiperparámetros óptimos para alcanzar la mejor clasificación para nuestro conjunto de datos debido a todas las posibles combinaciones que presenta.

AdaBoost Classifier

El AdaBoost (*Adaptive Boosting* [41])), desarrollado por Yoav Freund y Robert Schapire, es un algoritmo de ensamblaje que combina múltiples clasificadores débiles para formar un clasificador fuerte. AdaBoost ajusta el peso de cada clasificador basado en su precisión, poniendo más peso en los clasificadores que mejoran la predicción en cada iteración. Una característica clave es que AdaBoost puede utilizar cualquier clasificador como base, siempre que estos sean clasificadores débiles. Sin embargo, en la práctica, se suele utilizar árboles de decisión de profundidad uno, conocidos como "stumps".

AdaBoost sigue un proceso iterativo que se puede desglosar en cuatro etapas principales:

1. Inicialización de pesos
2. Entrenamiento de clasificadores débiles
3. Cálculo de parámetros de actualización a partir del error
4. Predicción final del modelo

Al inicio de cada iteración de entrenamiento, *AdaBoost* asigna un peso inicial igual a $\frac{1}{N}$ a cada muestra del conjunto de datos, donde N es el número total de muestras. Estos pesos representan la probabilidad de seleccionar cada muestra durante el muestreo.

$$w_i^{(1)} = \frac{1}{N} \quad \text{para cada muestra } i$$

En cada iteración t, se muestrea el conjunto de datos según los pesos actuales para formar un subconjunto de entrenamiento. Luego, se entrena un clasificador débil $h_t(x)$ utilizando este subconjunto.

Tras el entrenamiento, se calcula el error del clasificador t-ésimo sobre el conjunto de datos ponderado:

$$\epsilon_t = \sum_{i=1}^{N} w_i^{(t)} I(y_i \neq h_t(x_i))$$

donde I es una función indicadora que vale 1 si $y_i \neq h_t(x_i)$ y 0 en caso contrario. A partir de este error, se calcula el peso α_t asignado al clasificador t-ésimo:

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Este parámetro α_t refleja la confianza en el clasificador: cuanto menor es el error ϵ_t, mayor es el peso α_t.

Las muestras mal clasificadas reciben un mayor peso en la siguiente iteración para enfocar el entrenamiento en los ejemplos difíciles:

$$w_i^{(t+1)} = w_i^{(t)} \exp(\alpha_t I(y_i \neq h_t(x_i)))$$

Después, los pesos se normalizan para que sumen 1.

La predicción final de AdaBoost se obtiene mediante una combinación ponderada de las predicciones de todos los clasificadores débiles:

$$F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Para realizar una predicción, se toma el signo de $F(x)$:

$$\text{Predicción} = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

Para entender mejor cómo se deriva el parámetro α_t, consideramos la función de pérdida exponencial utilizada por AdaBoost:

$$E = \sum_{i=1}^{N} w_i^{(t)} \exp(-y_i F_{t-1}(x_i) - y_i \alpha_t h_t(x_i))$$

Donde $F_{t-1}(x)$ es la predicción acumulada hasta la iteración $t-1$. El objetivo es minimizar esta función de pérdida. La derivada de E respecto a α_t es:

$$\frac{\partial E}{\partial \alpha_t} = \sum_{i=1}^{N} w_i^{(t)} \left(-y_i h_t(x_i)\right) \exp(-y_i F_{t-1}(x_i) - y_i \alpha_t h_t(x_i))$$

Al establecer la derivada igual a cero y resolver para α_t, obtenemos:

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

AdaBoost es en resumen un clasificador fuerte y bastante flexible ya que puede trabajar sobre cualquier clasificador débil. Además, es resistente al sobreajuste, especialmente con conjuntos de datos limpios. Sin embargo, también presenta algunas desventajas. En conjuntos con datos con mucho ruido afecta negativamente el rendimiento, a su vez requiere clasificadores que funcionen de manera subóptima por sí mismos.

Extra Trees Classifier

El *Extra Trees Classifier*, (*Extremely Randomized Trees* [42]), es un método de ensamblaje utilizado tanto para tareas de clasificación como de regresión. Es similar a *Random Forest*, pero difiere en cómo se dividen los nodos. En lugar de buscar el mejor corte entre todas las características, *Extra Trees* selecciona los puntos de división de manera aleatoria.

Extra Trees sigue un proceso estructurado que incluye las siguientes etapas:

1. Aleatorización de las divisiones
2. Construcción del conjunto de árboles
3. Predicción conjunta

En cada nodo, en lugar de buscar el mejor punto de corte basado en una medida de calidad, *Extra Trees* selecciona aleatoriamente el punto de corte. Esto añade un nivel adicional de aleatoriedad comparado con los árboles de decisión tradicionales y *Random Forests*. Para un nodo m con característica j, el punto de corte s se elige aleatoriamente dentro del rango de los valores de j:

$$s \sim \text{Uniform}(\text{mín}(X_j), \text{máx}(X_j))$$

Se construyen múltiples árboles de decisión (B árboles), cada uno utilizando un subconjunto aleatorio de las características y aplicando la aleatorización de divisiones en cada nodo. La construcción de cada árbol T_b se realiza sobre un bootstrap sample del conjunto de datos de entrenamiento.

T_b se construye usando un conjunto de características aleatorio

Para realizar una predicción, cada árbol en el conjunto proporciona su predicción, y estas se combinan. En tareas de clasificación, se utiliza la votación mayoritaria, y en tareas de regresión, se calcula el promedio de las predicciones. Para la clasificación, la predicción combinada $\hat{y}(x)$ se define como:

$$\hat{y}(x) = \text{mode}\{T_b(x) \mid b = 1, \ldots, B\}$$

Para la regresión, la predicción combinada $\hat{y}(x)$ se define como:

$$\hat{y}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

Extra Trees tiene varias ventajas significativas. La aleatoriedad adicional ayuda a prevenir el sobreajuste, haciendo que el modelo sea más robusto a los datos ruidosos. Además, puede ser más rápido de entrenar en comparación con otros métodos debido a la simplicidad de la selección de cortes. Finalmente, la naturaleza aleatoria del algoritmo reduce la necesidad de ajuste preciso de hiperparámetros, haciéndolo menos sensible a configuraciones específicas y, por tanto, más fácil de implementar en la práctica.

CatBoost Classifier

El *CatBoost Classifier* [43] es un algoritmo de boosting de gradiente desarrollado por Yandex, diseñado para tareas de clasificación y regresión. *CatBoost* destaca por su capacidad de manejar variables categóricas de manera eficiente sin necesidad de codificación manual. Utiliza una técnica conocida como *Ordered Boosting* para abordar las dificultades asociadas con características categóricas de alta cardinalidad, permitiendo que CatBoost procese datos categóricos automáticamente y ahorrando tiempo y esfuerzo al usuario.

CatBoost utiliza boosting de gradiente, una técnica poderosa de aprendizaje por ensamblado que combina modelos de predicción débiles, a menudo árboles de decisión, para construir un modelo predictivo robusto. Este proceso funciona añadiendo iterativamente nuevos modelos al ensamblado, cada uno entrenado para corregir los errores de los modelos anteriores. La idea central es aumentar la precisión del modelo enfocándose en los ejemplos mal clasificados.

Una de las principales características de *CatBoost* es su capacidad para manejar características categóricas sin necesidad de preprocesamiento extenso. Implementa una técnica novedosa llamada *Ordered Boosting*, que genera una representación numérica al permutar las variables categóricas. Este método mantiene la información de las categorías mientras permite al modelo utilizar la poderosa técnica de boosting de gradiente.

El algoritmo *CatBoost* comienza con una suposición inicial, a menudo el promedio de la variable objetivo. Luego, construye gradualmente un ensamblado de árboles de decisión, donde cada árbol reduce los errores o residuales de los árboles anteriores. La regularización L2, también conocida como regularización de cresta, introduce un término de penalización en la función de pérdida para prevenir el sobreajuste y mejorar la capacidad de generalización del modelo. En el contexto de *CatBoost*, la regularización L2 es una característica clave que ayuda a controlar la complejidad de los árboles potenciados, añadiendo un término de regularización a la función de pérdida utilizada durante el proceso de entrenamiento:

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \lambda ||f_t||^2$$

donde l es la función de pérdida, λ es el término de regularización y f_t es el modelo en la iteración t.

El algoritmo construye iterativamente el ensamblado de árboles minimizando la función de pérdida utilizando el descenso de gradiente. En cada iteración, calcula el gradiente negativo de la función de pérdida con respecto a las predicciones actuales y ajusta un nuevo árbol al gradiente negativo. La tasa de aprendizaje determina el tamaño del paso tomado durante el descenso de gradiente. Este proceso se repite hasta que se hayan añadido un número predeterminado de árboles o se haya cumplido un criterio de convergencia. Al realizar predicciones, *CatBoost* combina las predicciones de todos los árboles en el ensamblado, lo que resulta en modelos altamente precisos y fiables.

Dado un conjunto de datos de entrenamiento con N muestras y M características, donde cada muestra se denota como (x_i, y_i), siendo x_i un vector de M características y y_i la variable objetivo correspondiente, CatBoost tiene como objetivo aprender una función $F(x)$ que prediga la variable objetivo y:

$$F(x) = F_0(x) + \sum_{m=1}^{M} \sum_{i=1}^{N} f_m(x_i)$$

donde, $F(x)$ representa la función de predicción global que *CatBoost* intenta aprender. Toma un vector de entrada x y predice la variable objetivo y. $F_0(x)$ es la suposición inicial o la predicción base. A menudo se establece como la media de la variable objetivo en el conjunto de datos de entrenamiento. Este término captura el comportamiento promedio general de la variable objetivo. $\sum_{m=1}^{M}$ representa la suma sobre el ensamblado de árboles. M denota el número total de árboles en el ensamblado. $\sum_{i=1}^{N}$ representa la suma sobre las muestras de entrenamiento. N denota el número total de muestras de entrenamiento. $f_m(x_i)$ representa la predicción del m-ésimo árbol para la i-ésima muestra de entrenamiento. Cada árbol en el ensamblado contribuye a la predicción global haciendo su propia predicción para cada muestra de entrenamiento.

La ecuación establece que la predicción global $F(x)$ se obtiene sumando la suposición inicial $F_0(x)$ con las predicciones de cada árbol $f_m(x_i)$ para cada muestra de entrenamiento. Esta suma se realiza para todos los árboles (m) y todas las muestras de entrenamiento (i).

CatBoost incorpora técnicas de regularización para prevenir el sobreajuste, introduciendo penalizaciones durante el proceso de entrenamiento para desalentar la complejidad excesiva del modelo y mejorar su capacidad de generalización. La combinación de estas técnicas avanzadas permite a *CatBoost* producir modelos altamente precisos y eficientes para una amplia gama de tareas de aprendizaje automático.

HistGradientBoostingClassifier

El *HistGradientBoostingClassifier* [44] es una implementación avanzada del algoritmo de *boosting* de gradiente que emplea histogramas para acelerar el proceso de entrenamiento de árboles de decisión. Esta técnica es altamente eficiente para manejar grandes conjuntos de datos, proporcionando una notable mejora en la velocidad de entrenamiento sin sacrificar la precisión del modelo.

El algoritmo sigue el enfoque general del *boosting* de gradiente, donde un modelo fuerte se construye combinando múltiples modelos débiles, típicamente árboles de decisión, entrenados de manera secuencial. La idea central es que cada modelo adicional se entrena para corregir los errores de los modelos anteriores.

La predicción de un modelo de *boosting* basado en histogramas se puede expresar como

$$\hat{y}_i = \sum_{t=1}^{T} \eta f_t(x_i)$$

donde $\hat{y}_i$ es la predicción para la i-ésima muestra, T es el número total de iteraciones o árboles en el modelo, η es la tasa de aprendizaje que controla el tamaño del paso durante el entrenamiento y $f_t(x_i)$ es la predicción del árbol en la iteración t para la muestra x_i.

El proceso de entrenamiento se inicia con una predicción inicial, a menudo la media de las etiquetas en el caso de regresión o la probabilidad a priori de las clases en el caso de clasificación. Luego, en cada iteración, el algoritmo ajusta un nuevo árbol de decisión a los residuos, que son la diferencia entre las predicciones actuales y los valores reales.

La función de pérdida $L(y_i, \hat{y}_i)$ utilizada es típicamente la entropía cruzada para problemas de clasificación y el error cuadrático medio para problemas de regresión. Estas funciones de pérdida se minimizan iterativamente para ajustar los modelos:

$$L = \sum_{i=1}^{N} l(y_i, \hat{y}_i)$$

donde N es el número total de muestras y $l(y_i, \hat{y}_i)$ es la función de pérdida individual para la i-ésima muestra.

Para mejorar la eficiencia del entrenamiento, el *HistGradientBoostingClassifier* utiliza histogramas para discretizar las características continuas. Este enfoque reduce significativamente el costo computacional asociado con la búsqueda de los mejores puntos de división en los árboles de decisión. Los valores de las características se agrupan en *bins*, y las mejores divisiones se encuentran examinando estos bins en lugar de los valores individuales de las características.

Formalmente, dado un conjunto de características X con M bins, la búsqueda del mejor punto de división en un nodo se realiza evaluando cada *bin* m para encontrar el punto de división s_m que minimiza la función de pérdida:

$$s_m = \arg\min_{s} \sum_{i \in \text{bin}(m)} L(y_i, \hat{y}_i)$$

donde $\text{bin}(m)$ denota el conjunto de muestras que caen en el m-ésimo *bin*.

El uso de histogramas permite una computación eficiente y paralelizable, lo que es especialmente beneficioso para grandes volúmenes de datos. Además, este método reduce la varianza en la selección de puntos de división, lo que contribuye a la estabilidad del modelo final.

En resumen, el *HistGradientBoostingClassifier* combina la potencia del boosting de gradiente con la eficiencia de los histogramas para ofrecer un rendimiento robusto y rápido en tareas de clasificación y regresión sobre grandes conjuntos de datos.

TabNetClassifier

TabNet [45] es un modelo de red neuronal diseñado para trabajar con datos tabulares. Utiliza una arquitectura basada en atención que permite al modelo

seleccionar de manera adaptativa las características más relevantes en cada paso de la red.

La arquitectura de *TabNet* se distingue por su capacidad para realizar una selección de características de manera dinámica y adaptativa a través de múltiples pasos de decisión. Esta capacidad es especialmente útil para datos tabulares, donde las relaciones entre características pueden ser complejas y no lineales. La atención permite que el modelo se enfoque en diferentes subconjuntos de características en diferentes momentos, mejorando así la precisión y la interpretabilidad del modelo.

La salida de la red *TabNet* se calcula como

$$y_i = f(x_i; \theta)$$

donde f es una función aprendida que representa la red neuronal, x_i es el vector de características de la i-ésima muestra, y θ son los parámetros del modelo que se ajustan durante el entrenamiento.

El proceso de entrenamiento implica la optimización de una función de pérdida, que mide la discrepancia entre las predicciones del modelo y las etiquetas verdaderas. Para problemas de clasificación, una función de pérdida comúnmente utilizada es la entropía cruzada, definida como

$$L = -\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

donde N es el número total de muestras, y_i es la etiqueta verdadera de la i-ésima muestra, y $\hat{y}_i$ es la predicción del modelo para esa muestra. La entropía cruzada penaliza fuertemente las predicciones incorrectas, guiando el proceso de entrenamiento para mejorar la precisión del modelo.

La arquitectura de *TabNet* incorpora varias capas de atención y decisión que trabajan juntas para refinar las predicciones. Cada capa de atención se enfoca en diferentes características del conjunto de datos, permitiendo que el modelo capture una variedad de patrones y relaciones. Las capas de decisión combinan esta información para generar la predicción final.

Formalmente, cada paso de atención y decisión puede ser representado como una transformación no lineal de las características de entrada, seguida de una combinación ponderada de estas transformaciones. Esto se puede expresar como

$$h_t = g_t(a_t(x_i, \theta_t))$$

donde h_t es la salida de la capa de decisión t, g_t es una función de activación, a_t es la función de atención para la capa t, y θ_t son los parámetros específicos de la capa. La salida final del modelo es una combinación de las salidas de todas las capas de decisión.

En resumen, *TabNet* es un modelo de red neuronal altamente eficiente y preciso para datos tabulares, gracias a su arquitectura basada en atención que permite una selección adaptativa de características. Este enfoque no solo mejora

la precisión de las predicciones, sino que también facilita la interpretación del modelo, permitiendo a los usuarios entender mejor cómo se toman las decisiones.

Gaussian Naive Bayes

El *Gaussian Naive Bayes* [46] es un clasificador probabilístico basado en el teorema de Bayes, que asume que las características siguen una distribución normal, (gaussiana). Este clasificador es ampliamente utilizado debido a su simplicidad y eficacia en una variedad de tareas de clasificación.

La probabilidad de que una muestra x pertenezca a una clase y se calcula utilizando el teorema de Bayes:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

donde $P(x|y)$ es la probabilidad condicional de x dado y, $P(y)$ es la probabilidad a priori de la clase y y $P(x)$ es la probabilidad marginal de la muestra x.

Para el *Gaussian Naive Bayes*, se asume que las características x_i son independientes entre sí y que siguen una distribución normal. Bajo esta suposición, la probabilidad condicional $P(x|y)$ se puede descomponer como el producto de las probabilidades individuales de cada característica:

$$P(x|y) = \prod_{i=1}^{n} P(x_i|y)$$

donde n es el número de características.

En el caso de una distribución normal, la probabilidad condicional $P(x_i|y)$ se calcula como:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

donde μ_y y σ_y son la media y la desviación estándar de la característica x_i para la clase y.

La media μ_y y la desviación estándar σ_y se estiman a partir del conjunto de datos de entrenamiento para cada clase y. Una vez estimados estos parámetros, el modelo puede calcular las probabilidades condicionales para nuevas muestras y asignarlas a la clase con la mayor probabilidad a posteriori.

El clasificador *Gaussian Naive Bayes* es eficiente y se comporta bien en problemas donde la suposición de independencia de características es razonable. Aunque esta suposición puede no ser realista en todos los casos, el clasificador aún puede ofrecer un rendimiento competitivo, especialmente en dominios con grandes conjuntos de datos y características relativamente simples.

La asignación de clase para una nueva muestra x se realiza seleccionando la clase y que maximiza la probabilidad a posteriori $P(y|x)$. Esto se puede expresar como:

$$\hat{y} = \arg\max_y P(y|x) = \arg\max_y \left(P(x|y)P(y)\right)$$

dado que $P(x)$ es constante para todas las clases y puede ser ignorada en la maximización.

En resumen, el *Gaussian Naive Bayes* es un clasificador basado en el teorema de Bayes y la suposición de que las características siguen una distribución normal, lo que permite una implementación eficiente y un buen rendimiento en muchos problemas de clasificación.

Stacking Classifier

El *Stacking*, o apilamiento [47], es una técnica de ensamblaje que combina múltiples modelos de aprendizaje automático para obtener una predicción final más robusta. La idea central es utilizar las predicciones de varios modelos base (también llamados modelos de nivel 0) como entradas para un modelo de nivel superior (también llamado meta-modelo o modelo de nivel 1), que aprende a hacer una predicción final basándose en estas predicciones.

Funcionamiento del Stacking:

1. **Entrenamiento de modelos base:**
   - Se entrenan varios modelos base $h_1, h_2, \ldots, h_M$ utilizando el conjunto de datos de entrenamiento original $\{(x_i, y_i)\}_{i=1}^N$.
2. **Generación de predicciones de modelos base:**
   - Una vez entrenados, los modelos base se utilizan para hacer predicciones sobre el conjunto de datos de entrenamiento mediante validación cruzada. Las predicciones hechas por estos modelos base sobre los datos de entrenamiento se utilizan como características para el meta-modelo.
   - Por ejemplo, si tenemos M modelos base y N instancias de entrenamiento, creamos un nuevo conjunto de características Z donde cada $z_i = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$.
3. **Entrenamiento del meta-modelo:**
   - Se construye un nuevo conjunto de datos $Z = \{(z_i, y_i)\}_{i=1}^N$ donde las características z_i son las predicciones hechas por los modelos base. Este nuevo conjunto de datos se utiliza para entrenar el meta-modelo g.
   - El meta-modelo aprende a combinar las predicciones de los modelos base de manera óptima minimizando una función de pérdida $L(y, \hat{y})$.
4. **Predicción final:**

- Para realizar una predicción final sobre un nuevo conjunto de datos, primero se hacen predicciones con los modelos base:

$$\hat{z}_i = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$$

- Estas predicciones se pasan al meta-modelo, que produce la predicción final:

$$\hat{y}_i = g(\hat{z}_i)$$

Supongamos que tenemos M modelos base y queremos predecir y a partir de x. Los modelos base hacen predicciones:

$$h_j(x_i) \quad \text{para} \quad j = 1, 2, \ldots, M$$

Estas predicciones se combinan en un vector:

$$z_i = [h_1(x_i), h_2(x_i), \ldots, h_M(x_i)]$$

El meta-modelo g toma estos vectores z_i y produce una predicción final:

$$\hat{y}_i = g(z_i)$$

La función de pérdida $L(y, \hat{y})$ que el meta-modelo minimiza puede ser, por ejemplo, la entropía cruzada para clasificación:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Voting Classifier

El *Voting*, o votación [48], es otra técnica de ensamblaje que combina las predicciones de varios modelos base para realizar una predicción final. Existen dos tipos principales de votación: votación dura (*hard voting*) y votación suave (*soft voting*).

Funcionamiento del voting:

1. **Entrenamiento de modelos base:**
   - Se entrenan varios modelos base $h_1, h_2, \ldots, h_M$ utilizando el conjunto de datos de entrenamiento original $\{(x_i, y_i)\}_{i=1}^{N}$.
2. **Predicción:**
   - Cada uno de los modelos base hace predicciones sobre el conjunto de datos de prueba.
3. **Votación:**
   - En la *votación dura* (*hard voting*), la clase final predicha es la que recibe la mayoría de los votos de los modelos base:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_M(x)\}$$

- En la *votación suave* (*soft voting*), se promedian las probabilidades de las clases predichas por los modelos base, y la clase con la mayor probabilidad promedio es la predicha:

$$\hat{p}(y = k|x) = \frac{1}{M} \sum_{j=1}^{M} p_j(y = k|x)$$

$$\hat{y} = \arg\max_k \hat{p}(y = k|x)$$

Para la votación dura, supongamos que cada modelo base h_j predice una clase $y \in \{1, 2, \ldots, K\}$:

$$h_j(x_i) \quad \text{para} \quad j = 1, 2, \ldots, M$$

La predicción final es la clase que recibe la mayoría de los votos:

$$\hat{y}_i = \text{mode}\{h_1(x_i), h_2(x_i), \ldots, h_M(x_i)\}$$

Para la votación suave, cada modelo base h_j produce una distribución de probabilidad sobre las clases:

$$p_j(y = k|x_i) \quad \text{para} \quad k = 1, 2, \ldots, K$$

La probabilidad promedio para cada clase es:

$$\hat{p}(y = k|x_i) = \frac{1}{M} \sum_{j=1}^{M} p_j(y = k|x_i)$$

La clase final es aquella con la mayor probabilidad promedio:

$$\hat{y}_i = \arg\max_k \hat{p}(y = k|x_i)$$

Con estos modelos, se espera obtener una comprensión más completa y robusta del rendimiento de los algoritmos de aprendizaje automático en el problema específico abordado en este trabajo.

5.2.5. Particiones de conjuntos de datos

La partición de conjuntos de datos en componentes de entrenamiento y prueba es fundamental en ML, permitiendo a los algoritmos aprender de una porción de los datos (entrenamiento, *train*) y luego validar su aprendizaje con otra porción independiente (prueba, *test*).

Estrategias de partición

En la investigación de datos biomédicos y en estudios donde las características individuales son críticas, es esencial elegir una estrategia de partición de datos adecuada. Las estrategias más comunes en estos contextos son la partición *subject-wise* y la partición *record-wise*. Cada enfoque presenta ventajas y desventajas en términos de cómo manejan la generalización del modelo, el sesgo, la eficiencia en el uso de los datos y el balance de los mismos [49].

La estrategia *subject-wise* consiste en dividir el conjunto de datos de manera que todos los registros pertenecientes a un sujeto específico se incluyan en un solo grupo, ya sea de entrenamiento o de prueba. Una de las principales ventajas de esta estrategia es que ayuda a reducir el sesgo de información, evitando que el modelo aprenda y reconozca injustamente detalles específicos de los sujetos vistos durante el entrenamiento. Además, al probar el modelo con sujetos completamente nuevos, se mejora su capacidad de generalización a individuos no vistos. Sin embargo, este método tiene desventajas significativas, como el desbalance de datos, ya que sujetos con mayor número de registros pueden sesgar los resultados. Además, se limita la cantidad de datos disponibles para el entrenamiento al asignar todos los registros de un sujeto a un solo conjunto.

Por otro lado, la partición *record-wise* distribuye cada registro individualmente entre el conjunto de entrenamiento y el de prueba, sin considerar a qué sujeto pertenecen. Esto incrementa la diversidad de datos en ambos conjuntos, permitiendo que el modelo aprenda patrones más complejos y generalizables. Además, este enfoque maximiza la utilización de los datos disponibles, aumentando la eficacia del proceso de entrenamiento. No obstante, esta estrategia también conlleva riesgos, como la contaminación de datos, donde el modelo podría memorizar características específicas de sujetos que aparecen en ambos conjuntos, inflando artificialmente su rendimiento. Asimismo, la capacidad de generalización del modelo puede verse afectada, ya que es evaluado con datos de sujetos que ya ha visto durante el entrenamiento.

Estas diferencias se presentan en una tabla comparativa que resume y compara las dos estrategias de partición de datos 5.1.

Aspecto	Partición *subject-wise*	Partición *record-wise*
Reducción de sesgo	Alta, evita el sesgo de información al no mezclar datos de sujetos.	Baja, riesgo de contaminación de datos entre entrenamiento y prueba.
Generalización	Mejorada, al evaluar con sujetos no vistos.	Reducida, debido a la evaluación en parte con sujetos ya vistos.
Diversidad de datos	Limitada, al agrupar todos los registros de un sujeto.	Alta, cada registro se maneja independientemente.
Eficiencia en uso de datos	Menor, restringe la cantidad de datos disponibles para entrenamiento.	Mayor, maximiza la cantidad de información disponible.
Balance de datos	Puede ser desbalanceado, afectando el rendimiento del modelo.	Más balanceado, al distribuir registros de manera individual.

Cuadro 5.1: Comparación de las estrategias de partición de datos

División de cada partición

Los esquemas de partición varían en función de las necesidades del proyecto y las características del conjunto de datos, dicha subdivisión es crítica para evaluar la generalización de un modelo a nuevos datos, previniendo problemas como el sobreajuste. A continuación, se presentan algunas de las particiones más comunes utilizadas en estudios de ML:

- **Partición 50/50**: Los datos se dividen equitativamente entre entrenamiento y prueba. Este esquema es útil para conjuntos de datos de tamaño considerable, asegurando que haya suficiente información tanto para el entrenamiento como para la validación. Un ejemplo de su uso se encuentra en el estudio de Skutt (2023) [6], donde esta partición fue seleccionada para equilibrar la capacidad de entrenamiento y la evaluación del modelo.

- **Partición 80/20**: Asigna el 80 % de los datos al entrenamiento y el 20 % a la prueba. Este enfoque es preferido cuando se dispone de un volumen de datos relativamente grande, permitiendo un entrenamiento exhaustivo y una validación efectiva. Hussain (2023) aplicó esta partición en su investigación sobre modelos de ML explicables [13].

- **Validación cruzada, (Cross-fold validation)**: Implica dividir el conjunto de datos en k subconjuntos, utilizando cada uno como conjunto de prueba en iteraciones sucesivas, mientras los restantes sirven de entrenamiento. Proporciona una evaluación comprensiva del modelo, aprovechando todos los datos para entrenamiento y prueba. Ejemplos de su aplicación incluyen la validación cruzada de 3 subconjuntos en el estudio de Daly (2023) [7] y de 5 subconjuntos en la investigación de Alirezaei (2017) sobre la detección de anomalías [10].

- **Leave-one-out subject**: Cada dato o sujeto se utiliza una vez como conjunto de prueba mientras que el resto forma el conjunto de entrenamiento [50]. Este método es particularmente útil para conjuntos de datos pequeños o en

situaciones donde cada caso individual es crítico.

Selección de la partición para el experimento

En nuestro estudio que emplea datos EEG en el ámbito de la salud, hemos seleccionado el enfoque de partición *subject-wise* debido a su relevancia en contextos médicos y biomédicos, donde es crucial poder generalizar a nuevos sujetos. Esta estrategia asegura que los datos de entrenamiento y prueba provienen de sujetos distintos, lo que es esencial para evaluar la efectividad del modelo en condiciones realistas y desconocidas, minimizando así el riesgo de sesgo en los resultados.

Dentro de este marco, hemos decidido implementar específicamente el método *Leave-one-out Subject* para la partición de nuestros datos. Este método es particularmente adecuado dado el limitado volumen de datos disponibles, permitiendo maximizar el aprovechamiento de cada observación individual para el entrenamiento y validación del modelo. Usar *Leave-one-out* proporciona una evaluación exhaustiva y detallada de la capacidad del modelo para operar eficazmente en aplicaciones médicas, donde cada dato puede tener un significado crítico. Este enfoque entrelaza la necesidad de precisión diagnóstica con la integridad metodológica, reforzando la aplicabilidad clínica de nuestros hallazgos y subrayando la importancia de la selección cuidadosa de métodos de partición en estudios de salud.

Capítulo 6

Análisis

En esta sección, se describen de manera detallada los requisitos funcionales y no funcionales identificados para el proyecto de análisis de señales EEG mediante técnicas de ML. Estos requisitos se han definido para garantizar que el sistema cumpla con los objetivos propuestos y asegure una alta calidad en su implementación y funcionamiento.

6.1. Requisitos funcionales

Los requisitos funcionales describen las funciones y tareas que se llevaron a cabo para cumplir con los objetivos del proyecto. A continuación se enumeran los requisitos funcionales definidos para este proyecto:

1. **RF1: Adquisición de datos EEG.** Se adquirieron señales EEG de 30 sujetos, 15 hombres y 15 mujeres, mientras realizaban cuatro actividades distintas. Se aseguró que el número de muestras fuese el mismo para cada sujeto y actividad.
2. **RF2: Preprocesamiento de datos.** Se normalizaron los datos EEG adquiridos para garantizar la homogeneidad y comparabilidad entre las muestras. Además, se implementaron técnicas de filtrado para eliminar ruido y artefactos de las señales EEG.
3. **RF3: Segmentación de señales.** Las señales EEG se segmentaron en ventanas de tiempo fijas, cada una considerada una muestra independiente para el análisis posterior.
4. **RF4: Extracción de características.** Se extrajeron características relevantes de las señales EEG preprocesadas, incluyendo medidas en el dominio del tiempo, frecuencia y tiempo-frecuencia.
5. **RF5: Construcción de modelos de ML.** Se construyeron distintos modelos utilizando las características extraídas, tales como SVM, KNN, árboles de decisión, algoritmos de ensamble, etc...

6. **RF6: Evaluación de modelos.** Se proporcionaron herramientas para evaluar el rendimiento de los modelos utilizando diversas métricas, incluyendo *accuracy*, sensibilidad y especificidad.

7. **RF7: Ranking y comparación.** Se generaron rankings de los modelos en base a su rendimiento, utilizando múltiples métricas para determinar cuál es el mejor bajo distintos criterios de evaluación.

6.2. Requisitos No Funcionales

Los requisitos no funcionales describen atributos de calidad del sistema y aspectos no relacionados directamente con las funcionalidades específicas, pero que son cruciales para el éxito del proyecto. A continuación se enumeran los requisitos no funcionales definidos para este proyecto:

1. **RNF1: Seguridad.** Se garantizó la confidencialidad y protección de los datos personales de los sujetos del estudio, cumpliendo con las normativas y regulaciones éticas relacionadas con la investigación biomédica y la protección de datos.

2. **RNF2: Rendimiento.** Se procesaron grandes volúmenes de datos EEG de manera eficiente, completando las operaciones de preprocesamiento, extracción de características y entrenamiento de modelos de manera modularizada. Esto permitió optimizar tiempos no repitiendo operaciones demandantes.

3. **RNF3: Usabilidad.** Los resultados del análisis se organizaron en múltiples carpetas con comparativas y otros datos relevantes, permitiendo a los investigadores acceder y visualizar la información de manera clara. Esto facilita que cada uno pueda realizar su propio análisis sin necesidad de conocimientos avanzados en programación.

Capítulo 7

Diseño e Implementación

Este capítulo detalla exhaustivamente el diseño metodológico y las fases de implementación del experimento realizado para el análisis de EEG recogidos mediante la diadema Muse. Como comentabamos en capítulos anteriores el experimento ha sido realizado a 15 hombres y 15 mujeres con 4 pruebas distintas de 3 minutos y una repetición de las mismas.

7.1. Diseño

7.1.1. Procedimiento de experimentación

Primeramente, recordamos la tabla donde se recopila la información de todos los estudios analizados previamente que podemos encontrar en la tabla 3.1.

El diseño experimental se adhiere a las prácticas establecidas en estudios anteriores y está centrado en asegurar la obtención de datos fiables y consistentes. Los parámetros, tales como el número de participantes, los modelos seleccionados y los tiempos de captura de la señal EEG, han sido escogidos en base a la media o mediana de estos valores encontrados en investigaciones previas. Este enfoque garantiza una comparación adecuada con la literatura existente y fortalece la fiabilidad de los resultados obtenidos. A continuación, se describen detalladamente los pasos del procedimiento:

1. **Ambiente controlado:** Todos los experimentos se llevarán a cabo en una habitación tranquila, configurada para mantener condiciones constantes y minimizar variables externas que puedan afectar los resultados.

2. **Consentimiento informado:** Los participantes serán informados sobre el propósito y la naturaleza del estudio. Se les pedirá que firmen un formulario de consentimiento acordando la recopilación y uso de sus datos para fines experimentales.

3. **Protección de datos personales:** Para gestionar y clasificar adecuadamente los datos recogidos, se ha implementado un sistema de nomencla-

tura específico para el almacenamiento de los archivos. Cada archivo se denomina siguiendo un formato estructurado, por ejemplo, '1_1_m1.csv'. El primer número en el nombre del archivo identifica al sujeto, asignando valores del 1 al 30 para preservar el anonimato de los participantes. El segundo número indica la repetición de la prueba, que puede ser 1 o 2. La letra subsequente señala la actividad específica realizada durante la sesión de EEG: 'c' para meditación, 'e' para pruebas matemáticas, 'm' para escuchar música, y 'l' para lectura. Finalmente, el último número refleja el minuto de la prueba, que varía del 1 al 3. Esta metodología de denominación facilita la organización, el acceso y la confidencialidad de los datos de los sujetos involucrados.

4. **Diversidad de participantes:** Se seleccionarán 30 individuos, incluyendo ambos sexos, entre las edades de 18 a 25 años. En concreto serán 15 hombres y 15 mujeres.

5. **Protocolo de pruebas:** Siguiendo los lineamientos de investigaciones previas, se establecen cuatro condiciones experimentales que se realizarán en estado sentado, cada una durante 3 minutos:

   - Estar relajado con los ojos cerrados. (Meditación).
   - Resolver problemas matemáticos.
   - Escuchar música.
   - Leer.

   La actividad de resolución de problemas matemáticos se alinea con investigaciones anteriores sobre la concentración, brindando datos valiosos sobre el estado cognitivo activo en situaciones que demandan atención y enfoque mental. De las otras 3 pruebas ya hemos mencionado estudios previos.

Figura 7.1: Pruebas del experimento

En concreto para la prueba de meditación se ha pedido al sujeto que se relaje y cierre los ojos. Además para ayudar al sujeto a entrar en este estado se han elegido dos micromeditaciones guiadas de youtube que se han dejado como ruido de fondo. Para la prueba matemática se ha preparado un documento en el que en la primera hoja aparecen sumas y restas aumentando en dificultad, en la segunda multiplicaciones y divisiones y en la última un sudoku de dificultad fácil. A los sujetos se les ha informado que pueden saltarse las operaciones que no sean capaces de realizar y además se les ha proporcionado asistencia antes de realizar el experimento explicando cualquier duda que tuvieran sobre como se resuelven estos problemas. En cada una de estas páginas el sujeto ha estado 1 minuto. Por si acaso alguno de ellos terminase antes del tiempo establecido se han preparado problemas auxiliares que no han sido necesarios en ninguna de las 30 pruebas. Para la prueba de música se le pide al sujeto que haga

una escucha activa de las canciones, permitiendole mover brazos y piernas, (mientras que sigan sentados), y que no realicen movimientos bruscos de cabeza para no perder la señal del dispositivo. Por último en la prueba de lectura se le han proporcionado a cada paciente tres textos. El primero contiene poemas, el segundo un capítulo de obra de teatro y el tercero un extracto de un texto narrativo. Se le ha pedido a cada sujeto que lea para él mismo los textos y en caso de terminarlos antes de que pase un minuto que vuelva a comenzar a leer el texto en el que se encuentre. Al igual que en la prueba matemática el investigador le indica al sujeto cuando puede proceder a avanzar a la siguiente subprueba.

6. **Aumento de datos:** Con el fin de conseguir un dataset más robusto se repetirá el experimento 2 veces a cada participante.

7. **Subpruebas:** Para conseguir un dataset que pueda ser utilizado en otro tipo de estudios las pruebas se dividirán en subpruebas de un minuto. Con esto, en otra investigación, se podría aislar la prueba musical e intentar distinguir entre géneros, lo mismo se podría intentar para la lectura o las matemáticas.

8. **Pausas:** Al principio, se tomará el tiempo que sea necesario para explicar debidamente el experimento y colocar correctamente la diadema MUSE. Se otorgarán 5-10 segundos de margen entre cada subprueba dentro de cada pruba. Asimismo, se incluirá una pausa de 15-30 segundos entre las pruebas. Por último, en la repetición del experimento se tomará un descanso de 1 minuto.

9. **Preguntas de control:** Aunque no se usen en el anáilisis que realizaremos posteriormente a cada sujeto se le iban haciendo preguntas de control a medida que profundizaba el experimento. Estas eran de caracter informativo y se centraban mayoritariamente en el grado de satisfacción alcanzado por el sujeto en cada subprueba o por ejemplo si había conseguido llegar a un estado de relajación.

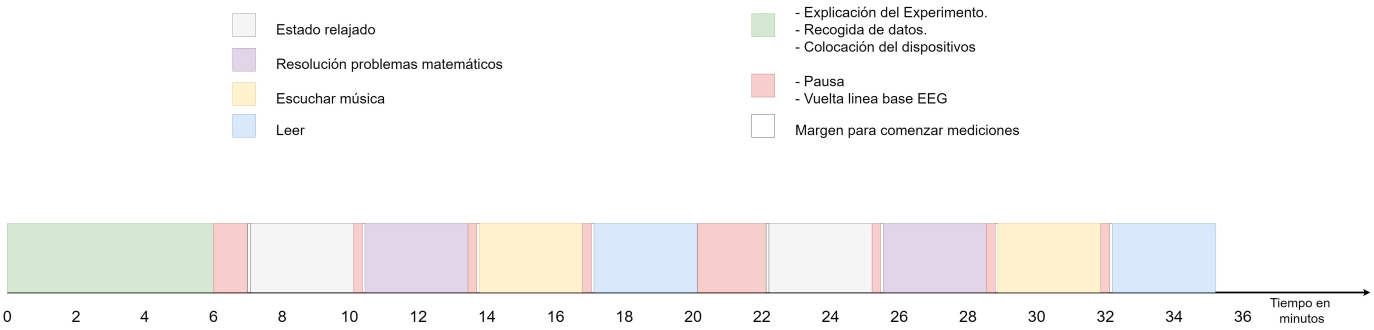


Figura 7.2: Proceso de experimentación con tiempo estimado

10. **Diversidad de la prueba:** En las repeticiones del experimento a los participantes se les proporcionará un nuevo texto, así como una nueva canción y problemas matemáticos a resolver. Preferiblemente se usarán géneros distintos tanto en texto como música a los ya experimentados por el paciente en la prueba anterior.

11. **Verificación del equipo:** Entre cada repetición de la prueba se incorporará un breve intervalo. Durante este tiempo, se verificará que el dispositivo de medición permanezca correctamente colocado y no cause molestias al participante.

Este protocolo está diseñado para respetar los derechos y la comodidad de los participantes al tiempo que se asegura la recopilación de datos de alta calidad para el análisis posterior. Si desea más información al respecto y observar los textos, preguntas de control o audios usados en el experimento puede encontrarlos en *Github*.

Cuadro 7.1: Especificaciones del experimento

Modelos	**Características extraídas**	**Participantes**	**Repeticiones**	**Aparato EEG**	**Tiempo**	**Reposo**
▪ *KNN* ▪ *SVM* ▪ *XGBoost* ▪ *Random Forest* ▪ *Stacking Classifier* ▪ *Voting Classifier* ▪ *Hist Gradient Boosting* ▪ *CatBoost* ▪ *Extra Trees* ▪ *AdaBoost* ▪ *MLP* ▪ *Gaussian Naives Bayes* ▪ *TabNet*	▪ Entropía basada en características ▪ Potencia máxima de cada banda de frecuencia ▪ Energía de cada banda de frecuencia ▪ Valor pico a pico de las bandas de frecuencia ▪ Valor medio de las bandas de frecuencia en el dominio temporal ▪ Ratio de energía de banda alfa a banda beta ▪ Entropía de muestra ▪ Entropía diferencial	30	2	MUSE	3 min	15 s / 1 min entre repeticiones

7.1.2. Diseño de casos de uso

En esta subsección se describen algunos de los casos de uso más relevantes del proyecto . Aunque no hay usuarios como tal, se considera al sistema como el actor principal.

1. **CU1: Extracción de datos EEG**
   - **Actor:** Sistema
   - **Descripción:** El sistema debe adquirir señales EEG de 30 sujetos, 15 hombres y 15 mujeres, mientras realizan cuatro actividades distintas.
   - **Precondiciones:** Disponibilidad del equipo de EEG y los sujetos para el estudio.
   - **Flujo principal:**
     a) El sistema ajusta el dispositivo y parámetros necesarios para realizar la extracción de señales.
     b) Se registran las señales EEG mientras los sujetos realizan las actividades.
     c) Se almacenan los datos adquiridos para su posterior análisis.
   - **Postcondiciones:** Se obtienen datos EEG de los 30 sujetos durante las cuatro actividades.
2. **CU2: Preprocesamiento de la señal**
   - **Actor:** Sistema
   - **Descripción:** El sistema debe preprocesar y normalizar los datos EEG adquiridos para asegurar la homogeneidad y comparabilidad entre las muestras.
   - **Precondiciones:** Los datos EEG en bruto están disponibles.
   - **Flujo principal:**
     a) El sistema aplica técnicas de filtrado para eliminar ruido y artefactos.
     b) El sistema normaliza los datos.
     c) El sistema segmenta las señales en ventanas de tiempo fijas.
   - **Postcondiciones:** Se obtienen datos EEG preprocesados y normalizados, listos para el análisis y la extracción de características.
3. **CU3: Extracción de siete características principales**
   - **Actor:** Sistema
   - **Descripción:** El sistema extrae 7 características relevantes de las señales EEG preprocesadas para cada banda y electrodo.
   - **Precondiciones:** Los datos EEG preprocesados por ventana temporal están disponibles.

- **Flujo principal:**
  a) El sistema hace un estudio sobre la literatura existente para determinar las características a extraer.
  b) El sistema extrae siete características de cada banda de frecuencia en cada uno de los electrodos del dispositivo.
- **Postcondiciones:** Se obtienen 7 características para cada una de las bandas de frecuencia en cada uno de los electrodos.

4. **CU4: Búsqueda de los mejores hiperparámetros**
   - **Actor:** Sistema
   - **Descripción:** El sistema debe buscar y optimizar los mejores hiperparámetros para los modelos utilizados en la clasificación de las señales EEG.
   - **Precondiciones:** Las características por ventana temporal de los datos EEG están disponibles.
   - **Flujo principal:**
     a) El sistema selecciona un modelo de ML.
     b) El sistema define un espacio de búsqueda para los hiperparámetros.
     c) El sistema aplica la técnica de optimización de *grid search*.
     d) El sistema evalúa el rendimiento de cada combinación de hiperparámetros utilizando validación cruzada.
     e) El sistema selecciona la mejor combinación de hiperparámetros basada en las métricas de rendimiento.
   - **Postcondiciones:** Se obtienen los mejores hiperparámetros para cada modelo.

5. **CU5: Evaluación comparativa**
   - **Actor:** Sistema
   - **Descripción:** El sistema debe evaluar y comparar el rendimiento de diferentes modelos y ventanas utilizando diversas métricas.
   - **Precondiciones:** Los modelos están entrenados y sus predicciones han sido almacenadas.
   - **Flujo principal:**
     a) El sistema recopila las métricas de rendimiento (*accuracy*, *balanced accuracy*, sensibilidad, especificidad) para cada modelo.
     b) El sistema organiza y presenta las métricas en tablas comparativas.
     c) El sistema genera tablas y visualizaciones para facilitar la comparación entre modelos.

- **Postcondiciones:** Se obtienen comparativas detalladas del rendimiento de los modelos, permitiendo identificar el modelo más efectivo.

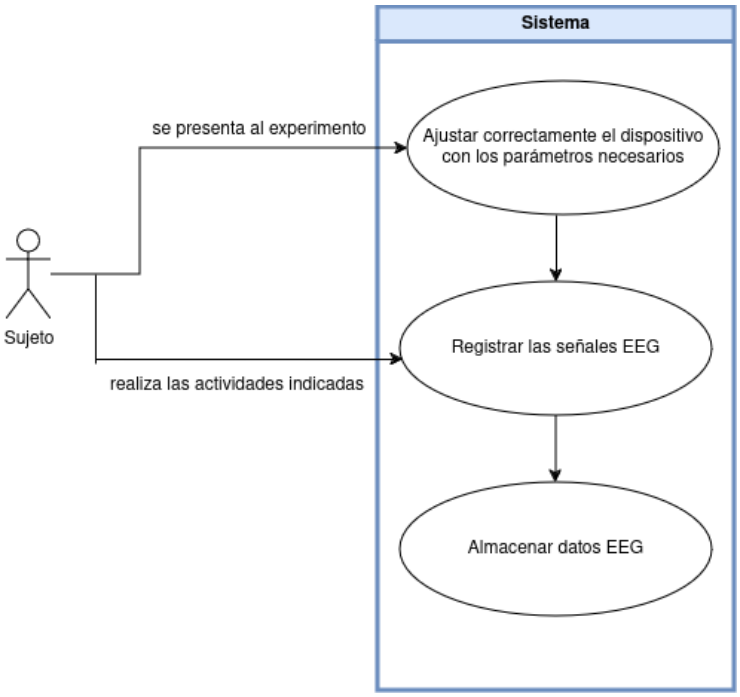


Figura 7.3: Ejemplo de CU1 en formato visual

7.2. Fases en la implementación

Para llevar a cabo este gran proyecto se llevaron a cabo distintas etapas. En esta sección hablaremos de la relativa a la implementación del código y al tratamiento de los datos, así como la obtención de resultados.

7.2.1. Preprocesamiento de los datos

Los datos recogidos mediante la diadema Muse de cada sujeto incluyen 24 archivos correspondientes a sus diferentes pruebas con las subpruebas correspondientes. Para clasificar correctamente estos datos se eligió guardar los archivos con el identificador comentado en 7.1.1 . Cada archivo contiene registros de al menos 1 minuto de duración extendiéndose por más de 3 minutos en total para cada experimento. El preprocesamiento de los datos se realizó siguiendo estos pasos:

1. Eliminación de filas con artefactos identificados por la diadema, tales como parpadeos.
2. Eliminación de columnas irrelevantes para el estudio como podría ser las marcas de tiempo.
3. Descarte de columnas de datos crudos (RAW) en favor de mediciones procesadas y correlacionadas.
4. Sustitución de valores nulos o infinitos detectados por la diadema por un valor estándar de 0.0, que corresponde a 50dB y es la frecuencia base de grabación.
5. Eliminación de filas cuyos valores son todos nulos, asegurando que cada archivo retenga un mínimo de 15360 filas, correspondiente a los datos recogidos durante 60 segundos a una tasa de 256 mediciones por segundo.
6. Los archivos con más de 15360 filas se truncaron para conservar únicamente las primeras 15360 filas.

Con este preprocesamiento hemos limpiado datos irrelevantes para nuestro estudio y nos quedamos con datos numéricos de igual formato para todos los archivos del conjunto.

7.3. Normalización y extracción de características

Una vez preprocesados los datos, se procedió a su normalización usando los valores máximos y mínimos observados en cada columna a través de todos los sujetos. Posteriormente, se extrajeron características según los métodos discutidos en el capítulo del estado del arte, resultando en un total de 140 características distintas.

- Se empleó el método `SelectKBest` con `f_classif` de la librería *Scikit-learn* para seleccionar las características más relevantes.
- Se exploraron diversas ventanas temporales de extracción (60, 30, 20 y 15 segundos) para optimizar la captura de información relevante y reducir el ruido introducido por fallos esporádicos de los sensores del dispositivo.

A lo largo de este proyecto el tiempo ha sido un factor determinante. La cantidad de datos abordada, aunque no suficiente para hacer un estudio de una muestra significativa de la población, es muy extensa para el procesamiento de un ordenador. Este hecho hizo que se priorizara la modularidad en el código y la ejecución por fases. En este sentido, para reducir tiempos se crearon archivos para almacenar las características extraídas por ventana de tiempo que luego se pasarán posteriormente a los modelos en vez de extraer las características

para cada entrenamiento. También quiero resaltar que con el preprocesamiento anterior sumado a la normalización y extracción de características, el conjunto de datos que obtenemos está completamente balanceado, es decir, cada una de las clases a modelar contiene las mismas instancias. Esto significa que ninguna clase está subrepresentada, por tanto, los modelos de ML no tenderán a priorizar alguna de las clases por encima del resto, ya que si lo hicieran, bajarían en rendimiento en las demas, traduciéndose en un peor rendimiento general del modelo.

7.3.1. Entrenamiento y Validación de Modelos

Para el entrenamiento y la validación de los modelos de ML se adoptó la técnica de Validación Cruzada LOO comentada previamente. Esta elección viene dada por la poca cantidad de datos, así conseguimos maximizar el conjunto de entrenamiento. Cada modelo se entrenó con datos de 29 sujetos y se validó con el sujeto restante. Esta estrategia asegura que el modelo pueda generalizar bien sobre datos no vistos previamente.

- Se configuraron los modelos básicos usando una búsqueda exhaustiva de hiperparámetros, con *Grid Search*, sobre las características seleccionadas.
- Una vez realizada se procedió a realizar su mejor ejecución para cada conjunto de características guardando tanto el tiempo como la matriz de confusión resultante de aplicar el modelo a cada sujeto.
- Se procedió a construir modelos basados en las predicciones de nuestros modelos básicos. Sobre estas se crea otro modelo para intentar mejorar las precisiones. Debido a su largo tiempo de cómputo se probaron sobre cierto número de características seleccionadas.

Con esta metodología usando *grid Search* no solamente encontramos los mejores hiperparámetros para cada uno de los modelos, sino que también obtenemos el número de características seleccionadas que más le beneficia. Además, al hacer esto para distintas ventanas temporales podemos, a su vez, obtener cual de las ventanas es la que nos da más información a la hora de clasificar correctamente las actividades. Cuanto menor sea la ventana antes podremos saber que está haciendo una persona en caso de leer las ondas en tiempo real.

7.3.2. Obtención de resultados

Para evaluar el desempeño de los modelos entrenados, se realizó un análisis comprensivo utilizando diversas métricas. Cada métrica proporciona información específica sobre la capacidad del modelo para clasificar correctamente las actividades. A continuación, se detalla cada métrica utilizada, su fórmula de cálculo y la información que aporta:

- ***Accuracy***: El *accuracy* es una medida global del rendimiento del modelo, calculada como la proporción de predicciones correctas sobre el total de predicciones. Matemáticamente, se expresa como:

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

  donde TP es el número de verdaderos positivos, TN es el número de verdaderos negativos, FP es el número de falsos positivos y FN es el número de falsos negativos. Un alto *accuracy* indica que el modelo tiene un buen rendimiento general.

- **Sensibilidad, *sensitivity* o *recall***: La sensibilidad es la capacidad del modelo para identificar correctamente los verdaderos positivos. Se calcula como:

$$\text{Sensibilidad} = \frac{TP}{TP + FN}$$

  Esta métrica es crucial cuando es importante minimizar los falsos negativos.

- **Especificidad, *specificity***: La especificidad mide la capacidad del modelo para identificar correctamente los verdaderos negativos. Se calcula como:

$$\text{Especificidad} = \frac{TN}{TN + FP}$$

  Es útil en contextos donde es importante minimizar los falsos positivos.

- ***Balanced accuracy***: En este estudio nos referiremos a la misma como sensibilidad media, esta métrica es útil cuando hay un desbalance en las clases del conjunto de datos. Es el promedio de la sensibilidad de cada clase, y se calcula como:

$$Balanced\ accuracy = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i}$$

  donde N es el número de clases. Esta métrica proporciona una evaluación más equitativa del rendimiento del modelo en todas las clases.

Para obtener estos resultados, se utilizó la matriz de confusión de cada modelo. La matriz de confusión se representa como una tabla $N \times N$, donde N es el número de clases, y cada entrada M_{ij} indica el número de instancias de la clase i que fueron clasificadas como clase j.

El análisis se complementa con un informe detallado que incluye las matrices de confusión normalizadas, permitiendo visualizar la proporción de aciertos y errores por clase. Este enfoque asegura que los resultados sean precisos y representativos, facilitando una comprensión profunda del rendimiento del modelo en la tarea de clasificación de actividades a partir de señales EEG.

Además, se evalúa el tiempo de ejecución de cada modelo, proporcionando una medida del rendimiento computacional. Este tiempo se incluye en el análisis para balancear la precisión del modelo con su eficiencia operativa, lo que es esencial para aplicaciones en tiempo real.

Finalmente, se seleccionan los mejores modelos según cada métrica clave: *accuracy*, sensibilidad y especifidad. Esta selección permite identificar no solo el modelo más preciso, sino también el que ofrece un mejor equilibrio entre las distintas clases y el que mantiene un rendimiento consistente en situaciones de desbalance de clases.

7.4. Pruebas y optimización de modelos

Durante el desarrollo del proyecto, se realizaron diversas pruebas y ajustes de los modelos de ML para optimizar su rendimiento, asegurando así la calidad del software propuesto, enlace al github. Esta sección describe las pruebas realizadas sin entrar en detalle, incluído el proceso de búsqueda de hiperparámetros y los ajustes específicos para cada modelo.

7.4.1. *Grid search*: selección de hiperparámetros y características

Para optimizar los modelos, se utilizó la técnica de *Grid Search*, una metodología que permite explorar sistemáticamente una gama de valores para los hiperparámetros de los modelos. El objetivo del *Grid Search* es identificar la combinación de hiperparámetros que proporciona el mejor rendimiento en términos de una métrica específica, como *accuracy* o *balanced accuracy*.

El proceso de *grid search* se realiza de la siguiente manera:

1. Se define un rango de valores posibles para cada hiperparámetro que se desea optimizar. Por ejemplo, para un modelo de clasificación, los hiperparámetros pueden incluir la tasa de aprendizaje, el número de árboles o la profundidad de los árboles, entre otros.

2. Se crea una cuadrícula de todas las combinaciones posibles de estos valores.

3. Se entrena y valida el modelo utilizando cada combinación de hiperparámetros. Para cada combinación, se evalúa el rendimiento del modelo utilizando una técnica de validación cruzada, en este caso se optó por 5-fold-cross validation para que fuese más eficiente.

4. Se selecciona la combinación de hiperparámetros que proporciona el mejor rendimiento en la métrica definida.

Como ejemplo, para el modelo XGBoost, se definieron varios hiperparámetros clave a optimizar:

- Número de árboles (*n_estimators*)

- Tasa de aprendizaje (*learning_rate*)
- Profundidad máxima de los árboles (*max_depth*)

El *grid search* evaluó todas las combinaciones posibles dentro de los rangos definidos para estos hiperparámetros. Al finalizar el proceso, se identificó la combinación óptima que ofrecía el mejor rendimiento en la clasificación de las señales EEG.

7.4.2. Meta-modelos

Para mejorar aún más el rendimiento de los modelos, se implementaron técnicas de ensamblaje mediante el uso de meta-modelos. Los meta-modelos combinan las predicciones de múltiples modelos básicos para crear un modelo más robusto y preciso. En este proyecto, se utilizaron dos técnicas principales de ensamblaje: Stacking y Voting.

Stacking

El Stacking combina varios modelos de diferentes tipos y utiliza un meta-modelo para aprender cómo combinar mejor sus predicciones. En este caso, se utilizaron los mejores modelos base obtenidos a partir del Grid Search y se emplearon como entradas para el meta-modelo. El proceso se realizó en dos niveles:

1. **Nivel 1, (modelos base)**: Se entrenaron varios modelos base, como *Random Forest*, *XGBoost*, *CatBoost*, entre otros, utilizando los hiperparámetros optimizados obtenidos en la fase previa.
2. **Nivel 2, (meta-modelo)**: Se utilizó un *Random Forest* como meta-modelo, que toma como entrada las predicciones de los modelos base y aprende a combinarlas para hacer la predicción final.

Voting

El *Voting* combina las predicciones de varios modelos mediante una votación mayoritaria, (*hard voting*), o ponderada ,(*soft voting*). En este proyecto, se utilizó el *hard voting*, donde cada modelo base realiza una predicción, y la clase que recibe la mayoría de los votos se selecciona como la predicción final.

Los clasificadores base utilizados se crearon utilizando los hiperparámetros optimizados para cada conjunto de datos. Para evaluar estos meta-modelos, se probó con distintos números de características seleccionadas, abarcando todo el espectro de los mejores modelos base.

El siguiente es un esquema general del proceso:

1. **Creación de clasificadores base**: Los clasificadores base se crean utilizando los mejores hiperparámetros obtenidos del *Grid Search*.

2. **Evaluación de meta-modelos**: Los meta-modelos se evalúan utilizando la técnica de validación cruzada LOO. Se prueban diferentes números de características seleccionadas para encontrar la combinación óptima que maximice el rendimiento del modelo para cada una de las ventanas temporales.

Este enfoque con meta-modelos, permitió combinar las fortalezas de diferentes modelos base, resultando en algunos casos en un rendimiento mejorado y más robusto en la clasificación de actividades a partir de señales EEG. Aunque como veremos en los resultados, no se consiguió mejorar en gran medida los resultados anteriores.

Capítulo 8

Resultados

En este capítulo se comentará acerca de los resultados obtenidos en distintas pruebas realizadas. En concreto, hablaremos del problema principal de diferenciar entre las 4 actividades y a raíz de los resultados obtenidos, nos centraremos en alguno de los problemas de clasificación binaria que podemos abordar.

8.1. Clasificación multiclase

El principal objetivo del proyecto fue desarrollar un modelo capaz de detectar y clasificar cuatro actividades distintas, (meditación, prueba matemática, música y lectura), con alta precisión y balance en las métricas de rendimiento. Para lograr esto, se evaluaron múltiples modelos de clasificación en distintas ventanas temporales, con diferente número k de características seleccionadas, todo esto utilizando tres métricas principales: *accuracy*, sensibilidad y especifidad. A continuación, se presentan los resultados obtenidos y su evaluación.

8.1.1. Mejores modelos

Para tener una visión general del rendimiento de los modelos, hemos creado un ranking basado en los resultados medios en las métricas que ha obtenido cada modelo en sus distintas ejecuciones.

Modelos	Accuracy	Sensibilidad	Specificidad
VotingClassifier	**0.5603**	**0.5603**	**0.8534**
Random Forest	0.5591	0.5591	0.8530
StackingClassifier	0.5572	0.5572	0.8524
CatBoost	0.5566	0.5566	0.8522
XGBoost	0.5517	0.5517	0.8506
HistGradientBoosting	0.5433	0.5433	0.8478
AdaBoost	0.5363	0.5363	0.8454
ExtraTrees	0.5258	0.5258	0.8419
SVM	0.4795	0.4795	0.8265
TabNet	0.3742	0.3742	0.7914
MLP	0.3455	0.3455	0.7818
KNN	0.3394	0.3394	0.7798
GaussianNB	0.3198	0.3198	0.7733

Cuadro 8.1: Valor medio de cada métrica para cada modelo

Como observamos en las tablas anteriores, son 4 los algoritmos que despuntan por encima del resto. Esto lo vemos en el prommedio de las puntuaciones de los ranking, con los resultados:

1. ***VotingClassifier***
2. ***Random Forest***
3. ***StackingClassifier***
4. ***CatBoost***

Hemos encontrado que *Voting Classifier* es el algoritmo que se comporta mejor para las 3 métricas estudiadas en las distintas ventanas temporales. Sin embargo, si tenemos en cuenta el factor tiempo, *voting* es la segunda peor técnica, con ordenes de magnitud por encima de cualquiera del resto de modelos estudiados, el único peor en cuanto a tiempo sería *stacking*. Por tanto, si lo que priorizamos es un estudio eficiente con resultados muy parecidos a los obtenidos por el modelo anterior, la mejor opción sería utilizar *Random Forest*. *Catboost* también es interesante ya que en el ranking general obtiene una puntuación muy similar, prácticamente idéntica, a la de *Random Forest*, con tiempos un poco inferiores.

8.1.2. Mejor ventana

Este experimento, como comentábamos anteriormente, se ha realizado con un número de ventanas variable. En concreto, tenemos que la ventana 1 son 60s; ventana 2, 30s; ventana 3, 20s; y 4, 15s. Estos son los intervalos escogidos de donde se han extraído las características con las que hemos creado los modelos. Así que ahora, analizaremos cual es la ventana temporal en la que los 4 mejores

algoritmos clasifican mejor. No tenemos en cuenta el resto de algoritmos, debido a que algunos, dependiendo de la ventana, obtienen resultados bastante malos que condicionarían los resultados. Para hacer esto calculamos el *accuracy*, la sensibilidad y especifidad promedio de dichos modelos.

Ventanas	ACC	Sensibilidad	Especificidad
4	**0.5608**	**0.5608**	**0.8536**
1	0.5593	0.5593	0.8531
3	0.5565	0.5565	0.8522
2	0.5556	0.5556	0.8519

Cuadro 8.2: Ranking de las distintas ventanas con la media de las distintas métricas

Como hemos podido observar para la ventana en la que se aprecian mejores resultados medios es la 4. Esto es bastante beneficioso porque a la hora de construir un programa en tiempo real, el cual nos indique la actividad que está realizando un usuario, cuanto menos segundos deba de procesar la señal EEG, antes podrá el programa adivinar la actividad que realiza el usuario.

8.1.3. Mejores resultados

Después de haber dado datos sobre los mejores modelos y ventanas temporales pasamos a presentar las ejecuciones de los modelos que mejores resultados han proporcionado.

Modelos con mejor *accuracy* y sensibilidad promedio En este caso hubo 2 modelos que mostraron los mejores resultados en estas métricas para 2 ventanas disintas. En concreto, uno de ellos fue el que presentó la **mejor especifidad promedio**:

- ***StackingClassifier* (Ventana: 1.0, K: 50.0)**:
  - *Accuracy*: **0.569444**
  - Sensibilidad promedio: **0.5694**
  - Especifidad promedio: 0.8565
  - Tiempo(s): 2030.3996, 33 min y 50.4s
- ***Random Forest* (Ventana: 2.0, K: 39.0)**:
  - *Accuracy*: **0.569444**
  - Sensibilidad promedio: **0.5694**
  - Especifidad promedio: **0.8565**
  - Tiempo(s): 60.9076, 1 min y 0.9s

Pasamos a estudiar ahora sus matríces de confusión normalizadas y extraemos información de cada clase en particular extrayendo datos como sensibilidad, especifidad. Por un lado tenemos:

***StackingClassifier* (Ventana: 1.0, K: 50.0)**:

	Música	Leer	Meditar	Matemáticas
Música	**0.561**	0.161	0.083	0.194
Leer	0.133	**0.578**	0.156	0.133
Meditar	0.022	0.133	**0.733**	0.111
Matemáticas	0.239	0.211	0.144	**0.406**

Cuadro 8.3: Matriz de confusión normalizada del *StackingClassifier*

	Música	**Leer**	**Meditar**	**Matemáticas**	**Promedio**
Sensibilidad	0.5611	0.5778	0.7333	0.4056	0.5694
Especificidad	0.8685	0.8315	0.8722	0.8537	0.8565

Cuadro 8.4: Matriz que muestra las distintas métricas asociadas a cada clase.

Por otro lado tenemos al otro modelo, que además de conseguir el mejor *accuracy* de entre todos los modelos, también presenta la mejor *especifidad media*:

***Random Forest* (Ventana: 2.0, K: 39.0)**:

	Música	Leer	Meditar	Matemáticas
Música	**0.581**	0.156	0.061	0.203
Leer	0.206	**0.506**	0.133	0.156
Meditar	0.064	0.069	**0.744**	0.122
Matemáticas	0.278	0.164	0.111	**0.447**

Cuadro 8.5: Matriz de confusión normalizada del *Random Forest*

	Música	**Leer**	**Meditar**	**Matemáticas**	**Media**
Sensibilidad	0.5806	0.5056	0.7444	0.4472	0.5694
Especificidad	0.8176	0.8704	0.8981	0.8398	0.8565

Cuadro 8.6: Matriz que muestra las distintas métricas asociadas a cada clase.

Analizando las tablas de ambos modelos descubrimos diferencias en su desempeño, a pesar de alcanzar un *accuracy* idéntico del 56.94 %. El *StackingClassifier* presenta un buen equilibrio en la clasificación de las actividades música, leer y meditar, pero tiene un desempeño inferior en matemáticas, como presentamos en las sensibilidades de 56.11 %, 57.78 %, 73.33 % y 40.56 %, respectivamente. Además, pese a requerir un tiempo altamente superior, pierde en las métrica de

especifidad. Esto se traduce en que *Random Forest* consiga una mejora en la sensibilidad para todas las actividades en comparación con el *StackingClassifier*, especialmente en meditación y matemáticas.

Analizando las métricas de sensibilidad y especificidad para cada clase, se identifican puntos fuertes y áreas de mejora para ambos modelos. En música, ambos modelos tienen sensibilidades similares. Para leer, la sensibilidad es mejor en el *StackingClassifier*, pero el *Random Forest* ofrece una especificidad superior. En la meditación, el *Random Forest* supera al *StackingClassifier* en todas las métricas, indicando una mejor capacidad para distinguir esta actividad. Aunque ambos modelos tienen sensibilidades bajas en la prueba de matemáticas, el *Random Forest* muestra una sensibilidad más alta, sugiriendo una mejor clasificación general para esta actividad.

Ambos modelos presentan las mismas debilidades a la hora de clasificar las actividades, pueden reconocer actividades como la meditación fácilmente, pero fallan demasiadas veces en la prueba de matemáticas. Sin embargo, debido a la leve mejora de resultados general y disminución en ordenes de magnitud en el tiempo empleado, *Random Forest* se podría establecer como el mejor algoritmo de los dos en cuanto a alcanzar mejor *accuracy*. Otro factor que provoca que Random Forest sea más interesante es que las ventanas temporales que utiliza para clasificar son de 30s en comparación con los 60s de Stacking. Si se hiciera un análisis en tiempo real, el algoritmo más realista sería *Random Forest*, debido en gran medida a la cantidad de tiempo necesaria para clasificar una muestra, así como su desempeño general superior.

8.1.4. Optimización de los resultados

Tras haber descubierto el modelo más óptimo para nuestro problema, *Random Forest*. Se optó por hacer una búsqueda todavía más extensa y en mayor profundidad, afinando los hiperparámetros del modelo al máximo para cada una de las distintas ventanas temporales. Esto produjo un ligero incremento en los resultados como se observa a continuación.

Ventana	**K**	**ACC**	**Sensibilidad**	**Especifidad**
1	130	0.5806	0.5806	0.8602
2	**82**	**0.5840**	**0.5840**	**0.8613**
3	134	0.5741	0.5741	0.858
4	129	0.5684	0.5684	0.8561

Cuadro 8.7: Resultados por ventana para el problema multiclase con *Random Forest* optimizado

Mostramos más datos relacionados con las métricas de cada una de las clases para las ventanas estudiadas.

	Métrica	Música	Leer	Meditar	Matemáticas	Promedio
Ventana 1	**Sensibilidad**	0.6167	0.4833	0.7278	0.4944	0.5806
	Especificidad	0.8556	0.9024	0.8852	0.8130	0.8602
Ventana 2	**Sensibilidad**	0.6056	0.5250	0.7389	0.4667	0.5840
	Especificidad	0.8407	0.8713	0.8991	0.8343	0.8613
Ventana 3	**Sensibilidad**	0.6000	0.4926	0.7278	0.4759	0.5741
	Especificidad	0.8284	0.8741	0.8846	0.8451	0.8580
Ventana 4	**Sensibilidad**	0.5833	0.4847	0.7222	0.4833	0.5684
	Especificidad	0.8241	0.8806	0.8903	0.8296	0.8561

Cuadro 8.8: Resultados de las métricas de cada clase en las cuatro ventanas

Podemos ver la clasifiación de las distintas actividades concuerda con los hallazgos anteriores. Un detalle a resaltar es que los mejores resultados se obtienen en la ventana 2, donde menos características se seleccionan. Destacamos la mejora en *accuracy* en un 2.5 % . Esta mejora, aunque haya sido la que menos características ha usado de las 4 ventanas, ha venido acompañada de un incremento sustancial en el número de características seleccionado para realizar la clasificación en comparación con el estudio anterior.

8.2. Clasificación binaria

Después de ver cómo se comportan los algoritmos en una clasificación multiclase y descubrir cuál de los modelos es el más óptimo en nuestro conjunto de datos, nos hemos preguntado qué pasaría si entrenasemos dicho modelo para distinguir únicamente entre dos actividades, cómo se comportaría?

8.2.1. Meditación y prueba matemática

Primeramente, seleccionamos actividades muy opuestas a priori entre sí, matemáticas y meditación. En la clasificación anterior, la actividad de meditación presentaba los mejores resultados en cuanto a sensibilidad, mientras que la prueba matemática los peores. Mostramos los resultados de las mejores ejecuciones del algoritmo *Random Forest* para cada ventana temporal y k características seleccionadas.

Ventana	K	ACC	Sensibilidad	Especifidad
1	116	0.7889	0.7889	0.7889
2	138	0.8097	0.8097	0.8097
3	115	**0.8148**	**0.8148**	**0.8148**
4	111	0.7993	0.7993	0.7993

Cuadro 8.9: Resultados por ventana para el problema meditación-matemáticas

Es interesante observar que en los resultados se obtiene la misma sensibilidad y especifidad media. Vemos como cambian estas métricas en las distintas ventanas temporales para cada clase. Para ello definimos como clase positiva la prueba de meditación, esto hará que los resultados para la clase de la prueba matemática se obtengan al intercambiar los resultados en sensibilidad y especifidad para cada ventana.

	Ventana 1	**Ventana 2**	**Ventana 3**	**Ventana 4**
Sensibilidad	0.7722	0.7694	0.7852	0.7833
Especificidad	0.8056	0.8500	0.8444	0.8153

Cuadro 8.10: Sensibilidad y especificidad para la clase positiva (meditación) en diferentes ventanas

Mostramos la matriz de confusión normalizada de la ventana 3, donde se presentan los mejores resultados.

	Meditación	**Matemáticas**
Meditación	**0.785**	0.215
Matemáticason normalizada de la ventana 3, donde se	0.156	**0.844**

Cuadro 8.11: Matriz de confusión normalizada por filas

Paradójicamente, al contrario que en los resultados multiclase, la acción mejor reconocida al enfrentar estas dos clases es la prueba matemática. Es curioso que al intentar distinguir entre varias actividades, la prueba matemática tenga estos resultados bastante inferiores, mientras que si la aislamos con la prueba de meditación los resultados se disparan. Igualmente, aunque con una sensibilidad un poco inferior meditación sigue siendo bastante reconocible por el modelo, acercándose a un 80 % de sensibilidad. En este caso, la mejor ventana en las evaluaciones ha sido la de 20 segundos, donde se ha logrado un accuracy de 81.48 %, el cual es bastante superior a los resultados multiclase. En este problema binario, sí que nos aproximamos a un modelo que puede distinguir correctamente entre actividades.

8.2.2. Meditación y música

Después de haber comparado pruebas muy opuestas como las anteriores, se ha decidido estudiar cómo se comporta el modelo al clasificar las clases mejor representadas en el problema multiclase. Para ello, al igual que en el caso anterior, entrenamos el modelo *Random Forest* con distintos parámetros y presentamos los mejores resultados.

Ventana	K	ACC	Sensibilidad	Especifidad
1	123	0.8611	0.8611	0.8611
2	61	**0.8639**	**0.8639**	**0.8639**
3	140	0.8556	0.8556	0.8556
4	133	0.8583	0.8583	0.8583

Cuadro 8.12: Resultados por ventana para el problema meditación-música

	Ventana 1	Ventana 2	Ventana 3	Ventana 4
Sensibilidad	0.8667	0.8583	0.8519	0.8583
Especificidad	0.8556	0.8694	0.8593	0.8583

Cuadro 8.13: Sensibilidad y especificidad para la clase positiva (meditación) en diferentes ventanas

Con los resultados anteriores, podemos concluir, que en la ventana 2, se presentan los mejores resultados. Mostramos entonces su matriz de confusión normalizada.

	Música	Meditación
Música	**0.869**	0.131
Meditación	0.142	**0.858**

Cuadro 8.14: Matriz de confusión normalizada por filas

En contraste con el caso anterior, aquí se observa que ambas actividades se clasifican con una sensibilidad prácticamente igual, sin una diferencia apreciable. En el caso de la ventana 2, la música es ligeramente superior. Cabe resaltar que estos han sido los mejores resultados en términos de clasificación de todas las pruebas realizadas. Es importante mencionar que para ventanas de 30 segundos, el *accuracy* es de aproximadamente 86.4 %. Esto se traduce en que el modelo clasifica correctamente entre ambas actividades mejor que en el resto de casos estudiados.

Capítulo 9

Conclusiones

Desde el inicio de este proyecto, nos hemos enfocado en cumplir con todos los objetivos establecidos en el capítulo 1. Se ha desarrollado una sólida base teórica sobre los métodos de procesamiento y análisis de señales EEG, acompañada de un estudio exhaustivo del estado del arte. Además, se ha creado un dataset con datos recolectados de 30 sujetos, se ha implementado un código eficaz para el preprocesamiento de los datos, y se han seleccionado y evaluado modelos de ML adecuados mediante la metodología de validación *leave-one-out* por sujeto. Esta evaluación ha permitido obtener una estimación realista de la capacidad de los modelos para generalizar a nuevos individuos. Finalmente, se han extraído conclusiones relevantes sobre la viabilidad de distinguir entre diferentes tipos de actividad cerebral utilizando dispositivos EEG de bajo costo y técnicas de ML, cumpliendo así con los objetivos específicos del estudio.

A la vista de los resultados presentados, podemos afirmar que los modelos de machine learning han logrado clasificar con un 56.94 % de precisión el problema multiclase. Este porcentaje, superior al de una clasificación aleatoria, que sería del 25 %, demuestra que los modelos están aprendiendo patrones reales de las actividades. Además, dado que las clases están balanceadas, podemos descartar que los resultados sean aleatorios. Sin embargo, este nivel de precisión no es suficiente para aplicaciones prácticas que buscan distinguir entre actividades a partir de un EEG. Otro de los problemas reside en que las ventanas de tiempo necesarias para la extracción de características, actualmente, son considerablemente grandes. Esto implicaría un tiempo excesivo en la clasificación en tiempo real de la actividad que realiza un paciente.

En lo que respecta a los problemas de clasificación binaria, los resultados son más prometedores, con una clasificación bastante aceptable, de 81.5 % y 86.4 %. Destaca el caso de diferenciar entre música y meditación, logrando la mejor precisión de clasificación. En este escenario, sería factible desarrollar un programa en tiempo real capaz de distinguir cuál de las dos actividades realiza el sujeto con una precisión considerable.

De los resultados extraídos a lo largo de las distintas pruebas, se puede extrapolar que el dispositivo MUSE, recoge bastante bien las ondas EEG relacionadas

con la meditación. Este hecho proporciona herramientas a los modelos de ML para realizar una buena clasificación al intentar distinguir esta actividad del resto.

9.1. Trabajo Futuro

Para mejorar los resultados y avanzar en futuras iteraciones del proyecto, consideraremos varias propuestas. En todas ellas, destacamos la reutilización de los datos del experimento presentado, ya sea como método comparativo o como dataset sobre el que estudiar el problema con un enfoque distinto.

Todas estas propuestas, surgen de las limitaciones que ha tenido el experimento. Aunque los resutados binarios son esparanzadores no podemos decir lo mismo del multiclase. Además, el experimento en sí, tiene algunas limitaciones que se dan, en gran parte, debido a la muestra de la población tomada. Una a destacar podría ser el enfoque cuadriculado que se ha seguido en la realización de los experimentos, pudiendo ir alternando el orden de las pruebas en los sujetos o incluso cambiando los tiempos de descanso. Por estas razones se proponen las siguientes líneas:

Mejorar la Calidad del Dispositivo Una posible mejora sería el uso de dispositivos EEG con electrodos conectados directamente al cuero cabelludo, lo que ofrecería una mayor calidad en las ondas registradas. Esto permitiría estudiar ventanas de menor tiempo con menor ruido, mejorando la precisión y eficiencia del modelo. Otra ventaja consistiría en la posibillidad de utilizar ventanas de tiempo similares a las ya consideradas, donde se podría comparar el rendimiento del dispositivo de alta calidad con el económico, evaluando cuánto mejora la precisión de los mismos modelos de aprendizaje automático al intentar distinguir actividades.

Explorar nuevas características de la señal Es posible que existan otras características de la señal EEG que sean más relevantes para distinguir entre las distintas actividades. Investigar y probar diferentes métodos de extracción de características podría llevar a mejoras significativas en la clasificación. En concreto, descubir características que caracterizen a una de las actividades podría aumentar en gran medida el porcentaje de acierto.

Uso de redes neuronales En la actualidad, la inteligencia artificial está en auge, con nuevos modelos y mejoras diarias. Entonces, explorar la capacidad de las redes neuronales, especialmente las redes neuronales profundas y las redes recurrentes, para clasificar señales EEG podría traducirse en una captura de patrones más complejos y una mejora de la precisión de la clasificación. No obstante, al contrario que en los métodos de ML, no se estudiaría con la misma exactitud cuales son las características de que los modelos funcionen, así como las características relevantes de las señales.

Aplicación en tiempo real Desarrollar y evaluar un prototipo de sistema en tiempo real que utilice los mejores modelos identificados. Este sistema debería ser capaz de procesar las señales EEG, extraer y seleccionar características y proporcionar predicciones sobre la actividad del sujeto. El objetivo principal sería intentar reducir el tiempo que tarda el sistema en realizar una predicción manteniendo unos porcentajes de clasificación elevados.

Ampliación del Conjunto de Datos Realizar nuevos experimentos para recopilar un conjunto de datos más grande y diverso. Esto ayudaría a mejorar la generalización del modelo y a reducir el riesgo de sobreajuste. Se podrían realizar las pruebas ya practicadas a los 30 sujetos a un número mucho mayor que pueda aportar más información y relevancia al estudio.

En conclusión, aunque los resultados actuales muestran que es posible distinguir entre diferentes actividades utilizando señales EEG, hay un amplio margen de mejora. En este capítulo se ha intentado dar una visión global de cuales podrían ser los mejores caminos a seguir para profundizar en el tema, consiguiendo en un futuro poder distinguir entre múltiples actividades simplemente con una señal EEG.

Bibliografía

[1] C. D. Castaño Román, J. C. Cadavid Ruiz et al., «La transformación de las metodologías de desarrollo y la tendencia ágil,» 2018.

[2] C. Boaventura José, E. Peña Herrera, P. Verdecia Vicet e Y. Fustiel Alvarez, «Elección entre una metodología ágil y tradicional basado en técnicas de soft computing,» *Revista Cubana de Ciencias Informáticas*, vol. 10, págs. 145-158, 2016.

[3] R. G. Figueroa, C. J. Solís y A. A. Cabrera, «Metodologías tradicionales vs. metodologías ágiles,» *Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación*, vol. 9, n.º 1, págs. 1-10, 2008.

[4] A. N. Cadavid, J. D. F. Martínez y J. M. Vélez, «Revisión de metodologías ágiles para el desarrollo de software,» *Prospectiva*, vol. 11, n.º 2, págs. 30-39, 2013.

[5] M. C. Solís, «Una explicación de la programación extrema (XP),» *Willi. Net*, 2003.

[6] C. Skutt, «Use of EEG-Based Machine Learning to Predict Music-Related Brain Activity,» 2023.

[7] I. Daly, «Neural decoding of music from the EEG,» *Scientific Reports*, vol. 13, n.º 1, pág. 624, 2023.

[8] K. Tsekoura y A. Foka, «Classification of EEG signals produced by musical notes as stimuli,» *Expert Systems with Applications*, vol. 159, pág. 113 507, 2020.

[9] D. Devi, K. Kaveena, S. Keerthana y M. Rangeetha V, «Human Activity Recognition Using Machine Learning Technique,» en *2022 International Conference on Futuristic Technologies (INCOFT)*, 2022, págs. 1-5. DOI: 10.1109/INCOFT55651.2022.10094437.

[10] M. Alirezaei y S. H. Sardouie, «Detection of human attention using EEG signals,» en *2017 24th National and 2nd International Iranian Conference on biomedical engineering (ICBME)*, IEEE, 2017, págs. 1-5.

[11] K. Kunze, Y. Shiga, S. Ishimaru y K. Kise, «Reading Activity Recognition Using an Off-the-Shelf EEG – Detecting Reading Activities and Distinguishing Genres of Documents,» en *2013 12th International Conference on Document Analysis and Recognition*, 2013, págs. 96-100. DOI: 10.1109/ICDAR.2013.27.

[12] A. Salehzadeh, A. P. Calitz y J. Greyling, «Human activity recognition using deep electroencephalography learning,» *Biomedical Signal Processing and Control*, vol. 62, pág. 102 094, 2020, ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2020.102094. dirección: https://www.sciencedirect.com/science/article/pii/S1746809420302500.

[13] I. Hussain, R. Jany, R. Boyer et al., «An Explainable EEG-Based Human Activity Recognition Model Using Machine-Learning Approach and LIME,» *Sensors*, vol. 23, n.º 17, pág. 7452, 2023.

[14] G. Ayala y F. Montes, *Probabilidad Básica*. Valencia: Universidad de Valencia, 2024.

[15] N. Cristianini y J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[16] T. Kirschstein y R. Köhling, «What is the source of the EEG?» *Clinical EEG and neuroscience*, vol. 40, n.º 3, págs. 146-149, 2009.

[17] H. Berger, «Zwanzig Jahre EEG: Zum Andenken an den Entdecker des menschlichen Elektrencephalogramms,» *Archiv für Psychiatrie und Nervenkrankheiten*, vol. 183, n.º 1-2, págs. 1-1, 1949.

[18] C. S. Herrmann, D. Strüber, R. F. Helfrich y A. K. Engel, «EEG oscillations: From correlation to causality,» *International Journal of Psychophysiology*, vol. 103, págs. 12-21, 2016, Research on Brain Oscillations and Connectivity in A New Take-Off State, ISSN: 0167-8760. DOI: https://doi.org/10.1016/j.ijpsycho.2015.02.003. dirección: https://www.sciencedirect.com/science/article/pii/S0167876015000331.

[19] K. Blinowska y P. Durka, «Electroencephalography (eeg),» *Wiley encyclopedia of biomedical engineering*, 2006.

[20] P. Ahmadian, S. Cagnoni y L. Ascari, «How capable is non-invasive EEG data of predicting the next movement? A mini review,» *Frontiers in human neuroscience*, vol. 7, pág. 124, 2013.

[21] O. Azuara y Z. Gillette, «Using Machine Learning to Determine Optimal Sleeping Schedules of Individual College Students,» en oct. de 2022, págs. 13-25, ISBN: 978-3-031-17901-3. DOI: 10.1007/978-3-031-17902-0_2.

[22] C. M. Wilkinson, J. I. Burrell, J. W. Kuziek, S. Thirunavukkarasu, B. H. Buck y K. E. Mathewson, «Application of the Muse portable EEG system to aid in rapid diagnosis of stroke,» *medRxiv*, págs. 2020-06, 2020.

[23] O. E. Krigolson, M. R. Hammerstrom, W. Abimbola et al., «Using Muse: Rapid mobile assessment of brain performance,» *Frontiers in Neuroscience*, vol. 15, pág. 634 147, 2021.

[24] H. Habehh y S. Gohel, «Machine learning in healthcare,» *Current Genomics*, vol. 22, n.º 4, pág. 291, 2021.

[25] S. Marshall, «Cortical Activation during Mobility in an Indoor Real-World Environment: A Mobile EEG Study,» 2023.

[26] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman y Hall/CRC, 2011.

[27] V. S. Kushwah y A. Bajpai, «Machine learning and its algorithms: A Research,» *nternational Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, n.º 12S2, 2019.

[28] A. L'heureux, K. Grolinger, H. F. Elyamany y M. A. Capretz, «Machine learning with big data: Challenges and approaches,» *Ieee Access*, vol. 5, págs. 7776-7797, 2017.

[29] R. Ocaña, I. Agudo-Ruiz, F. J. López-Muñoz et al., «Detección de fraude en transacciones Blockchain usando procesos de Machine Learning, una Aproximación al Estado del Arte.,» 2023.

[30] S. Sah, «Machine Learning: A Review of Learning Types,» *Preprints*, jul. de 2020. DOI: 10.20944/preprints202007.0230.v1. dirección: https://doi.org/10.20944/preprints202007.0230.v1.

[31] A. Singh, N. Thakur y A. Sharma, «A review of supervised machine learning algorithms,» en *2016 3rd international conference on computing for sustainable global development (INDIACom)*, Ieee, 2016, págs. 1310-1315.

[32] F. Hahne, W. Huber, R. Gentleman, S. Falcon, R. Gentleman y V. Carey, «Unsupervised machine learning,» *Bioconductor case studies*, págs. 137-157, 2008.

[33] L. P. Kaelbling, M. L. Littman y A. W. Moore, «Reinforcement learning: A survey,» *Journal of artificial intelligence research*, vol. 4, págs. 237-285, 1996.

[34] S. Ardabili, A. Mosavi y A. R. Várkonyi-Kóczy, «Advances in machine learning modeling reviewing hybrid and ensemble methods,» en *International conference on global research and education*, Springer, 2019, págs. 215-227.

[35] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh et al., «Effects of distance measure choice on k-nearest neighbor classifier performance: a review,» *Big data*, vol. 7, n.º 4, págs. 221-248, 2019.

[36] M.-L. Zhang y Z.-H. Zhou, «A k-nearest neighbor based algorithm for multi-label classification,» en *2005 IEEE international conference on granular computing*, IEEE, vol. 2, 2005, págs. 718-721.

[37] V. K. Chauhan, K. Dahiya y A. Sharma, «Problem formulations and solvers in linear SVM: a review,» *Artificial Intelligence Review*, vol. 52, n.º 2, págs. 803-855, 2019.

[38] J. Brownlee, *XGBoost With python: Gradient boosted trees with XGBoost and scikit-learn*. Machine Learning Mastery, 2016.

[39] A. Parmar, R. Katariya y V. Patel, «A review on random forest: An ensemble classifier,» en *International conference on intelligent data communication technologies and internet of things (ICICI) 2018*, Springer, 2019, págs. 758-763.

[40] T. Windeatt, «Accuracy/diversity and ensemble MLP classifier design,» *IEEE Transactions on Neural Networks*, vol. 17, n.º 5, págs. 1194-1211, 2006.

[41] A. J. Ferreira y M. A. Figueiredo, «Boosting algorithms: A review of methods, theory, and applications,» *Ensemble machine learning: Methods and applications*, págs. 35-85, 2012.

[42] D. Sharma, R. Kumar y A. Jain, «Breast cancer prediction based on neural networks and extra tree classifier using feature ensemble learning,» *Measurement: Sensors*, vol. 24, pág. 100 560, 2022.

[43] A. A. Ibrahim, R. L. Ridwan, M. M. Muhammed, R. O. Abdulaziz y G. A. Saheed, «Comparison of the CatBoost classifier with other machine learning methods,» *International Journal of Advanced Computer Science and Applications*, vol. 11, n.º 11, 2020.

[44] M. Maftoun, N. Shadkam, S. S. S. Komamardakhi, Z. Mansor y J. H. Joloudari, «Malicious URL Detection using optimized Hist Gradient Boosting Classifier based on grid search method,» *arXiv preprint arXiv:2406.10286*, 2024.

[45] A. S. Pillai, «Cardiac disease prediction with tabular neural network,» 2022.

[46] H. Kamel, D. Abdulah y J. M. Al-Tuwaijari, «Cancer classification using gaussian naive bayes algorithm,» en *2019 international engineering conference (IEC)*, IEEE, 2019, págs. 165-170.

[47] S. Džeroski y B. Ženko, «Is combining classifiers with stacking better than selecting the best one?» *Machine learning*, vol. 54, págs. 255-273, 2004.

[48] D. Ruta y B. Gabrys, «Classifier selection for majority voting,» *Information fusion*, vol. 6, n.º 1, págs. 63-81, 2005.

[49] S. Saeb, L. Lonini, A. Jayaraman, D. Mohr y K. Kording, «Voodoo Machine Learning for Clinical Predictions,» *BioRxiv*, jun. de 2016. DOI: 10.1101/059774.

[50] W. Mao, X. Mu, Y. Zheng y G. Yan, «Leave-one-out cross-validation-based model selection for multi-input multi-output support vector machine,» *Neural Computing and Applications*, vol. 24, págs. 441-451, 2014.