



*ugr*

Universidad  
de **Granada**

TRABAJO FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA

# Aprendizaje Automático en Física de Partículas

---

**Optimizando el Análisis de Datos Simulados con MLOps**

**Autor**

Rubén Morillas López

**Tutores**

Alberto Guillén Perales  
Bruno Zamorano García



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—

Granada, Junio de 2024









*ugr* | Universidad  
de **Granada**

# Aprendizaje Automático en Física de Partículas

---

**Optimizando el Análisis de Datos Simulados con MLOps**

**Autor**

Rubén Morillas López

**Tutores**

Alberto Guillén Perales  
Bruno Zamorano García



# Aprendizaje Automático en Física de Partículas: Optimizando el Análisis de Datos Simulados con MLOps

Rubén Morillas López

**Palabras clave:** Identificación de partículas, Aprendizaje automático, Redes neuronales convolucionales, Piones, Kaones, Análisis de regresión, API de clasificación

## Resumen

La identificación de partículas subatómicas es fundamental en la física de partículas. Este estudio analiza la clasificación de piones y kaones en una cámara de proyección temporal de argón líquido, comparando métodos tradicionales de aprendizaje automático con redes neuronales.

Los resultados de este análisis revelan como las redes neuronales convolucionales reconocen mejor los patrones y extraen características complejas, superando los métodos tradicionales de Aprendizaje Automático, llegando a obtener un 0.937 de accuracy en la clasificación de las trazas de las partículas.

También se desarrollo un estudio de regresión de la energía de entrada, medida en GeV. Para ello se usaron técnicas tradicionales de Aprendizaje Automático. Además, se desarrolló una API que permite a los usuarios identificar partículas y su energía de entrada, facilitando su integración en trabajos de investigación.

Con este estudio vemos la eficacia de las Redes Convolucionales para identificar y clasificar las trazas de las partículas dentro del argón liquido, y la implementación de la API subraya su aplicabilidad práctica para la comunidad científica.



# **Machine Learning in Particle Physics: Optimising Simulated Data Analysis with MLOps**

Rubén Morillas López

**Keywords:** Particle identification, Machine learning, Convolutional neural networks, Pions, Kaons, Regression analysis, Classification APIs

## **Abstract**

The identification of subatomic particles is fundamental in particle physics. This study focuses on the classification of pions and kaons in a liquid argon time projection chamber, comparing traditional machine learning methods with convolutional neural networks.

The results demonstrate that CNNs better recognise patterns and extract complex features, outperforming traditional machine learning methods and achieving an accuracy of 0.937 in particle track classification.

Furthermore, a regression study was conducted to estimate the input energy, measured in GeV, using traditional machine learning techniques. An API was developed to allow users to identify particles and determine their input energy, facilitating integration into research work.

This study highlights the effectiveness of convolutional networks for identifying and classifying particle tracks in liquid argon and underscores the practical applicability of the developed API for the scientific community.



Yo, **Rubén Morillas López**, alumno de la titulación TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informáticas y de Telecomunicación de la Universidad de Granada, con DNI 74539338-H, 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:** Rubén Morillas López

Granada a 10 de Junio de 2024



**D. Alberto Guillén Perales**, Catedrático de Universidad del Área Departamento de Ingeniería de Computadores, Automática y Robótica de la Universidad de Granada.

**D. Bruno Zamorano García**, Profesor Contratado Doctor Indefinido del Departamento de Departamento de Física Teórica y del Cosmos de la Universidad de Granada.

**Informan:**

Que el presente trabajo titulado **Aprendizaje Automático en Física de Partículas, Optimizando el Análisis de Datos Simulados con MLOps**, ha sido realizado bajo su supervisión por **Rubén Morillas López**, 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 X de Junio 2024.

**Los directores:**

**Alberto Guillén Perales**

**Bruno Zamorano García**



# **Agradecimientos**

Me gustaría dedicar este trabajo a todas las personas que me han acompañado en este camino hasta el día de hoy, en especial a mis padres, mi hermana, mis abuelos, mi tito Florencio, mi prima Carmen y a Raquel. Sin ellos llegar hasta aquí hubiera sido muy difícil.

De este camino me llevo todas las vivencias y experiencias que me ha dado la vida, así como el trabajo continuo y el esfuerzo que he tenido que dedicar mientras compaginaba mis estudios con la vida laboral.

Como decía mi abuelo José, que no pudo ver como llegaba a ser ingeniero pero que seguro que él junto con mi abuela Rosario están muy orgullosos de que yo haya conseguido todos mis propósitos, "Niño estudia, que la vida está muy mala". Aquí quiero decirles que todavía no ha acabado el camino, pero que seguro que seguiré consiguiendo todo lo que me proponga. Gracias a ellos por sus sabias palabras y consejos.

A mis padres por haberme apoyado tanto anímica como económicamente. Han sido un pilar fundamental en mi formación y en mi vida, ya que jamás me han puesto algún impedimento a la hora de estudiar y si han tenido que hacer un esfuerzo tanto por mí, como por mi hermana lo han hecho. Esto también es parte de vosotros.

Agradecer a Fran, por haberme hecho el camino universitario más llevadero, tanto dentro como fuera de las aulas.





Índice	1
--------	---

---

# Índice

Índice	1
Índice de cuadros	4
Índice de figuras	5
<b>0. Objetivos</b>	<b>8</b>
0.1. Objetivo Principal . . . . .	8
0.2. Objetivos extras . . . . .	8
<b>1. Introducción</b>	<b>1</b>
1.1. Simulación de los datos . . . . .	7
<b>2. Descripción de los Datos</b>	<b>8</b>
2.1. Correlación de los datos . . . . .	10
2.2. Distribución de los datos . . . . .	14
2.2.1. Variable: <i>hitX</i> . . . . .	15
2.2.2. Variable: <i>hitY</i> . . . . .	16
2.2.3. Variable: <i>hitZ</i> . . . . .	17
2.2.4. Variable: <i>hitInteg</i> . . . . .	18
2.3. Variable: <i>trueE</i> . . . . .	19
2.3.1. Media, desviación, valor mínimo y máximo . . . . .	20
<b>3. Separación de los Datos</b>	<b>20</b>
<b>4. Métricas de error</b>	<b>25</b>
4.1. Problema de clasificación . . . . .	25
<b>5. Planificación del Trabajo</b>	<b>28</b>
<b>6. Problema de Clasificación</b>	<b>29</b>
6.1. Modelos Tradicionales de Aprendizaje Automático . . . . .	29
6.1.1. Adaptación del conjunto de datos . . . . .	29

6.1.2. Elección del número de hits por suceso . . . . .	30
6.1.3. Modelos Aprendizaje Automático Tradicionales . . . . .	32
6.2. Elección de los hiperparámetros para los distintos modelos . . . . .	36
6.2.1. Entrenamiento . . . . .	39
6.2.2. Resultados del entrenamiento . . . . .	40
6.2.3. Entrenamiento por Validación Cruzada . . . . .	45
6.3. Modelos de Redes Neuronales . . . . .	51
6.3.1. Adaptación del conjunto de datos para el entrenamiento .	51
6.3.2. Arquitecturas para el entrenamiento . . . . .	54
6.3.3. Entrenamiento . . . . .	58
6.3.4. Resultados de entrenamiento . . . . .	60
6.3.5. Comparación de entrenamientos . . . . .	66
6.3.6. Entrenamiento del mejor modelo . . . . .	69
6.3.7. Resultados finales . . . . .	72
<b>7. Problema de Regresión</b>	<b>74</b>
7.1. Adaptación del conjunto de datos al entrenamiento . . . . .	74
7.2. Elección del número de hits por suceso . . . . .	75
7.3. Modelos tradicionales de aprendizaje automático para la regresión	75
7.4. Entrenamiento . . . . .	78
7.5. Resultados de Entrenamiento . . . . .	80
7.5.1. Resultados Entrenamiento con el conjunto de datos para la clasificación . . . . .	80
7.5.2. Resultados entrenamiento con el tipo de partícula añadido	84
7.5.3. Comparación de Rendimiento del Modelo Random Forest con Diferentes Conjuntos de Entrada . . . . .	88
7.6. Entrenamiento del mejor modelo con mejores resultados . . . . .	93
7.7. Gráficas de errores . . . . .	93
7.7.1. Gráfico de Errores Residuales . . . . .	94
7.7.2. Gráfico de Dispersion de Errores Residuales . . . . .	95
7.7.3. Gráfico de Predicciones contra Valores Reales . . . . .	96
7.7.4. Conclusión . . . . .	97
7.8. Resultado final . . . . .	97

<b>8. Análisis de los Resultados</b>	<b>98</b>
8.1. Etapa de Clasificación . . . . .	98
8.2. Etapa de Regresión . . . . .	99
<b>9. Despliegue del Modelo</b>	<b>101</b>
9.1. Montaje de FastAPI . . . . .	101
9.2. Script de Python . . . . .	101
9.2.1. Main . . . . .	101
9.2.2. Script para reentrenar los modelos . . . . .	102
9.2.3. Script para la clasificación de las partículas . . . . .	102
9.2.4. Script para la regresión de la energía de entrada . . . . .	103
9.3. La API de FastAPI . . . . .	103
9.3.1. Clasificación . . . . .	104
9.3.2. Regresión . . . . .	105
9.3.3. Reentrenamiento de los modelos . . . . .	107
<b>10. Presupuesto</b>	<b>108</b>
<b>11. Conclusiones</b>	<b>110</b>
11.1. Trabajos Futuros . . . . .	111

## Índice de cuadros

1.	Descripción numérica de las características . . . . .	20
2.	Distribución del conjunto de datos . . . . .	25
3.	Matriz de Confusión [3] . . . . .	26
4.	Parámetros de los modelos de Aprendizaje Automático . . . . .	35
5.	Tabla resumen entrenamiento modelos de Aprendizaje Automático	41
6.	Mejor N para cada modelo . . . . .	45
7.	Resultados en términos de accuracy del entrenamiento por Validación Cruzada . . . . .	47
8.	Resumen resultados Accuracy . . . . .	49
9.	Arquitectura Red Convolucional . . . . .	55
10.	Arquitectura de Redes Recurrentes con capas LSTM (RNN-LSTM)	55
11.	Arquitectura de Redes Recurrentes con capas GRU (RNN-GRU) .	56
12.	Resultados de accuracy en los distintos tipos de energías . . . . .	72
13.	Parámetros de los modelos de Aprendizaje Automático para la regresión . . . . .	78
14.	Resumen de resultados del problema de la regresión de la energía de entrada . . . . .	93
15.	Resultados problema regresión energía de entrada . . . . .	97
16.	Presupuesto del proyecto . . . . .	110

## Índice de figuras

1.	Comparación entre Neuronas Biológicas y Redes Neuronales Artificiales [16]. . . . .	2
2.	Esquema del Experimento DUNE: Del Acelerador de Protones a la Detección Subterránea [10] . . . . .	4
3.	Esquema del Detector Subterráneo del Experimento DUNE [13] .	6
4.	Montaje del Detector de Partículas para el Experimento DUNE [13]	7
5.	Estructura inicial de los datos . . . . .	9
6.	Matriz de correlación inicial . . . . .	11
7.	Matriz de correlación con los datos de entrenamiento . . . . .	13
8.	Gráfica distribución variable <i>hitX</i> . . . . .	15
9.	Gráfica distribución variable <i>hitY</i> . . . . .	16
10.	Gráfica distribución variable <i>hitZ</i> . . . . .	17
11.	Gráfica distribución variable <i>hitInteg</i> . . . . .	18
12.	Gráfica distribución variable <i>trueE</i> . . . . .	19
13.	Separación del número de sucesos en los distintos grupos . . . . .	21
14.	Número de sucesos de cada partícula en el conjunto de entrenamiento . . . . .	22
15.	Número de sucesos de cada partícula en el conjunto de validación	23
16.	Número de sucesos de cada partícula en el conjunto de prueba .	24
17.	Estructura de datos para el problema de clasificación con modelos tradicionales de Aprendizaje Automático . . . . .	30
18.	Distribución del número de hits por sucesos . . . . .	30
19.	Diagrama de caja de la distribución del número de hits por suceso	31
20.	Comparativa de accuracy de todos los modelos tradicionales de Aprendizaje Automático . . . . .	41
21.	Comparativa de los tiempos de entrenamiento de los modelos de Aprendizaje Automático . . . . .	42
22.	Matriz de confusión del entrenamiento de LightGBM . . . . .	43
23.	Matriz de confusión del entrenamiento de Random Forest . . . . .	43
24.	Matriz de confusión del entrenamiento de Gradient Boosting . . .	44
25.	Matriz de confusión del entrenamiento de XGBoost . . . . .	44

26.	Comparativa de accuracy en el entrenamiento de los modelos de Aprendizaje Automático por Validación Cruzada . . . . .	46
27.	Función de pérdida del entrenamiento de XGBoost . . . . .	48
28.	Resultados en términos de Accuracy de todos los conjuntos de entrenamiento . . . . .	50
29.	Frecuencia de la longitud de las cadenas de hits por sucesos . . . .	52
30.	Longitud de las cadenas de hits por sucesos . . . . .	53
31.	Longitud de las cadenas de hits por sucesos . . . . .	54
32.	Comportamiento de la función de perdida entrenamiento arquitectura CNN . . . . .	60
33.	Comportamiento del accuracy en el entrenamiento arquitectura CNN	61
34.	Comportamiento de la función de perdida entrenamiento arquitectura LSTM . . . . .	62
35.	Comportamiento del accuracy en el entrenamiento arquitectura LSTM . . . . .	63
36.	Comportamiento de la función de perdida entrenamiento arquitectura GRU . . . . .	64
37.	Comportamiento del accuracy en el entrenamiento arquitectura GRU	65
38.	Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de entrenamiento . . . . .	66
39.	Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de validación . . . . .	67
40.	Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de entrenamiento . . . . .	67
41.	Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de validación . . . . .	68
42.	Función de pérdida del entrenamiento del mejor modelo (CNN) .	70
43.	Accuracy del entrenamiento del mejor modelo (CNN) . . . . .	71
44.	Resultados de accuracy en los distintos tipos de energías . . . .	73
45.	Estructura de datos para el problema de regresión con modelos tradiciones de Aprendizaje Automático . . . . .	75
46.	Error Absoluto Medio . . . . .	80
47.	Error Cuadrático Medio . . . . .	81

48.	Coeficiente de Determinación . . . . .	82
49.	Error Absoluto Medio Normalizado . . . . .	82
50.	Error Cuadrático Medio Normalizado . . . . .	83
51.	Error Absoluto Medio . . . . .	85
52.	Error Cuadrático Medio . . . . .	85
53.	Coeficiente de Determinación . . . . .	86
54.	Error Absoluto Medio Normalizado . . . . .	87
55.	Error Cuadrático Medio Normalizado . . . . .	87
56.	Comparativa para Random Forest del Error Absoluto Medio . . .	89
57.	Comparativa para Random Forest del Error Cuadrático Medio . .	90
58.	Comparativa para Random Forest del Coeficiente de Determinación	90
59.	Comparativa para Random Forest del Error Absoluto Medio Normalizado . . . . .	91
60.	Comparativa para Random Forest del Error Cuadrático Medio Normalizado . . . . .	92
61.	Errores Residuales . . . . .	94
62.	Errores Residuales . . . . .	95
63.	Errores Residuales . . . . .	96
64.	Servicio cargado satisfactoriamente . . . . .	103
65.	Interfaz de usuario de FastAPI . . . . .	104
66.	End-Point para la clasificación . . . . .	104
67.	Ejemplo de clasificación de evento . . . . .	105
68.	End-Point para la regresión . . . . .	106
69.	Ejemplo de regresión de evento . . . . .	106
70.	End-Point para el reentrenamiento de los modelos . . . . .	107
71.	Ejemplo de reentrenamiento de los modelos . . . . .	107
72.	Coste máquina virtual con GPU . . . . .	109

## 0. Objetivos

En esta sección veremos los distintos tipos de objetivos que abarcará este TFG. Estos objetivos se han clasificado en tres categorías principales: objetivos generales, objetivos específicos y objetivos extras.

### 0.1. Objetivo Principal

El objetivo de este Trabajo de Fin de Grado (TFG) es desarrollar un sistema de análisis y clasificación de sucesos en experimentos de física de partículas utilizando técnicas avanzadas de aprendizaje automático. En particular se implementarán dos enfoques principales:

1. **Redes Neuronales para Clasificación de sucesos Complejos:** Este enfoque empleará redes neuronales avanzadas para clasificar sucesos complejos. Para manejar sucesos de tamaño variable, se utilizarán técnicas de padding, permitiendo que los modelos procesen secuencias de datos de longitud variable sin perder información relevante. Los datos se procesarán en forma de hits individuales, cada uno con su identificador de suceso, maximizando así la utilización de la información recogida por los experimentos.
2. **Modelos Clásicos de Aprendizaje Automático:** Este enfoque agrupará los hits por suceso en una estructura predefinida, lo cual puede implicar una pérdida de información debido a la variabilidad en el número de hits por suceso. Se determinará un número fijo  $N$  de hits por suceso. Si un suceso tiene menos hits que  $N$ , se llenará con ceros; si tiene más, se seleccionarán los hits más recientes según la variable *hitTime*.

Se evaluarán ambos enfoques para determinar cuál captura más eficazmente la información relevante y proporciona mejores resultados en la clasificación de partículas, como kaones y piones.

### 0.2. Objetivos extras

Este proyecto tiene como objetivo principal desarrollar y desplegar un modelo de clasificación de partículas que, además de identificar el tipo de partícula, pueda

estimar su energía de entrada expresada en GeV. Para ello, se utilizarán dos modelos: uno de clasificación supervisada para identificar el tipo de partícula (como kaones, piones, protones, etc.), y otro de regresión para estimar su energía de entrada.

Ambos modelos se integrarán en una API utilizando FastAPI, permitiendo a cualquier usuario procesar eventos de partículas a partir de archivos CSV. La API leerá los datos de los eventos, ejecutará la clasificación y regresión, y devolverá una respuesta indicando el tipo de partícula y su energía de entrada, por ejemplo: "Se trata de un kaon y lo hace con una energía de entrada de 1.5 GeV."

Esta herramienta será valiosa para la comunidad científica, facilitando la clasificación y análisis de partículas de manera precisa y eficiente.

## 1. Introducción

El Aprendizaje Automático se utiliza para resolver problemas de clasificación y regresión en diversos campos de nuestra vida cotidiana. A principios del siglo XXI aparece una rama del Aprendizaje Automático llamada Aprendizaje Profundo. La diferencia con el Aprendizaje Automático es que usa redes neuronales que funcionan de forma muy parecida a las conexiones neuronales biológicas de nuestro cerebro.

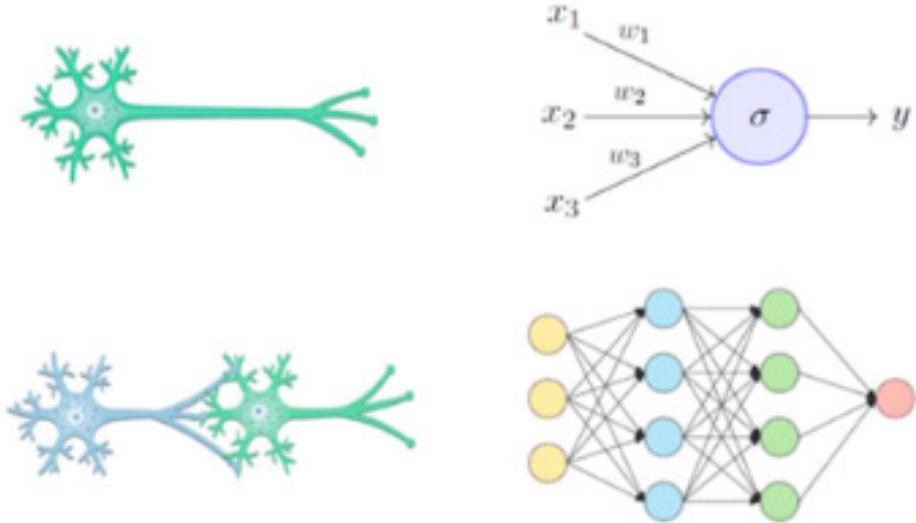


Figura 1: Comparación entre Neuronas Biológicas y Redes Neuronales Artificiales [16].

La física, como ciencia experimental, siempre ha requerido analizar los datos de diversos experimentos. Por esa razón, la física ha recurrido constantemente a herramientas de la ciencia de datos. Como factor importante y requerible para poder aplicar la ciencia de datos dentro de la física, se cumple con el principal requisito que es el enorme volumen de datos de que se dispone, lo que hace que la física sea un excelente laboratorio para probar modelos y herramientas de la ciencia de datos, así como el desarrollo de nuevos algoritmos [30] [31].

En estos modelos, se utilizan como entrada los datos y las características más relevantes de experimentos previos. Estos datos han sido recopilados y etiquetados adecuadamente. A continuación, entrenaremos los modelos con estos datos y sus respectivas etiquetas para que puedan aprender los patrones presentes en las entradas y predecir las etiquetas con el menor margen de error posible. De este modo, el modelo podrá generalizar un patrón para futuros datos [30] [31].

En los modelos de Aprendizaje Automático tradicionales los resultados suelen

ser interpretables, mientras que los modelos de Aprendizaje Profundo no permiten comprender cómo se llega al resultado final, lo que dificulta su interpretación. El uso de estos modelos en muchas áreas ha generado problemas éticos y morales relacionados con la falta de interpretabilidad. En cambio, en el ámbito de la física, este problema no surge por cuestiones éticas, sino porque no basta con saber que algo sucede; también es necesario comprender el porqué de los fenómenos observados.

La física ha evitado el uso de Aprendizaje Profundo debido a la falta de interpretabilidad de los resultados. Sin embargo, ahora se está discutiendo su adopción porque no queda más remedio, dada la complejidad y el volumen de los datos actuales.

El objetivo dentro de nuestro problema será la clasificación binaria de dos tipos de partículas: kaones y piones. Nos enfocaremos en la vía de desintegración  $p^- \rightarrow k^+ + \nu^-$ . La identificación del protón a través de esta vía se basa en detectar kaones en un entorno de argón líquido. Para ello, desarrollaremos métodos y análisis específicos para una correcta identificación de estas partículas.

Los kaones y los piones son tipos de mesones, partículas subatómicas que desempeñan roles cruciales dentro de la física de partículas [9] [44]:

- Los kaones son un tipo de mesón, partículas subatómicas compuestas por un quark y un antiquark. Tienen una masa intermedia entre los mesones más ligeros, como los piones, y los más pesados. Los kaones son inestables y se desintegran en otras partículas subatómicas. Estos mesones se utilizan en experimentos de física de partículas para estudiar la violación de simetrías fundamentales en la naturaleza y explorar las interacciones fuertes y débiles, lo cual es crucial para entender mejor el comportamiento de las partículas y las fuerzas que actúan entre ellas a nivel subatómico.
- Por otro lado, los piones son otro tipo de mesón y son las partículas mesónicas más ligeras. También están compuestos por un quark y un antiquark. Los piones juegan un papel crucial en la interacción fuerte que mantiene

unidos a los protones y neutrones en el núcleo atómico. Estas partículas son fundamentales en el estudio de las fuerzas nucleares y en la compresión de la estructura del núcleo atómico. Los piones y los kaones son esenciales en la física de partículas y se estudian para comprender mejor las fuerzas fundamentales y las interacciones que gobiernan la materia a nivel subatómico.

A altas energías, distinguir un kaon de otras partículas subatómicas, como puede ser un píón, es en general más sencillo ya que su masa es 3.5 veces mayor y sus distintos productos de la desintegración. Sin embargo, a bajas energías puede llegar a confundirse con piones con más facilidad.

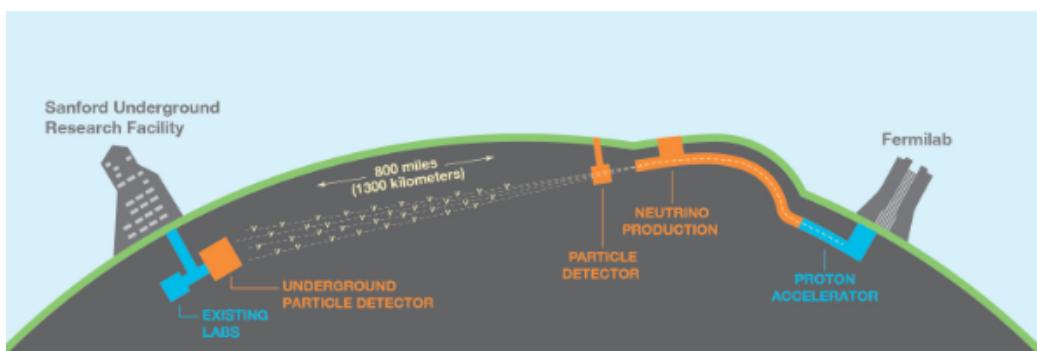


Figura 2: Esquema del Experimento DUNE: Del Acelerador de Protones a la Deteción Subterránea [10]

El experimento DUNE [10] [14] (*Deep Underground Neutrino Experiment*) es un proyecto internacional avanzado en el campo de la física de neutrinos. DUNE se centrará en los neutrinos, las partículas de materia más abundantes en el universo, para investigar cuestiones fundamentales sobre la naturaleza de la materia y la evolución del universo. El experimento contará con dos detectores en el haz de neutrinos más intenso del mundo. Uno de ellos estará a más de un kilómetro de profundidad en el Laboratorio de Investigación Subterránea de Sanford en Dakota del Sur a 1,300 kilómetros de distancia. Estos detectores permitirán a los científicos buscar nuevos fenómenos subatómicos y transformar nuestra comprensión de

los neutrinos.

En el CERN (*Centro de Investigación Europeo*), hay dos prototipos de los detectores lejanos. El primero comenzó a recolectar datos en septiembre de 2018 y el segundo está en construcción. La línea de luz es llamada LBNF (*Long Baseline Neutrino Facility*) proporcionará el haz de neutrinos y la infraestructura para los detectores de DUNE. La excavación y construcción en el Sanford Lab comenzó el 21 de julio de 2017.

En el experimento DUNE, el argón líquido se utiliza como medio detector en los detectores de neutrinos. El argón líquido es crucial ya que los neutrinos son partículas extremadamente difícil de detectar porque interactúan muy débilmente con la materia. El argón líquido permite la detección de estos sucesos raros mediante el uso de cámaras de proyección temporal. Cuando un neutrino interactúa con un núcleo de argón, produce partículas cargadas que atraviesan el argón líquido y generan electrones de ionización.

El argón líquido tiene una alta densidad y pureza, lo que maximiza la probabilidad de interacción de los neutrinos. Esto ayuda a contener y detectar eficientemente las trayectorias de las partículas resultantes. Cuando los electrones de ionización se mueven en el campo eléctrico, generan luz de centello y señales eléctricas que se pueden recoger y analizar. Esto proporciona una imagen clara y precisa de la interacción del neutrino y permite reconstruir las trayectorias de las partículas con gran detalle.

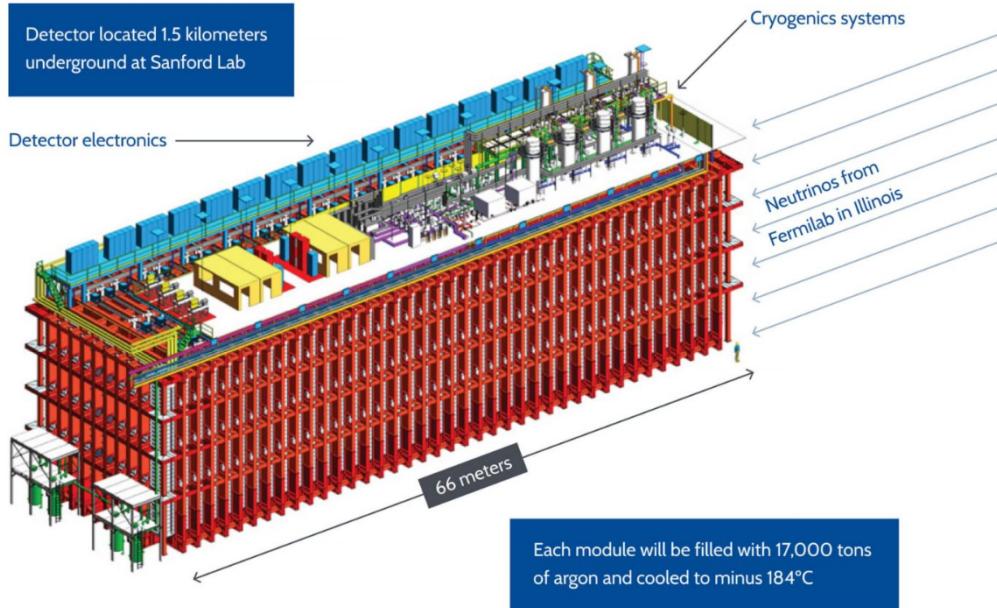


Figura 3: Esquema del Detector Subterráneo del Experimento DUNE [13]

La imagen anterior es uno de los cuatro módulos del detector. Los cuatro módulos contendrán casi 70,000 toneladas de argón líquido ultrapuro. El argón es un elemento que está en el aire y resulta idóneo para el estudio de los neutrinos: el gran tamaño de sus átomos hace más probable la interacción.

Cuando un neutrino choca con el núcleo de un átomo de argón, produce partículas que desprenden electrones en el argón líquido. Un alto voltaje atrae estos electrones hacia planos de cables instalados en el interior de cada módulo detector. El resultado es una señal distintiva y precisa que proporcionará información importante sobre la interacción de los neutrinos y permite reconstruir en 3D las trayectorias de las partículas cargadas producidas durante el proceso.



Figura 4: Montaje del Detector de Partículas para el Experimento DUNE [13]

El experimento DUNE se encuentra actualmente en construcción, por lo que no es posible trabajar con los datos reales del experimento. Es por ello por lo que los datos han sido obtenidos mediante simulaciones.

### 1.1. Simulación de los datos

Para la obtención de los datos para el experimento, se utilizó el paquete GEANT4 [6] para la simulación de partículas y radiaciones a través de la materia. GEANT4 permite desarrollar aplicaciones que simulan una amplia variedad de configuraciones experimentales y detectores, junto con diversas fuentes de radiación. Estas aplicaciones registran las magnitudes físicas seleccionadas, que resultan de las interacciones de las partículas fuente y secundarias con el material presente en la configuración simulada. GEANT4 es una herramienta aplicada a áreas como la física de altas energías, nuclear y médica.

Por otro lado, el software LArSoft [27] se utilizó para la simulación, reconstrucción y análisis de datos en detectores TPC de gran escala. Basado en el marco de trabajo de las Herramientas de Análisis y Reconstrucción (ART) del Fermilab, LArSoft permite el acceso eficiente a los registros de sucesos y la construcción de colecciones de sucesos. La combinación de GEANT4 y LArSoft proporcionó una solución efectiva para simular las condiciones y sucesos esperados en los detectores del experimento DUNE, abordando con precisión la interacción de partículas y radiación con la materia y la respuesta del detector de argón líquido. Esta combinación asegura que los datos obtenidos sean adecuados para el estudio de los sucesos previstos en DUNE.

Cabe destacar que los datos utilizados en este experimento fueron generados por Bruno Zamorano, uno de los tutores de este TFG, quien me los proporcionó para su análisis y experimentación de este trabajo.

## 2. Descripción de los Datos

Para la estructura de los datos, asumimos que cada vez que una partícula penetra en el volumen de argón líquido se origina un suceso que incluye múltiples puntos de impacto, denominados hits. Estos hits resultan de la llegada de electrones libres a los planos anódicos. A partir de esta premisa, los datos se organizan en un conjunto de datos que comprende de 8 características, por tanto, este conjunto de datos tendrá 8 columnas, las cuales albergan información esencial para llevar a cabo el análisis de clasificación entre piones y kaones. Cada hit se representa en una fila y las filas que pertenecen al mismo suceso están ubicadas de forma consecutiva.

eventID	PDGcode	trueE	hitX	hitY	hitZ	hitTime	hitInteg
0	221	...	...	...	...	...	...
0	221	...	...	...	...	...	...
0	221	...	...	...	...	...	...
...	...	...	...	...	...	...	...
19999	321	...	...	...	...	...	...

Figura 5: Estructura inicial de los datos

Las 8 características que componen el conjunto de datos son:

- *eventID*: Identificador del suceso. Es un número común a todos los hits pertenecientes al mismo suceso
- *PDGcode*: Es el número que identifica inequívocamente a cada partícula. En nuestro caso tendremos un *PDGcode* = 321 para los kaones y un *PDGcode* = 211 para los piones.
- *trueE*: Corresponde a la energía total de entrada de la partícula expresada en GeV. Cada suceso tiene el mismo valor de *trueE*.
- *hitX*: Coordenada del eje X expresada en cm
- *hitY*: Coordenada del eje Y expresada en cm
- *hitZ*: Coordenada del eje Z expresada en cm
- *hitTime*: tiempo relativo del hit
- *hitInteg*: integral de la señal del hit en el plano de colección expresada en ADC x tick, equivalente aproximadamente a 1.9 KeV. Cada hit tiene un valor distinto.

Hay que destacar que la variable *trueE* no se puede usar para entrenar los distintos modelos de clasificación ya que es un valor desconocido en las mediciones reales. Debido a este motivo, después del estudio del clasificador en diferentes

partículas, ya sea kaones o piones, se va a realizar otro estudio de regresión de esta variable.

En la relación de coordenadas espaciales, *hitX*, *hitY* y *hitZ*, hay que tener en cuenta que las partículas han sido inyectadas en el punto  $(-100, 0, 150)$  para garantizar que están correctamente contenidas en el detector. Para un mejor entendimiento se ha transformada estos puntos de todos los hits al punto  $(0, 0, 0)$  para que sea más fácil y comprensible su representación.

Se simularon 20.000 sucesos de cada partícula, pero no llegaron a generar 496 sucesos de piones y 52 de kaones. Por tanto, tendremos 19.504 y 19.948 partículas de piones y kaones respectivamente. Además, como los datos son simulacros de una simulación virtual no es necesario ningún tratamiento adicional al respecto.

## 2.1. Correlación de los datos

Se ha llevado un estudio de los datos para conocer la relación entre las distintas variables. Este estudio se realiza mediante una matriz de correlación [38]. Una matriz de correlación muestra el grado de correlación entre las múltiples intersecciones de medida como una matriz de celdas rectangulares. Cada celda de la matriz representa la intersección de dos medidas y el color de la celda el grado de correlación entre esas dos medidas.

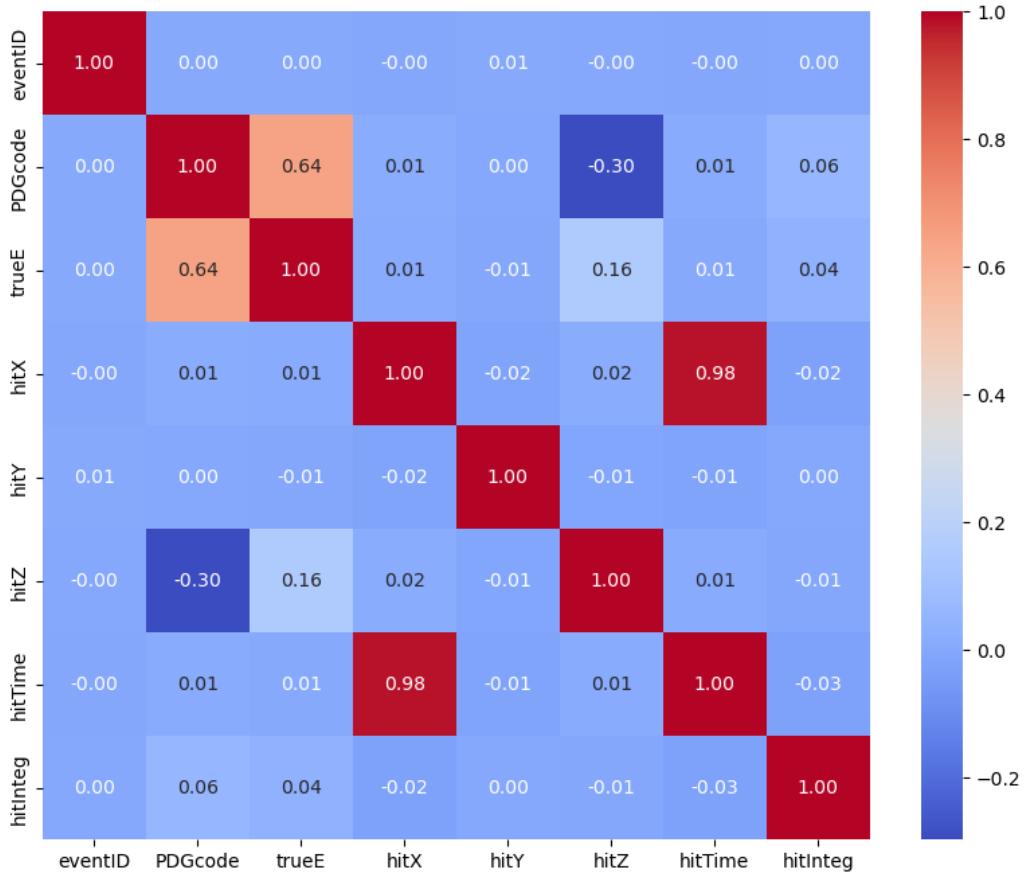


Figura 6: Matriz de correlación inicial

Como podemos ver en la figura 6, podemos observar distintas interrelaciones entre las variables involucradas en el estudio de clasificación entre piones y kaones. Una observación crucial es la alta correlación entre las variables *hitTime* y *hitX*, que es evidencia por un coeficiente de correlación de 0.98. Esta fuerte correlación sugiere que una de las variables podría ser redundante, ya que una parece ser derivada o estar muy influenciada por la otra.

En contextos de modelado predictivo y análisis de datos, la presencia de variables altamente correlacionadas puede llevar a problemas de multicolinealidad, [42]. La multicolinealidad puede también incrementar la varianza de los coeficientes estimados, haciéndolos inestables y sensibles a cambios en el modelo.

Dado este alto nivel de correlación y entendiendo que la redundancia entre *hitTime* y *hitX* puede ser problemática, se recomienda eliminar *hitTime* del conjunto de datos. Esta decisión se basa en la premisa de que mantener ambas variables no aporta información adicional significativa y, por el contrario, puede complicar el modelo sin mejorar su capacidad predictiva. Además, eliminar *hitTime* puede simplificar el análisis posterior sin comprometer la integridad del mismo, permitiendo así concentrarse en otras variables que aportan información única para la clasificación entre piones y kaones.

Finalmente, la matriz con los datos de entrenamiento quedaría como en la figura 7

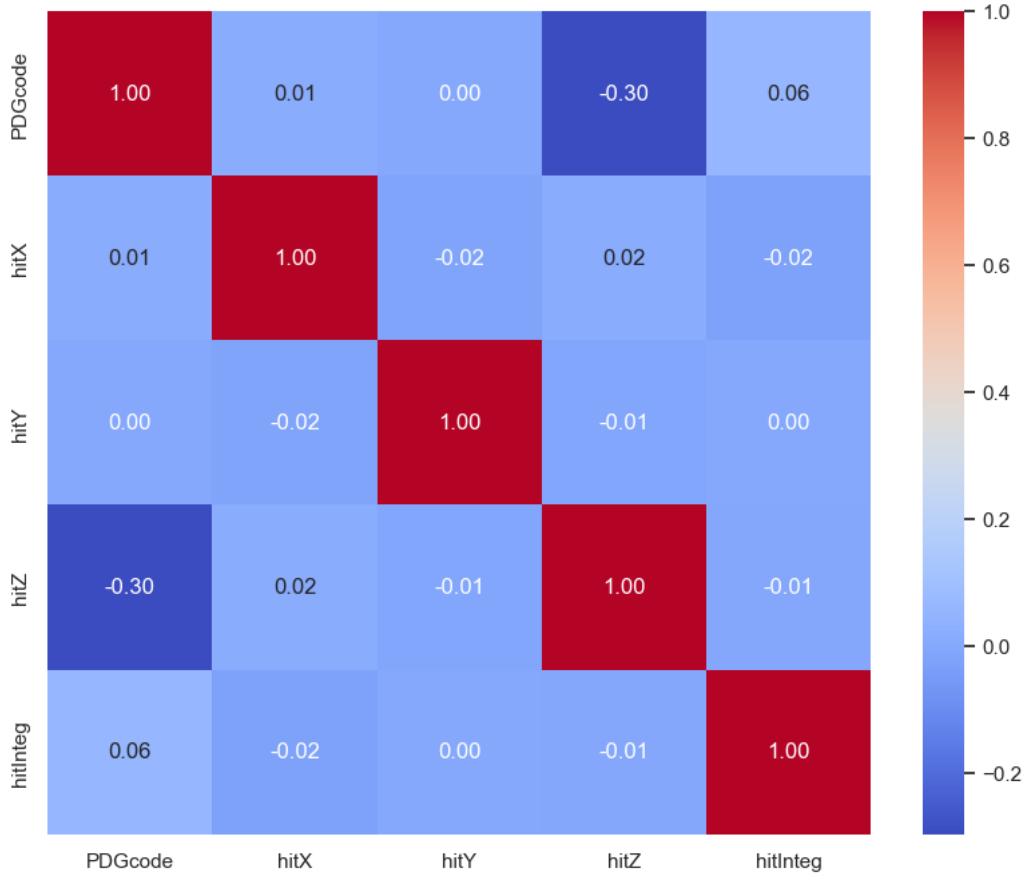


Figura 7: Matriz de correlación con los datos de entrenamiento

La mayoría de las correlaciones son muy débiles, lo que indica que las variables son mayormente independientes unas de otras. Esto es útil en los modelos predictivos, ya que sugiere que cada variable aporta información única sin redundancia significativa.

La correlación de *PDGcode* y *hitZ* es notable. Por lo que podemos ver los piones dejan trazas más largas, por tanto llegan a *Z* más altos. Esto es totalmente consistente con la física, ya que los piones tienen una masa menor y por lo tanto depositan menos energía por cada centímetro recorrido.

Dado que las variables son en su mayoría independientes, los modelos predic-

tivos basados en estas no deberían sufrir de multicolinealidad, lo que mejora la estabilidad y la interpretación de los modelos.

## 2.2. Distribución de los datos

El estudio de la distribución de los datos [40] en cualquier análisis científico o de datos es crucial para comprender la naturaleza de las variables involucradas, identificar posibles irregularidades, y determinar técnicas estadísticas más apropiadas para su análisis. La distribución de los datos puede ser relevante para encontrar patrones subyacentes, tendencias, y asociaciones que son esenciales para tomar decisiones. Cuando se analiza la distribución de los datos, algunos aspectos clave a considerar incluyen la centralidad, la dispersión, la forma de la distribución.

En el contexto de nuestro problema, la física de partículas, como en el análisis de datos de detección de partículas, estudiar la distribución puede ayudar a diferenciar entre diferentes tipos de partículas y entender cómo interactúan en diferentes condiciones. A continuación realizaremos un estudio de las 4 variables que vamos a usar para resolver los diferentes problemas.

### 2.2.1. Variable: $hitX$

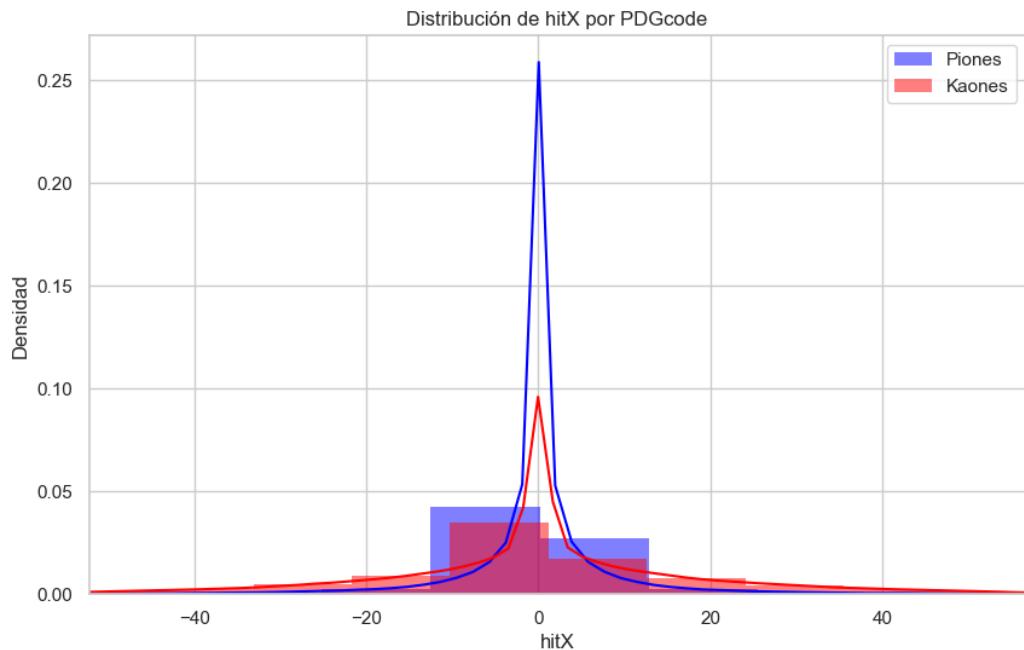


Figura 8: Gráfica distribución variable  $hitX$

La figura 8 presenta la distribución de la variable  $hitX$ .

Los piones como podemos ver tienen un pico muy pronunciado alrededor de  $hitX = 0$  esto es debido a que en este punto es donde se lanza las partículas ya que ha sido modificado los valores, como se ha indicado anteriormente. Esto indica que la mayoría de los hits de piones se concentran cerca del origen, con una alta densidad en esta zona. Esto puede indicar una precisión más alta en el proceso de detección ya que hace que los hits de los piones sean más predecibles y concentrados.

Los kaones tienen una distribución más dispersa y menos centrada. No muestran un pico tan definido como los piones, lo que indica una variabilidad mayor entre las posiciones de sus hits. Los kaones, sin embargo, muestran un máximo más pequeño que los piones y una base más ancha, reflejando una mayor dispersión.

sión en sus valores de  $hitX$

### 2.2.2. Variable: $hitY$

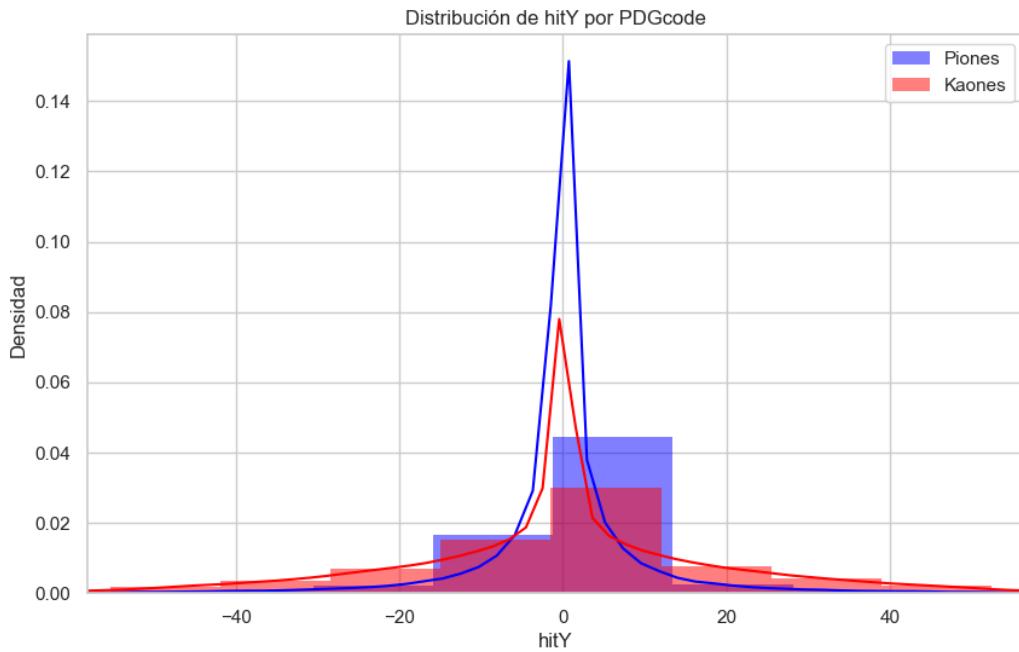


Figura 9: Gráfica distribución variable  $hitY$

Como podemos ver en la figura 9 tanto los piones como los kaones presentan picos pronunciados alrededor de  $hitY = 0$ . Esto indica que la mayoría de los sucesos de detección de ambas partículas tienden a centrarse cerca de este punto. Ambas distribuciones muestran una simetría notable alrededor del eje central. Esto puede indicar, que en el proceso de detección, los hits de ambas partículas se distribuyen de manera uniforme hacia la izquierda y la derecha del centro

El pico de los piones es notablemente más agudo que el de los kaones, lo que sugiere que los piones tienen una distribución más concentrada en comparación con los kaones. Los piones y los kaones exhiben colas que se extienden hacia los extremos de la gráfica, aunque las colas de los piones caen más rápidamente que

la de los kaones. Esto implica que es menos probable que los piones se detecten a grandes distancias del centro, en comparación con los kaones.

### 2.2.3. Variable: *hitZ*

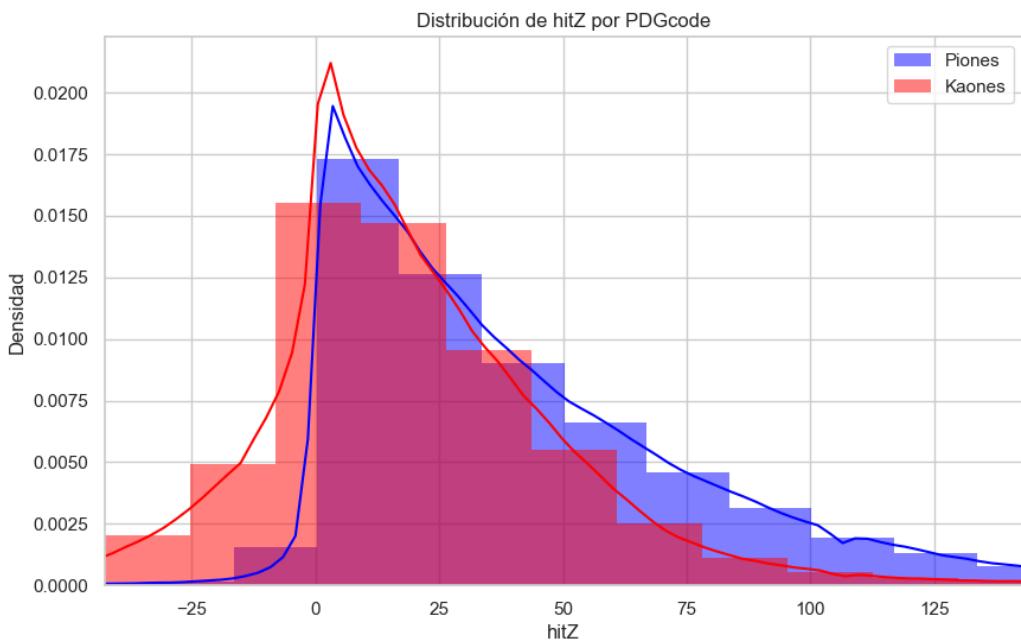


Figura 10: Gráfica distribución variable *hitZ*

Como podemos ver en la figura 10, la distribución de los piones tiene un pico bastante pronunciado alrededor de  $hitZ = 0$ . Esto sugiere que los piones tienden a ser detectados cerca del centro del detector en la dirección Z. La distribución es relativamente simétrica alrededor de este pico. En cambio, la distribución de los kaones también muestra un pico en la misma zona que los piones pero, es menos pronunciado y más ancho, indicando una mayor dispersión en la posición de los hits a lo largo del eje Z. Los kaones tienden a tener un rango más extenso de interacciones a lo largo del eje Z en comparación con los piones.

Ambas distribuciones tienen colas que se extienden hacia valores más altos

de  $hitZ$ , pero los piones muestran una cola más larga que los kaones. Esto puede indicar que los piones son capaces de viajar distancias más largas dentro del detector antes de interactuar o ser detectados, en comparación con los kaones, como ya se ha comentado anteriormente.

#### 2.2.4. Variable: $hitInteg$

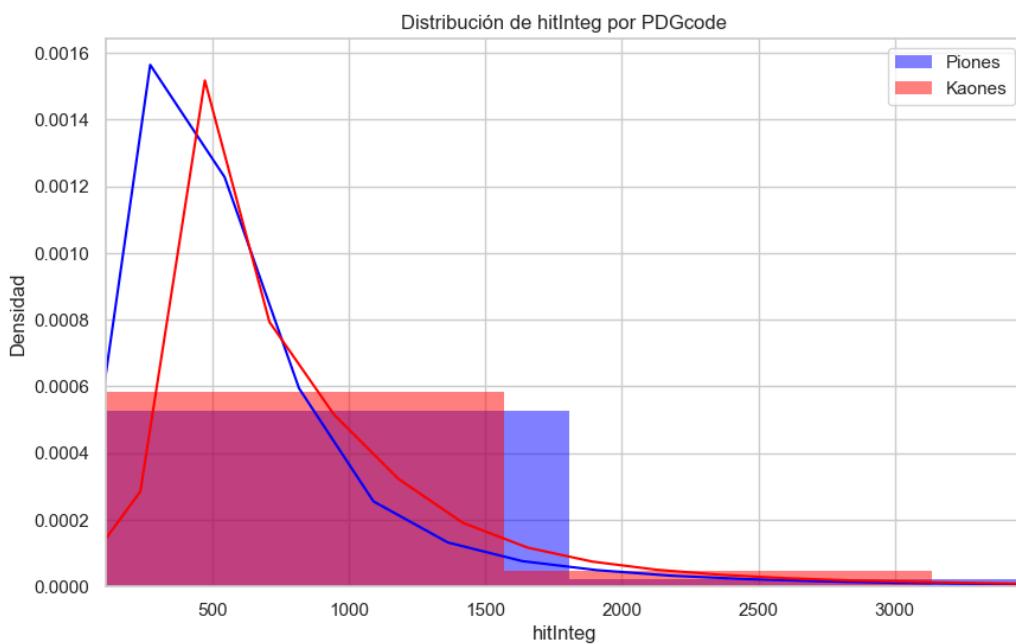


Figura 11: Gráfica distribución variable  $hitInteg$

Como podemos ver en la figura 11, la distribución de los piones muestra un pico más agudo y está centrado alrededor de un valor más bajo de  $hitInteg$  en comparación con los kaones. Esto indica que los piones, en promedio, generan hits con menor intensidad integrada. La distribución de los kaones es más ancha y el pico está desplazado hacia un valor más alto de  $hitInteg$ . Esto es esperable según la física, ya que tienen una masa mayor y su ionización es más alta en promedio.

Los piones muestran una dispersión menor y una caída más rápida de la densidad a medida que  $hitInteg$  aumenta, lo que implica que es menos probable que los piones generen hits de muy alta intensidad. Los kaones, por otro lado, tienen una cola más larga, indicando que hay una mayor probabilidad de encontrar kaones con valores más altos de  $hitInteg$ .

### 2.3. Variable: $trueE$

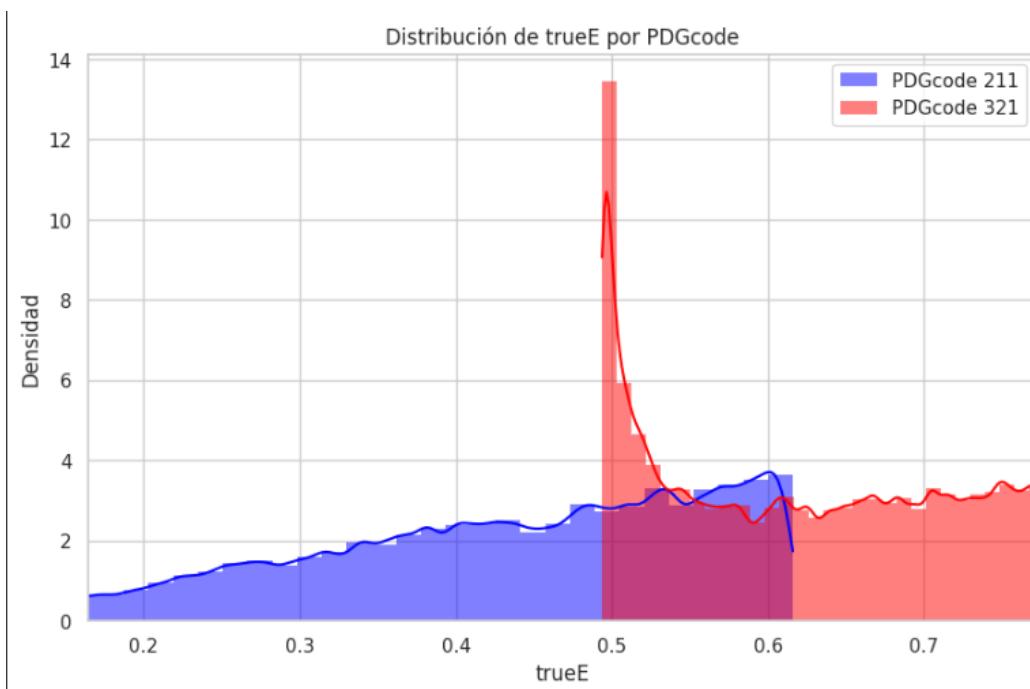


Figura 12: Gráfica distribución variable  $trueE$

Como podemos ver en la figura 12, la distribución de los piones (PDGcode 211) muestra un pico más agudo y está centrado alrededor de un valor más bajo de  $trueE$  en comparación con los kaones (PDGcode 321). Esto indica que los piones, en promedio, generan hits con menor intensidad integrada. La distribución de los kaones es más ancha y el pico está desplazado hacia un valor más alto de  $trueE$ . Esto es esperable según la física, ya que los kaones tienen una masa mayor y su ionización es más alta en promedio.

Los piones muestran una dispersión menor y una caída más rápida de la densidad a medida que `trueE` aumenta, lo que implica que es menos probable que los piones generen hits de muy alta intensidad. Los kaones, por otro lado, tienen una cola más larga, indicando que hay una mayor probabilidad de encontrar kaones con valores más altos de `trueE`.

Cabe destacar, que esta variable no será útil para el entrenamiento de los modelos de clasificación, ya que este valor no es conocido en mediciones reales.

### 2.3.1. Media, desviación, valor mínimo y máximo

Variable	Media	Desviación estándar	Valor mínimo	Valor máximo
<b>hitX</b>	-99.54213	18.04539	-201.718	183.831
<b>hitY</b>	-0.1310142	19.41624	-235.299	203.689
<b>hitZ</b>	177.6367	35.30203	-1.33675	504.1
<b>hitInteg</b>	782.3906	745.2574	1.189	55182.6
<b>trueE</b>	0.5434	0.1393	0.1395	0.7769

Cuadro 1: Descripción numérica de las características

## 3. Separación de los Datos

Para llevar a cabo nuestra fase de trabajo, se realiza una sólida separación del conjunto de datos en tres categorías distintas: entrenamiento, validación y test. Esta división del conjunto de datos se hace siguiendo una proporción, un 70 % para el conjunto de entrenamiento, un 20 % para el conjunto de validación y el 10 % restante para el conjunto de prueba [28].

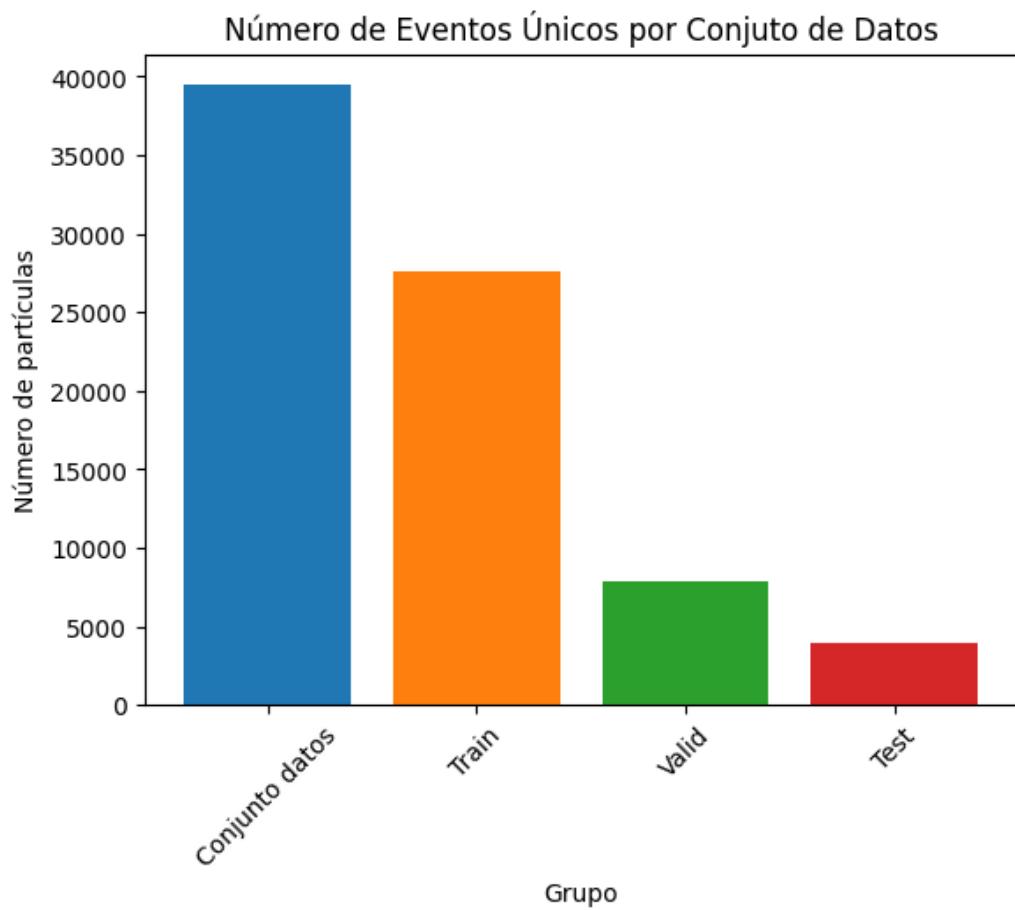


Figura 13: Separación del número de sucesos en los distintos grupos

Como premisa fundamental dentro de cada conjunto de datos, es que estén balanceadas el número de partículas por sucesos.

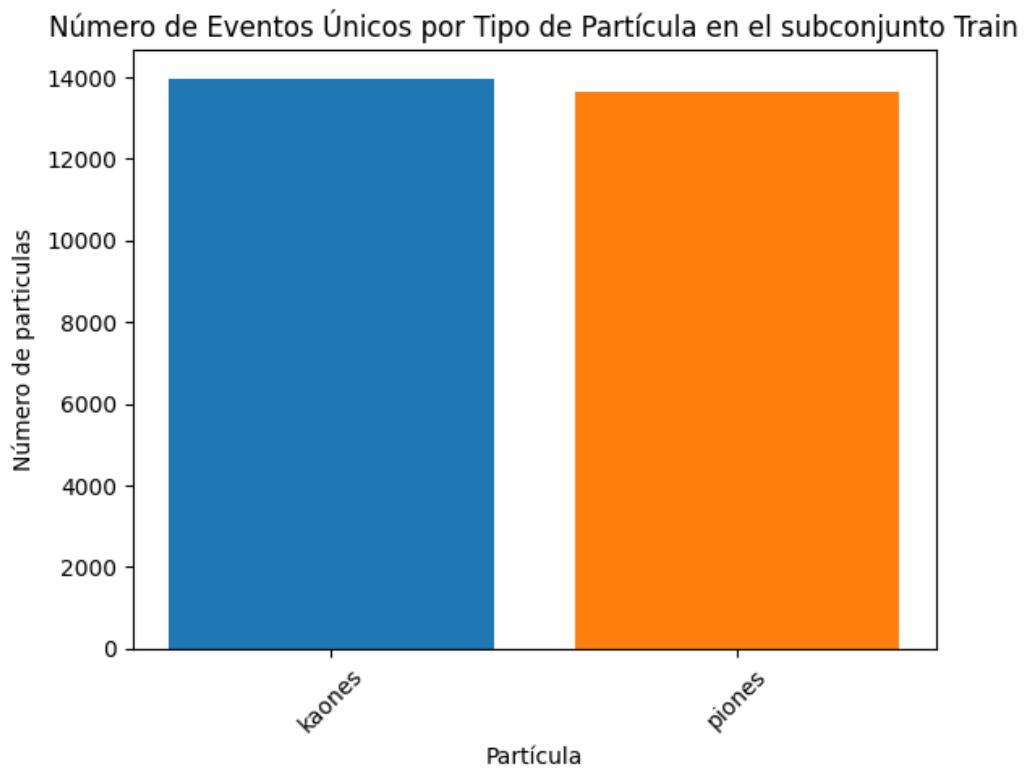


Figura 14: Número de sucesos de cada partícula en el conjunto de entrenamiento

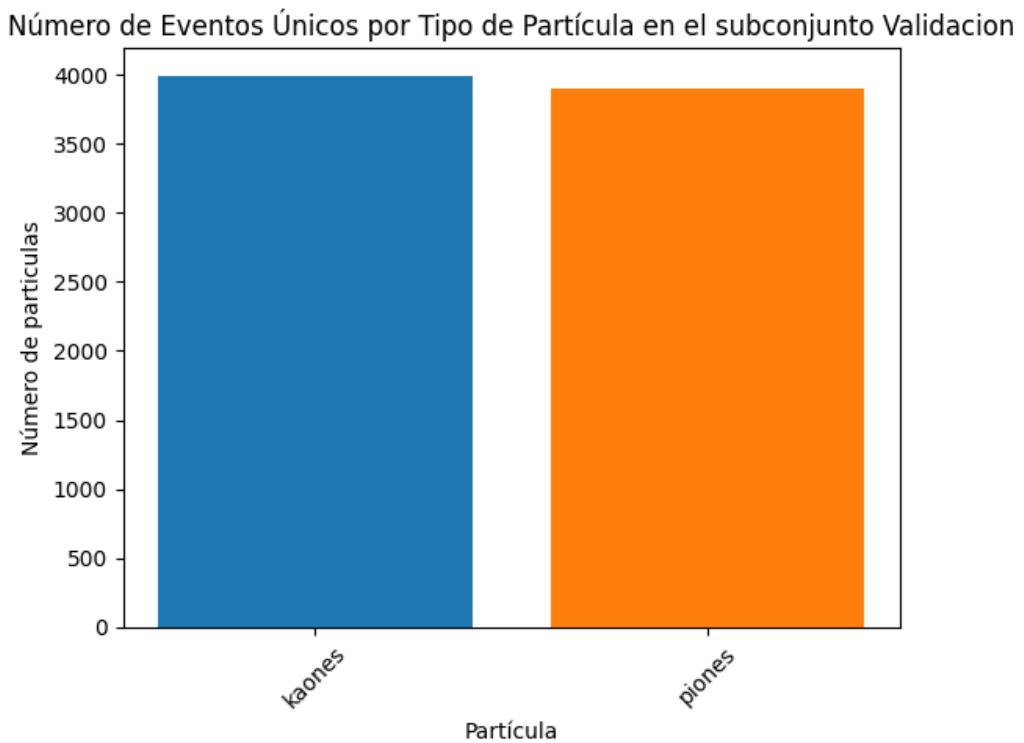


Figura 15: Número de sucesos de cada partícula en el conjunto de validación

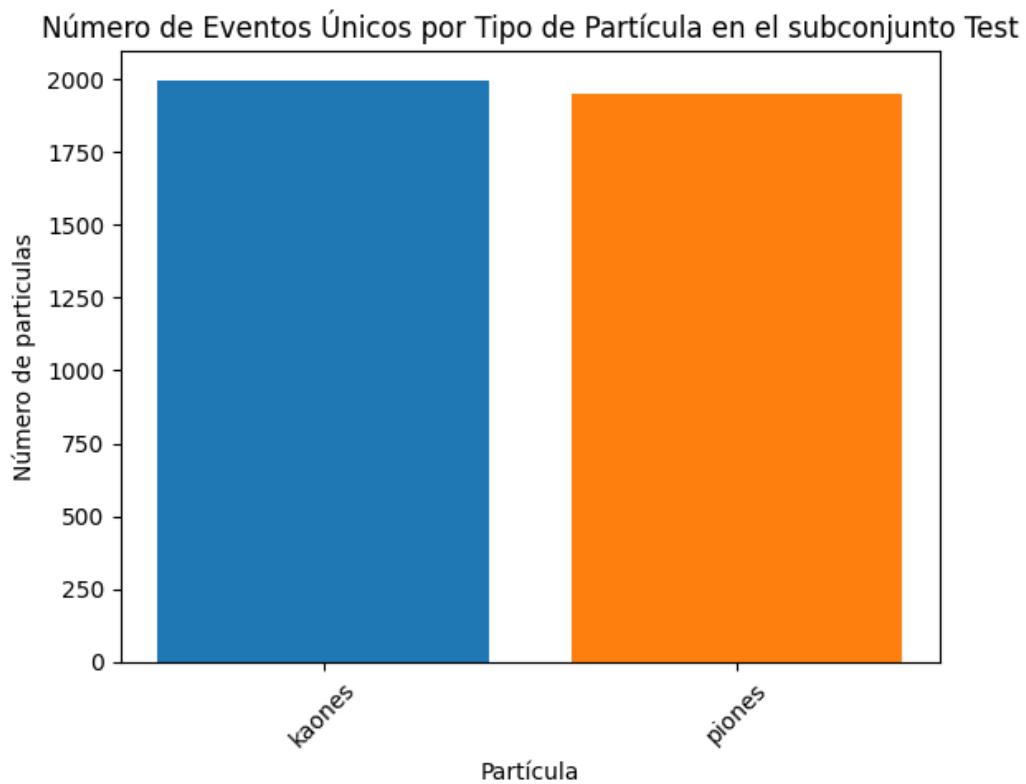


Figura 16: Número de sucesos de cada partícula en el conjunto de prueba

El conjunto de entrenamiento (figura 14) es fundamental en el desarrollo de modelos de aprendizaje automático, ya que permite a los algoritmos aprender de los patrones y las relaciones presentes en los datos. Esta fase de aprendizaje es crucial para capacitar a los modelos en decisiones automáticamente sin necesidad de una programación explícita para tareas específicas como la clasificación.

El conjunto de validación (figura 15) es esencial para la evaluación y el afinamiento del modelo previamente entrenado con el conjunto de entrenamiento. Con este conjunto, es posible ajustar y regular los parámetros del modelo, evaluando su efectividad y afinando los hiperparámetros. Este proceso facilita la compresión del aprendizaje y adaptación del modelo a los datos, permitiendo hacer mejoras significativas antes de aplicar el modelo al conjunto de prueba.

El conjunto de prueba (figura 16) representa un conjunto que no se le ha expuesto al modelo. Su función es dar una evaluación imparcial y final del modelo. Estos datos representan el mundo real y como se comportará el modelo ante datos que nunca antes ha visto, el conjunto de pruebas permite evaluar al modelo final como desempeñará su trabajo ante datos desconocidos

Conjunto de datos	Nº de sucesos	Nº de Kaones	Nº de Piones
Entrenamiento	27615	13963	13652
Validación	7891	3990	3901
Prueba	3946	1995	1951

Cuadro 2: Distribución del conjunto de datos

## 4. Métricas de error

### 4.1. Problema de clasificación

Para el problema de clasificación es natural evaluar el rendimiento de los modelos a través de distintas métricas. Las métricas de error en los modelos de aprendizaje automático son fundamentales para evaluar cómo de precisos son los modelos al hacer predicciones. Permiten determinar la efectividad del modelo al comparar las predicciones generadas con los valores reales de los datos. Mediante el uso de esta métrica podemos realizar ajustes para mejorar el modelo. Esto es esencial no sólo para optimizar el modelo, sino también para asegurar que cumpla con los estándares requeridos para su aplicación práctica, evitando problemas en decisiones basadas en sus predicciones [19].

La primera métrica se trata de una tabla de frecuencias donde las filas pertenecen a la clase predicha y las columnas a la clase real, representando el número de predicciones de cada clase mutuamente excluyentes, la tabla se conoce como Matriz de Confusión y se representa de la siguiente manera:

VP: Verdadero Positivo	FN: Falso Negativo
FP: Falso Positivo	VN: Verdadero Negativo

Cuadro 3: Matriz de Confusión [3]

- **VP:** Corresponde a los valores que son clasificados como positivos y verdaderamente pertenecen a la clase positiva
- **VN:** Representa los valores que son clasificados como negativos y verdaderamente pertenecen a la clase negativa
- **FN:** Representa los valores que son clasificados como negativos y verdaderamente pertenecen a la clase positiva
- **FP:** Representa los valores que son clasificados como positivos y verdaderamente pertenecen a la clase negativa

Para entender mejor cómo de eficaz es el modelo en las distintas predicciones, todas las matrices de confusión que se muestran en el estudio serán normalizadas. En este caso los valores se representan como porcentajes del total de casos en lugar de conteos absolutos. Esto es útil para comparar modelos sobre conjuntos de datos de diferentes tamaños o para entender mejor las proporciones de clasificación correcta o incorrecta.

Para evaluar la calidad de la clasificación se suele emplear una serie de métricas entre las que destacan las siguientes:

- **Accuracy:** Es la precisión del modelo, se puede definir como la relación entre la predicción correcta y el número total de instancias de entrada.

$$\text{Accuracy} = \frac{VP + VN}{VP + FN + FP + VN}$$

- **Recall:** La precisión se define como el número de resultados correctos, dividido por el número de aciertos de las clases predichas por el modelo de predicción.

$$\text{Recall} = \frac{VP}{VP + FN}$$

- **F1-Score:** Es la media armónica entre la precisión y el recall. Muestra la robustez del modelo de predicción.

$$F1 - Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

- **Roc Auc:** Sirve para evaluar el rendimiento de los algoritmos de clasificación binaria. El Auc representa el grado o medida de separabilidad entre clases, siendo 1.0 el valor ideal que indica una perfecta distinción, y 0.5 indica el rendimiento no mejor que el azar. Se genera calculando la tasa de verdaderos positivos contra la tasa de falsos positivos. [39]

Dentro de la física las métricas de *precisión* y *recall* se suelen denominar *pureza* y *eficiencia*, respectivamente. Tal y como se ha explicado, el *recall* es una métrica que mide la capacidad del modelo para identificar todas las instancias positivas del conjunto de datos. No obstante, una eficiencia alta puede venir acompañada de un aumento de falsos positivos. Por otro lado, la *pureza* indica la precisión de las predicciones positivas, pero un valor alto de *pureza* puede resultar en una reducción de la eficiencia. Esto se debe a que el clasificador podría volverse más cauteloso al realizar predicciones positivas, lo que podría llevar a no clasificar algunas instancias que son positivas como tal. En esencia, es común cuando se intenta mejorar una de estas métricas, la otra puede disminuir. Por eso es importante alcanzar un equilibrio adecuado entre *eficiencia* y *pureza*.

En el contexto de nuestro problema, al realizar la clasificación binaria entre las dos partículas se ha asegurado que la distribución del número de sucesos por partícula es equitativa entre los distintos conjuntos de datos: entrenamiento, validación y prueba. Dado que la penalización derivada de clasificar falsos positivos y falsos negativos es idéntica en nuestro caso, se opta por usar el índice de accuracy como la métrica principal para evaluar y comparar el rendimiento de los diferentes modelos propuestos.

## 5. Planificación del Trabajo

Para llevar a cabo esta tarea, se implementarán dos enfoques principales dentro del espectro del Aprendizaje Automático. El primero emplea técnicas avanzadas de Redes Neuronales para la clasificación de sucesos complejos, utilizando técnicas de padding para manejar sucesos de tamaño variable. Paralelamente, se explorará un enfoque de Aprendizaje Automático que agrupa 'hits' por suceso en una estructura predefinida, aunque este método puede implicar una pérdida de información cuando los sucesos varían en número de hits.

De otra forma, el segundo enfoque de Redes Neuronales experimentará con una estructura de entrada diferente, donde los datos se presentarán por sucesos completos en lugar de hits individuales. Este método enfrentará desafíos como el ajuste de tamaños variables de sucesos mediante técnicas de *padding*, para estandarizar las entradas al modelo. Se compararán ambos enfoques para determinar cuál captura más eficazmente la información relevante y se evaluarán sus resultados.

Respecto al método de clasificación mediante modelos de Aprendizaje Automático sí perderemos información ya que a los modelos hay que pasarle una estructura final de hits por sucesos. Como hemos comentado, no todos los sucesos tienen el mismo número de hits. Por tanto, habrá que elegir un N que representará el número de hits por suceso. En el caso de que el N sea mayor que el número de hits por sucesos, nos quedaremos con los todos los hits del suceso y rellenaremos con 0 todos los huecos hasta completar y si N es menor que el número de sucesos los ordenaremos por tiempo ya que nos lo indica la variable '*hitTime*' y nos quedaremos con los últimos que hayan ocurrido.

Cuando obtengamos resultados de los distintos métodos de clasificación, evaluaremos los resultados obtenidos en relación al conjunto de validación y nos quedaremos con el mejor modelo. Este modelo le aplicaremos el conjunto de test, ya que este será nuestro resultado final.

El mejor modelo de clasificación se llevará a una fase de despliegue para que pueda ser usado por cualquier usuario. Este despliegue será en la API de *FastAPI*, mediante el *framework* asociado a Python. La API se encargará de leer un csv donde se encuentran los distintos sucesos y se obtendrán los resultados de la clasificación entre los dos tipos de partículas (kaones o piones) y la energía verdadera que presenta ese suceso. Estos sucesos tendrán un preprocesado que vendrá dado por una Pipeline de la biblioteca de *sklearn* y después será predecida por el modelo.

El código fuente desarrollado en cuadernos de Jupyter, los distintos script de Python, despliegue y automatización de tareas se encuentran en mi repositorio de *GitHub* [enlace].

Toda la planificación correspondiente a la realización de este TFG, está totalmente detallada en el siguiente enlace de *Notion*[enlace].

## 6. Problema de Clasificación

### 6.1. Modelos Tradicionales de Aprendizaje Automático

#### 6.1.1. Adaptación del conjunto de datos

Como primera aproximación al problema de clasificación usaremos métodos tradicionales de Aprendizaje Automático. Para ello, tendremos que realizar una nueva estructura de los datos, ya que originalmente los datos vienen representados como en la figura 5.

Para la entrada de los distintos modelos tradicionales de Aprendizaje Automático tendremos que realizar algunos cambios. La entrada del modelo corresponde con cada una de las filas de la matriz. Por tanto, la entrada de nuestro modelo viene representada por la siguiente estructura:

EventID	hitX			hitY			hitZ			hitInteg						
	hitX0	hitX1	...	hitXN	hitY0	hitY1	...	hitYN	hitZ0	hitZ1	...	hitZN	hitInteg0	hitInteg1	...	hitIntegN
event0																
event1																
event2																
event3																
event4																

Figura 17: Estructura de datos para el problema de clasificación con modelos tradicionales de Aprendizaje Automático

Como podemos observar, entrenaremos nuestros modelos con solo 4 tipos de características, *hitX*, *hitY*, *hitZ* y *hitInteg*.

### 6.1.2. Elección del número de hits por suceso

Antes de realizar la estructura de datos anterior se debe realizar un estudio del número de hits que debemos seleccionar para entrenar el modelo. En las siguientes imágenes veremos una gráfica de frecuencia para ver el número de hits por suceso.

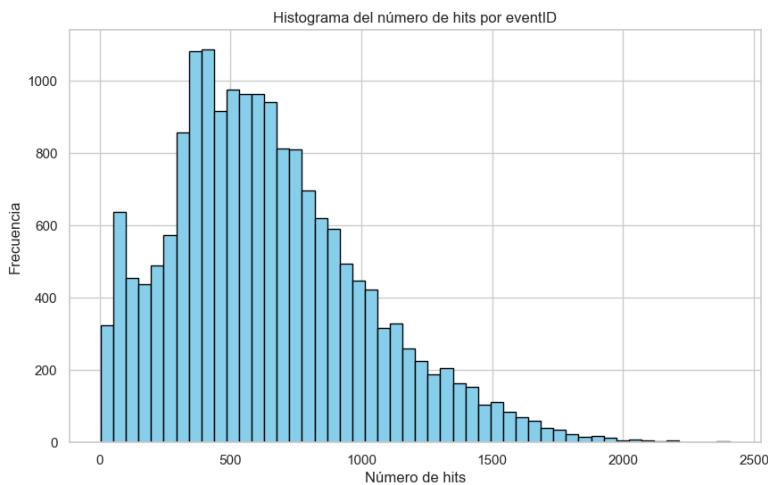


Figura 18: Distribución del número de hits por sucesos

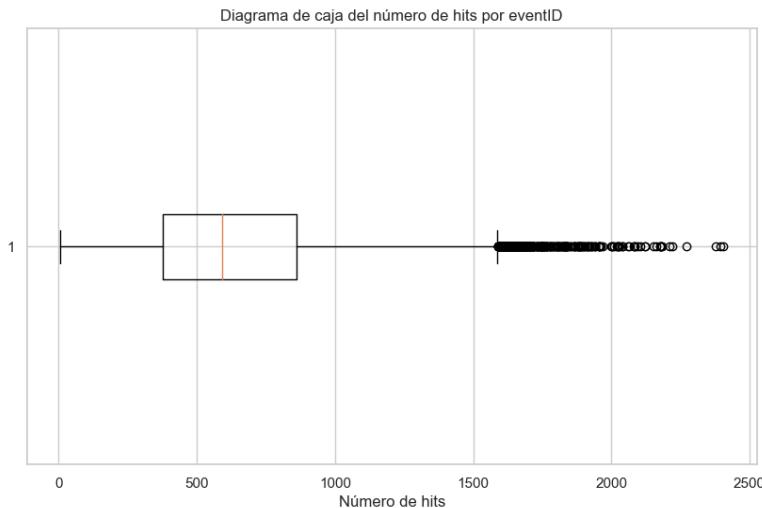


Figura 19: Diagrama de caja de la distribución del número de hits por suceso

Observando los histogramas 18, podemos ver que la mayoría de los sucesos tienen un número de hits que cae dentro de una amplia gama. El primer histograma de 18 muestra una distribución con picos aproximados entre 500 y 1000 hits por suceso. El segundo histograma 18 recalca que hay una alta concentración de sucesos en el rango menor, lo cual disminuye progresivamente después de 1000 hits.

Para un modelo de Aprendizaje Automático, seleccionar un rango de datos que sea representativo de la mayoría de los casos (sin incluir extremos que podrían sesgar el modelo) es crucial. Los histogramas 18 sugieren que el rango de 400 a 800 hits por suceso es central y muy representativo de la distribución general, abarcando la parte densa antes de que la frecuencia comience a disminuir notablemente.

El diagrama de caja 19 muestra que la mediana de los datos está cerca de los 750 hits, lo cual cae dentro del rango seleccionado. Esto es positivo ya que nos indica que la mediana de los datos no está sesgada hacia los extremos bajos o altos. La parte superior de la caja del diagrama 19 se extiende significativamente más allá del tercer cuartil, indicando la presencia de valores atípicos altos. Esto reafirma la importancia de limitar el rango superior a 800 para evitar incluir muchos de estos valores extremos en el entrenamiento del modelo.

Según todo esto se eligió seleccionar un rango de hits por sucesos entre 400 y 800 para entrenar los modelos de Aprendizaje Automático. Se está optando por un rango que evita los valores extremadamente bajos, los cuales podrían no tener suficiente información para el aprendizaje eficaz de los modelos, y también evitar valores muy altos, que podrían representar casos atípicos o ruidosos.

Este rango asegura que el modelo se entrene en un subconjunto de datos que es estadísticamente significativo y común, lo que podría mejorar la capacidad del modelo para generalizar a nuevos datos que se parecen a los más frecuentes en el conjunto de entrenamiento.

Esta elección está bien fundamentada por las distintas gráficas 18 y 19. Este rango incluye la mediana y cubre la parte más densa y central de la distribución, minimizando el impacto de los valores atípicos en el aprendizaje del modelo. Esto debería reflejar un entrenamiento más robusto y generalizable, crucial para el éxito en aplicaciones de Aprendizaje Automático.

### 6.1.3. Modelos Aprendizaje Automático Tradicionales

Ahora veremos cada uno de los distintos algoritmos de aprendizaje supervisado y no supervisado que usaremos para la clasificación entre los sucesos de kaones y piones.

- **Regresión Logística:** [15] La Regresión Logística es el conjunto de modelos estadísticos utilizados cuando se desea conocer las relaciones entre una variable dicotómica con una o más variables explicativas independientes, llamadas covariables, ya sean cualitativas o cuantitativas. Este modelo tiene tres finalidades: cuantificar la importancia de relación entre cada una de las covariables y la variable independiente, clarificar la existencia de interacción y confusión entre covariables respecto a la variable dependiente y clasificar dentro de las categorías de la variable dependiente.

- **Random Forest:** [5] Es un poderoso algoritmo de aprendizaje automático que utiliza múltiples árboles de decisión y selección aleatoria de características para mejorar la predicción y la robustez de las predicciones en problemas de clasificación y regresión, en nuestro caso lo usaremos para el problema de la clasificación. Este algoritmo combina múltiples árboles de decisión en un 'bosque' para tomar decisiones. Cada árbol en el bosque se construye utilizando un subconjunto aleatorio de características del conjunto de datos, y las decisiones se toman por la mayoría de votos de los árboles individuales
- **Gradient Boosting** [32] Construyen secuencialmente modelos débiles altamente relacionados con el gradiente negativo de la función de pérdida. Son altamente adaptables, permitiendo la elección de diversas funciones de pérdida y han demostrado éxito en una variedad de aplicaciones. En comparación con otras técnicas de ensemble, como los Random Forest, Gradient Boosting utiliza una estrategia constructiva de formación de conjunto, agregando modelos secuencialmente para mejorar la precisión predictiva.
- **SVM:** [22] Son algoritmos que clasifican datos encontrando una línea o hiperplano óptimo que maximiza la distancia entre cada clase en un espacio N-dimensional. Se usa en problemas de clasificación, comúnmente, diferenciando entre dos clases al encontrar el hiperplano óptimo que maximiza el margen entre los puntos de datos más cercanos de clases opuestas. Estos algoritmos pueden manejar tanto tareas de clasificación lineal como no lineal, utilizando funciones kernel para transformar los datos a un espacio dimensional superior cuando no son separables linealmente. La elección de la función kernel depende de las características de los datos y el uso específico.
- **Decision Tree:** [37] Un árbol de decisión es un clasificador que divide recursivamente el espacio de instancias en subespacios a través de pruebas sobre los valores de atributos. Los nodos representan pruebas sobre atributos, mientras que las hojas representan las clases objetivo. Cada instancia se clasifica navegando desde la raíz del árbol hasta una hoja, siguiendo el

resultado de las pruebas a lo largo del camino. Los árboles de decisión pueden interpretarse geométricamente como una colección de hiperplanos de árbol se controla mediante criterios de parada y métodos de poda, y se mide por el número total de nodos, profundidad del árbol y número de atributos utilizados. Además, los árboles de decisión pueden transformarse en reglas de manera sencilla, lo que permite su interpretación y simplificación para mejorar su comprensión y precisión.

- **K-NN:** [20] Este algoritmo es un clasificador de aprendizaje no supervisado que utiliza la proximidad para realizar clasificaciones o predicciones sobre datos individuales. En problemas de clasificación, asigna una etiqueta de clase basada en un voto mayoritario entre los  $k$  vecinos más cercanos. Aunque es simple y preciso, su eficiencia disminuye en conjuntos de datos grandes. Se utiliza en sistemas de recomendación, reconocimiento de patrones, entre otros.
- **XGBoost:** [12] Es un método de Aprendizaje Automático supervisado para la clasificación y regresión que mejora sobre otros algoritmos como Random Forest y Gradient Boosting. Utiliza árboles de decisión y varios métodos de optimización para trabajar con conjuntos de datos grandes y complejos. Este algoritmo ajusta el dataset de entrenamiento mediante la construcción de árboles de decisión basados en residuos, utilizando una puntuación de similitud y regularización para evitar sobreajuste. Además, implementa técnicas como la poda de ramas innecesarias, el boceto de cuantil ponderado y el aprendizaje paralelo para mejorar la eficiencia y el rendimiento.
- **LightGBM:** [24] Utiliza la técnica de Gradient Boosting. Con este método los árboles se construyen de manera secuencial y cada uno que se agrega aporta su granito de arena para refinar la predicción anterior. Es decir, se comienza con un valor constante y cada árbol nuevo se entrena para predecir el error en la suma de todas las predicciones de los árboles anteriores. Una vez terminado el proceso, las predicciones se calculan sumando los resultados de todos los árboles que se construyeron. El efecto que tiene esto es que cada vez que se agrega un árbol nuevo se le presta atención a las muestras

en las que el modelo está funcionando peor y se trabaja para mejorar ese aspecto.

Es importante tener en cuenta que todos los algoritmos tienen hiperparámetros que influyen directamente en los resultados obtenidos. Por lo tanto, no solo es crucial elegir el algoritmo más adecuado, sino también configurarlo correctamente. Además, puede ser necesario ajustar el valor N para cada algoritmo. El objetivo principal es encontrar la combinación entre el algoritmo utilizando sus hiperparámetros y el valor de N, con el fin de obtener resultados posibles.

A continuación veremos los parámetros elegidos para cada algoritmo:

Algoritmo	Parámetros
Regresión logística	penalty: l2, solver: lbfgs, random_state: 42, max_iter: 500, multiclass: ovr
Random Forest	n_estimators: 50, criterion: gini, max_depth: None, max_features: sqrt, random_state: 42
Gradient Boosting	loss: log_loss, n_estimators: 100, criterion: friedman_mse, max_features: sqrt, random_state: 42, learning_rate: 0.1
SVM	kernel: rbf, gamma: scale, random_state: 42, probability: True
Decision Tree	criterion: gini, splitter: best, random_state: 42
K-NN	n_neighbors: 5, weights: uniform, metric: minkowski
XGB	booster: gbtree, max_depth: 6, gamma: 0, learning_rate: 0.3, subsample: 1, colsample_bytree: 1, objective: binary: logistic, eval_metric: logloss, seed: 42
LGBM	boosting_type: gbdt, max_depth: 6, learning_rate: 0.3, subsample: 1, colsample_bytree: 1, random_state: 42

Cuadro 4: Parámetros de los modelos de Aprendizaje Automático

## 6.2. Elección de los hiperparámetros para los distintos modelos

Ahora veremos la justificación de la elección de los parámetros en los modelos de aprendizaje automático basándonos en los fundamentos teóricos y prácticos:

### 1. Regresión logística:

- **penalty: l2:** Se utiliza la regularización L2 (también conocida como ridge) para evitar el sobreajuste, ya que penaliza los coeficientes de características de gran magnitud.
- **solver: lbfgs:** Es un algoritmo de optimización eficiente que funciona bien con la regularización L2.
- **random\_state: 42:** Fijar la semilla del generador de números aleatorios asegura que los resultados sean reproducibles.
- **max\_iter: 500:** Aumentar el número máximo de iteraciones garantiza que el algoritmo tenga suficiente tiempo para converger, especialmente en casos en que el problema es complejo.
- **multiclasse: ovr:** Es una estrategia estándar para extender la regresión logística a problemas de clasificación multiclasse.

### 2. Random Forest:

- **n\_estimators: 50:** Utilizar 50 árboles suele ser un buen punto de partida para obtener un equilibrio entre rendimiento y tiempo de computación.
- **criterion: gini:** El índice Gini es una medida estándar de impureza para la clasificación.
- **max\_depth: None:** Permitir que los árboles crezcan completamente ayuda a capturar todas las relaciones posibles en los datos.
- **max\_features: sqrt:** Utilizar la raíz cuadrada del número total de características es una práctica común que mejora la diversidad de los árboles y reduce el sobreajuste.
- **random\_state: 42:** Fijar la semilla para la reproducibilidad de los resultados.

### 3. Gradient Boosting:

- **loss: log\_loss:** La pérdida logarítmica es adecuada para problemas de clasificación.
- **n\_estimators: 100:** Utilizar 100 árboles es un buen equilibrio entre rendimiento y tiempo de entrenamiento.
- **criterion: friedman\_mse:** Este criterio mejora la precisión del modelo al reducir la varianza.
- **max\_features: sqrt:** Ayuda a evitar el sobreajuste y mejorar la generalización.
- **random\_state: 42:** Fijar la semilla para la reproducibilidad.
- **learning\_rate: 0.1:** Un learning rate bajo mejora la estabilidad y la capacidad de los modelos para generalizar, permitiendo que cada árbol contribuya de manera gradual.

### 4. SVM:

- **kernel: rbf:** El kernel de función base radial (*RBF*) es efectivo para problemas de clasificación no lineales.
- **gamma: scale:** Escalar la gamma automáticamente en función del número de características asegura una selección apropiada de la gamma.
- **probability: True:** Permite calcular las probabilidades de las predicciones, lo cual es útil para ciertas aplicaciones como la evaluación de la incertidumbre.

### 5. Decision Tree:

- **Criterion: gini:** El índice de Gini es una medida estándar de impureza utilizada en clasificación.
- **splitter: best:** Selecciona la mejor división en cada nodo, optimizando el criterio de división.
- **random\_state: 42:** Asegura la reproducibilidad de los resultados.

## 6. K-NN:

- **n\_neighbors: 5:** Utilizar 5 vecinos es un punto de partida que balancea la complejidad y el riesgo de sobreajuste.
- **weights: uniform:** Asigna el mismo peso a todos los vecinos, lo cual simplifica el modelo
- **rmetric: minkowski:** Generaliza varias métricas de distancia, donde con  $p = 2$  se convierte en la distancia euclídea, una elección común para K-NN.

## 7. XGBoost:

- **booster: gbtree:** Utiliza árboles de decisión como base, lo cual es efectivo para la mayoría de problemas de clasificación y regresión.
- **max\_depth:6:** Limitar la profundidad del árbol ayuda a controlar el sobreajuste al impedir que los árboles se vuelvan demasiado complejos.
- **gamma: 0:** No penaliza la partición de los nodos, permitiendo una mayor flexibilidad en la construcción de los árboles.
- **learning\_rate: 0.3:** Un learning rate relativamente alto acelera el proceso de entrenamiento, pero puede requerir una validación cuidadosa para evitar el sobreajuste.
- **subsample: 1:** Utiliza todas las muestras en cada iteración, asegurando que el modelo tenga a todos los datos disponibles.
- **colsample\_bytree: 1:** Utiliza todas las características en cada iteración, asegurando que no se omita ninguna característica relevante.
- **objective: binary: logistic:** Específico para problemas de clasificación binaria, modelando la probabilidad de pertenecer a una clase.
- **eval\_metric: logloss:** La pérdida logarítmica es adecuada para medir la precisión de las predicciones de probabilidad.
- **random\_state: 42:** Asegura la reproducibilidad de los resultados.

## 8. LightGBM:

- **boosting\_type: gbdt:** El método tradicional de boosting basado en árboles es efectivo y ampliamente utilizado.
- **max\_depth: 6:** Limitar la profundidad del árbol ayuda a controlar el sobreajuste y a mantener la simplidad del modelo.
- **learning\_rate: 0.3:** Un learning rate relativamente alto permite un entrenamiento más rápido, aunque puede requerir ajustes para evitar el sobreajuste.
- **subsample: 1:** Utiliza todas las muestras en cada iteración, asegurando un uso completo de los datos disponibles.
- **solsample\_bytree: 1:** Utiliza todas las características en cada iteración, asegurando un análisis exhaustivo de todas las variables.
- **random\_state: 42:** Asegura la reproducibilidad de los resultados

### 6.2.1. Entrenamiento

Antes de entrenar los distintos modelos, hay que llevar a cabo la fase de procesamiento de datos para formar la estructura que hemos visto en la figura 17. Para ello definiremos una función, llamada *filtrado\_datos* que seguirá la siguiente política:

1. **Ordenaremos el conjunto de datos:** Los datos se ordenan primero por *eventID* en orden ascendente y luego por *hitTime* en orden descendente. Este paso es crucial para asegurar que los datos estén agrupados y que los impactos más recientes se prioricen en el procesamiento posterior.
2. **Agrupación por suceso:** Se agrupan los datos ordenados por *eventID*, lo que permite manejar los datos de cada suceso individualmente en los pasos siguientes.
3. **Filtrado por selección de *PDGcode*:** Dentro de cada grupo de sucesos, se identifican los *PDGcode* únicos presentes. Para cada *PDGcode*, se filtran

los datos correspondientes y se seleccionan los primeros N registros, donde N es un parámetros de entrada.

4. **Aplicación de Padding:** Dado que diferentes sucesos pueden tener un número variable de impactos, se aplica un padding para estandarizar la longitud de los vectores de características. Se crea un array de ceros para cada característica y se llenan con los valores reales hasta completar N elementos. Esto asegura que todos los vectores tengan la misma longitud, facilitando el análisis comparativo posterior.
5. **Reorganización y almacenamiento de datos:** Los datos procesados de cada grupo se concatenan en un solo vector y se almacenan en una lista. Este paso finaliza la preparación de los datos que caracterizan cada impacto por tipo de partícula y suceso.
6. **Etiquetado de datos:** Paralelamente, se asigna una etiqueta numérica a cada tipo de partícula basado en el código de la partícula, transformando el código de partículas a una forma más simplificada para análisis binario o clasificadorio.

La función devuelve una matriz con los datos para el entrenamiento y un array con las etiquetas.

Dado que los datos provienen de una simulación y están libres de errores o valores anómalos, se introducirán directamente en el modelo sin requerir un preprocesamiento adicional. Durante el análisis de los valores óptimos para el parámetro N se determinó que el rango apropiado es entre 400 y 800 hits por suceso, con un incremento de 40 en 40 unidades. Por lo tanto, se realizaron un total de 11 entrenamientos para cada modelo, abarcando los distintos valores de N en este intervalo.

### 6.2.2. Resultados del entrenamiento

La comparativa entre los resultados de los distintos modelos en términos de accuracy es la siguiente:

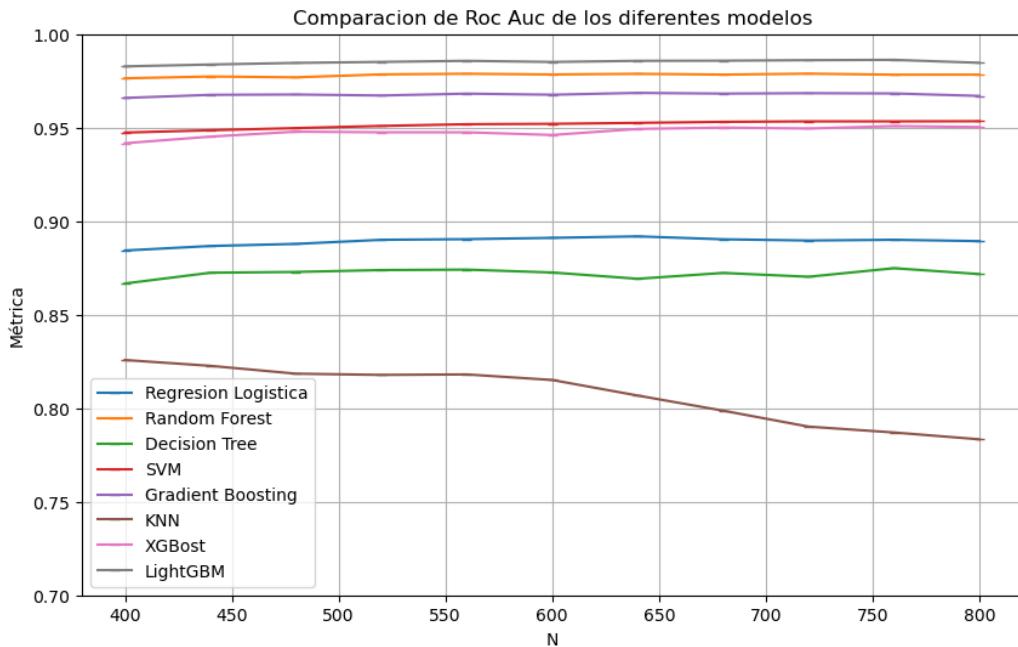


Figura 20: Comparativa de accuracy de todos los modelos tradicionales de Aprendizaje Automático

También veremos una tabla resumen de la métrica de accuracy de los distintos modelos y 4 mejores valores en cuanto a accuracy:

Modelo	Max Accuracy	N Max	Min Accuracy	N Min	Diferencia
Regresión Logística	0.8422	640	0.8322	400	0.0100
Random Forest	<b>0.9361</b>	560	0.9305	480	0.0055
Decision Tree	0.8749	560	0.9305	480	0.0082
SVM	0.8938	760	0.8666	400	0.0082
Gradient Boosting	<b>0.9251</b>	800	0.8807	400	0.0130
K-NN	0.7228	400	0.6574	800	0.0653
XGBoost	<b>0.9508</b>	760	0.9417	400	0.0091
LightGBM	<b>0.9510</b>	720	0.9414	400	0.0096

Cuadro 5: Tabla resumen entrenamiento modelos de Aprendizaje Automático

Otro factor importante a la hora de elegir un buen modelo, es el tiempo de entrenamiento. Los modelos que son muy complejos pueden ofrecer un mejor rendimiento en términos de precisión, pero también más tiempo para entrenarse,

lo que puede ser un problema en aplicaciones que necesitan respuestas en tiempo real o cuando se trabaja con una gran cantidad de datos. Por lo tanto, es esencial equilibrar la precisión con la eficiencia operativa. Así, la elección de un algoritmo debe estar guiada no solo por su capacidad predictiva, sino también por su viabilidad práctica en el entorno de uso previsto. A continuación veremos una imagen donde mostraremos los tiempos de entrenamiento en función de la N:

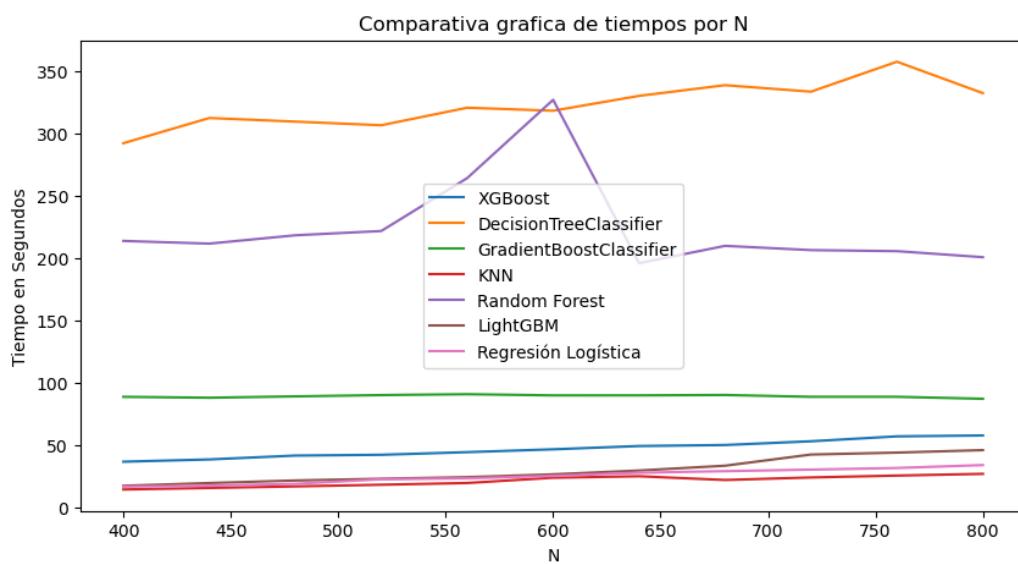


Figura 21: Comparativa de los tiempos de entrenamiento de los modelos de Aprendizaje Automático

A continuación veremos una matriz de confusión de los mejores 4 modelos:

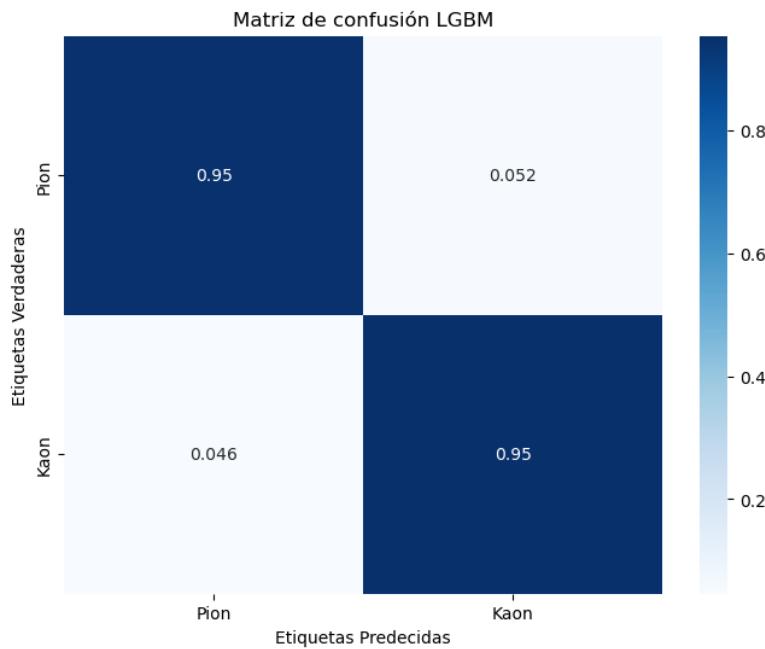


Figura 22: Matriz de confusión del entrenamiento de LightGBM

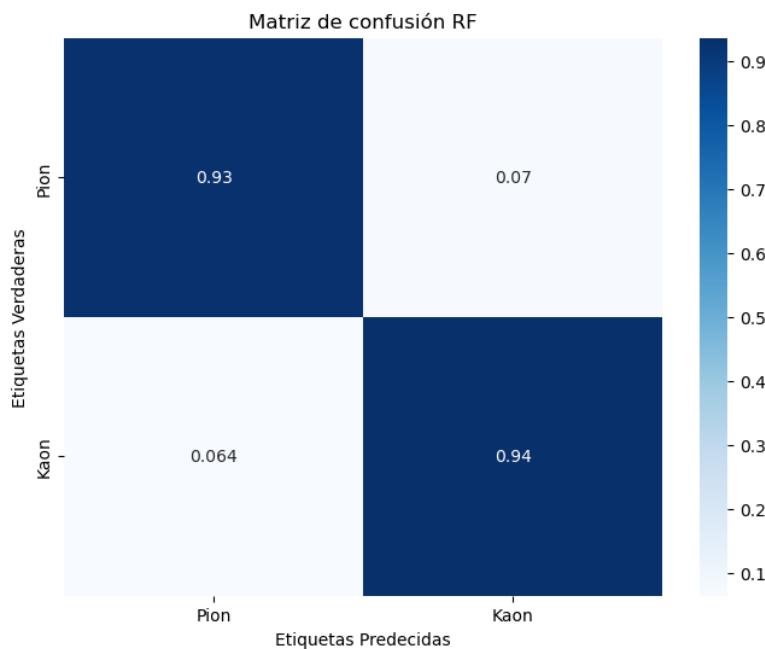


Figura 23: Matriz de confusión del entrenamiento de Random Forest

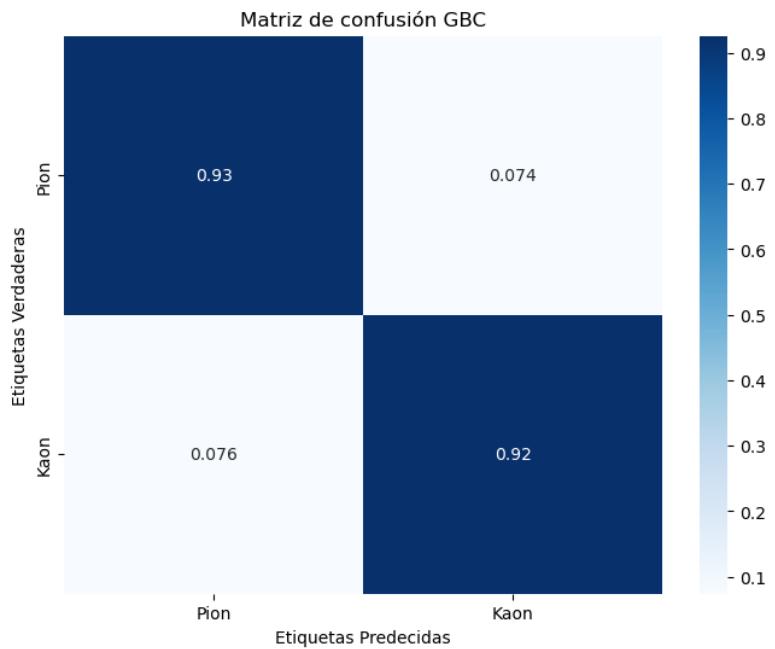


Figura 24: Matriz de confusión del entrenamiento de Gradient Boosting

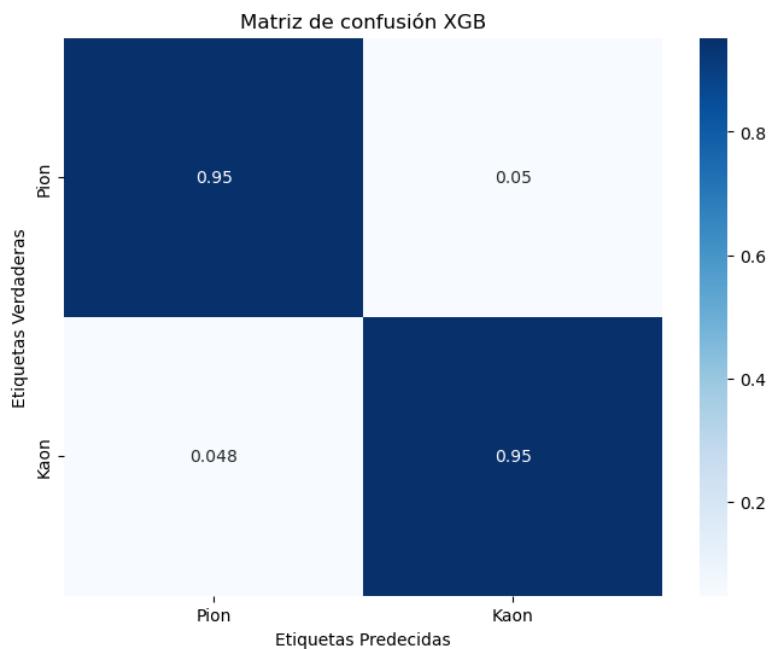


Figura 25: Matriz de confusión del entrenamiento de XGBoost

Tanto LightGBM como XGBoost destacan por su alta precisión y bajo error, haciendo que ambos modelos sean altamente efectivos para este tipo de clasificación. Random Forest y Gradient Boosting muestran un rendimiento ligeramente inferior en comparación con LightGBM y XGBoost, especialmente en la tasa de errores.

Es cierto que los 4 modelos han mostrado un rendimiento competente, pero LightGBM y XGBoost sobresalen particularmente en términos de precisión y minimización de errores. Estos resultados sugieren que tanto LightGBM como XGBoost son opciones robustas para aplicaciones que requieren alta precisión en la clasificación de partículas.

A continuación cogeremos los 4 mejores modelos y veremos cual es la mejor  $n$  para cada modelo, para posteriormente realizar un entrenamiento mediante Validación Cruzada. Los mejores modelos con su respectiva  $N$  son:

XGBoost	$N = 760$
Gradient Boosting	$N = 760$
Random Forest	$N = 400$
LightGBM	$N = 720$

Cuadro 6: Mejor  $N$  para cada modelo

### 6.2.3. Entrenamiento por Validación Cruzada

La Validación Cruzada [35] es un método estadístico para evaluar y comparar algoritmos de aprendizaje al dividir los datos en dos segmentos: uno para entrenar o aprender un modelo y otro para validar el modelo. En la forma básica de la validación cruzada, los conjuntos de entrenamiento y validación se superponen en rondas sucesivas para que cada punto de datos tenga la oportunidad de ser validado. En la validación cruzada  $k$ -fold, los datos se dividen en  $k$  segmentos de igual tamaño y realizan  $k$  iteraciones de entrenamiento y validación, utilizando un segmento diferente para la validación en cada interacción, mientras que los demás segmentos se utilizan para la validación en cada iteración, mientras que los demás segmentos se utilizan para el aprendizaje. Se utiliza para evaluar o compa-

rar algoritmos de aprendizaje mediante la realización de múltiples iteraciones y el seguimiento del rendimiento de cada algoritmo en cada iteración utilizando una métrica de rendimiento predeterminada, en nuestro caso será el accuracy. Los resultados de estas iteraciones se pueden utilizar para obtener una medida agregada o realizar pruebas de hipótesis estadísticas para demostrar la superioridad de un algoritmo sobre otro.

Para realizar nuestra etapa de entrenamiento por Validación Cruzada, uniremos nuestros datos de entrenamiento y validación, para que el modelo tenga un mayor número de datos para entrenar y validar. Estos son los resultados en términos de Accuracy:

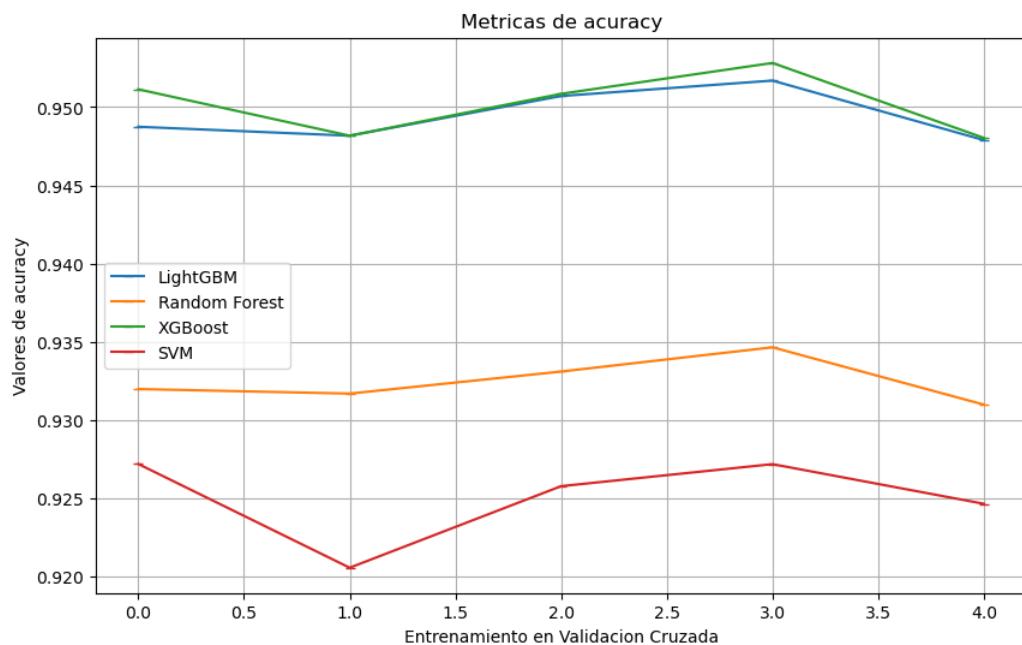


Figura 26: Comparativa de accuracy en el entrenamiento de los modelos de Aprendizaje Automático por Validación Cruzada

Estos son los resultados numéricos en la tabla resumen:

Modelo	Iteración 1	Iteración 2	Iteración 3	Iteración 4	Iteración 5
Random Forest	0.9319	0.9316	0.9331	0.9346	0.9309
Gradient Boosting	0.9272	0.9205	0.9257	0.9271	0.9246
LightGBM	0.9487	0.9481	0.9507	0.9516	0.9478
XGBoost	0.9511	0.9481	0.9508	<b>0.9528</b>	0.9480

Cuadro 7: Resultados en términos de accuracy del entrenamiento por Validación Cruzada

Según la figura 26 y que representa la comparativa entre las distintas iteraciones de cada modelo, podemos ver que los 4 modelos son bastante estables en los resultados. Podemos observar como el moldeo XGBoost presenta un pico en cuanto a la métrica de accuracy. Por tanto, XGBoost será el modelo que usaremos para llevarlo a producción y someterlo a una etapa de explicabilidad.

Como el modelo XGBoost es el modelo que mejores resultados nos ha dado en el estudio, será importante ver una gráfica de la función de pérdida durante el entrenamiento.

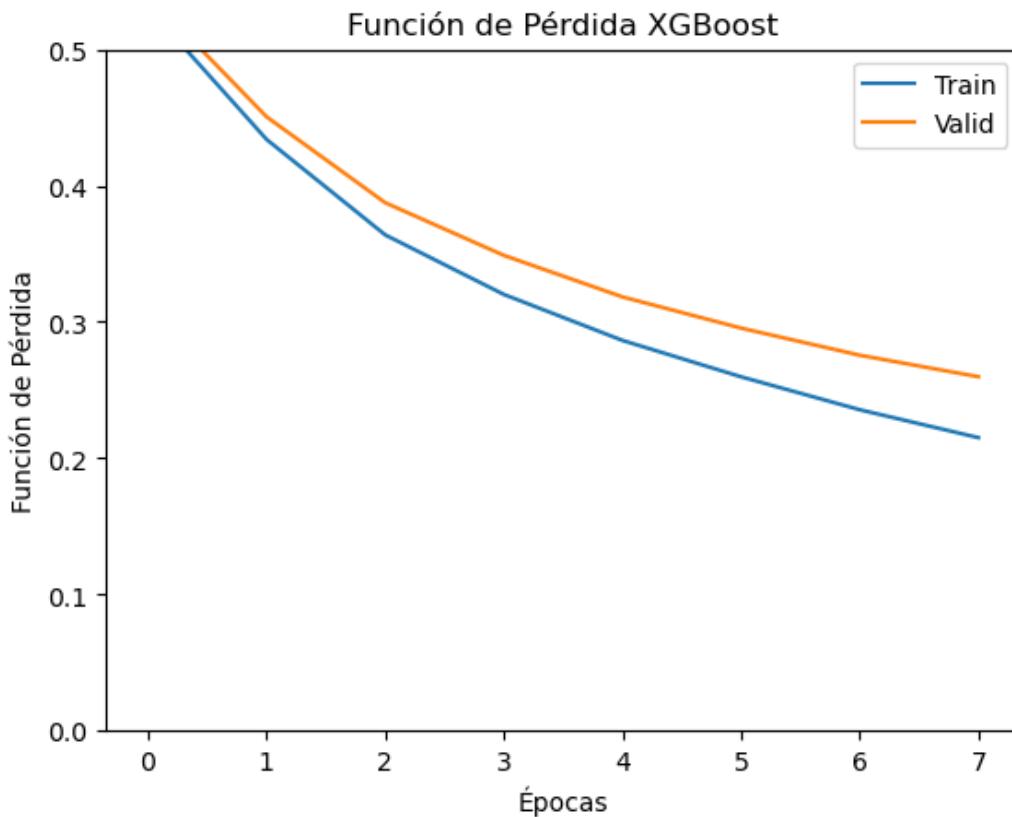


Figura 27: Función de pérdida del entrenamiento de XGBoost

La imagen nos muestra el desempeño del modelo XGBoost durante el proceso de entrenamiento, representado por la función de pérdida logarítmica para los conjuntos de entrenamiento y validación. En la gráfica, ambas líneas tienen una tendencia descendente, lo cual sugiere que la pérdida logarítmica está disminuyendo a medida que el modelo procesa más épocas. Esto indica que el modelo está aprendiendo de forma efectiva. La separación de las líneas muestra que el modelo está generalizando bien. No hay una gran brecha que podría indicar que el modelo se comporta de manera muy eficiente en los datos no vistos en comparación con los datos de entrenamiento, esto es indicativo de que el modelo no está sobreajustando.

En la fase de explicabilidad del modelo, es crucial examinar cómo este clasifi-

ca las observaciones en distintos rangos de energía total de entrada de la partícula, identificados por la variable *trueE*. Es relevante mencionar que *trueE* es constante para todos los hits dentro de un mismo suceso. Por consiguiente, adaptaremos la función de filtrado de datos para que incluyan como nuevo output, la energía de entrada correspondiente a cada suceso, a esta función modificada la denominaremos *filtrado\_datos\_energía*.

Para categorizar los datos en tres niveles de energía, entre baja, media y alta, es necesario ordenar los valores de *trueE* y segmentar este conjunto en tres grupos iguales. De acuerdo con esta metodología, los umbrales que definen los cortes entre los rangos de baja y media energía, y entre media y alta energía, son respectivamente 0.42793 GeV y 0.559643 GeV.

Según esto, es importante agregar que antes de mostrar los resultados de prueba, se ha realizado un entrenamiento con todos los datos del conjunto de entrenamiento y validación, para poder entrenar el modelo con más datos y que este generalice de mejor forma. Los resultados han sido los siguientes:

Conjunto	Accuracy	Accuracy baja	Accuracy media	Accuracy alta
Entrenamiento	0.94117	0.95888	0.93613	0.92846
Validación	0.92141	0.94630	0.92154	0.89655
Prueba	0.91763	0.93768	0.92155	0.89369

Cuadro 8: Resumen resultados Accuracy

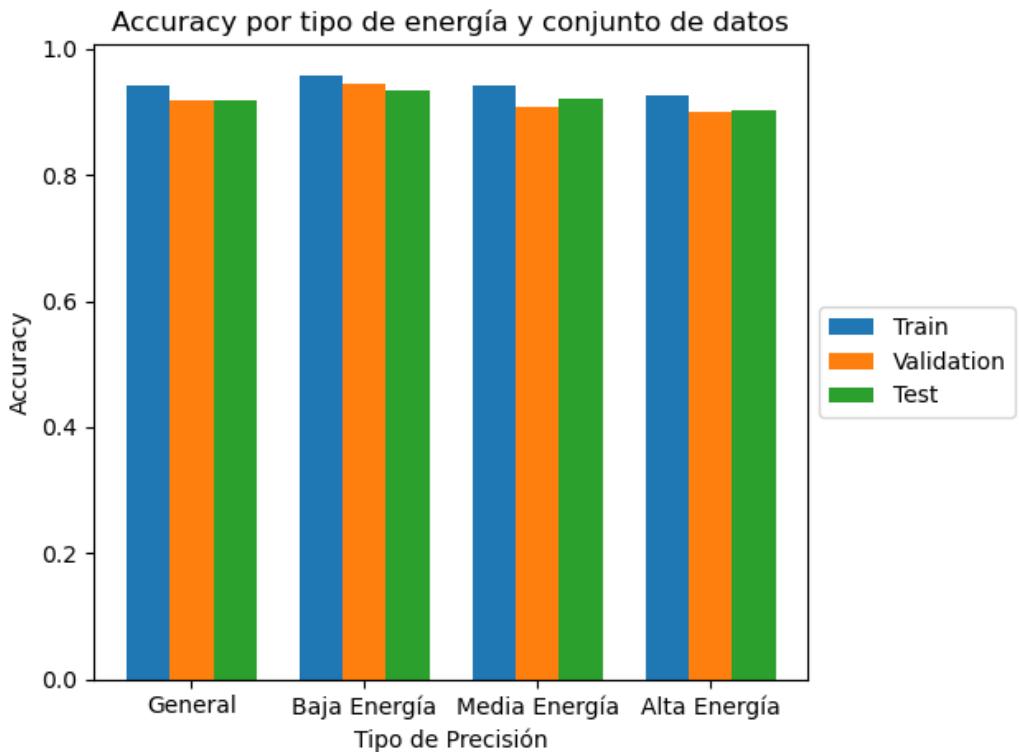


Figura 28: Resultados en términos de Accuracy de todos los conjuntos de entrenamiento

Como podemos observar en los resultados de los entrenamientos, vemos una consistencia notable en la precisión entre los conjuntos de entrenamiento, validación y prueba a través de los tipos de energía. Esto indica que el modelo generaliza bien y no presenta un sobreajuste al conjunto de entrenamiento. La capacidad del modelo para generalizar los datos no vistos para mantener niveles comparables de precisión en datos no vistos es un indicativo de su robustez.

Los resultados por rangos de energía muestran una consistencia notable. Esto es positivo porque indica que el modelo maneja uniformemente bien los distintos niveles de energía, sin mostrar una preferencia o un rendimiento deficiente en ninguno en particular, también hay que añadir que para altas energías es un poco más baja que para el resto. Esto es debido a que hemos truncado el numero de hits por evento, debido a esto a alta energía tiene una mayor transcendencia.

La alta precisión en los datos de entrenamiento y la capacidad de mantener una precisión similar en los conjuntos de validación y prueba es un indicativo fuerte de que el modelo no está sobreajustado. Esto es crucial para garantizar que el modelo será efectivo en entornos reales y no solo en condiciones controladas de entrenamiento.

### 6.3. Modelos de Redes Neuronales

En la siguiente fase de nuestro proyecto, implementaremos una arquitectura basada en redes neuronales para abordar la clasificación dentro de nuestro conjunto de datos. Además de alcanzar el objetivo principal de clasificación, es crucial minimizar la pérdida de información durante la transformación de la estructura de datos original. La estructura de datos se ilustra en la figura mencionada previamente 5. Para conservar la máxima información posible, optamos por seleccionar el suceso con el mayor número de hits en el conjunto de entrenamiento. Este criterio garantiza que, si en futuros conjuntos de datos se identifica un suceso con una cantidad superior de hits, se priorizan aquellos que ocurrieron más recientemente, es decir, aquellos con un valor de *hitTime* más alto. Para los de menor tamaño, emplearemos una técnica de relleno o ‘padding’ para estandarizar las dimensiones de entrada de la red.

Este enfoque no solo optimiza la precisión de la clasificación al reducir la pérdida de datos críticos, sino que también estandariza la entrada para la arquitectura de red neuronal, facilitando así el procesamiento y mejorando potencialmente la eficacia del modelo en tareas de aprendizaje profundo.

#### 6.3.1. Adaptación del conjunto de datos para el entrenamiento

El conjunto de datos de entrenamiento, antes de transformarlo tiene la siguiente forma:

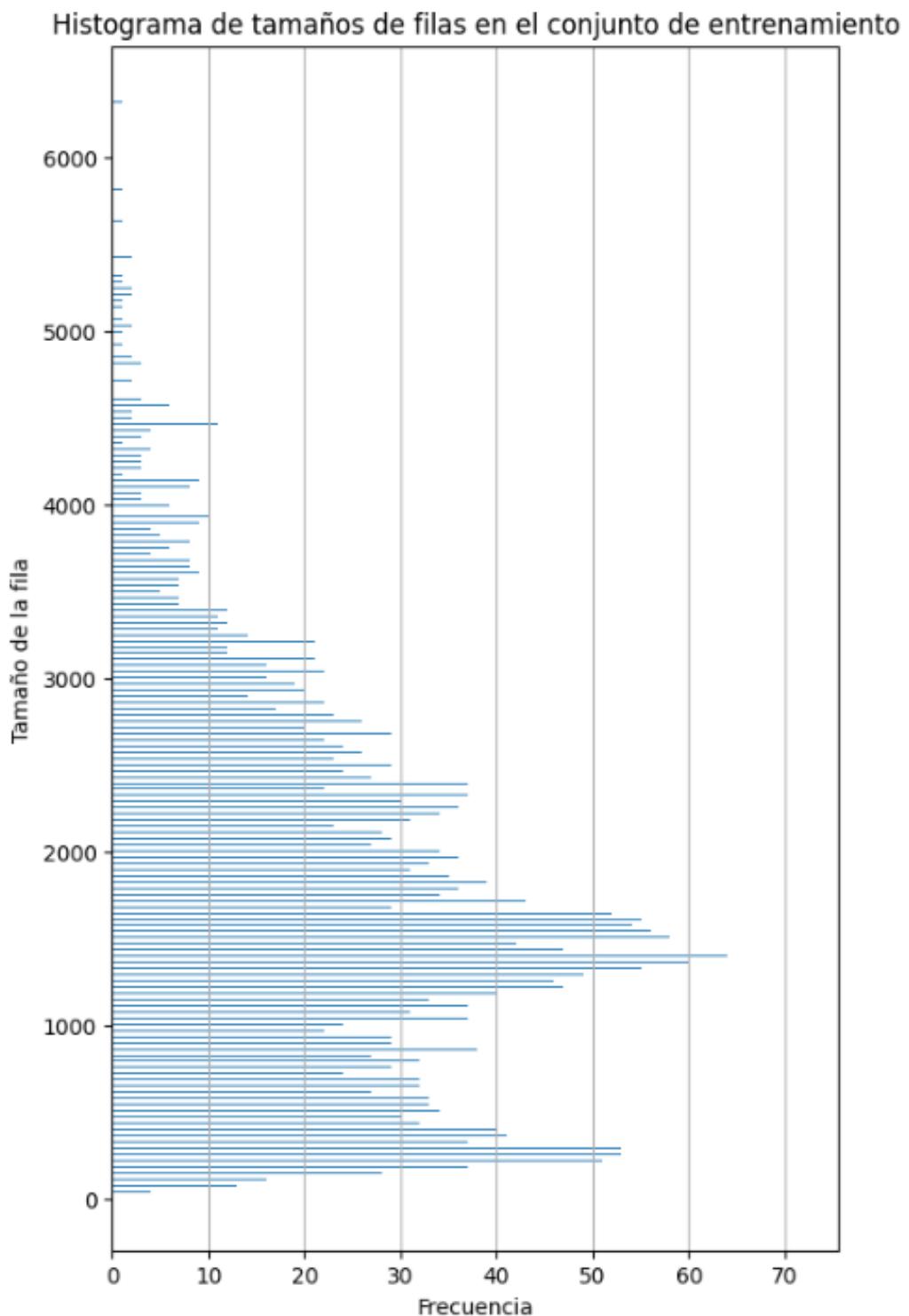


Figura 29: Frecuencia de la longitud de las cadenas de hits por sucesos

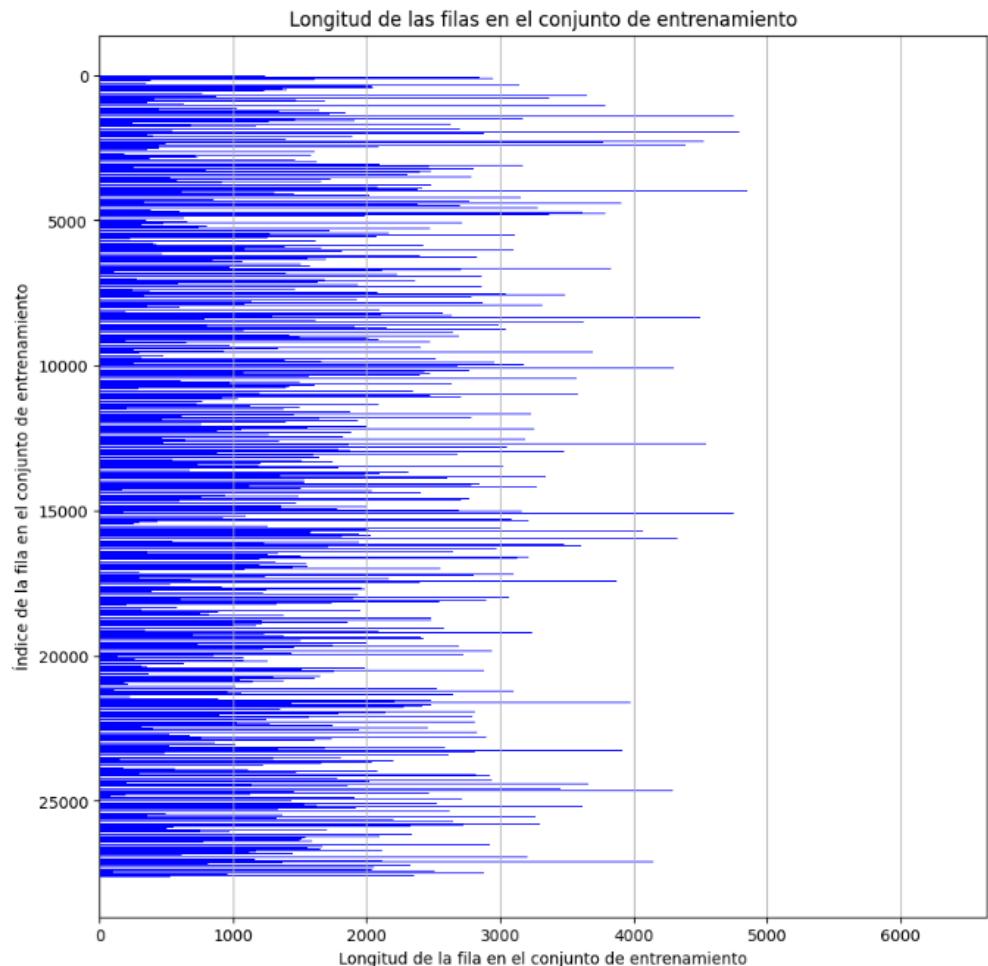


Figura 30: Longitud de las cadenas de hits por sucesos

La cantidad de hits del suceso con mayor número de hits en el conjunto de entrenamiento asciende a 6328 hits. Con ese valor organizaremos los hits en filas como vemos en la imagen 31 y completamos con un padding de ceros hasta completar cada fila, como podemos ver en la siguiente imagen:

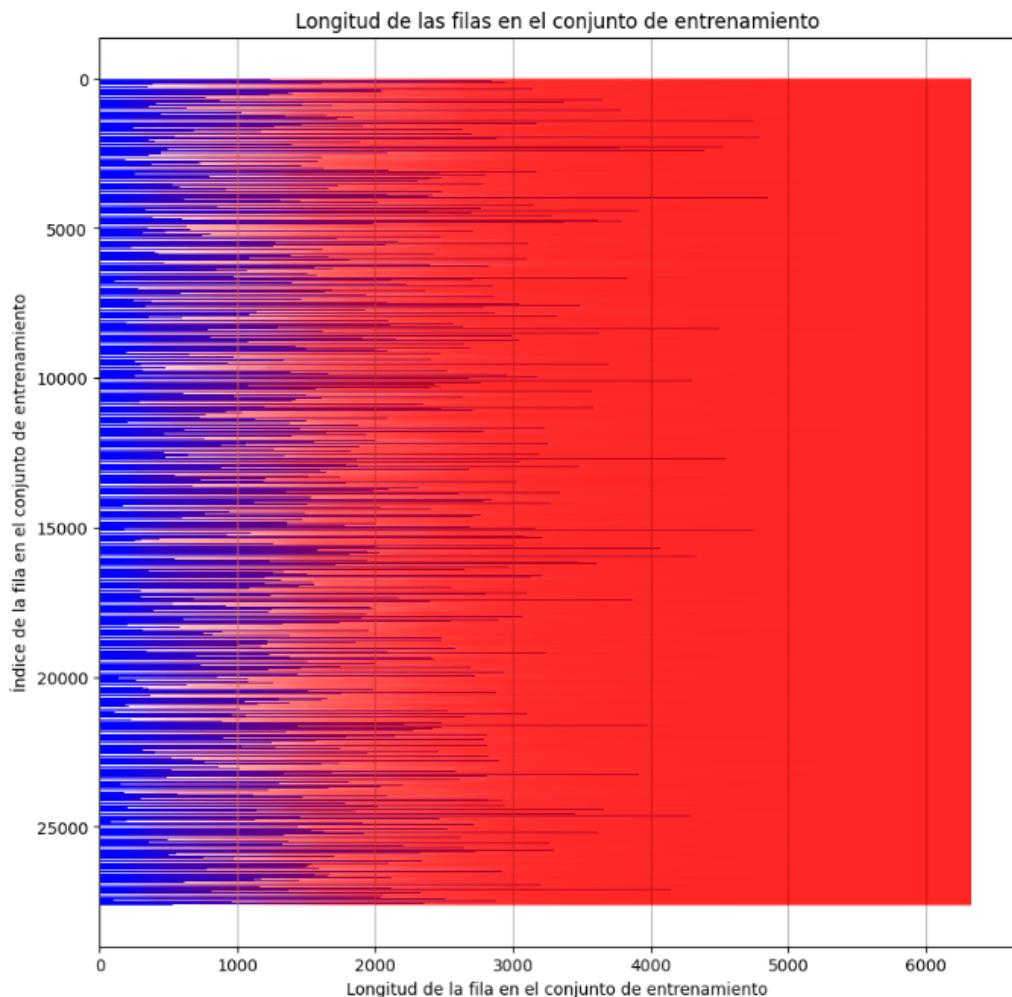


Figura 31: Longitud de las cadenas de hits por sucesos

### 6.3.2. Arquitecturas para el entrenamiento

En esta sección veremos las tres arquitecturas propuestas para la clasificación de los distintos modelos, estas son las siguientes:

Arquitectura convolucional (CNN):

Tipo de capa	Tamaño del kernel	Dimensiones de entrada	Dimensiones de Salida	Canales de entrada	Canales de salida
Conv1D	10	(6328,1)	(6319,128)	1	128
ReLU	-	(6319,128)	(6319,128)	-	-
Conv1d	8	(6319, 128)	(6312, 256)	128	256
ReLU	-	(6312, 256)	(6312, 256)	-	-
Global Average Pooling	-	(6312, 256)	(1, 256)	-	-
Dense	-	(256,)	(256,)	-	-
ReLU	-	(256,)	(256,)	-	-
Dense	-	(256,)	(128,)	-	-
ReLU	-	(128,)	(128,)	-	-
Dense	-	(128,)	(1,)	-	-
Sigmoid	-	(1,)	(1,)	-	-

Cuadro 9: Arquitectura Red Convolucional

Modelo de Redes Neuronales Recurrentes con capas LSTM (RNN-LSTM):

Tipo de capa	Tamaño del kernel	Dimensiones de entrada	Dimensiones de Salida	Canales de entrada	Canales de salida
LSTM	-	(6328,1)	(6328,128)	1	128
Global Average Pooling	-	(6328,128)	(128,)	-	-
Dense	-	(128,)	(256,)	-	-
ReLU	-	(256,)	(256,)	-	-
Dense	-	(256,)	(128,)	-	-
ReLU	-	(128,)	(128,)	-	-
Dense	-	(128,)	(1,)	-	-
Sigmoid	-	(1,)	(1,)	-	-

Cuadro 10: Arquitectura de Redes Recurrentes con capas LSTM (RNN-LSTM)

Modelo de Redes Neuronales Recurrentes con capas GRU (RNN-GRU):

Tipo de capa	Tamaño del kernel	Dimensiones de entrada	Dimensiones de Salida	Canales de entrada	Canales de salida
GRU	-	(6328,1)	(6328,128)	1	128
Global Average Pooling	-	(6328,128)	(128,)	-	-
Dense	-	(128,)	(256,)	-	-
ReLU	-	(256,)	(256,)	-	-
Dense	-	(256,)	(128,)	-	-
ReLU	-	(128,)	(128,)	-	-
Dense	-	(128,)	(1,)	-	-
Sigmoid	-	(1,)	(1,)	-	-

Cuadro 11: Arquitectura de Redes Recurrentes con capas GRU (RNN-GRU)

Las capas usadas en los distintos modelos son las siguientes:

- **Convoluciones:** Las capas de convolución en el análisis de series temporales y señales unidimensionales, aplican una operación de convolución a lo largo de la dimensión temporal de los datos de entrada. El funcionamiento de las capas convolucionales es el siguiente: desliza el kernel a lo largo de una secuencia de datos. El kernel tiene un tamaño fijo predefinido y se aplica al subconjunto contiguos de entrada. A medida que el kernel se desplaza a lo largo de la entrada, realiza un producto escalar entre el filtro y el segmento de la entrada sobre el que se encuentra. Los filtros están diseñados para capturar características locales importantes dentro de la secuencia. Al aplicar la convolución la dimensión de los datos se puede reducir lo cual es útil para disminuir la complejidad computacional y controlar el sobreajuste. [25]
- **ReLU:** Las capas ReLU son una función de activación ampliamente utilizadas en las redes neuronales para introducir no linealidad al modelo, factor muy importante a la hora de aprender y modelar relaciones complejas en los datos. Es una capa muy sencilla y efectiva. Viene definida matemáticamente como  $f(x) = \max(0, x)$ . [26]
- **Global Average Pooling:** Es una técnica de reducción de dimensiones apli-

cada a datos unidimensionales en el contexto de redes neuronales convolucionales. Al reducir cada canal del mapa de características a un solo valor promedio, GlobalAveragePooling1D efectivamente colapsa la dimensión temporal, pasando de un mapa de características 1D a un vector de características. Esto elimina la necesidad de capas densas completamente conectadas al final de la red, reduciendo significativamente el número total de parámetros. Con menos parámetros para aprender, el modelo es menos propenso a sobreajustarse a los datos de entrenamiento, lo que puede mejorar la generalización en datos no vistos. [29]

- **Dense:** Esta capa es también conocida como capas completamente conectadas, con un componente fundamental en las redes neuronales profundas, especialmente en las arquitecturas de redes neuronales. Cada neurona en una capa densa recibe entradas de todas las neuronas de la capa anterior, lo que significa que las características de todas las neuronas anteriores son utilizadas. Esta capa puede aprender patrones complejos combinando características de todas las entradas, lo que las hace muy efectivas para hacer predicciones basadas en un conjunto grande y diverso de características. Son esenciales para tareas de aprendizaje supervisado donde se necesita una salida específica basada en características de alto nivel extraídas por capas convolucionales, como en nuestro caso. [26]
- **Sigmoid:** Esta función es usada en redes neuronales, ya que especialmente es conocida por su capacidad de mapear valores de entrada arbitrarios a un rango cerrado entre 0 y 1. Esto lo hace útil para problemas de clasificación binaria donde es necesario interpretar la salida de la red como una probabilidad. Viene definida por la fórmula  $f(x) = \frac{1}{1+e^{-x}}$ . La salida de esta capa se puede interpretar fácilmente como probabilidades.
- **LSTM:** Es un tipo especializado de unidad recurrente utilizada en redes neuronales para procesar y predecir secuencias. Las LSTMs son eficaces para recuperar los problemas de dependencias a largo plazo que afectan a las redes neuronales. La estructura de la LSTM puede ser capaz de decidir qué información es relevante y debe ser descartada del estado de la celda,

actualiza el estado con nueva información y determina qué parte del estado de la celda actual va al siguiente tiempo y la salida de la red. A diferencia de las Redes Neuronales tradicionales, las LSTM pueden aprender y recordar información durante períodos prolongados, lo que es crucial para muchas tareas de procesamiento secuencial donde el contexto histórico influye en la interpretación presente.[17]

- **GRU:** Esta capa es un tipo avanzado y recurrente usada en redes neuronales para manejar secuencia de datos. Esta capa se diseña como una alternativa más simple y eficiente computacionalmente a la LSTM. Esta capa es capaz de determinar cuánta información del paso anterior se debe llevar a cabo al estado actual y decidir cuando el estado anterior debe ser olvidado. Al tener menos funciones que la LSTM la hace más fácil de computar y entrenar, lo que las hace adecuadas para conjuntos de datos grandes. Aunque sean más simples pueden llegar a tener un rendimiento similar a las LSTM [8]

### 6.3.3. Entrenamiento

Antes de entrenar las distintas arquitecturas, hay que llevar a cabo una fase de procesamiento de datos para formar la estructura que hemos visto en la figura 31. Para ello definiremos una función llamada `filtrado_datos_variable`, que se diferenciará de la función `filtrado_datos` en lo siguiente:

A diferencia de `filtrado_datos`, esta función no impone una longitud fija a las secuencias. En lugar de ello, mantiene la longitud natural de cada secuencia, lo que resulta en secuencias de longitud variable. Después se la añadirá un padding hasta completar con la secuencia de mayor tamaño en el caso del conjunto de datos de entrenamiento.

Aquí tampoco realizaremos un procesamiento de datos adicional, ya que los datos vienen de una simulación virtual y están libres de errores y valores anómalos, por tanto los introduciremos directamente en las distintas arquitecturas. Todos los entrenamientos que se van a realizar se van hacer con los siguientes requisitos:

- **Función de pérdida:** La función de perdida será *binary cross entropy*. Es

una función de pérdida utilizada comúnmente en problemas de clasificación binaria. La fórmula es la siguiente:

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i * \log(p(y_i)) + (1 - y_i) * \log(1 - p(y_i))$$

Esta función de pérdida castiga más severamente las predicciones incorrectas con alta confianza. En nuestro caso usaremos la que viene por defecto en las bibliotecas de TensorFlow a la hora de compilar el modelo. [23]

- **Early Stopping:** Es una técnica usada en el entrenamiento de los modelos. Su función principal es parar el entrenamiento cuando el modelo empieza a sobreajustarse demasiado a los valores del conjunto de entrenamiento. Durante el entrenamiento, se monitorea la métrica de la función de pérdida en el conjunto de datos de validación. Se define una paciencia, en nuestro caso será de 2, este número de épocas son las que el modelo puede seguir entrenando si no mejora la métrica de la función de pérdida en validación. Una vez se ha parado el entrenamiento, guarda el estado del modelo en el punto donde obtuvo la mejor métrica de evaluación. Esto asegura que el modelo finalizado sea el que tuvo el mejor rendimiento durante el entrenamiento. Usaremos la biblioteca de callbacks de TensorFlow y ajustaremos los parámetros según hemos comentado. [1]
- **ReduceLROnPlateau:** Es una función que ajusta la tasa de aprendizaje (lr) durante el entrenamiento del modelo, cuando la métrica monitoreada ha dejado de mejorar. Esta función evita las situaciones donde pueden producirse mínimos locales debido a una tasa de aprendizaje demasiado alta. La métrica que vamos a monitorear es la función de pérdida en el conjunto de validación, con un factor de 0.5, es decir, se reducirá a la mitad la tasa de aprendizaje cada vez que la métrica monitoreada deje de mejorar. Usaremos una paciencia de 0, ya que la reducción ocurrirá automáticamente después de que se detecte que una métrica no ha mejorado en la última época.

### 6.3.4. Resultados de entrenamiento

El resultado del entrenamiento obtenido por la arquitectura convolucional es el siguiente:

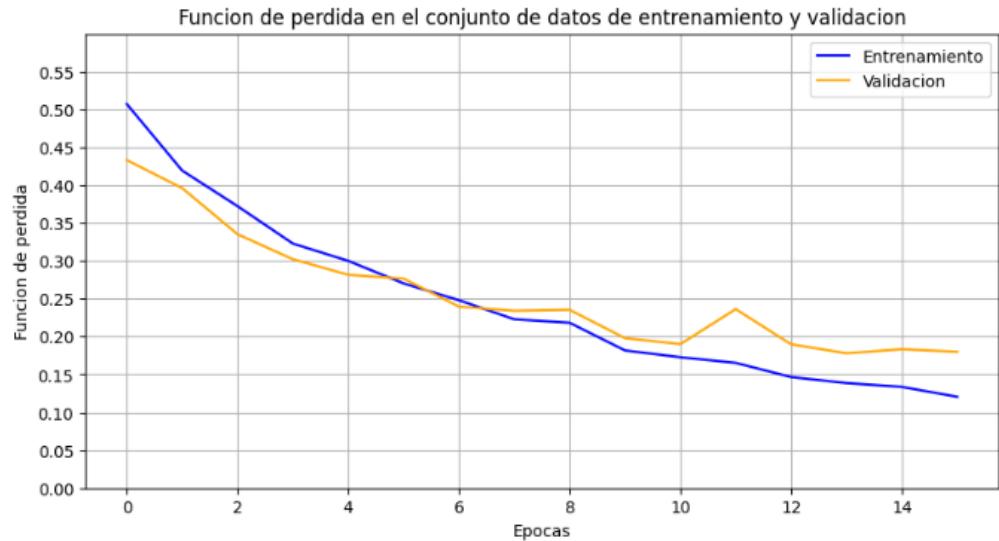


Figura 32: Comportamiento de la función de perdida entrenamiento arquitectura CNN

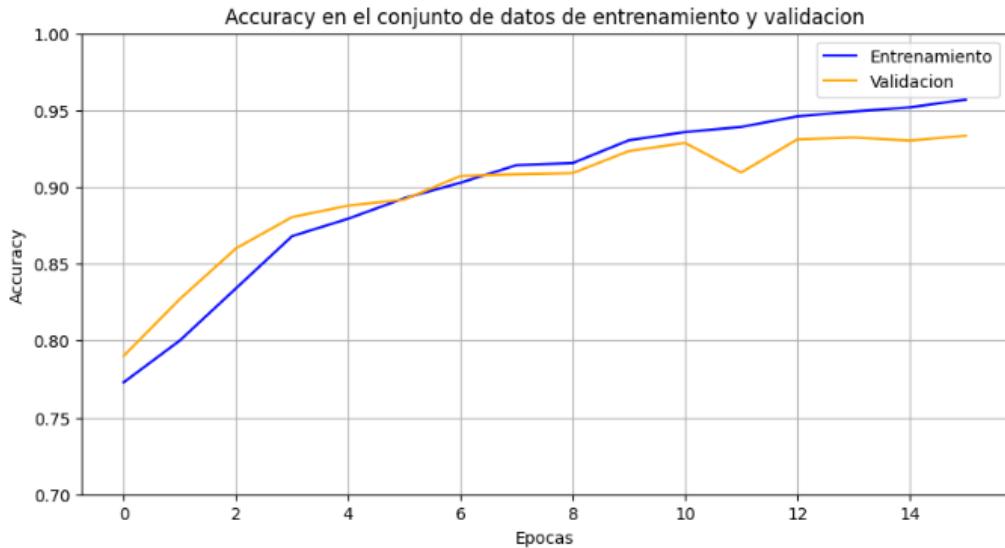


Figura 33: Comportamiento del accuracy en el entrenamiento arquitectura CNN

Según la figura 32 la función de pérdida en el conjunto de datos de entrenamiento disminuye constantemente desde aproximadamente 0.35 hasta cerca de 0.20 en las 7 primeras épocas. Aunque hay pequeñas fluctuaciones, la tendencia general es descendente, indicando que el modelo está aprendiendo y ajustando sus pesos correctamente. La misma evolución la tendríamos en la curva que representa el conjunto de datos de validación. Las pérdidas de entrenamiento y validación son bastante similares, lo que sugiere que el modelo no está sobreajustando significativamente a los datos de entrenamiento.

En la figura 33 la precisión en el conjunto de datos de entrenamiento aumenta consistentemente de alrededor de 0.85 a más de 0.9 en las primeras 7 épocas. La precisión muestra una tendencia positiva y estable, indicando que el modelo está mejorando su capacidad para clasificar correctamente los datos de entrenamiento. En la curva que representa el conjunto de datos de validación, la precisión comienza alrededor de 0.88 y aumenta hasta más de 0.90. La tendencia en esta curva es similar a la del conjunto de entrenamiento. Como la curva de precisión en ambos conjuntos de datos son similares, sugiere que el modelo generaliza bien y no está sobreajustando a los datos de entrenamiento. Como podemos ver, el modelo ha si-

do entrenado de manera efectiva para la tarea de clasificación de ambas partículas.

Ahora veremos un resumen gráfico del entrenamiento de la arquitectura compuesta por capas de LSTM.

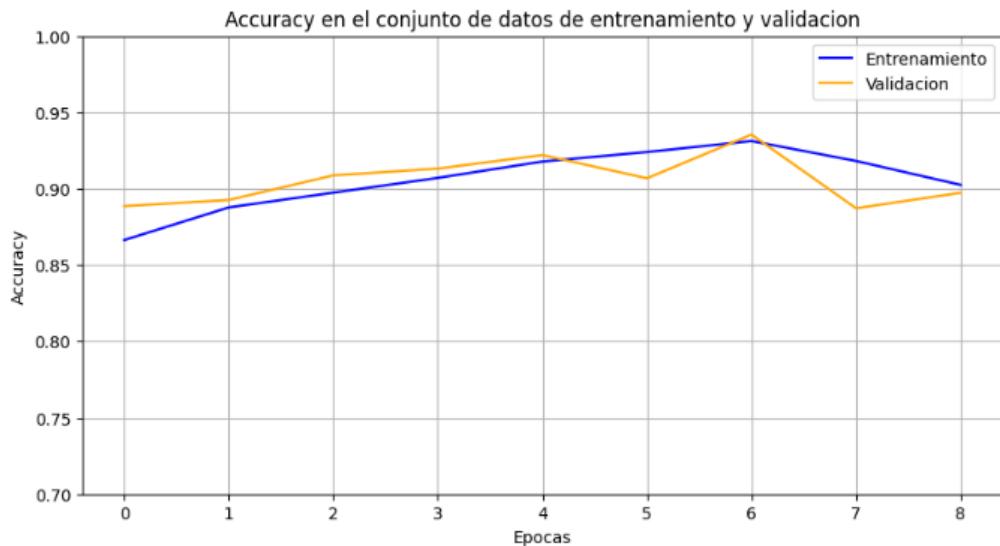


Figura 34: Comportamiento de la función de perdida entrenamiento arquitectura LSTM

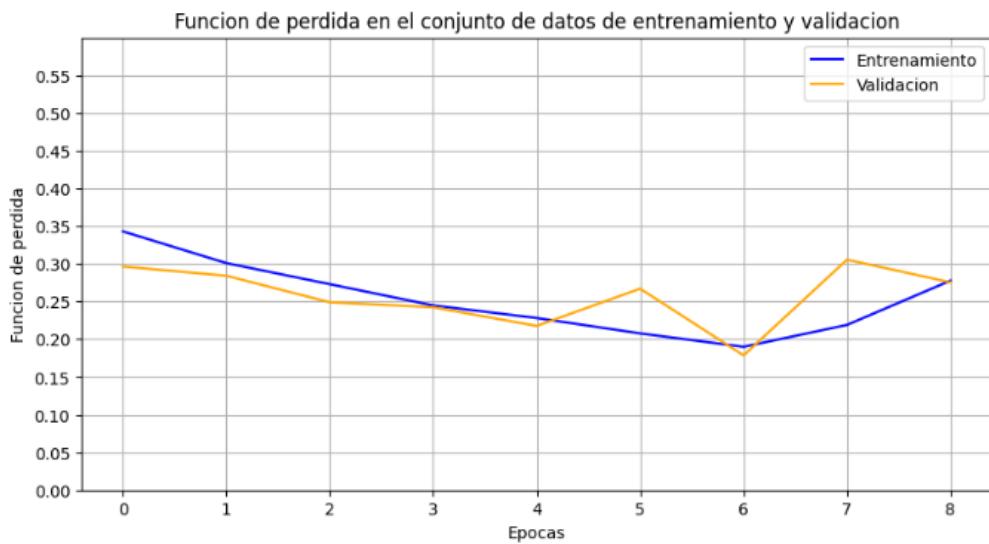


Figura 35: Comportamiento del accuracy en el entrenamiento arquitectura LSTM

Como podemos ver en la gráfica de la función de pérdida 34 la tendencia de la curva que indica la pérdida en el conjunto de datos de entrenamiento disminuye de manera constante desde aproximadamente 0.35 hasta alrededor de 0.20 en las primeras épocas. Hay pequeñas fluctuaciones, pero la tendencia general es descendente, lo que indica que el modelo está aprendiendo y ajustando sus pesos correctamente. La curva que indica la función de pérdida en el conjunto de datos de validación también muestra una tendencia general a la baja desde aproximadamente 0.30 hasta alrededor de 0.20. Se observan fluctuaciones donde la pérdida en validación aumenta antes de disminuir, lo que puede ser un indicativo de un sobreajuste temporal.

Como podemos ver en la gráfica que refleja el comportamiento del modelo en términos de accuracy mediante el entrenamiento por épocas en la figura 35, la precisión con el conjunto de datos de entrenamiento aumenta de manera constante desde aproximadamente 0.85 hasta 0.90 en sus primera épocas. La precisión muestra una tendencia positiva y estable, indicando que el modelo está mejorando su capacidad para clasificar correctamente los datos de entrenamiento. De forma similar ocurre en la curva que indica el comportamiento en términos de accuracy

en el conjunto de validación.

A continuación veremos el entrenamiento generado por la arquitectura formada por capas GRU:

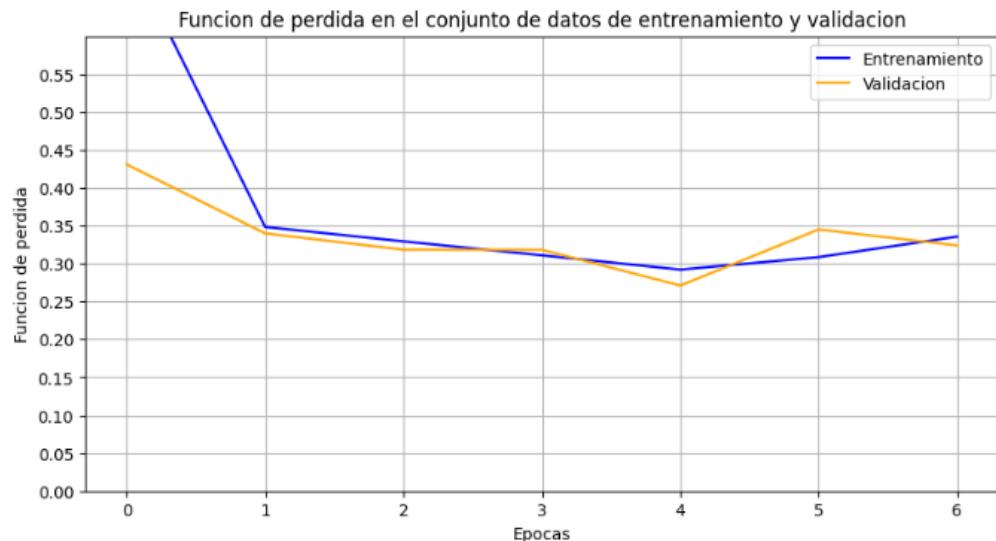


Figura 36: Comportamiento de la función de perdida entrenamiento arquitectura GRU

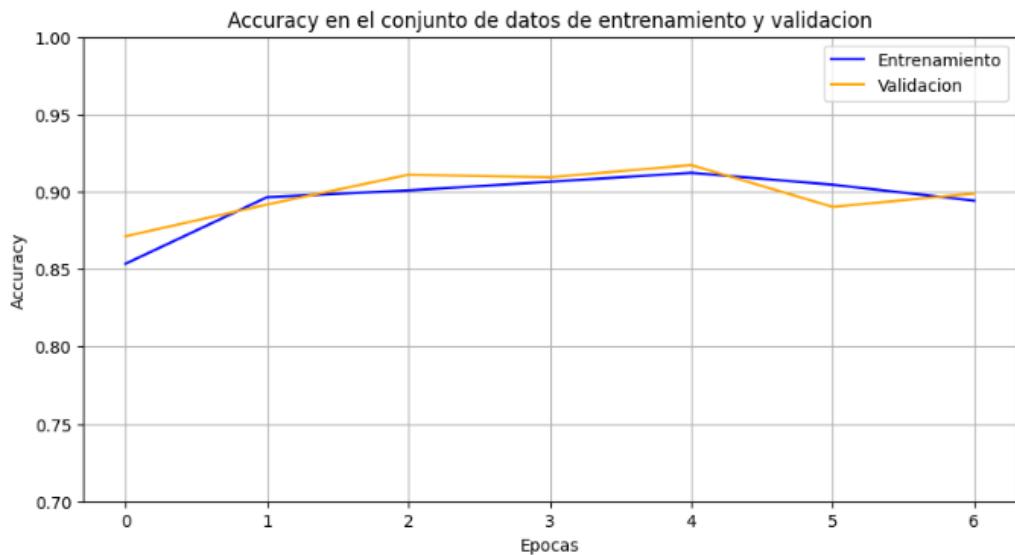


Figura 37: Comportamiento del accuracy en el entrenamiento arquitectura GRU

Según la gráfica de la función de pérdida 36 muestra un entrenamiento de 7 épocas. En el conjunto de datos de entrenamiento es descendente, ya que disminuye bruscamente desde aproximadamente 0.55 hasta alrededor de 0.35. Luego continúa disminuyendo hasta estabilizarse alrededor de 0.30. Como la tendencia es descendente, indica que el modelo está aprendiendo de manera efectiva. En el conjunto de datos de validación disminuye de manera constante desde 0.45 hasta estabilizarse cerca de 0.30 en las 5 primeras épocas. Como las funciones de pérdida en el conjunto de datos de entrenamiento y validación son bastante similares después de la primera época, lo que sugiere que el modelo no está sobreajustando y que está generalizando bien.

Si ahora nos fijamos en la gráfica de accuracy 37 vemos cómo la tendencia en el conjunto de datos de entrenamiento es ascendente consistentemente desde 0.85 hasta 0.9 en las cuatro primeras épocas. La precisión muestra una tendencia positiva estable, indicando que el modelo está mejorando su capacidad para clasificar correctamente los datos de entrenamiento. En el conjunto de datos de validación la precisión comienza alrededor de 0.88 y se mantienen cercada a 0.90. Las dos

curvas son muy cercanas, lo que nos indica que el modelo está generalizando bien y no está sobreajustando a los datos de entrenamiento.

En el caso del entrenamiento de la arquitectura compuesta por las capas de LSTM y GRU, al final del entrenamiento se ve una fluctuación bastante negativa en lo que al aprendizaje se refiere, por lo que las funciones de EarlyStopping hace que el entrenamiento pare quedándose con el estado que presenta unos valores de la función de pérdida en el conjunto de validación más prometedor en cuanto al rendimiento del modelo.

### 6.3.5. Comparación de entrenamientos

A continuación veremos una comparativa gráfica del entrenamiento de los tres modelos para poder definir el mejor modelo:

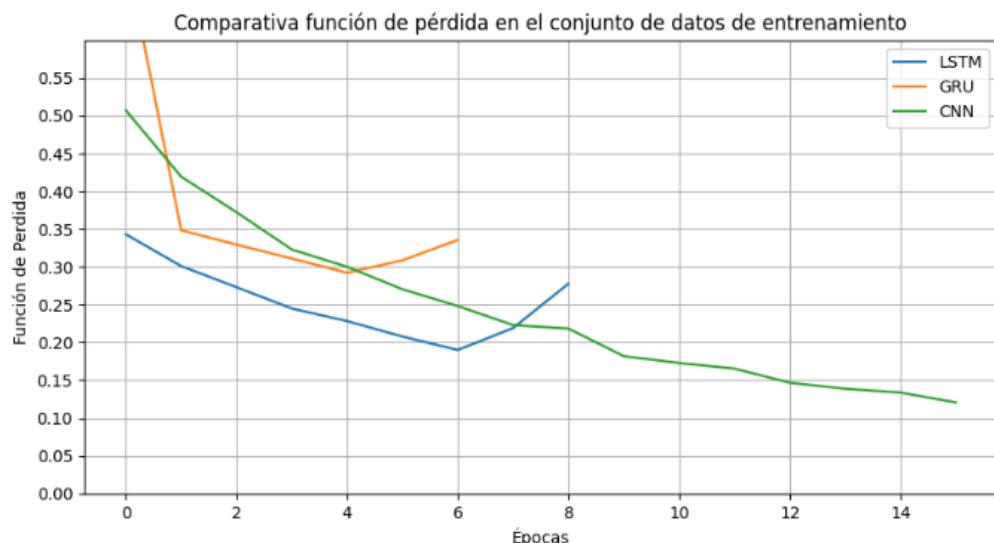


Figura 38: Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de entrenamiento

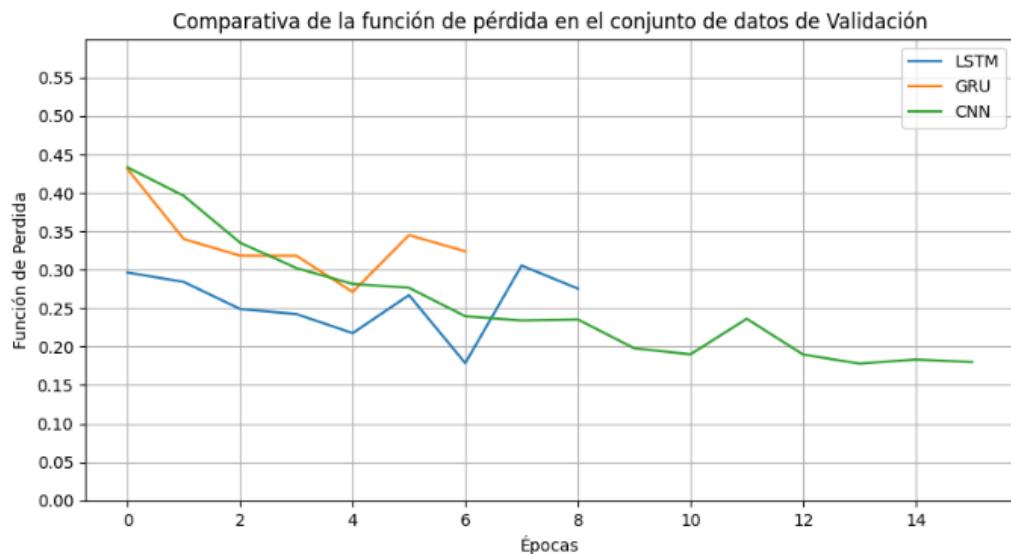


Figura 39: Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de validación

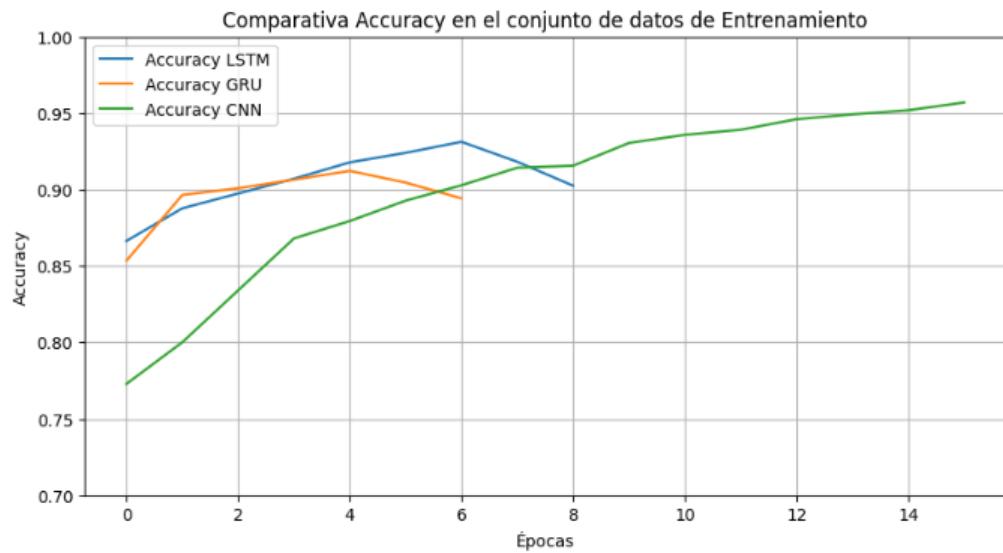


Figura 40: Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de entrenamiento

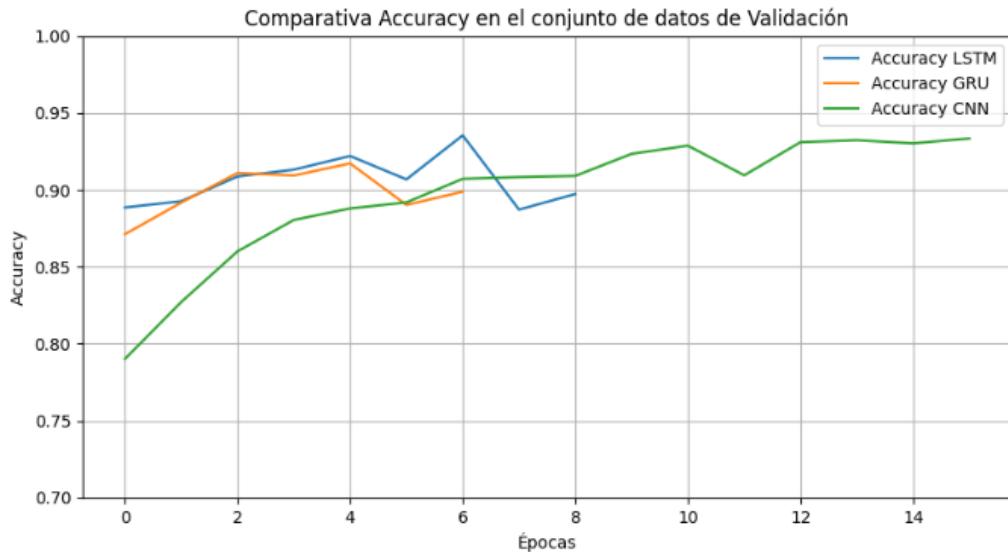


Figura 41: Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de validación

En cuanto al accuracy en el conjunto de datos de validación, vemos cómo la red compuesta por capas convolucionales muestra un aumento constante y alcanza un nivel estable alrededor de la época 6. A partir de ahí la precisión se mantiene en torno a 0.90. La red compuesta por la capa LSTM es consistente y alcanza valores muy próximos a la red compuesta por capas convolucionales, pero muestra unas fluctuaciones notables en las siguientes épocas. El modelo compuesto por capas GRU alcanza buenos valores de accuracy pero muestra una ligera disminución y estabilización en torno a 0.90 a partir de la época 4.

Según la gráfica de accuracy en el conjunto de datos de entrenamiento, la red compuesta por capas convolucionales muestra un aumento constante y alcanza alrededor de 0.95 en las épocas finales, mostrando una mejora continua sin signos de sobreajuste. La precisión de la arquitectura compuesta por capas LSTM alcanza un pico de 0.95, similar al modelo anterior compuesto por capas convolucionales, pero muestra más fluctuaciones en comparación con el modelo de redes convolucionales. La arquitectura compuesta por capas GRU alcanza un valor de accuracy entorno a 0.90 y luego muestra una tendencia a estabilizarse, aunque tiene más

fluctuaciones que los demás modelos.

La función de pérdida en el conjunto de datos de validación muestra cómo la red convolucional disminuye de manera constante y se estabiliza alrededor de 0.20, mostrando una mejora continua. La red compuesta por las capas LSTM muestra una disminución significativa, aunque con algunas fluctuaciones. La arquitectura compuesta por las capas GRU disminuye rápidamente al principio, pero luego se estabiliza y muestra fluctuaciones alrededor de 0.30.

En cuanto al conjunto de datos de entrenamiento, la función de pérdida muestra una disminución consistente a lo largo de las épocas para la red convolucional, indicando una tendencia a la baja que se estabiliza en niveles bajos hacia las últimas épocas. La arquitectura con capas LSTM también presenta una disminución constante, aunque con más fluctuaciones en comparación con la red convolucional. Por su parte, la arquitectura con capas GRU disminuye de manera significativa inicialmente, pero luego se estabiliza mostrando algunas fluctuaciones.

La red neuronal muestra un rendimiento muy sólido con una precisión alta y establece en el conjunto de entrenamiento como en el de validación, y una función de pérdida que disminuye consistentemente. Su estabilizador y tendencia a mejorar sin muchas fluctuaciones lo hacen una opción muy robusta. La red compuesta por la capa LSTM tiene un rendimiento alto, pero muestra más fluctuaciones en comparación con CNN, especialmente en las últimas épocas. La red compuesta por la capa GRU tiene un buen rendimiento inicial, pero tiene fluctuaciones y una función ligeramente más alta en comparación con la red convolucional y las LSTM, por lo que la hacen menos preferible.

### 6.3.6. Entrenamiento del mejor modelo

El modelo de red convolucional parece ser el mejor de los tres en términos de estabilidad, precisión y menor pérdida en el conjunto de entrenamiento como en el de validación. Por tanto, entrenaremos este modelo con todos los datos de

entrenamiento y validación para que generalice mejor.

Para dar el valor de accuracy en el conjunto de datos de prueba, se ha entrenado todo el modelo desde cero con el conjunto de datos de entrenamiento y validación para que el modelo pueda aprovechar toda la información disponible. Este entrenamiento se realizó con los mejores hiperparámetros identificados durante la fase de validación, asegurando así que el modelo esté optimizado para su desempeño. Una vez completado el entrenamiento, el modelo fue evaluado utilizando el conjunto de datos de prueba para medir su precisión y capacidad de generalización. Los resultados obtenidos de accuracy en el conjunto de datos de prueba proporcionan una medida objetiva de la efectividad del modelo y permiten confirmar que la red convolucional es la opción más adecuada para el problema abordado.

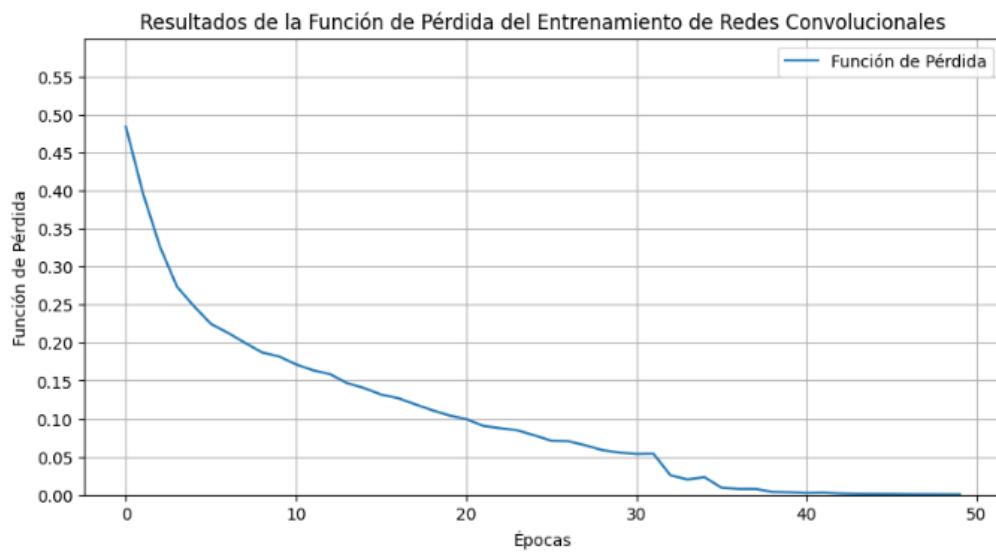


Figura 42: Función de pérdida del entrenamiento del mejor modelo (CNN)

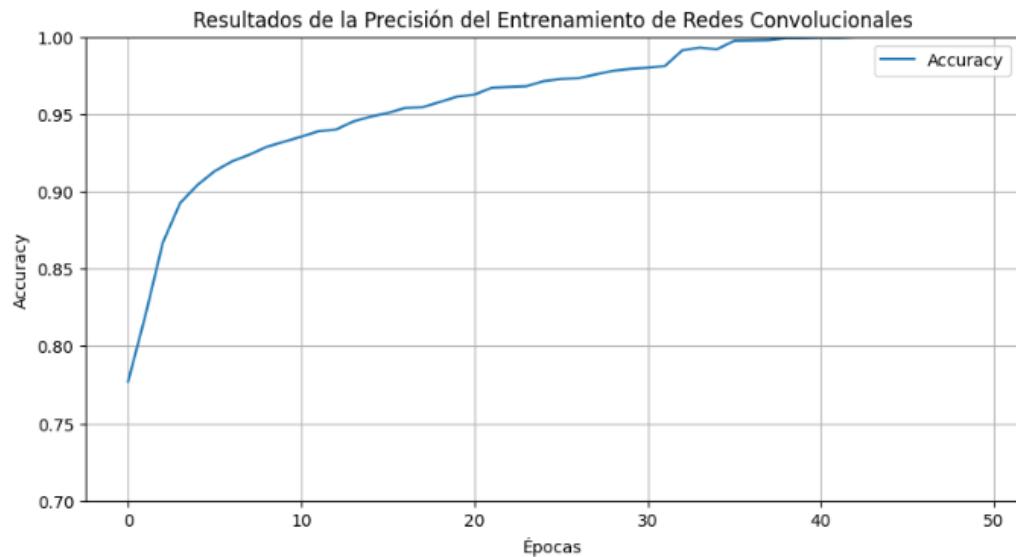


Figura 43: Accuracy del entrenamiento del mejor modelo (CNN)

El entrenamiento duró 50 épocas utilizando los datos de entrenamiento y validación. A lo largo del proceso de entrenamiento se registraron tanto la función de pérdida como la de accuracy para evaluar el rendimiento del entrenamiento.

En la figura de la función de pérdida 42, se observa la evolución de la función de pérdida a lo largo de las épocas. Inicialmente, la pérdida es relativamente alta, pero disminuye de manera constante a medida que el modelo aprende a partir de los datos de entrenamiento. Al final del entrenamiento, la pérdida alcanza valores muy bajos, lo que indica que el modelo ha ajustado sus parámetros para minimizar los errores en las predicciones.

En la imagen de la curva de accuracy 43 muestra la precisión del modelo durante el entrenamiento. Se puede observar que la precisión aumenta rápidamente en las primeras épocas, superando el 0.90 después de aproximadamente 10 épocas. Posteriormente, el aumento de la precisión es más gradual, alcanzando valores cercanos a 1 al finalizar el entrenamiento. Este comportamiento sugiere que el modelo no solo ha aprendido a reducir los errores, sino que también ha mejorado

su capacidad para hacer predicciones correctas de manera consistente.

Con esta metodología aseguramos que el modelo ha aprovechado al máximo la información disponible durante el entrenamiento y está en la mejor posición para generalizar sus predicciones a nuevos datos.

### 6.3.7. Resultados finales

Como parte final del entrenamiento del modelo veremos cómo se comporta en términos de accuracy general y en los tres niveles de energía, baja, media y alta. Para categorizar los datos en tres niveles, entre baja, media y alta, es necesario ordenar los valores de *trueE* y segmentar este conjunto en tres grupos iguales. De acuerdo con esta metodología, los umbrales que definen los cortes entre los rangos de baja y media energía, y entre media y alta energía, son respectivamente 0.42793 GeV y 0.559643 GeV.

Según esto, es importante agregar que antes de mostrar los resultados de prueba, se ha realizado un entrenamiento con todos los datos del conjunto de entrenamiento y validación, para poder entrenar el modelo con más datos y que se generalice de mejor forma. Los resultados han sido los siguientes:

Conjunto de datos	Accuracy	Accuracy bajas energías	Accuracy medias energías	Accuracy altas energías
Entrenamiento	0.95375	0.96230	0.94904	0.94991
Validación	0.93131	0.95001	0.92450	0.91941
Prueba	0.93791	0.94756	0.93907	0.92710

Cuadro 12: Resultados de accuracy en los distintos tipos de energías

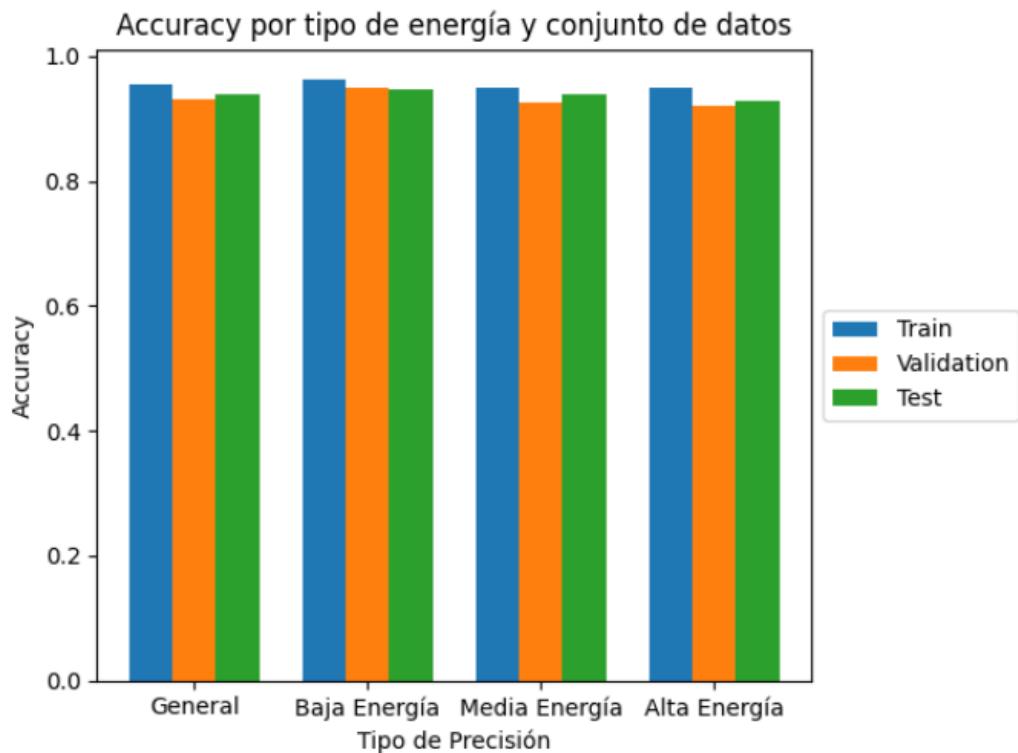


Figura 44: Resultados de accuracy en los distintos tipos de energías

El modelo de la red convolucional entrenado demuestra ser efectivo y estable, logrando alta precisión en la clasificación de energías bajas, medias y altas en todos los conjuntos de datos evaluados. Este desempeño indica que el modelo ha generalizado bien a partir de los datos de entrenamiento y es capaz de realizar predicciones precisas en datos no vistos anteriormente.

Este análisis final confirma la robustez del modelo y su adecuación para tareas de clasificación en diferentes niveles de energía, cumpliendo con los objetivos planteados al inicio del proyecto.

## 7. Problema de Regresión

Dentro de este apartado resolveremos el problema de regresión donde calcularemos el valor de la energía de entrada de la partícula, que viene representado en la variable *trueE*, ya que este valor no se conoce mientras se produce el experimento, si no que se conoce a posteriori. Por tanto, realizaremos una regresión para calcularlo. En esta sección trataremos de abordar la solución para resolver el problema.

### 7.1. Adaptación del conjunto de datos al entrenamiento

Como primera aproximación, cogeremos la misma estructura que usamos para la clasificación binaria por medio de modelos de Aprendizaje Automático tradicionales, cogeremos de cada suceso el valor correspondiente a la variable *trueE*. Este valor, como se dijo en el apartado de clasificación es el mismo para un mismo suceso. Por tanto, a la función que devuelve la entrada a los modelos de aprendizaje automático devolveremos también el valor correspondiente a la energía de entrada de la partícula, ya que este valor será nuestra etiqueta.

Los datos quedarán como la figura 17. Por lo tanto como primera aproximación entrenaremos con solo 4 tipos de características, *hitX*, *hitY*, *hitZ* y *hitInteg*.

Además de probar con el conjunto de datos 17, probaremos con esta misma estructura, pero al principio de cada lista de hits le indicaremos el tipo de partícula que es. Esto lo haremos a partir de los modelos de clasificación ya que hemos obtenido unos buenos resultados (más de un 0.90 de accuracy).

Los datos quedarán quedará de la siguiente manera:

	hitX				hitY				hitZ				hitInteg				
EventID	Tipo particula	hitX0	hitX1	...	hitXN	hitY0	hitY1	...	hitYN	hitZ0	hitZ1	...	hitZN	hitInteg0	hitInteg1	...	hitIntegN
event0																	
event1																	
event2																	
event3																	
event4																	

Figura 45: Estructura de datos para el problema de regresión con modelos tradicionales de Aprendizaje Automático

Como podemos observar, a parte de los entrenamientos anteriores, entrenaremos nuestros modelos para el problema de la regresión con 5 características, *tipoParticula*, *hitX*, *hitY*, *hitZ* y *hitInteg*

## 7.2. Elección del número de hits por suceso

Para elegir el número de hits que debemos coger para cada modelo, haremos como en el problema de clasificación, y cogeremos un rango de hits por sucesos entre 400 y 800 para entrenar los modelos de regresión. Este rango evita los valores extremadamente bajos, los cuales podrían no tener suficiente información y los valores extremadamente altos que podrían contener casos ruidosos ya que habría que completar aquellos hits con ruido.

Este rango asegura que el modelo está entrenando con un conjunto de datos que es estadísticamente significativo, común y lo suficientemente grande como para aprender, lo que puede mejorar la capacidad del modelo para generalizar a nuevos datos que se parecen a los más frecuentes en el conjunto de entrenamiento.

Todo este estudio está fundamentado por las figuras 18 y 19.

## 7.3. Modelos tradicionales de aprendizaje automático para la regresión

En este apartado veremos los distintos algoritmos de aprendizaje supervisado y no supervisado que se usarán para la regresión de la energía de entrada. Los

algoritmos son los siguientes:

- **Regresión lineal:** Predice el valor de una variable dependiente basándose en una variable independiente. Este análisis estima los coeficientes de una ecuación lineal que mejor se ajusta a los datos, utilizando el método de mínimos cuadrados para minimizar las discrepancias entre los valores predichos y reales. Los modelos de regresión lineal son sencillos y aplicables a múltiples campos ya que proporcionan una fórmula matemática fácil de interpretar y entrenar rápidamente. [21]
- **Regresión Ridge:** Es una técnica de regularización utilizada en regresión lineal para abordar problemas de multicolinealidad y sobreajuste. Se añade un término de penalización a la función de coste que es proporcional al cuadrado de la magnitud de los coeficientes. Este término de regularización está controlado por el hiperparámetro alfa ( $\alpha$ ), también conocido como parámetro de regularización, que determina la cantidad de penalización aplicada. Es especialmente útil cuando se trabaja con datos donde las variables pueden estar altamente correlacionadas. [18]
- **Regresión Lasso:** es una técnica de regularización y selección de variables en regresión lineal. A diferencia de la regresión ridge, que penaliza la suma de los cuadrados de los coeficientes, Lasso penaliza la suma de los valores absolutos de los coeficientes. Lasso elimina aquellas variables que no contribuyen al modelo. [41]
- **Random Forest:** Es una técnica de aprendizaje supervisado que se utiliza tanto para la clasificación como para la regresión. Fue desarrollada para mejorar la precisión y robustez de los modelos individuales de árboles de decisión. Random Forest construye múltiples árboles de decisión durante el entrenamiento y produce la media de las predicciones individuales de los árboles (para la regresión) o la mayoría de las predicciones (para la clasificación) para tomar la decisión final. Es especialmente útil cuando se trabaja con grandes conjuntos de datos con muchas características. Una de las principales ventajas de Random Forest es su capacidad para manejar

datos faltantes y mantener la precisión con conjuntos de datos grandes, a la vez que reduce el riesgo de sobreajuste. [5]

- **Bayesian Ridge Regression:** es una variante de la regresión ridge que incorpora un enfoque bayesiano para la estimación de los coeficientes del modelo. Mientras que la regresión ridge minimiza una función de coste que incluye un término de penalización para reducir la magnitud de los coeficientes, la regresión bayesian ridge añade una capa adicional de incertidumbre modelando los coeficientes como variables aleatorias con distribuciones probabilísticas. [11]
- **Stochastic Gradient Descent:** Es un método de optimización iterativo usado para minimizar funciones de pérdida en modelos de aprendizaje automático. A diferencia del descenso de gradiente tradicional, que computa el gradiente utilizado todo el conjunto de datos, el SGD actualiza los parámetros del modelo basándose en gradientes calculados a partir de muestras individuales o pequeños lotes de muestras. Esto permite una convergencia más rápida y eficiente, especialmente en contextos con grandes volúmenes de datos. [4]

Al igual que en el caso de clasificación, existen hiperparámetros que influyen directamente en los resultados obtenidos por los algoritmos. Por lo tanto, no solo es crucial elegir el algoritmo correcto, sino también configurarlo adecuadamente a través de los hiperparámetros.

Para poder elegir los parámetros de los modelos correctamente usaremos un framework que facilita la optimización de hiperparámetros. Es un framework de optimización automática de hiperparámetros que utiliza algoritmos de optimización bayesiana para encontrar la mejor combinación de hiperparámetros en modelos de aprendizaje automático. Permite la definición flexible de espacios de búsqueda y utiliza técnicas avanzadas para optimizar de manera eficiente. Optuna facilita la integración con diversas bibliotecas de aprendizaje automático. [33]

Los hiperparámetros elegidos son los siguientes:

Modelo	Hiperparámetros
Regresión Lineal	fit_intercept: true, copy_X: true, positive: true
Regresión Ridge	alpha: 1.5, fit_intercept: True, copy_X: False, solver: sag, positive: False
Regresión Lasso	alpha: 1.5, fit_intercept: False, max_iter: 1500, warm_start: True, positive: False, selection: cyclic
Elastic Net	alpha: 0.5, fit_intercept: True, max_iter: 500, warm_start: False, positive: False, selection: cyclic
Random Forest	n_estimators: 50, max_depth: 22, min_samples_split: 2, min_samples_leaf: 2, max_features: sqrt
Bayesian Ridge	compute_score: True, fit_intercept: True, max_iter: 300
Stochastic Gradient Descent	penalty: elasticnet, fit_intercept: False, max_iter: 1200, learning_rate: invscaling, warm_start: True

Cuadro 13: Parámetros de los modelos de Aprendizaje Automático para la regresión

## 7.4. Entrenamiento

Para entrenar este modelo lo haremos con los hiperparámetros que nos ha proporcionado Optuna como los mejores parámetros. Para evaluar los entrenamientos lo haremos con las siguientes métricas de error.

- **Error Absoluto Medio (MAE):** Es una métrica simple, pero poderosa que se utiliza para evaluar la precisión de los modelos de regresión. MAE mide la diferencia absoluta promedio entre los valores predichos y los valores reales objetivo. A diferencia de otras métricas, el MAE no eleva al cuadrado los errores, lo que significa que da igual peso a todos los errores. Al dar el mismo peso a todos los errores, ofrece una visión equilibrada de la magnitud de las desviaciones entre las predicciones y los valores reales. La fórmula es la siguiente:  $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$  [36]
- **Error Cuadrático Medio (MSE):** Es una métrica que mide la diferencia promedio entre los valores predichos por el modelo y los valores reales. Esta métrica es especialmente útil porque penaliza los errores grandes más severamente que los pequeños, debido a que los errores están elevados al cuadrado. Por ese motivo otorga más peso a los errores grandes, lo que

puede ser útil si los grandes errores son particularmente indeseables. La formula es la siguiente:  $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$  [7]

- **Coeficiente de Determinación ( $R^2$ ):** Es una métrica fundamental para evaluar el rendimiento de los modelos de regresión. Mide la proporción de la varianza total en la variabilidad dependiente que es explicada por las variables independientes del modelo. Un valor alto indica un buen modelo, pero también puede significar que el modelo está sobreajustando a los datos de entrenamiento. La formula es la siguiente:  $R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$  [2]

Hay que tener en cuenta, que el rango de energía de entrada, en el conjunto de entrenamiento va desde 0.13957 y 0.776955, como la diferencia del rango de los valores es pequeña, siendo de 0.637385, el MSE y el MAE no serán valores muy grandes. Por tanto, es importante interpretar estos valores en el rango de los datos. Un MSE y un MAE bajo indica que, en promedio las predicciones del modelo están cerca de los valores reales, lo cual es deseable.

En este rango tan reducido, pequeñas desviaciones pueden ser significativas. Es crucial reconocer que un MSE y MAE bajo este contexto puede indicar una buena precisión, pero la interpretación debe hacerse en relación con el rango total de los datos.

Por ese motivo se van a mostrar también valores normalizados de MSE y MAE, para ver el error relativo respecto al rango de los datos. Este añadido ayudará a obtener una mejor comprensión del rendimiento del modelo en el contexto de la escala de los datos.

El  $R^2$  es una métrica complementaria que puede proporcionar información sobre la proporción de la variabilidad total de los datos que es explicada por el modelo. Un  $R^2$  alto, junto con un MSE y MAE bajos, puede confirmar la precisión del modelo en el contexto de un rango reducido de datos.

## 7.5. Resultados de Entrenamiento

En esta sección veremos los resultados con las dos entradas de datos y nos quedaremos con el resultado del mejor modelo para cada entrada y después los compararemos.

### 7.5.1. Resultados Entrenamiento con el conjunto de datos para la clasificación

La comparativa entre los resultados de los distintos modelos en términos de MAE, MSE,  $R^2$ , MSE normalizada y MAE normalizada son los siguientes:

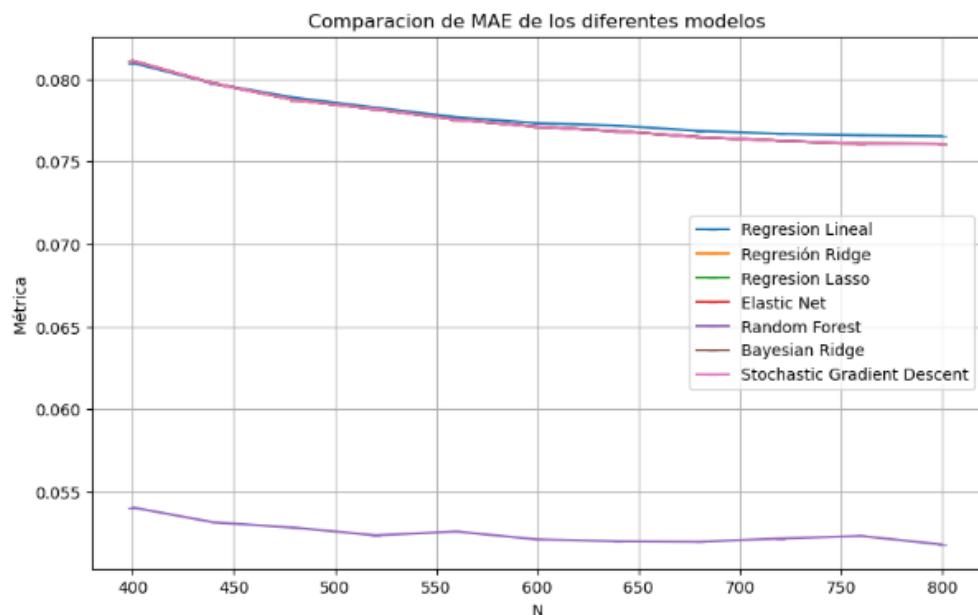


Figura 46: Error Absoluto Medio

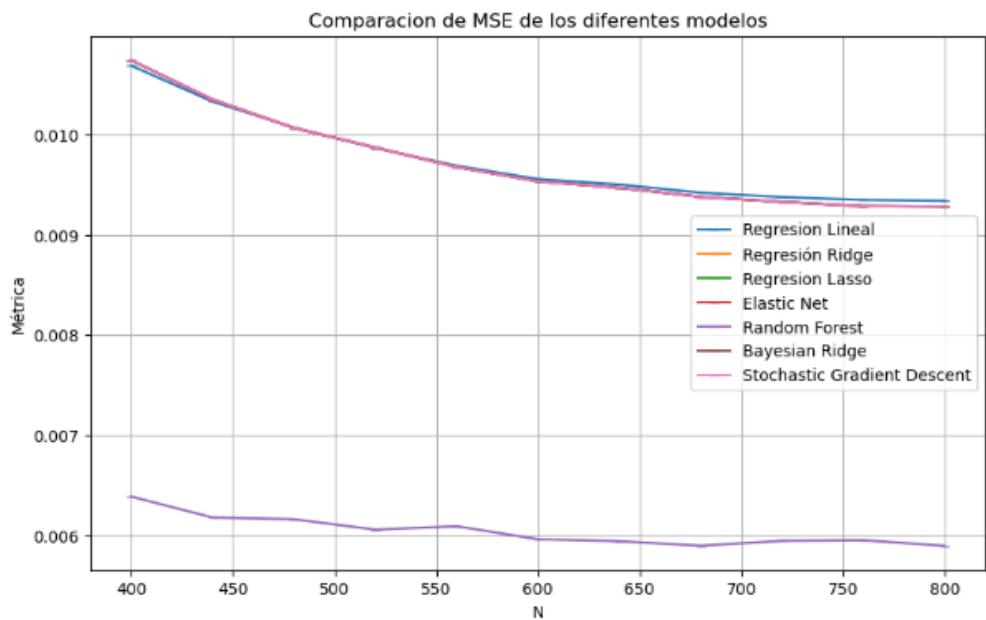


Figura 47: Error Cuadrático Medio

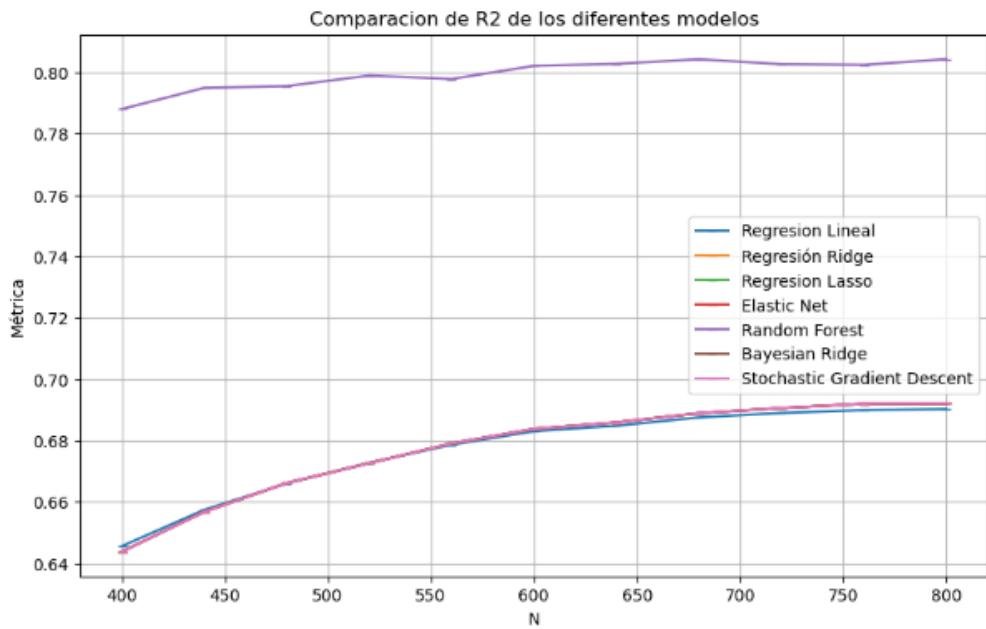


Figura 48: Coeficiente de Determinación

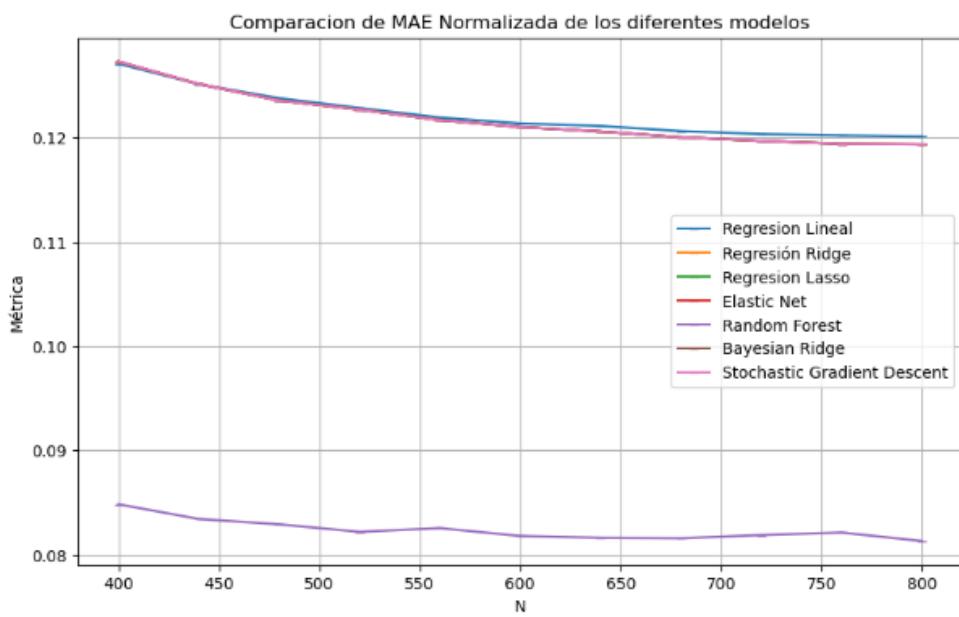


Figura 49: Error Absoluto Medio Normalizado

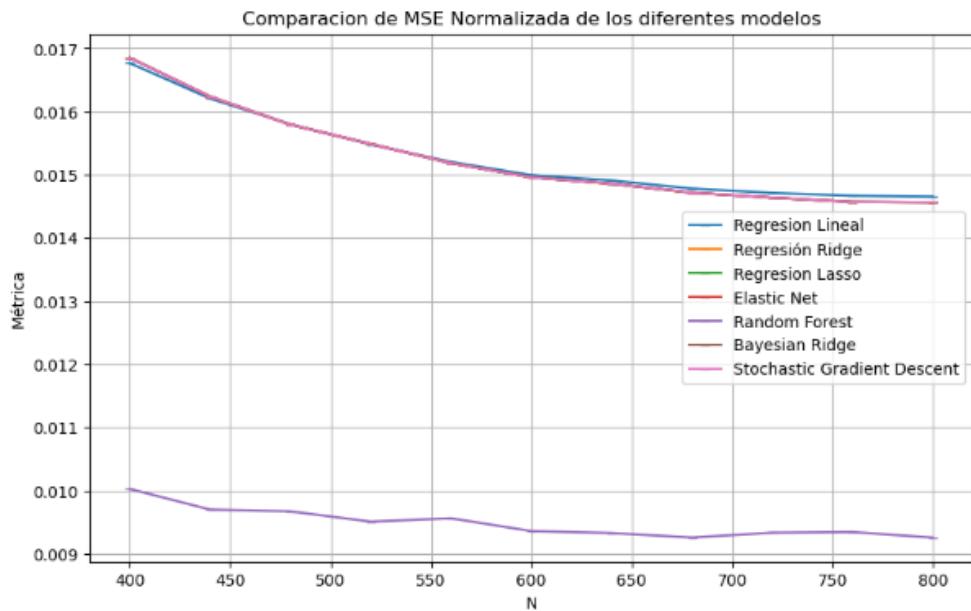


Figura 50: Error Cuadrático Medio Normalizado

El MAE y su versión normalizada (figuras 46 y 49) son métricas que indican el promedio de los errores absolutos entre las predicciones y los valores reales. Según nuestros resultados el modelo de Random Forest presente los valores más bajos de MAE y MAE normalizado. Esto indica que, en promedio, las predicciones del modelo de Random Forest está más cerca de los valores reales en comparación con los otros modelos. Los demás modelos atienen un rendimiento muy similar, con valores de MAE y MAE normalizado ligeramente superiores a los de Random Forest. Esto nos indica que son también modelos efectivos, tienen un rendimiento ligeramente inferior al modelo de Random Forest en términos de error absoluto.

El MSE y su versión normalizada (figuras 47 y 50) son métricas que miden el promedio de los errores al cuadrado entre las predicciones y los valores reales. Por tanto, esta métrica penaliza los errores más grandes que el MAE, proporcionando una medida más estricta del rendimiento del modelo. El modelo de Random Forest nuevamente presenta los valores más bajos de MSE y MSE normalizado.

Esto indica que el modelo no solo optimiza los errores pequeños sino que también maneja los errores grandes, haciendo sus predicciones más confiables. Los demás modelos presentan un caso similar al caso del MAE. Esto refuerza la observación de que Random Forest tiene un rendimiento superior en términos de minimizar errores significativos.

Los coeficientes de determinación  $R^2$  (figura 48) es una medida de qué tan bien se ajustan las predicciones a los valores reales, por tanto, un  $R^2$  cercano a 1 indica un modelo que explica bien la variabilidad de los datos. En nuestro caso, Random Forest vuelve a mostrar un  $R^2$  consistentemente alto, manteniéndose cerca de 0.80. Esto sugiere que el modelo es muy efectivo para capturar la variabilidad de los datos y en hacer predicciones precisas. Los demás modelos de  $R^2$  se estabilizan alrededor de 0.60-0.70, lo cual es notable pero inferior al modelo de Random Forest, esto indica que los modelo capturan menos variabilidad en los datos en comparación con Random Forest.

Según todo esto, se concluye que Random Forest es el más adecuado para el problema en cuestión. Este modelo no solo minimiza los errores absolutos y cuadrados, sino que también captura la mayor parte de variabilidad en los datos, haciendo de sus predicciones más precisas y confiables.

### 7.5.2. Resultados entrenamiento con el tipo de partícula añadido

La comparativa entre los resultados de los distintos modelos en términos de MAE, MSE,  $R^2$ , MSE normalizada y MAE normalizada son los siguientes:

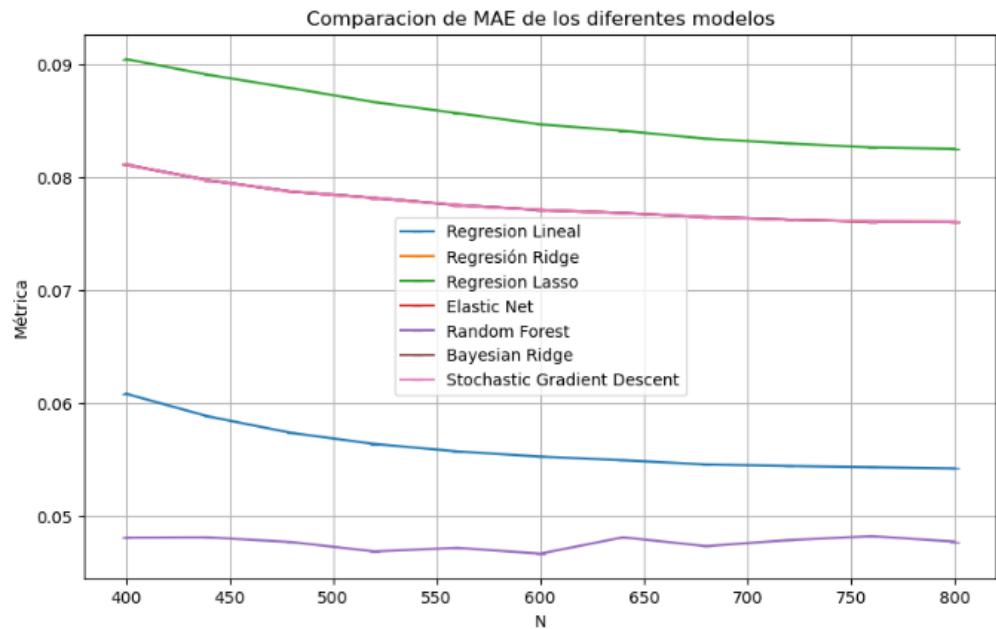


Figura 51: Error Absoluto Medio

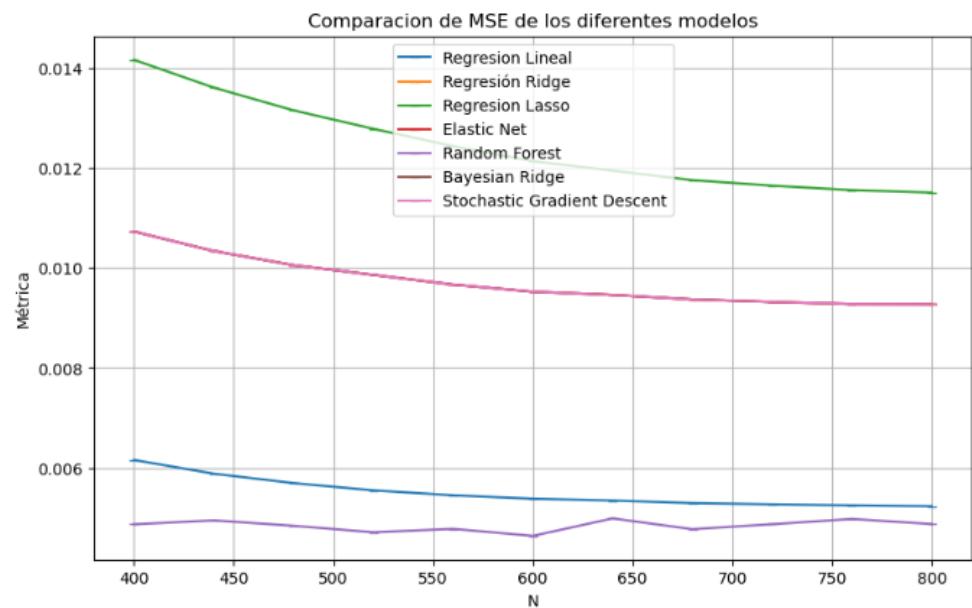


Figura 52: Error Cuadrático Medio

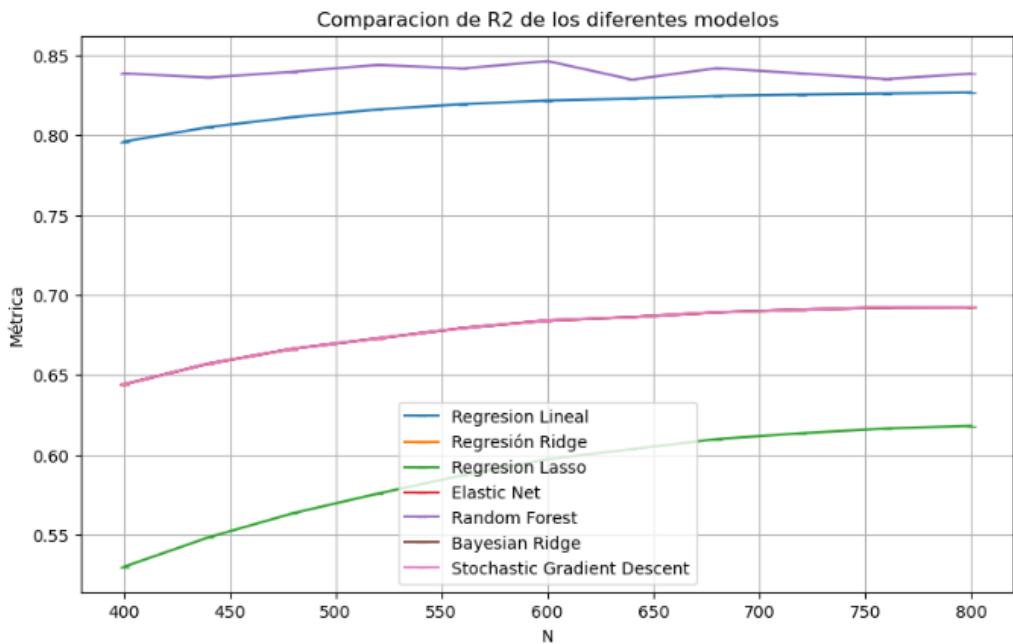


Figura 53: Coeficiente de Determinación

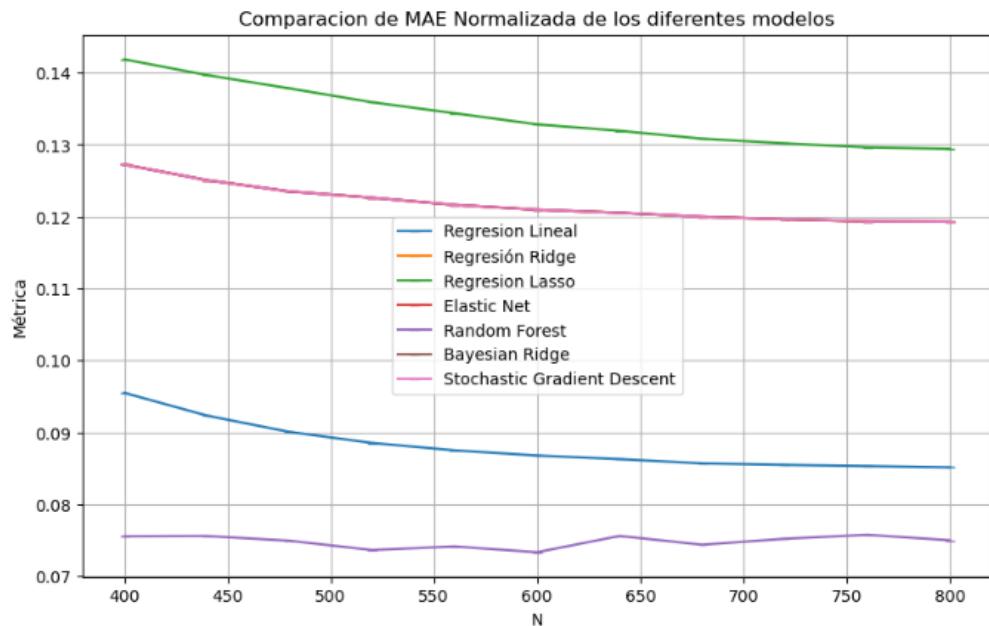


Figura 54: Error Absoluto Medio Normalizado

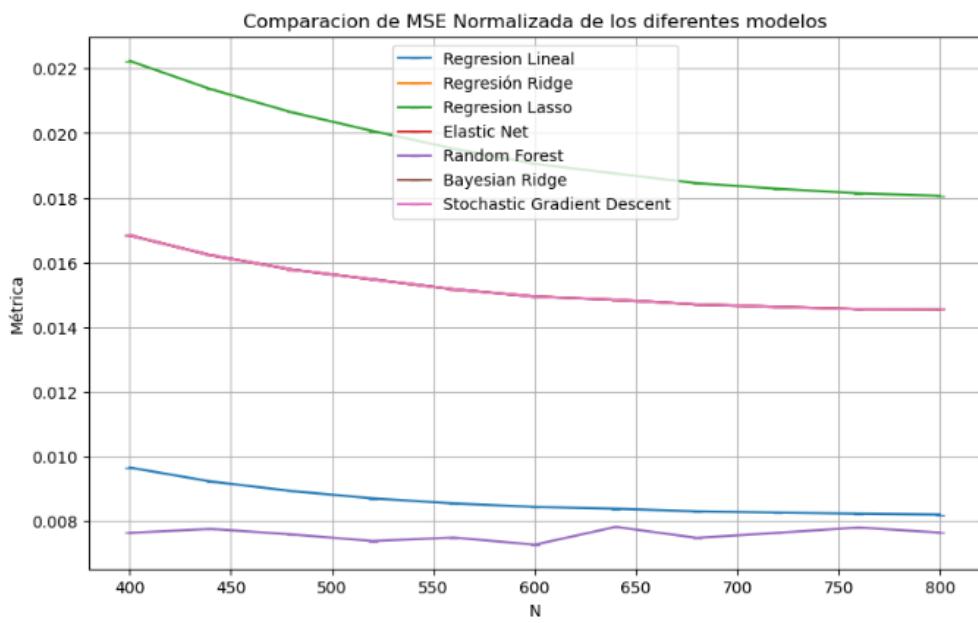


Figura 55: Error Cuadrático Medio Normalizado

Para comparar las gráficas de los diferentes modelos, podemos observar el rendimiento en términos de MAE, MAE normalizado, MSE, MSE normalizado y  $R^2$ , a medida que aumenta el tamaño de la N.

Para MAE y MAE normalizado (figuras 51 y 54), Random Forest tiene el valor más bajo, indicando un mejor rendimiento entre los modelos representados. Los otros modelos, tienen un MAE y MAE normalizado más alto en comparación con Random Forest.

Para MSE y MSE normalizado (figuras 52 y 55), Random Forest sigue liderando, ya que tienen el valor más bajo, lo que destaca su capacidad para minimizar los errores al cuadrado y con esto confirma su eficacia. Los demás modelo presentan un rendimiento inferior.

En cuanto a  $R^2$  (figura 53), Random Forest tiene el valor de  $R^2$  más alto, cercano a 0.85, lo que indica que explica mejor la variabilidad de los datos. Los otros modelos tienen valores de  $R^2$  más bajos.

En resumen, Random Forest es consistentemente el mejor modelo en todas las métricas, mostrando el menor error y la mayor capacidad para explicar la variabilidad de los datos en comparación con los otros modelos evaluados.

### **7.5.3. Comparación de Rendimiento del Modelo Random Forest con Diferentes Conjuntos de Entrada**

Como hemos visto en los dos apartados anteriores, el modelo Random Forest es el que mejores resultados da en cuanto a la regresión de la energía de entradas. Por tanto, vamos a comparar el entrenamiento del mismo modelo con las dos entradas distintas de datos.

A continuación veremos la comparativa de los entrenamientos de los datos:

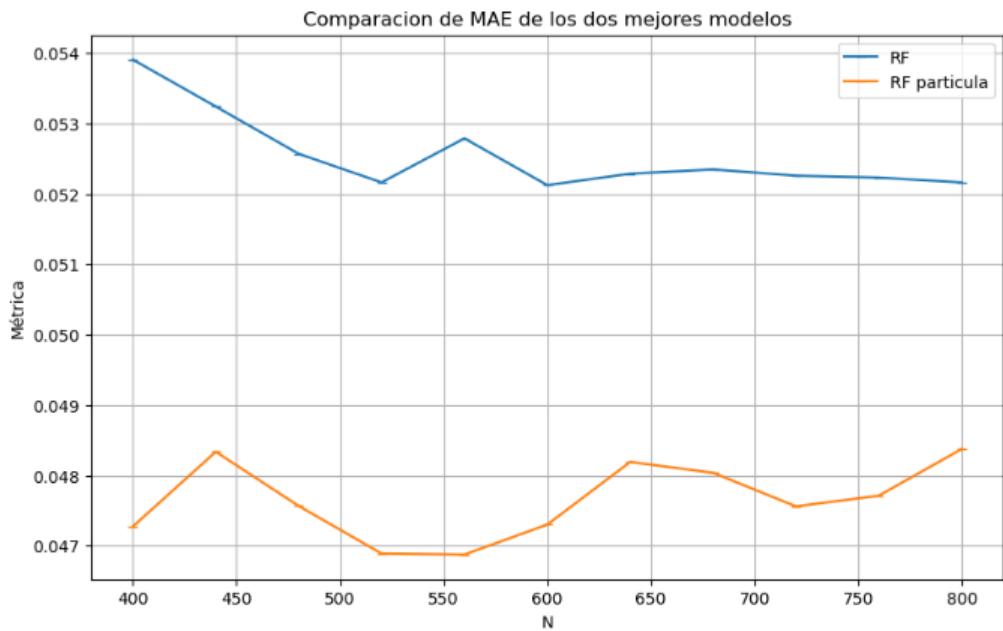


Figura 56: Comparativa para Random Forest del Error Absoluto Medio

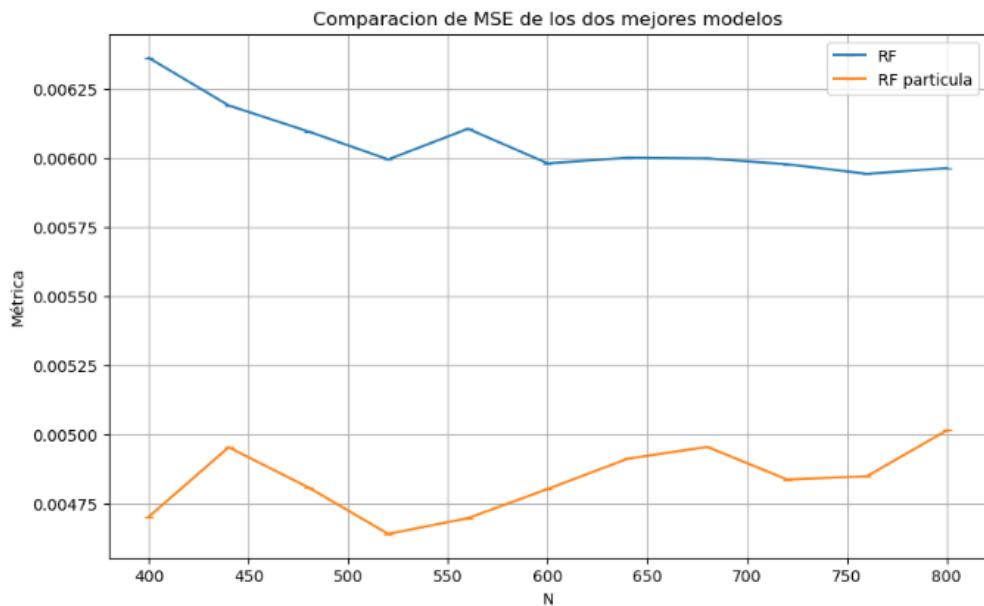


Figura 57: Comparativa para Random Forest del Error Cuadrático Medio

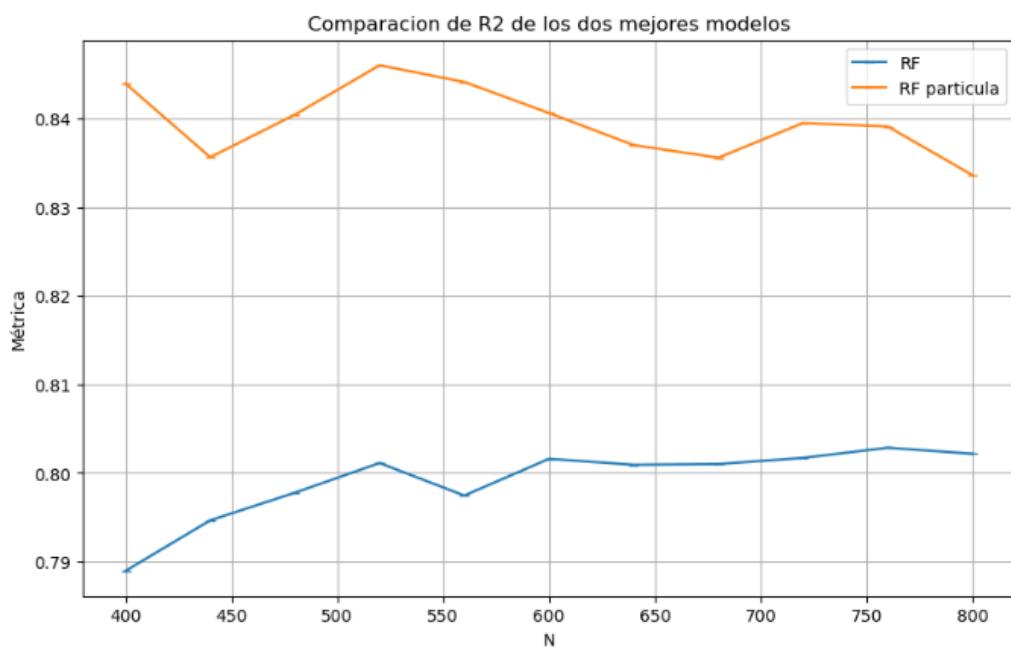


Figura 58: Comparativa para Random Forest del Coeficiente de Determinación

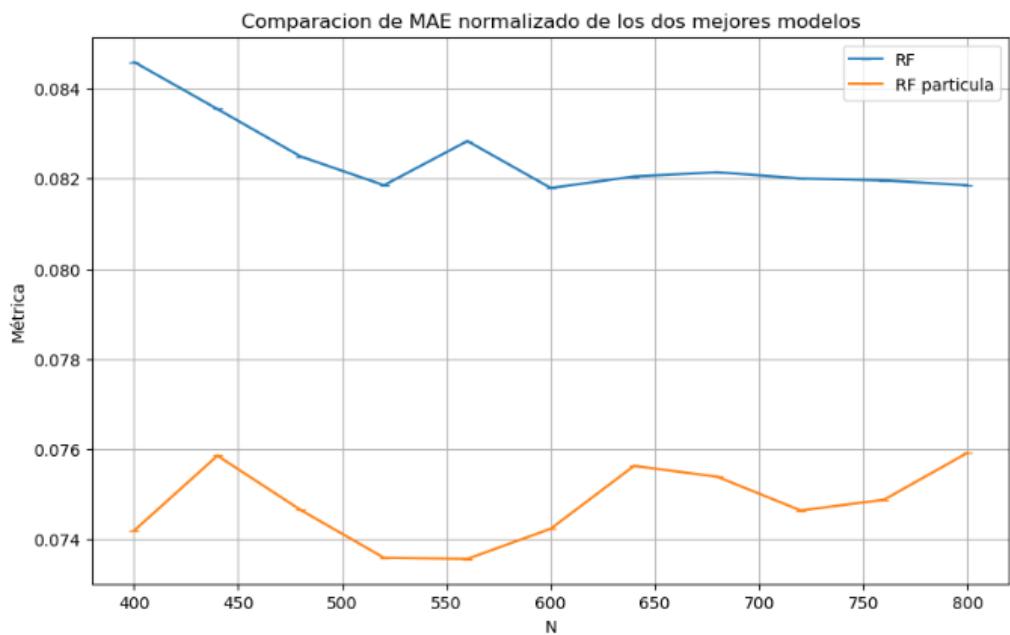


Figura 59: Comparativa para Random Forest del Error Absoluto Medio Normalizado

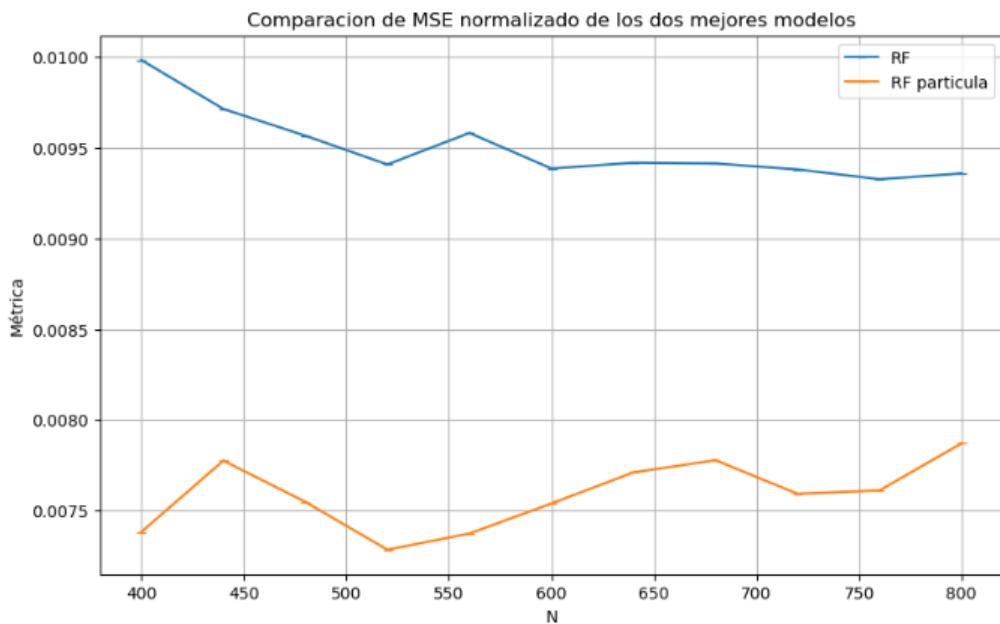


Figura 60: Comparativa para Random Forest del Error Cuadrático Medio Normalizado

Comparando el MSE normalizado (figura 60) vemos como para la entrada en la que le incluimos el tipo de partícula es consistentemente más bajo que la entrada que no le indicamos el tipo de partícula. Esto sugiere que la adición de la información del tipo de partícula mejora la capacidad del modelo para minimizar los errores al cuadrado, reflejando una mayor precisión en la predicción. En el análisis del MSE (figura 57) pasa igual, lo que refuerza la conclusión de que el modelo es más efectivo al predecir con la información del tipo de partícula, ya que logra minimizar los errores al cuadrado de forma más eficiente.

De manera similar, RF cuando le indicamos el tipo de partícula mantiene un MAE (figura 56) inferior al de RF, corroborando la superioridad del modelo con la entrada adicional. La inclusión del tipo de partícula permite al modelo hacer predicciones más precisas, reduciendo el error absoluto promedio. De igual forma pasa en la comparación de MAE normalizada (figura 59), lo que sugiere que las predicciones del modelo son más cercanas a los valores reales cuando se incluye

esta variable adicional.

En términos de  $R^2$  (figura 58), el modelo que le indicamos el tipo de partícula que es tiene un valor más alto que el otro que tiene la otra entrada. Esto incide que el modelo con la entrada adicional del tipo de partícula explica mejor la variabilidad de los datos, proporcionando una mayor capacidad predictiva y un ajuste más cercano a los datos observados.

Por tanto, la inclusión del tipo de partícula como entrada en el modelo Random Forest mejora significativamente su rendimiento en todas las métricas evaluadas. Esto demuestra que agregar esta variable adicional proporciona un valor predictivo sustancial, resultando en predicciones más precisas y un modelo más robusto.

## 7.6. Entrenamiento del mejor modelo con mejores resultados

En esta sección veremos que N es mejor para el modelo con la entrada de datos indicando el tipo de partícula que es. Los resultados han sido los siguientes:

Métrica	Mínimo	N Mínimo	Máxima	N Máxima	Media	Desviación
MSE	0.00464	520	0.00501	800	0.00483	0.00011
MAE	0.04687	560	0.04837	800	0.04764	0.00051
$R^2$	0.83363	800	0.84603	520	0.83961	0.00378
MSE normalizado	0.00728	520	0.00787	800	0.00758	0.000178
MAE normalizado	0.07356	560	0.07591	800	0.0747	0.00080

Cuadro 14: Resumen de resultados del problema de la regresión de la energía de entrada

Como podemos ver, el valor de N que más se repite como mejor resultado del entrenamiento del mejor modelo con mejores resultados es  $N = 800$ . Por tanto, será el que usaremos en el siguiente apartado para dar los resultados finales.

## 7.7. Gráficas de errores

Ahora veremos gráficamente los errores del modelo abordando la interpretación y la relevancia de cada uno para evaluar el modelo de regresión.

### 7.7.1. Gráfico de Errores Residuales

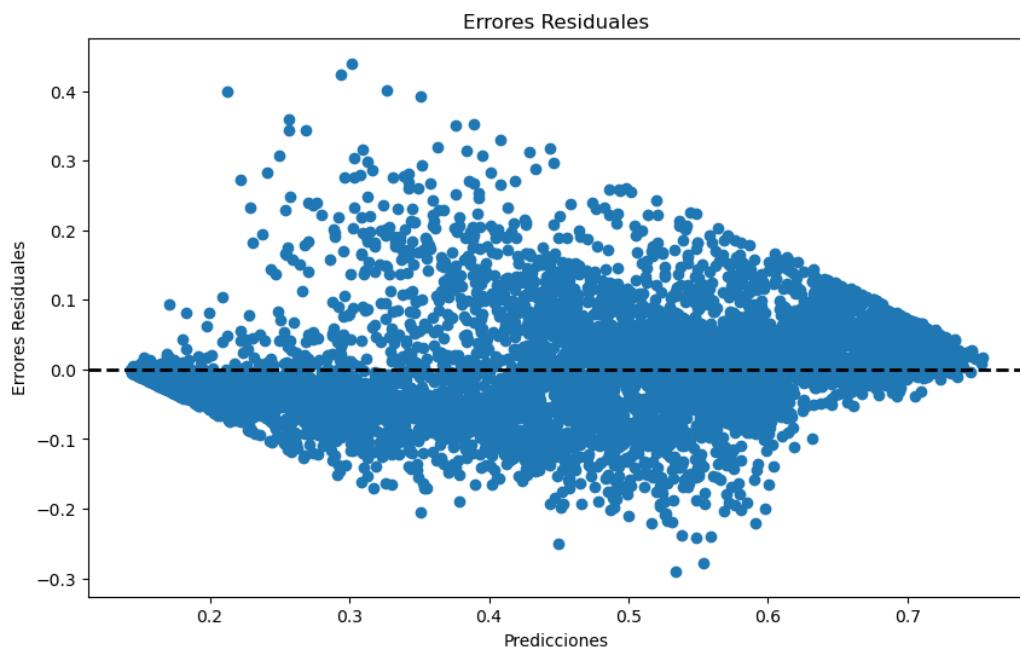


Figura 61: Errores Residuales

En la figura 63 muestra los errores residuales, que son las diferencias entre los valores reales y las predicciones del modelo, en el eje y, y las predicciones del modelo en el eje x. La línea horizontal en el valor 0 representa el punto donde no hay error, es decir, donde la predicción coincide perfectamente con el valor real. En un modelo ideal, los errores residuales deben estar distribuidos aleatoriamente alrededor de esta línea horizontal sin formar patrones. La presencia de cualquier patrón, como una forma de embudo o una curva, puede indicar problemas con la varianza constante de los errores o sugerir que el modelo no está capturando adecuadamente alguna estructura de los datos. En tu gráfico, se observa una cierta dispersión que se ensancha conforme se incrementan las predicciones, lo que podría indicar que el modelo tiene mayor variabilidad en sus errores para ciertos rangos de predicciones.

### 7.7.2. Gráfico de Dispersion de Errores Residuales

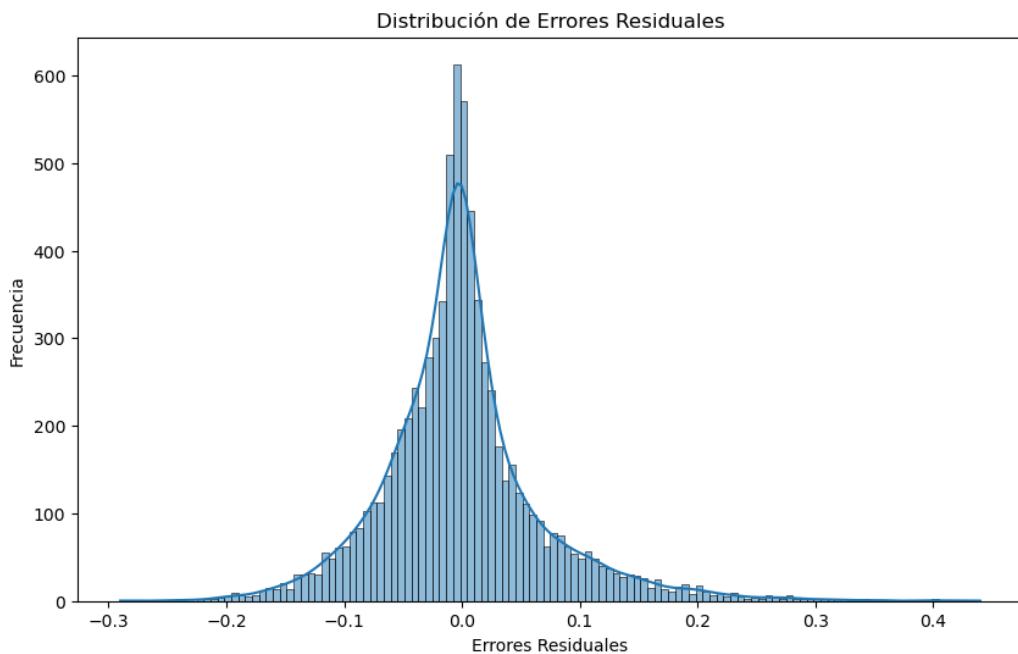


Figura 62: Errores Residuales

El histograma de errores residuales muestra la distribución de estos errores. Idealmente, los errores residuales deben seguir una distribución normal centrada en cero, lo que indicaría que el modelo no tiene sesgo y que los errores son aleatorios. En el histograma proporcionado, se observa una distribución que parece aproximadamente normal, con una gran concentración de errores cerca de cero y una disminución gradual en ambos extremos. Sin embargo, cualquier desviación significativa de la forma normal puede indicar problemas en el modelo, como un sesgo sistemático en las predicciones. El hecho de que la mayoría de los errores residuales se centren en torno a cero es una buena señal, ya que sugiere que las predicciones son generalmente precisas, aunque la distribución de los extremos puede ser analizada más a fondo para entender cualquier posible anomalía.

### 7.7.3. Gráfico de Predicciones contra Valores Reales

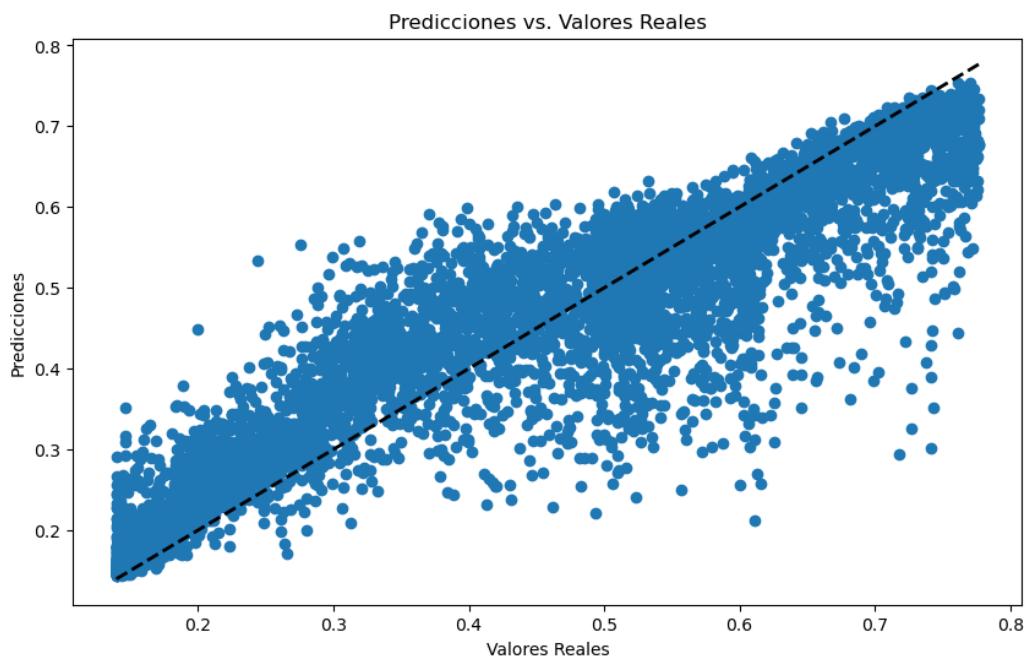


Figura 63: Errores Residuales

El gráfico de dispersión de predicciones contra los valores reales es crucial para evaluar el rendimiento de un modelo de regresión. En este gráfico, los valores reales se colocan en el eje x y las predicciones del modelo en el eje y. La línea diagonal ( $y = x$ ) representa las predicciones perfectas; cualquier punto sobre esta línea indica una predicción exacta. Los puntos dispersos alrededor de la línea diagonal muestran cuán cerca o lejos están las predicciones de los valores reales. Un buen modelo debería tener los puntos cerca de la línea diagonal. En el gráfico proporcionado, los puntos parecen estar bien alineados a lo largo de la línea diagonal, lo que indica que el modelo realiza predicciones razonablemente precisas. No obstante, una mayor dispersión a medida que aumentan los valores puede sugerir que el modelo es menos preciso para valores mayores, lo que podría ser un área para mejorar.

Métrica	Entrenamiento	Validación	Prueba
MSE	0.00106	0.00478	0.00459
MAE	0.02164	0.04726	0.04643
$R^2$	0.96456	0.84135	0.84889
MSE normalizado	0.00167	0.00750	0.00721
MAE normalizado	0.03396	0.07416	0.07292

Cuadro 15: Resultados problema regresión energía de entrada

#### 7.7.4. Conclusión

Los tres gráficos proporcionan una visión integral del rendimiento del modelo de regresión. En general, el modelo parece tener un buen rendimiento con una distribución de errores residuales cercana a cero y predicciones alineadas con los valores reales. Sin embargo, la dispersión de los errores residuales y su mayor variabilidad en ciertos rangos de predicciones destacan áreas donde el modelo podría mejorar. Ajustes adicionales y refinamientos en el modelo podrían ayudar a reducir estos errores y mejorar la precisión general, especialmente para predicciones en los extremos del rango de datos.

### 7.8. Resultado final

Como parte final del entrenamiento del modelo, es crucial evaluar su comportamiento utilizando diversas métricas de rendimiento, tales como el Error Cuadrático Medio (MSE), el Error Cuadrático Medio Normalizado (MSE Normalizado), el Error Absoluto Medio (MAE), el Error Absoluto Medio Normalizado (MAE Normalizado) y el Coeficiente de Determinación ( $R^2$ ). Estas métricas proporcionan una visión completa de la precisión y la capacidad predictiva del modelo.

Para llevar a cabo esta evaluación, analizaremos los resultados obtenidos en los conjuntos de datos de entrenamiento, validación y test. Cada uno de estos conjuntos desempeña un papel fundamental en la validación del modelo y en la evaluación de su capacidad de generalización.

El modelo de Random Forest entrenado ha demostrado un rendimiento sólido en todas las fases del proceso. Los valores de MSE y MAE bajos indican que el

modelo comete pocos errores en sus predicciones, mientras que los altos valores de  $R^2$  confirman que el modelo explica bien la variabilidad de los datos.

No hay indicios claros de sobreajuste de los datos, ya que el rendimiento del modelo se mantiene consistentemente en los datos de validación y prueba, aunque ligeramente inferior al conjunto de entrenamiento, lo cual es esperado y aceptable en la mayoría de los casos. Esto sugiere que el modelo tiene una buena capacidad de generalización y puede predecir con precisión en datos no vistos previamente

## 8. Análisis de los Resultados

### 8.1. Etapa de Clasificación

En las secciones anteriores se han visto distintos enfoques para abordar el problema de clasificación propuesto. Uno de ellos trata de resolverlo mediante modelos tradicionales de Aprendizaje Automático, y otro más avanzado que recurre a la aplicación de arquitecturas de Redes Neuronales. En la figura 28 y en la tabla 8 podemos ver los resultados obtenidos por el mejor modelo de clasificación dentro de todos los modelos de Aprendizaje Automático tradicionales. En la figura 44 y en la tabla 12 podemos ver los resultados que hemos obtenido por la mejor arquitectura de Redes Neuronales dentro de todas las arquitecturas.

La decisión de llevar un modelo a producción en un problema de clasificación se fundamenta en la evaluación exhaustiva de su desempeño en múltiples métricas y conjuntos de datos. En este caso, se compararon dos modelos basados en su accuracy y por categorías específicas (bajas energías, medias energías y altas energías) en los conjuntos de datos de entrenamiento, validación y prueba. Como hemos podido observar en las figuras y las tablas mencionadas anteriormente, se ha obtenido muy buenos resultados en el propósito de la clasificación binaria de la traza de las dos distintas partículas, por encima de 0.90 en accuracy con los dos modelos.

Tras la comparación de ambos modelos, se opta por llevar a producción el

modelo implementado mediante un enfoque de Aprendizaje Automático tradicional, ya que el modelo de Redes Neuronales ha sido ejecutado en un servidor de Google Colab con una tarjeta gráfica más potente que el ordenador local y para el despliegue no podemos contar con este modelo.

En bajas energías, el modelo de Aprendizaje Automático tuvo una ligera ventaja en el conjunto de entrenamiento, y aunque el modelo de Redes Neuronales lo superó en los conjuntos de validación y prueba, la necesidad de recursos de hardware especializados limita su implementación. El modelo de Aprendizaje Automático también tuvo un rendimiento consistente en estos subconjuntos, lo cual es crucial para asegurar un desempeño robusto en diferentes escenarios de energía. Además, la consistencia del modelo elegido es un indicador fuerte de su fiabilidad y estabilidad. Esto es esencial para aplicaciones en producción donde la variabilidad en el rendimiento puede llevar a resultados impredecibles y potencialmente dañinos.

En conclusión, se ha decidido llevar a producción el modelo de Aprendizaje Automático tradicional debido a su menor dependencia de recursos computacionales avanzados, asegurando así una implementación más práctica y sostenible.

## 8.2. Etapa de Regresión

En esta sección, se presentan y analizan los resultados de la etapa de regresión de la energía de entrada que predice el modelo, evaluados a través de múltiples métricas en los conjuntos de datos de entrenamiento, validación y prueba. Las métricas consideradas incluyen el Error Cuadrático Medio (MSE), el Error Cuadrático Medio Normalizado (MSE normalizado), el Error Absoluto Medio (MAE), el Error Absoluto Medio Normalizado (MAE Normalizado) y el Coeficiente de Determinación ( $R^2$ ). Los resultados finales se pueden ver en la tabla 15.

En el conjunto de entrenamiento, el modelo presenta un MSE y un MAE bajo, lo que indica que los errores al cuadrado y el error absoluto medio son bajos lo

que nos hace confirmar que tiene una buena precisión en las predicciones realizadas por el modelo. El coeficiente de determinación es bueno, ya que indica que el modelo es capaz de explicar una gran parte de la variabilidad de los datos de entrenamiento.

En el conjunto de validación, el modelo presenta un MSE y un MAE algo más grande que en el conjunto de entrenamiento, pero aún así sigue siendo bajo, lo que indica una mayor variabilidad en los errores al cuadrado cuando se enfrentan a datos no vistos previamente. Estos valores de error todavía son aceptables por el modelo. El  $R^2$  es menor que en el de entrenamiento, pero aún muestra una buena capacidad de generalización.

En el conjunto de prueba, el MSE y el MAE son similares al conjunto de validación, lo que sugiere una buena generalización y una capacidad del modelo para predecir datos nuevos, reafirmando la precisión del modelo en datos no vistos. El  $R^2$  indica que el modelo una buena cantidad de los datos de prueba, lo que es consistente con el resultado de validación y muestra buena capacidad de predicción.

El análisis de estas métricas a través de los diferentes conjuntos de datos revela que el modelo de regresión ha sido capaz de aprender adecuadamente de los datos de entrenamiento, generalizar bien a los datos de validación y mantener un rendimiento consistente en el conjunto de prueba. Los resultados obtenidos, especialmente en el conjunto de prueba, son críticos para determinar la aplicabilidad práctica del modelo en escenarios del mundo real, asegurando su robustez y precisión en diversas situaciones y conjuntos de datos.

En conclusión, se ha decidido llevar a producción el modelo de Random Forest indicándole en la entrada de datos el tipo de partícula a la que corresponde, previamente siendo predecida por el modelo de clasificación, ya que como hemos visto ha dado buenos resultados.

## 9. Despliegue del Modelo

En este TFG se llevará a cabo una etapa de despliegue del modelo para poder simular un entorno real. Este despliegue al no disponer de un servidor público, se realizará en local. Para ello usaremos el framework de FastAPI [34] para Python.

FastAPI está basado en Starlette y Pydantic, lo que permite manejar muchas solicitudes por segundo con baja latencia. Nos proporciona una validación automática de datos usando Pydantic, asegurando que las entradas y salidas sean siempre del tipo esperado. Nos ofrece una generación automática de documentación y soporta operaciones asíncronas de forma nativa, lo que permite realizar múltiples operaciones simultáneamente, mejorando la eficiencia del despliegue.

### 9.1. Montaje de FastAPI

Para descargar todas la dependencias de FastAPI escribiremos en el terminal de Linux: *pip install fastapi uvicorn*. Para ejecutar la API localmente, se utilizará *Uvicorn* [43]. Desde la línea de comandos, se puede iniciar el servidor con el siguiente comando *uvicorn main:app --reload*.

Una vez esté el servicio corriendo se podrá interactuar con la API en *http://127.0.0.1:8000* y la documentación interactiva estará en *http://127.0.0.1:8000/docs*

### 9.2. Script de Python

#### 9.2.1. Main

En el scrip principal implementa una API web utilizando el framework FastAPI para la clasificación del tipo de partícula y la regresión de la energía de entrada de la partícula. La API permite subir archivos CSV que contienen los datos necesarios para ser procesados por los modelos de Aprendizaje Automático previamente entrenados.

La API dispone de dos funcionalidades principales:

- **Clasificación de partículas:** Mediante de un modelo de clasificación XG-Boost, se predice si una partícula es un kaon o un pión.
- **Clasificación de partículas y regresión de la energía de entrada:** Utiliza el clasificador para predecir el tipo de partícula y después utiliza esa predicción y otro conjunto de datos para predecir la energía de entrada basada en las características proporcionadas por el archivo CSV.

La aplicación proporciona un punto de entrada que da la bienvenida a los usuarios y dos endpoints para procesar los archivos subidos, retornando las predicciones generadas por los modelos.

### **9.2.2. Script para reentrenar los modelos**

El reentrenamiento de los modelos en producción es una práctica esencial para mantener y mejorar el rendimiento de los sistemas de aprendizaje automático, ya que los datos pueden evolucionar con el tiempo, y el modelo necesita adaptarse a nuevas tendencias y patrones para mantenerse preciso. También, el incorporar nuevos datos puede ayudar a mejorar la precisión y robustez del modelo, ya que, pueden ayudar a reducir sesgos que el modelo original pudiera haber tenido.

La frecuencia del reentrenamiento, en nuestro caso será de forma manual, es decir, el administrador entrenará el modelo y lo guardará en un archivo para después leer ese modelo para predecir. Es importante conocer que este reentrenamiento se puede configurar para que sea haga automáticamente.

### **9.2.3. Script para la clasificación de las partículas**

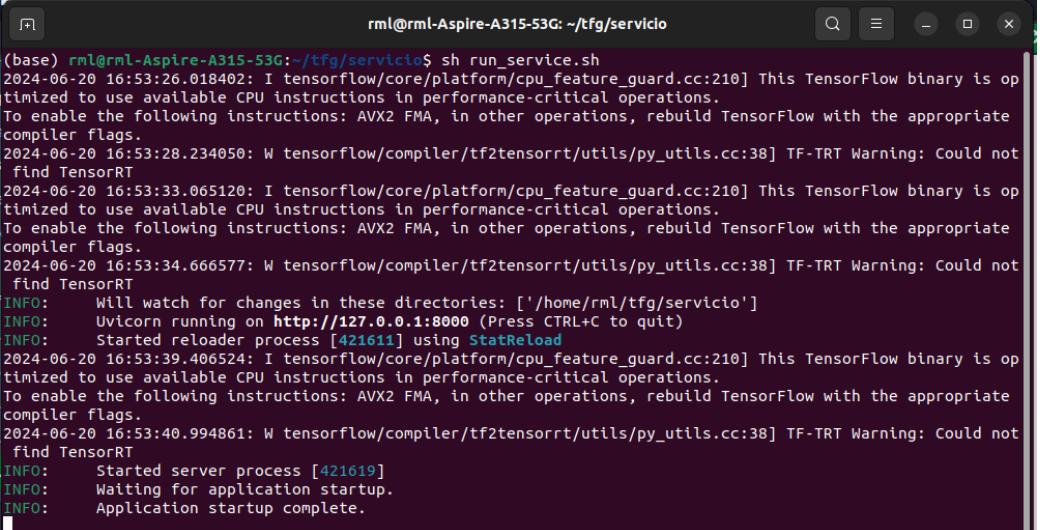
Este script desarrollará las funciones necesarias para cuando el usuario haga una llamada al servicio de clasificación, desde el *main* se llame a la función de clasificación. Esta será la encargada de leer el modelo que está escrito en un archivo y predecir el tipo de partícula según los datos de entrada.

### 9.2.4. Script para la regresión de la energía de entrada

Este script desarrollará las funciones necesarias para cuando el usuario haga una llamada al servicio de regresión, desde el *main* se llame a la función de regresión. Esta se encargará de leer el modelo que está escrito en un archivo y predecir el valor de energía de entrada de la partícula medida en GeV.

## 9.3. La API de FastAPI

Dentro de la API tendremos los tres módulos diseñados en el apartado anterior. A continuación, veremos el uso y aplicación dentro de la API, para ello, tendremos que ejecutar nuestro script *run\_service.sh* con el comando *sh*, es decir, escribir en nuestro terminal *sh run\_service.sh*.



```
rml@rml-Aspire-A315-53G:~/tfg/servicio$ sh run_service.sh
2024-06-20 16:53:26.018402: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-06-20 16:53:28.234050: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2024-06-20 16:53:33.065120: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-06-20 16:53:34.666577: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
INFO:     Will watch for changes in these directories: ['/home/rml/tfg/servicio']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [421611] using StatReload
2024-06-20 16:53:39.406524: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-06-20 16:53:40.994861: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
INFO:     Started server process [421619]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Figura 64: Servicio cargado satisfactoriamente

Como vemos en la figura 64 el servicio se ha cargado satisfactoriamente, ahora abriremos un navegador y escribiremos la siguiente dirección: <http://127.0.0.1:8000/docs>. Con esto abrimos la interfaz de usuario generada por FastAPI, conocida como la documentación interactiva de OpenAPI. Esto permite a los usuarios explorar y probar los endpoints de la API de una manera interactiva y visual. A continuación veremos una imagen de la interfaz de usuario:

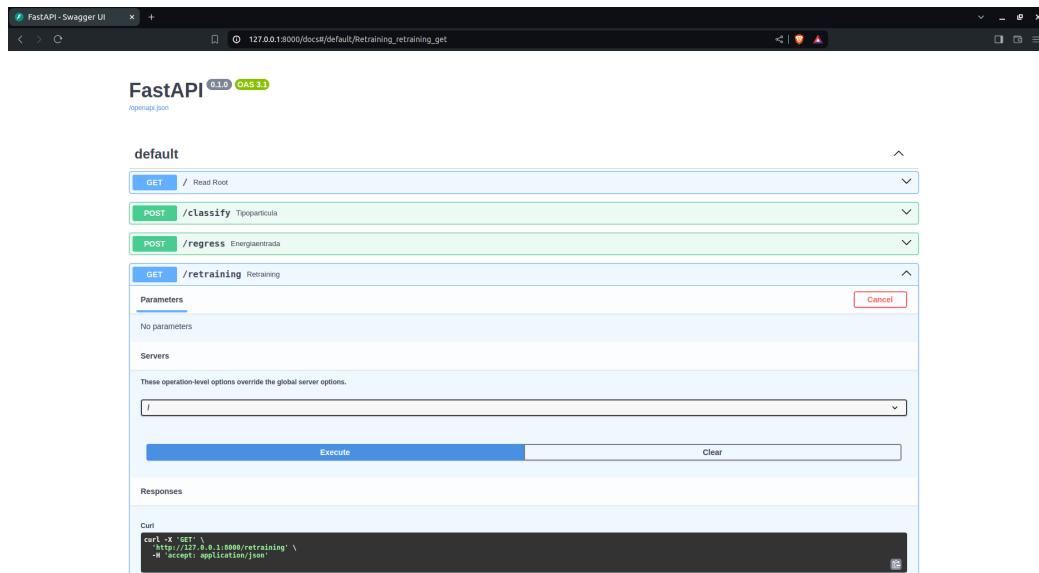


Figura 65: Interfaz de usuario de FastAPI

Como podemos ver en la figura 65 viene los 3 end-points diseñados. En las siguientes secciones detallaremos el funcionamiento de cada uno de ellos.

### 9.3.1. Clasificación

El end-point para la clasificación está en la ruta `/classify` donde conoceremos el tipo de partícula. Pulsamos y se abrirá lo siguiente:

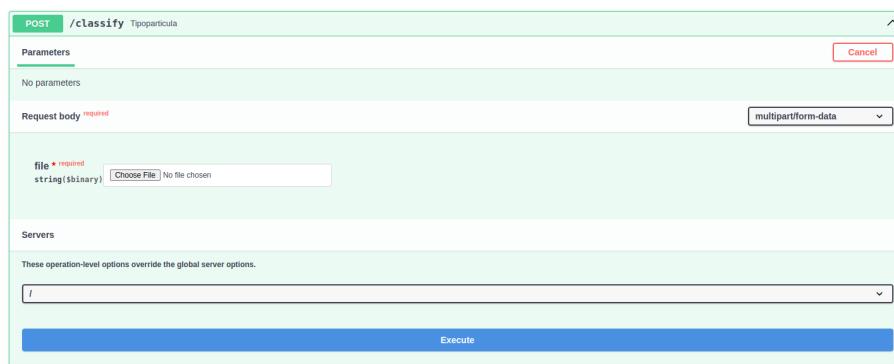


Figura 66: End-Point para la clasificación

Haga clic en el botón *Choose File* para abrir el explorador de archivos. Luego, seleccione el archivo que contiene los 'hits' del evento que desea analizar. Esto permitirá a la aplicación determinar si el evento corresponde a un kaon o un pion. A continuación mostraremos un ejemplo:

The screenshot shows a 'Responses' section from a Swagger UI interface. It includes a 'Curl' block with the following command:

```
curl -X 'POST' \
'http://127.0.0.1:8000/classify' \
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'file=@filtered_kaons1_prueba.csv;type=text/csv'
```

A 'Request URL' block shows the endpoint: `http://127.0.0.1:8000/classify`. Below it, a 'Server response' block shows a 200 status code. The 'Details' tab is selected, displaying the 'Response body' as a JSON object:

```
{ "predictions": [ "It's a pion" ] }
```

There are 'Copy' and 'Download' buttons next to the response body. The 'Response headers' section lists:

```
content-length: 31
content-type: application/json
date: Thu, 20 Jun 2024 15:17:09 GMT
server: uvicorn
```

Figura 67: Ejemplo de clasificación de evento

Como podemos ver en la figura 67 vemos como nuestro clasificador ha predicho un pion.

### 9.3.2. Regresión

El end-point para la regresión está en la ruta `/regress` donde conoceremos el tipo de partícula y la energía de entrada con la que lo hace. Pulsaremos y se abrirá lo siguiente:

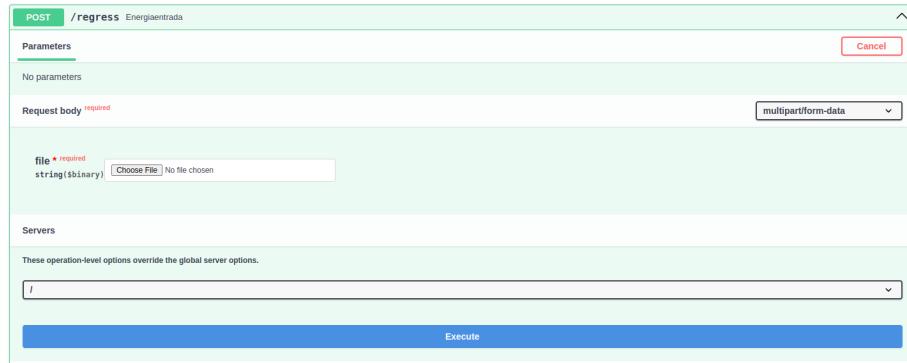


Figura 68: End-Point para la regresión

De forma similar a el end-point de clasificación, haremos clic en el botón *Choose File* para abrir el explorador de archivos. Luego, seleccionaremos el archivo que contiene los 'hits' del evento que desea analizar. Esto permitirá a la aplicación determinar si el evento corresponde a un kaon o un pion. A continuación mostraremos un ejemplo:



Figura 69: Ejemplo de regresión de evento

Como podemos ver en la figura 69 ha predicho un pion lo hace con 0.32405116 GeV de energía de entrada.

### 9.3.3. Reentrenamiento de los modelos

Con el fin de mantener y mejorar el rendimiento de los sistemas de Aprendizaje Automático, ya que los datos pueden evolucionar con el tiempo y los modelos pueden degradar. En nuestro caso se hará de forma manuela, es decir, con el endpoint diseñado. El end-point para el reentrenamiento de los modelos está en la ruta `/retraining`. Pulsaremos y se abrirá lo siguiente:

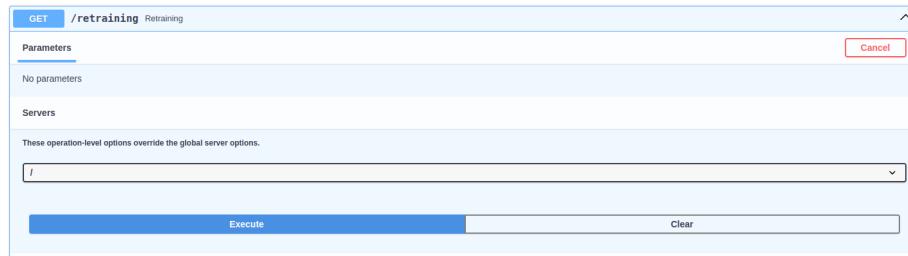


Figura 70: End-Point para el reentrenamiento de los modelos

Para reentrenar los modelos haremos clic en el botón *Execute* y se reentrenará los modelos. Cuando haya finalizado mostrará el siguiente texto por la casilla de salida *Response body*.

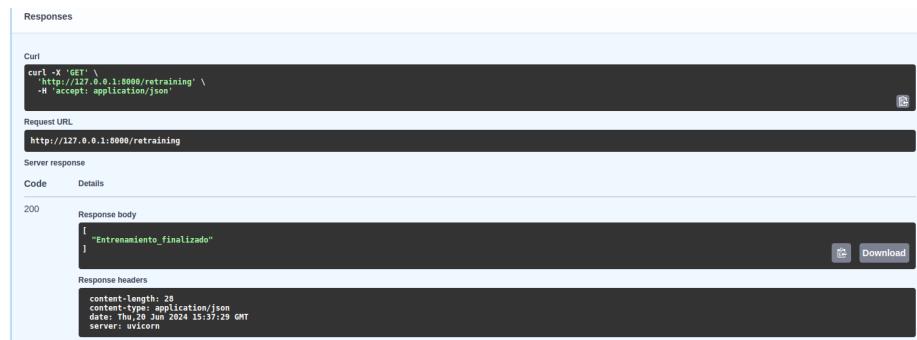


Figura 71: Ejemplo de reentrenamiento de los modelos

Como podemos ver en la figura 71 se han reentrenado los dos modelos correctamente.

## 10. Presupuesto

A modo de información, veremos lo que supone el coste de este trabajo realizado. Para llevar a cabo este trabajo es necesario de un equipo con GPU y que cuente con la potencia suficiente para realizar las pruebas, así como tener a una persona realizando el estudio.

El tiempo dedicado a la realización de este trabajo han sido unas 300 horas. Es importante indicar que esto es una aproximación. En cuanto al equipo empleado se va hacer la estimación de precio usando servicios en la nube que nos permite crear máquinas virtuales con las especificaciones indicadas concretamente.

También hay que indicar las horas de trabajo de los dos tutores. Se ha estimado que el tutor Alberto Guillén ha dedicado aproximadamente 10 horas de trabajo. Estas horas incluyen diversas sesiones de tutoría, así como la búsqueda de información y la recomendación de artículos relevantes para complementar y enriquecer el contenido de este TFG. La orientación proporcionada ha sido fundamental para el desarrollo y finalización exitosa de este proyecto.

Se han estimado también las horas de trabajo de Bruno Zamorano que ha dedicado aproximadamente 8 horas de trabajo. Estas horas incluyen las diferentes sesiones de tutoría, la búsqueda de información y la generación de los datos en el simulador. Su orientación ha sido muy útil para el desarrollo y finalización exitosa de este proyecto.

### Cost Estimate Summary

As of 30 may 2024 • 11:56

Prices in USD

COMPUTE	56,00 \$
Cloud GPU	56,00 \$
Service type	Cloud GPU
GPU-time	100 Hours
GPU Model	NVIDIA L4
Region	Iowa (us-central1)
Number of GPUs	0.1
Total estimated cost	56,00 \$ / mo

Figura 72: Coste máquina virtual con GPU

Según la figura 72 el coste son 56 dólares al mes, para la realización de este TFG, con el plazo de un mes sería suficiente para realizar todas las pruebas oportunas. Estos 56 dólares son 51,77€.

También será necesario un equipo con CPU para ejecutar el resto de algoritmos y desplegar el modelo. En mi caso se ha realizado en mi equipo personal que tiene las siguientes características:

- **Memoria RAM:** 16 GB
- **Procesador:** Intel Core i5
- **Almacenamiento:** 1TB SSD

Este equipo tiene un coste de 479.00€.

Cantidad	Concepto	Importe	Total
10	Horas Alberto	52 €	520 €
8	Horas Bruno	52 €	416 €
300	Horas Rubén	15.62 €	4687.5 €
1	Máquina con GPU	51.77 €	51.77 €
1	Equipo con CPU	479.00 €	479.00 €
<b>Subtotal</b>		<b>6154.27 €</b>	
<b>I.V.A.</b>		<b>21 %</b>	<b>1292.40 €</b>
<b>Total</b>			<b>7446.67 €</b>

Cuadro 16: Presupuesto del proyecto

Las licencias de fastAPI, Scikit-Learn y TensorFlow son licencias de código abierto por tanto estas licencias son gratuitas.

## 11. Conclusiones

Este proyecto ha abordado la problemática de la identificación y clasificación de partículas subatómicas, específicamente piones y kaones, utilizando técnicas de Aprendizaje Automático, comenzando con métodos tradicionales y luego empleando redes neuronales. A lo largo del desarrollo del Trabajo Fin de Grado (TFG), se han implementado y evaluado múltiples modelos y enfoques con el objetivo de optimizar la precisión y la capacidad predictiva en la tarea propuesta.

En la etapa de clasificación, se compararon modelos tradicionales de aprendizaje automático con arquitecturas avanzadas de redes neuronales. Los resultados mostraron que, aunque los modelos tradicionales proporcionan una precisión aceptable, las redes neuronales, especialmente las convolucionales, tienden a superar en términos de precisión y capacidad de generalización. Esto se debe a su habilidad para capturar características complejas y patrones dentro de los datos.

La fase de regresión se centró en predecir la energía de entrada de las partículas. Se utilizó el modelo de Random Forest debido a su robustez y consistencia en los resultados obtenidos. Las métricas de evaluación, incluyendo el MSE, MAE y el coeficiente de determinación ( $R^2$ ), indicaron un buen rendimiento del modelo,

tanto en los conjuntos de entrenamiento como en los de validación y prueba. Esto sugiere que el modelo es capaz de generalizar adecuadamente y proporcionar predicciones precisas en datos no vistos previamente.

Además, se destacó la importancia de los recursos computacionales en el entrenamiento de modelos avanzados. Mientras que las redes neuronales requieren hardware especializado para alcanzar su máximo potencial, los modelos tradicionales de aprendizaje automático ofrecen una solución más práctica y sostenible para su implementación en producción, especialmente cuando se dispone de recursos limitados.

El despliegue del modelo se realizó utilizando FastAPI, lo que permitió crear un servicio eficiente y accesible. La implementación de scripts en Python para el reentrenamiento y clasificación de partículas aseguró que el sistema fuese adaptable y capaz de mejorar continuamente con nuevos datos.

Finalmente, la colaboración y dedicación de los tutores fueron fundamentales para el éxito de este TFG. La orientación proporcionada, así como la búsqueda y recomendación de artículos relevantes, enriquecieron significativamente el contenido y la calidad del trabajo realizado.

En resumen, este TFG no solo logró implementar soluciones efectivas para la clasificación y regresión de partículas subatómicas, sino que también demostró la viabilidad y eficiencia de combinar técnicas tradicionales y avanzadas de aprendizaje automático en aplicaciones prácticas. Los resultados obtenidos proporcionan una base sólida para futuras investigaciones y desarrollos en el campo de la física de partículas y el aprendizaje automático.

## **11.1. Trabajos Futuros**

En el transcurso de este Trabajo de Fin de Grado se han alcanzado los objetivos principales. Estos objetivos incluyen la clasificación binaria de la traza de kaones y piones mediante técnicas de Aprendizaje Automático Tradicionales y Técnicas

de Redes Neuronales, donde el coste computacional es considerablemente mayor. Como objetivos adicionales, nos hemos centrado en resolver un problema de regresión para predecir la energía de entrada de la partícula, cuya energía no se puede usar para entrenar los distintos modelos de clasificación ya que es un valor desconocido en las mediciones reales. Otro objetivo adicional ha sido diseñar una API para que estos modelos puedan ser desplegados en un servidor y servir a la comunidad científica que trabaja con este tipo de cuestiones físicas. Sin embargo, como en toda investigación, se han identificado ciertos límites y áreas que requieren una exploración adicional para consolidar y expandir los hallazgos presentados.

La sección de trabajos futuros tiene como objetivo delinear las posibles direcciones que podrían seguirse para superar las limitaciones encontradas y profundizar en los aspectos que han surgido como relevantes durante el desarrollo de este estudio. Al considerar estos posibles caminos de investigación, se pretende fortalecer las bases teóricas y prácticas del trabajo realizado.

Una de las limitaciones que encontramos dentro de los objetivos adicionales fue al diseñar la API con FastAPI. El mejor clasificador que obtuvimos fue el de Redes Neuronales Convolucionales. Al usar un servidor con una tarjeta gráfica más potente que la de mi ordenador local y con los servicios de CUDA instalados, no pude montar este clasificador en la API, ya que la API fue montada en mi ordenador local y tuvimos que conformarnos con el modelo de XGBoost, que había dado buenos resultados, pero no tan buenos como las Redes Neuronales Convolucionales.

Otra de las limitaciones que encontramos fue montar este servicio en un servidor público para que pueda ser usado por la comunidad científica. Para montar este servicio sería conveniente desarrollar un frontend que sea más intuitivo para los usuarios. Para esto podríamos usar Gradio, una herramienta de código abierto diseñada para crear interfaces web interactivas de manera rápida y sencilla.

Otro trabajo futuro puede ser ajustar mejor los modelos de regresión. Aunque

los resultados obtenidos estaban dentro de lo esperado, considero que se pueden afinar un poco más las soluciones. Para ello, sería conveniente volver a hacer un ajuste con Optuna pero con más parámetros. Otro enfoque podría ser intentar buscar una solución con modelos de Redes Neuronales, aunque esto puede aumentar el coste computacional y, si el modelo fuera necesario desplegarlo en un servidor, necesitaríamos una GPU para realizar las predicciones.