



ugr | Universidad
de Granada

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

Aprendizaje Automático en Física de Partículas

Optimizando el Análisis de Datos Experimentales 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

Resumen

La identificación de partículas subatómicas es esencial en la física de partículas. Este estudio se enfoca en la clasificación de las trazas de piones y kaones generadas en una cámara de proyección temporal de argón líquido. Se exploran dos enfoques principales para la identificación de estas trazas: uno utilizando métodos tradicionales de aprendizaje automático y otro basado en redes neuronales. Los resultados obtenidos muestran que, las técnicas tradicionales de Aprendizaje Automático proporcionan una buena precisión, las redes neuronales superan significativamente en términos de precisión. Las redes neuronales convolucionales demostraron ser especialmente efectivas para la tarea de clasificación debido a su alta capacidad para capturar características complejas de las trazas. Adicionalmente, se incluye un análisis de regresión para predecir la energía de entrada de las partículas.

Abstract

The identification of subatomic particles is a fundamental aspect of particle physics. This study examines the classification of pion and kaon traces generated in a liquid argon time projection chamber. Two primary approaches for the identification of these traces are investigated: one utilising traditional machine learning methods and the other based on neural networks. The outcomes demonstrate that, while traditional machine learning techniques offer satisfactory accuracy, neural networks significantly outperform in terms of accuracy. Convolutional neural networks were found to be particularly effective for the classification task due to their high ability to capture complex features of the traces. Additionally, a regression analysis was included to predict the input energies of the particles.

Índice	0
--------	---

Índice

Índice	0
Índice de cuadros	0
Índice de figuras	1
1. Introducción	1
1.1. Simulación de los datos	5
2. Descripción de los Datos	6
2.1. Correlación de los datos	7
2.2. Distribución de los datos	10
2.2.1. Variable: <i>hitX</i>	11
2.2.2. Variable: <i>hitY</i>	12
2.2.3. Variable: <i>hitZ</i>	13
2.2.4. Variable: <i>hitInteg</i>	14
2.2.5. Media, desviación, valor mínimo y máximo	15
3. Separación de los Datos	15
4. Métricas de error	20
4.1. Problema de clasificación	20
5. Planificación del Trabajo	22
6. Problema de Clasificación	23
6.1. Modelos Tradicionales de Aprendizaje Automático	23
6.1.1. Adaptación del conjunto de datos	23
6.1.2. Elección del número de hits por evento	24
6.1.3. Modelos Tradiciones de Aprendizaje Automático	27
6.1.4. Entrenamiento	30
6.1.5. Resultados del entrenamiento	31
6.1.6. Entrenamiento por Validación Cruzada	36
6.2. Modelos de Redes Neuronales	41
6.2.1. Adaptación del conjunto de datos para el entrenamiento	41
6.2.2. Arquitecturas para el entrenamiento	44
6.2.3. Entrenamiento	48
6.2.4. Resultados de entrenamiento	49
6.2.5. Comparación de entrenamientos	55

6.2.6. Entrenamiento del mejor modelo	58
6.2.7. Resultados finales	60
7. Problema de Regresión	62
7.1. Adaptación del conjunto de datos al entrenamiento	62
7.2. Elección del número de hits por evento	63
7.3. Modelos tradicionales de aprendizaje automático para la regresión	63
7.4. Entrenamiento	65
7.5. Resultados de Entrenamiento	67
7.5.1. Resultados Entrenamiento con el conjunto de datos para la clasificación	67
7.5.2. Resultados entrenamiento con el tipo de partícula añadido	71
7.5.3. Comparación de Rendimiento del Modelo Random Forest con Diferentes Conjuntos de Entrada	75
7.6. Entrenamiento del mejor modelo con mejores resultados	80
7.7. Resultado final	80
8. Análisis de los Resultados	81
8.1. Etapa de Clasificación	81
8.2. Etapa de Regresión	83
9. Despliegue del Modelo	84
9.1. Montaje de FastAPI	84
9.2. Script de Python	84
9.2.1. Main	84
9.2.2. Script para reentrenar los modelos	85
9.2.3. Script para la clasificación de las partículas	85
9.2.4. Script para la regresión de la energía de entrada	85
10. Presupuesto	86
11. Conclusiones	88

Índice de cuadros	0
-------------------	---

Índice de cuadros

1. Descripción numérica de las características	15
2. Distribución del conjunto de datos	20
3. Matriz de Confusión [3]	20
4. Parámetros de los modelos de Aprendizaje Automático	30
5. Tabla resumen entrenamiento modelos de Aprendizaje Au- tomático	32
6. Mejor N para cada modelo	36
7. Resultados en términos de accuracy del entrenamiento por Va- lidación Cruzada	37
8. Resumen resultados Accuracy	39
9. Arquitectura Red Convolucional	45
10. Arquitectura de Redes Recurrentes con capas LSTM (RNN- LSTM)	45
11. Arquitectura de Redes Recurrentes con capas GRU (RNN-GRU)	46
12. Resultados de accuracy en los distintos tipos de energías . .	61
13. Parámetros de los modelos de Aprendizaje Automático para la regresión	65
14. Resumen de resultados del problema de la regresión de la energía de entrada	80
15. Resultados problema regresión energía de entrada	81
16. Horas de trabajo de los tutores	88

Índice de figuras

1.	Comparación entre Neuronas Biológicas y Redes Neuronales Artificiales	1
2.	Esquema del Experimento DUNE: Del Acelerador de Protones a la Detección Subterránea [9]	3
3.	Diagrama del Detector Subterráneo del Experimento DUNE [12]	4
4.	Montaje del Detector de Partículas para el Experimento DUNE	5
5.	Estructura inicial de los datos	6
6.	Matriz de correlación inicial	8
7.	Matriz de correlación con los datos de entrenamiento	9
8.	Gráfica distribución variable <i>hitX</i>	11
9.	Gráfica distribución variable <i>hitY</i>	12
10.	Gráfica distribución variable <i>hitZ</i>	13
11.	Gráfica distribución variable <i>hitInteg</i>	14
12.	Separación del número de eventos en los distintos grupos . . .	16
13.	Número de eventos de cada partícula en el conjunto de entrenamiento	17
14.	Número de eventos de cada partícula en el conjunto de validación	18
15.	Número de eventos de cada partícula en el conjunto de prueba	19
16.	Estructura de datos para el problema de clasificación con modelos tradicionales de Aprendizaje Automático	24
17.	Distribución del número de hits por eventos	25
18.	Diagrama de caja de la distribución del número de hits por evento	26
19.	Comparativa de accuracy de todos los modelos tradicionales de Aprendizaje Automático	32
20.	Comparativa de los tiempos de entrenamiento de los modelos de Aprendizaje Automático	33
21.	Matriz de confusión del entrenamiento de LightGBM	34
22.	Matriz de confusión del entrenamiento de Random Forest . .	34
23.	Matriz de confusión del entrenamiento de Gradient Boosting .	35
24.	Matriz de confusión del entrenamiento de XGBoost	35
25.	Comparativa de accuracy en el entrenamiento de los modelos de Aprendizaje Automático por Validación Cruzada	37
26.	Función de perdida del entrenamiento de XGBoost	38
27.	Resultados en términos de Accuracy de todos los conjuntos de entrenamiento	40
28.	Frecuencia de la longitud de las cadenas de hits por eventos .	42

29.	Longitud de las cadenas de hits por eventos	43
30.	Longitud de las cadenas de hits por eventos	44
31.	Comportamiento de la función de perdida entrenamiento arquitectura CNN	50
32.	Comportamiento del accuracy en el entrenamiento arquitectura CNN	50
33.	Comportamiento de la función de perdida entrenamiento arquitectura LSTM	52
34.	Comportamiento del accuracy en el entrenamiento arquitectura LSTM	52
35.	Comportamiento de la función de perdida entrenamiento arquitectura GRU	53
36.	Comportamiento del accuracy en el entrenamiento arquitectura GRU	54
37.	Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de entrenamiento	55
38.	Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de validación	56
39.	Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de entrenamiento	56
40.	Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de validación	57
41.	Función de pérdida del entrenamiento del mejor modelo (CNN)	59
42.	Accuracy del entrenamiento del mejor modelo (CNN)	59
43.	Resultados de accuracy en los distintos tipos de energías	61
44.	Estructura de datos para el problema de regresión con modelos tradiciones de Aprendizaje Automático	63
45.	Error Absoluto Medio	67
46.	Error Cuadrático Medio	68
47.	Coeficiente de Determinación	69
48.	Error Absoluto Medio Normalizado	69
49.	Error Cuadrático Medio Normalizado	70
50.	Error Absoluto Medio	72
51.	Error Cuadrático Medio	72
52.	Coeficiente de Determinación	73
53.	Error Absoluto Medio Normalizado	74
54.	Error Cuadrático Medio Normalizado	74
55.	Comparativa para Random Forest del Error Absoluto Medio .	76
56.	Comparativa para Random Forest del Error Cuadrático Medio .	76
57.	Comparativa para Random Forest del Coeficiente de Determinación	77

Índice de figuras	0
-------------------	---

58. Comparativa para Random Forest del Error Absoluto Medio Normalizado	78
59. Comparativa para Random Forest del Error Cuadrático Medio Normalizado	79
60. Coste maquina virtual con GPU	87

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 algoritmos distintos ya que usa redes neuronales que funcionan de forma muy parecida a las conexiones neuronales biológicas de nuestro cerebro.

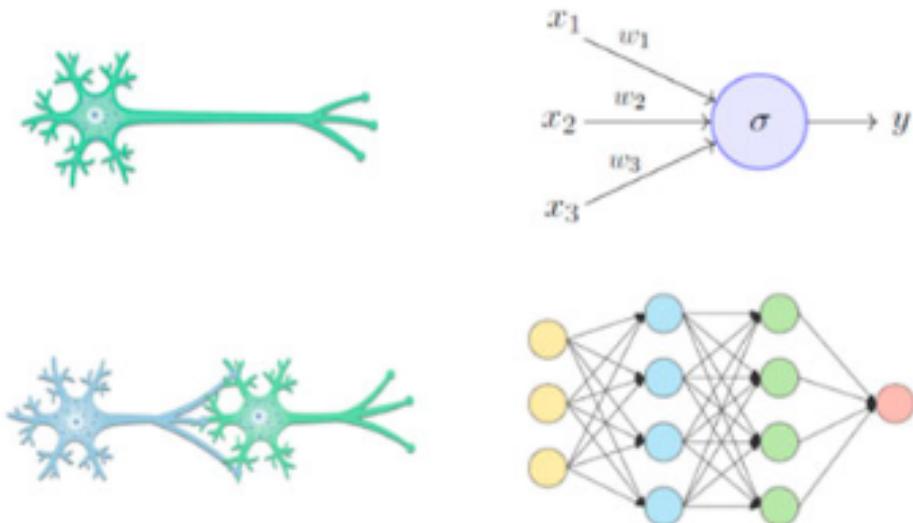


Figura 1: Comparación entre Neuronas Biológicas y Redes Neuronales Artificiales

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 que se dispone, lo que hace que la física sea un laboratorio para probar modelos y herramientas de la ciencia de datos, así como el desarrollo de nuevos algoritmos.

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.

Uno de los problemas a la hora de implementar soluciones en Aprendizaje Profundo es que a diferencia en problemas de Aprendizaje Automático, es que suelen ser los resultados interpretables, en cambio los modelos de Aprendizaje Profundo no es posible comprender cómo el modelo llega al resultado final, lo que dificulta su interpretación. El uso de estos modelos en muchas áreas ha destacado 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.

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.

A altas energías, los muones y los piones se diferencian claramente de los kaones por su masa 3.5 veces menor y por sus productos de desintegración, que son distintos. Sin embargo, a energías bajas, pueden confundirse con kaones. Por esta razón, enfocamos nuestro estudio en diferenciar kaones de piones.

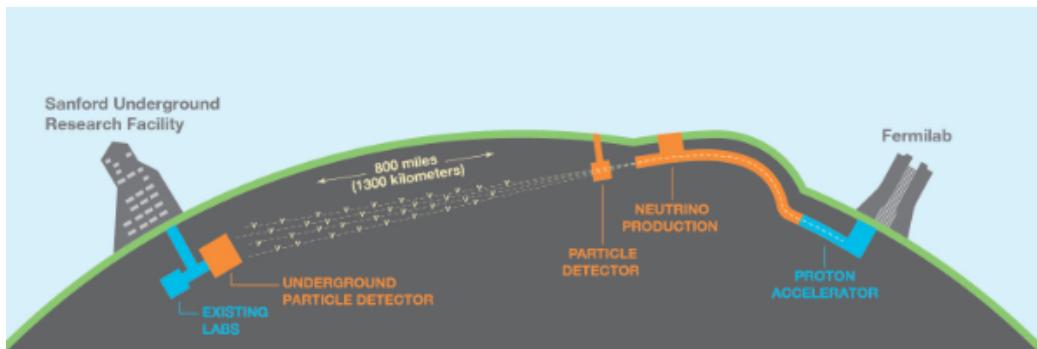


Figura 2: Esquema del Experimento DUNE: Del Acelerador de Protones a la Detección Subterránea [9]

En el experimento DUNE [9] [13] (*Deep Underground Neutrino Experiment*) es un proyecto internacional avanzado en el campo de la ciencia 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 compresión 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.

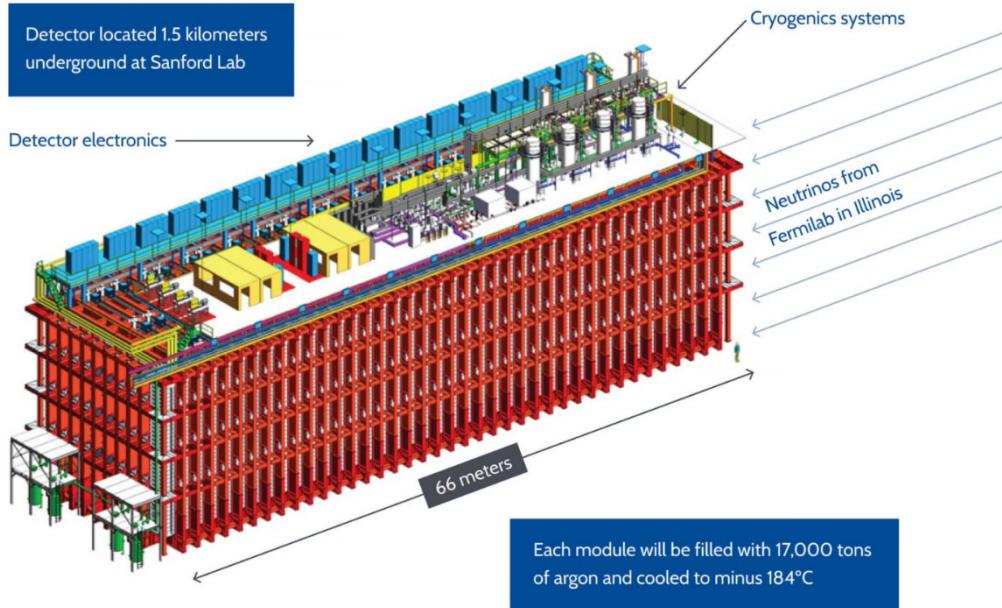


Figura 3: Diagrama del Detector Subterráneo del Experimento DUNE [12]

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.

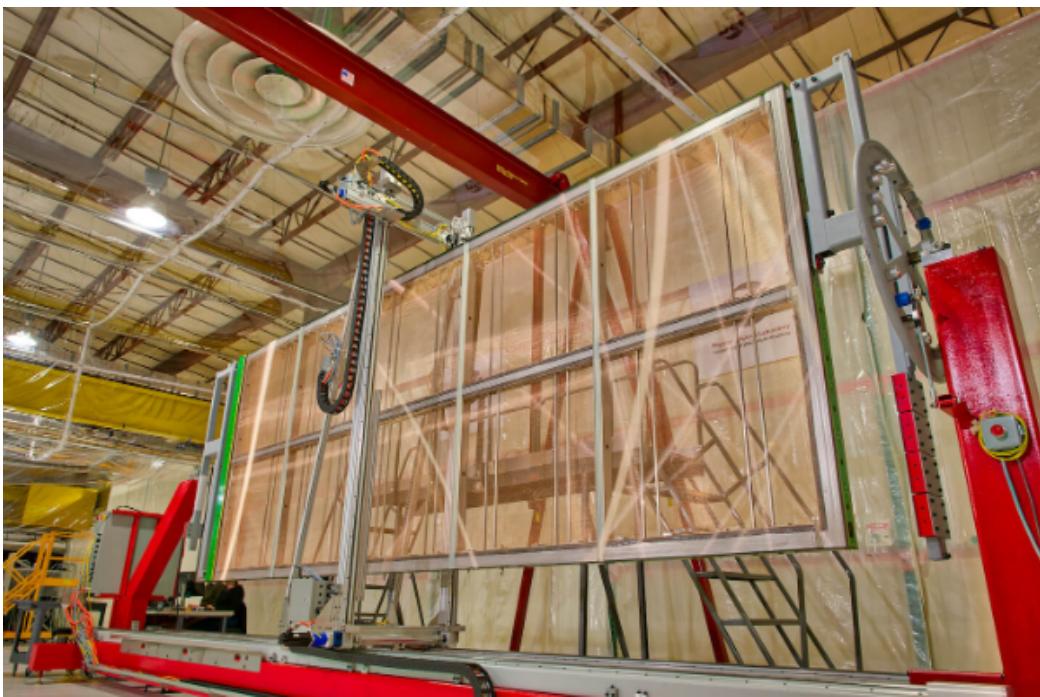


Figura 4: Montaje del Detector de Partículas para el Experimento DUNE

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 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 eventos y la construcción de colecciones de eventos. La combinación de GEANT4 y LArSoft proporcionó una solución efectiva para simular las condiciones y eventos 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.

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 evento que incluye múltiples puntos de impacto, denominado 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 hits se representa en una fila y las hilas que pertenecen al mismo evento 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 evento 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 hits 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 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 eventos de cada partícula, pero no llegaron a generar 496 eventos 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 sacados 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.[33]. 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.

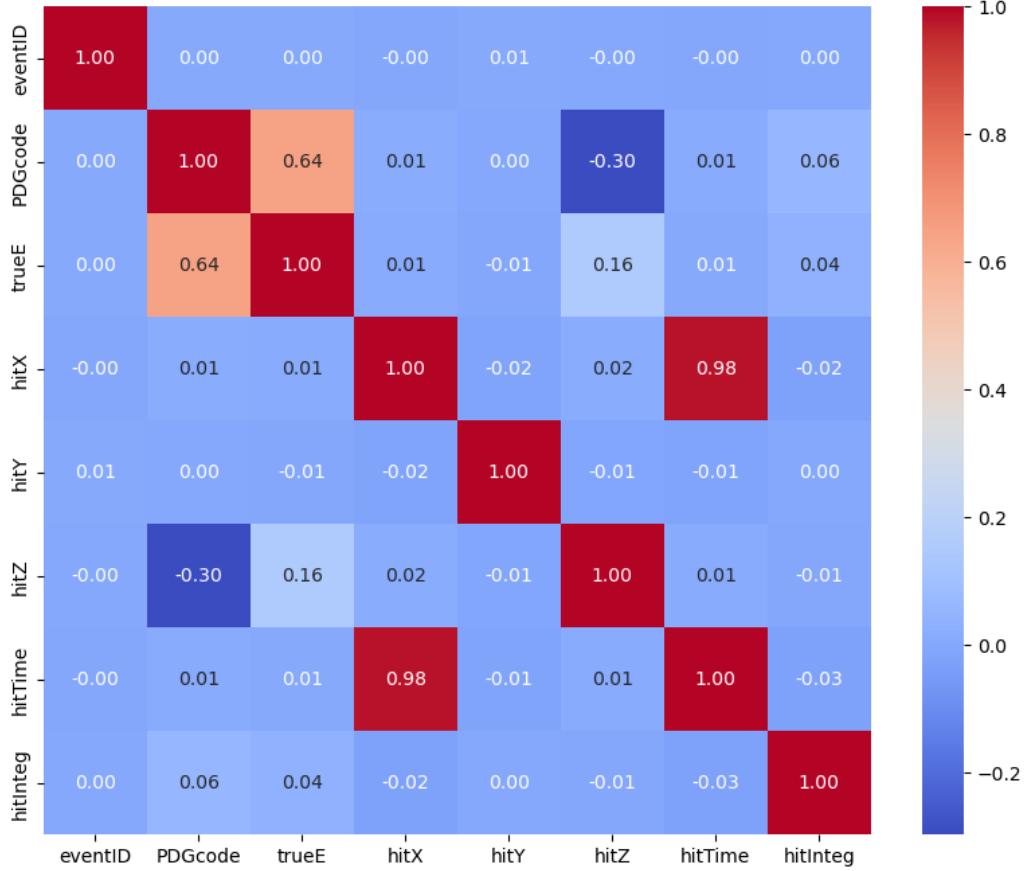


Figura 6: Matriz de correlación inicial

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 evidenciada 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, [37]. La multicolinealidad puede también incrementar la varianza

de los coeficientes estimado, 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

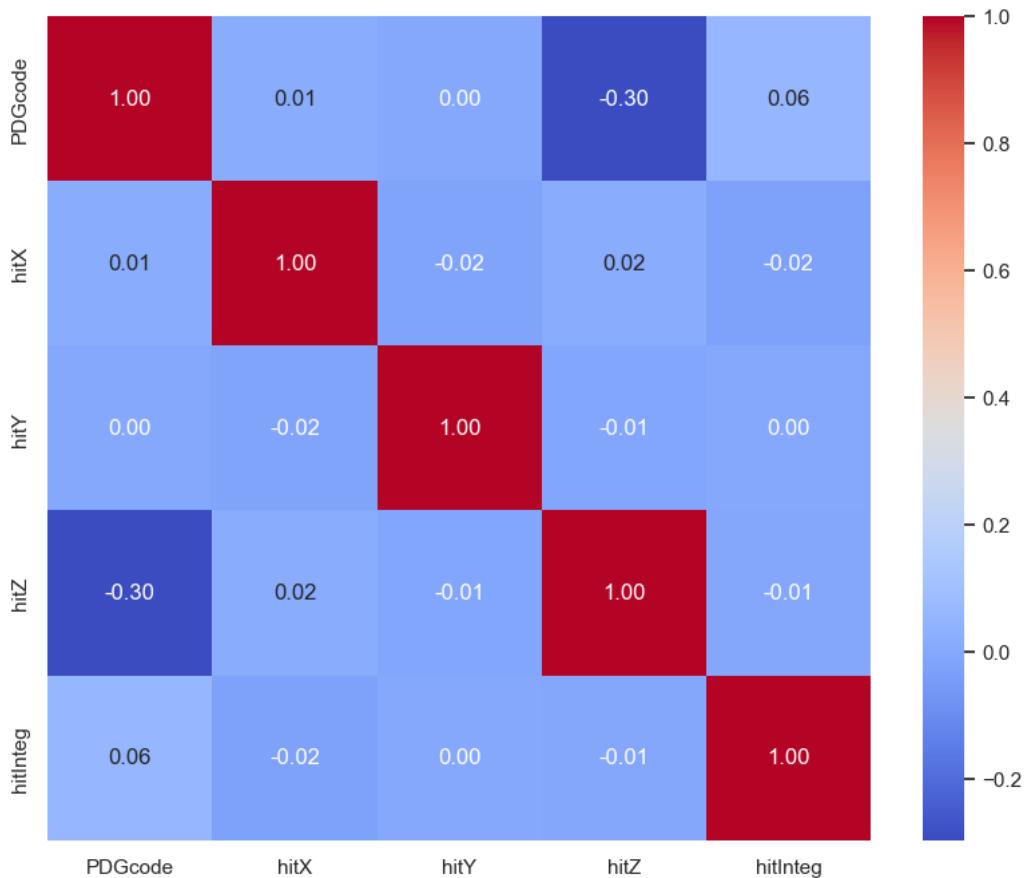


Figura 7: Matriz de correlación con los datos de entrenamiento

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. Esto sugiere que existe una relación lineal negativa moderada entre el *PDGcode* y la coordenada Z (*hitZ*). Esto puede implicar que ciertos tipos de partículas tienden a interactuar en regiones específicas del detector.

Dado que las variables son en su mayoría independientes, los modelos predictivos 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 [35] 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: $hitX$

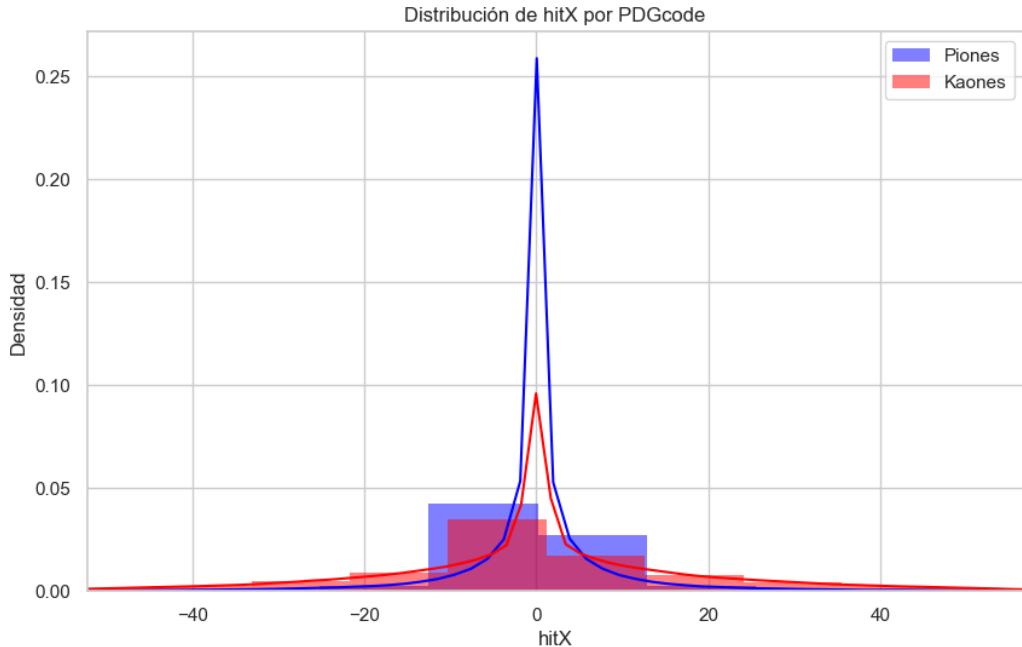


Figura 8: Gráfica distribución variable $hitX$

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

Los piones como podemos ver tiene un pico muy pronunciado alrededor de $hitX = 0$ esto es debido a que en este punto es donde se lanza el experimento 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 en sus valores de $hitX$.

2.2.2. Variable: $hitY$

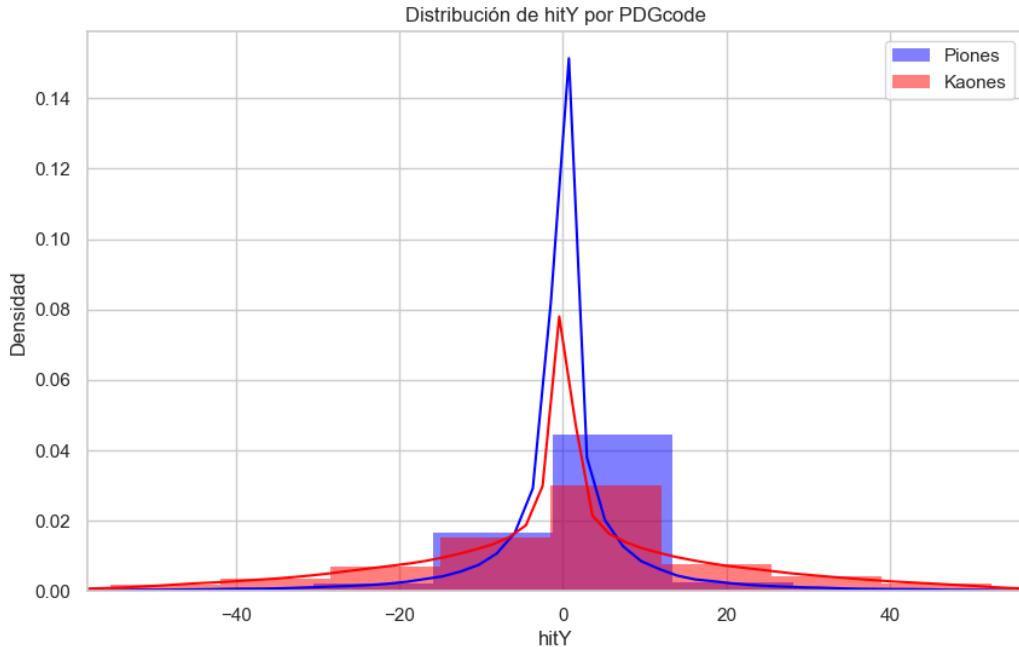


Figura 9: Gráfica distribución variable $hitY$

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 eventos 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: $hitZ$

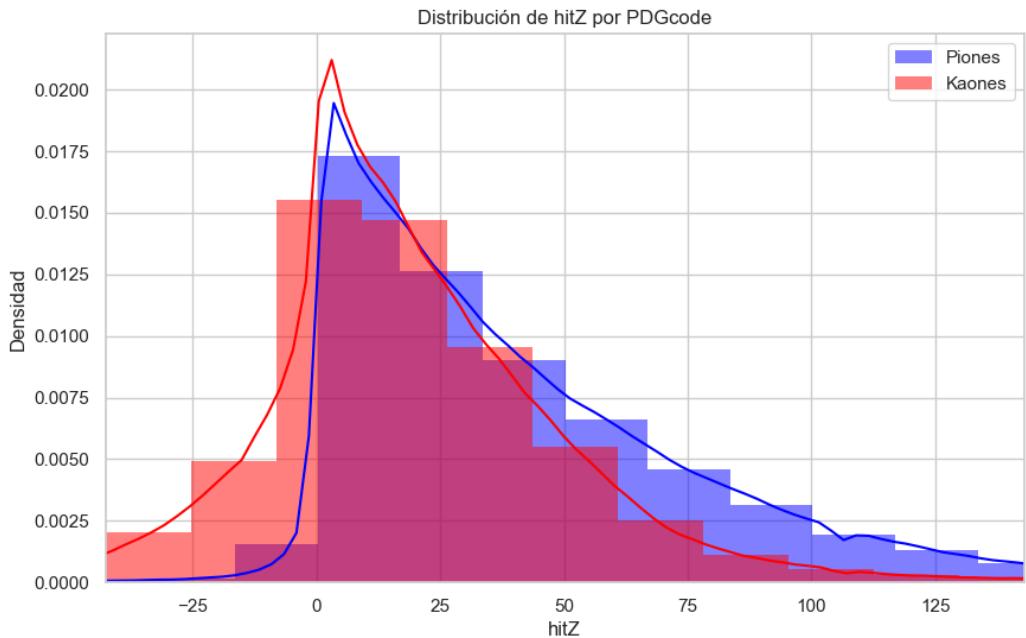


Figura 10: Gráfica distribución variable $hitZ$

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 kaones muestran una cola más larga que los piones. Esto puede indicar que los kaones son capaces de viajar distancias más largas dentro del detector antes de interactuar o ser detectados, en comparación con los piones.

2.2.4. Variable: *hitInteg*

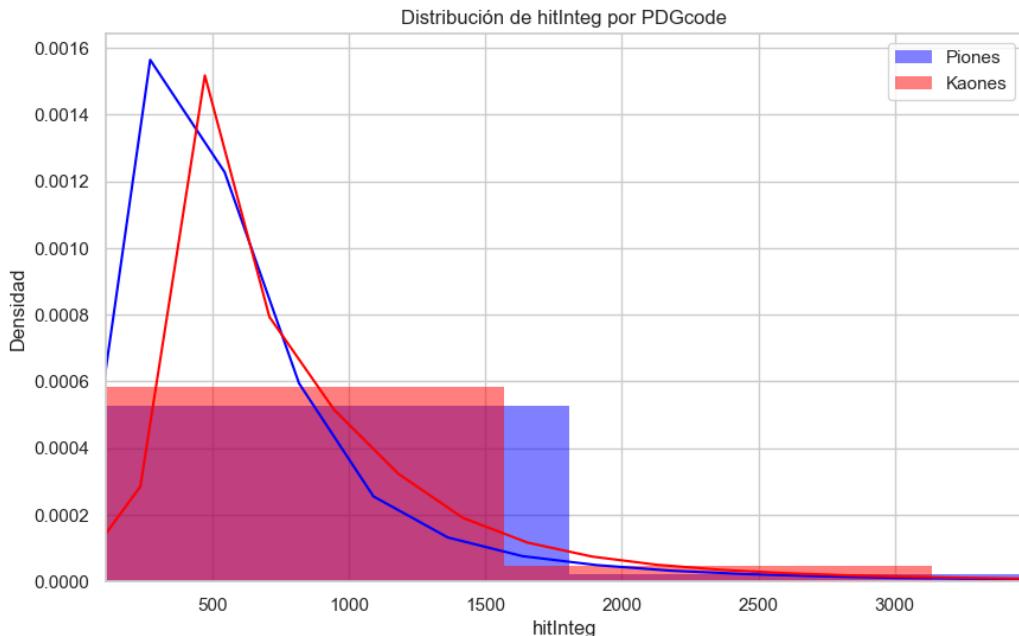


Figura 11: Gráfica distribución variable *hitInteg*

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 quiere decir que los kaones tienden a generar hits con mayor intensidad integrada.

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*.

2.2.5. Media, desviación, valor mínimo y máximo

Variable	Media	Desviación estndar	Valor mímico	Valor máximo
hitX	-99.54213	18.04539	-201.718	183.831
hitY	-0.1310142	19.41624	-2 35.299	203.689
hitZ	177.6367	35.30203	-1.33675	504.1
hitInteg	782.3906	745.2574	1.189	55182.6

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 [25].

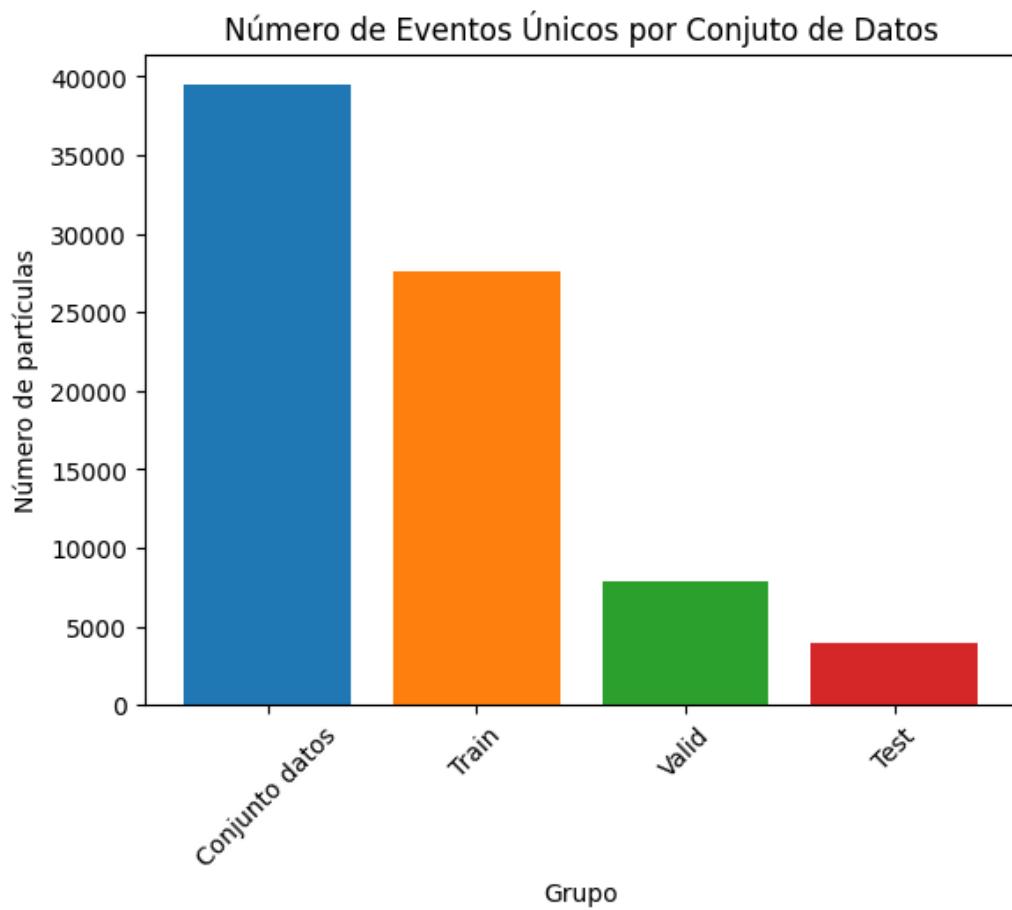


Figura 12: Separación del número de eventos en los distintos grupos

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

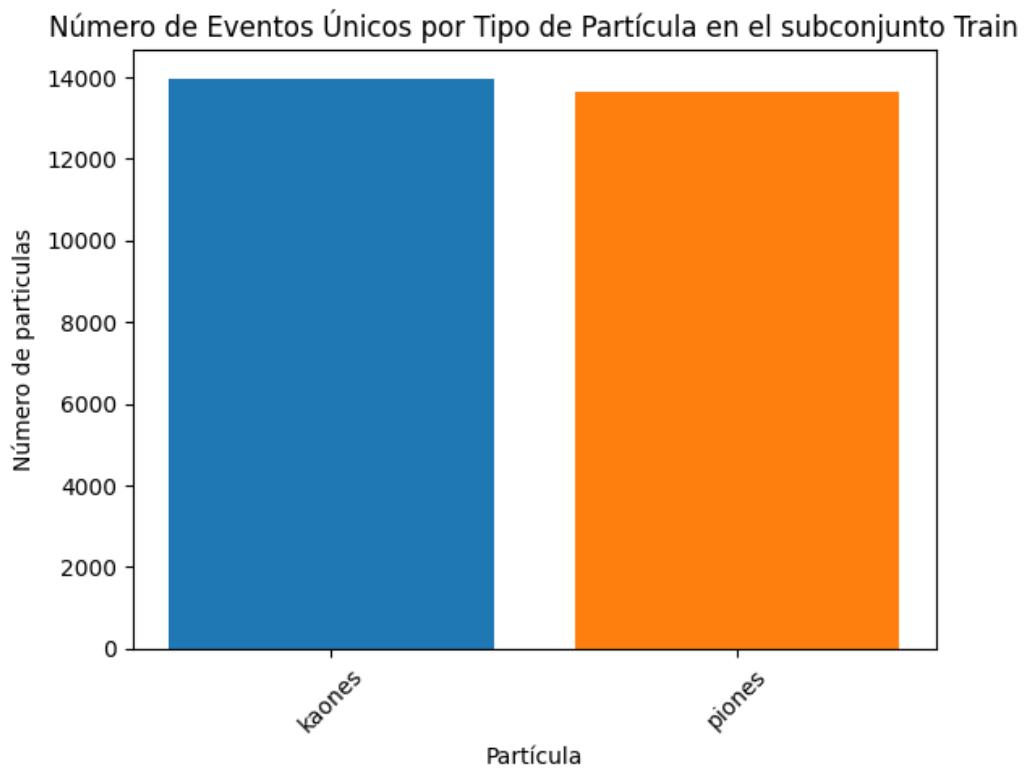


Figura 13: Número de eventos de cada partícula en el conjunto de entrenamiento

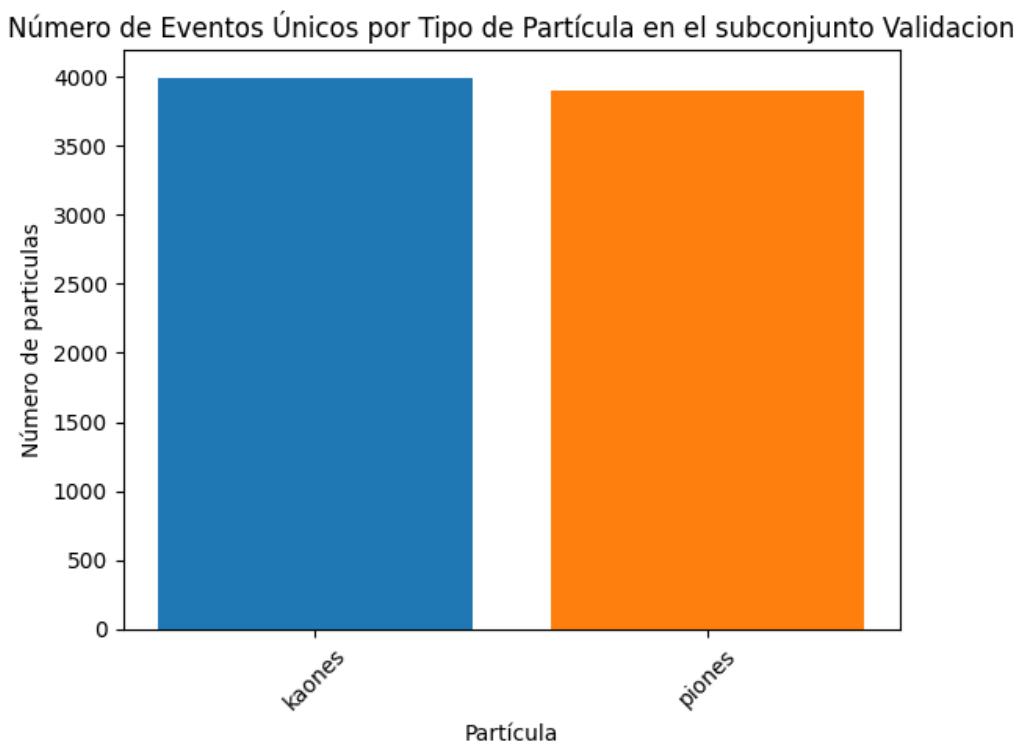


Figura 14: Número de eventos de cada partícula en el conjunto de validación

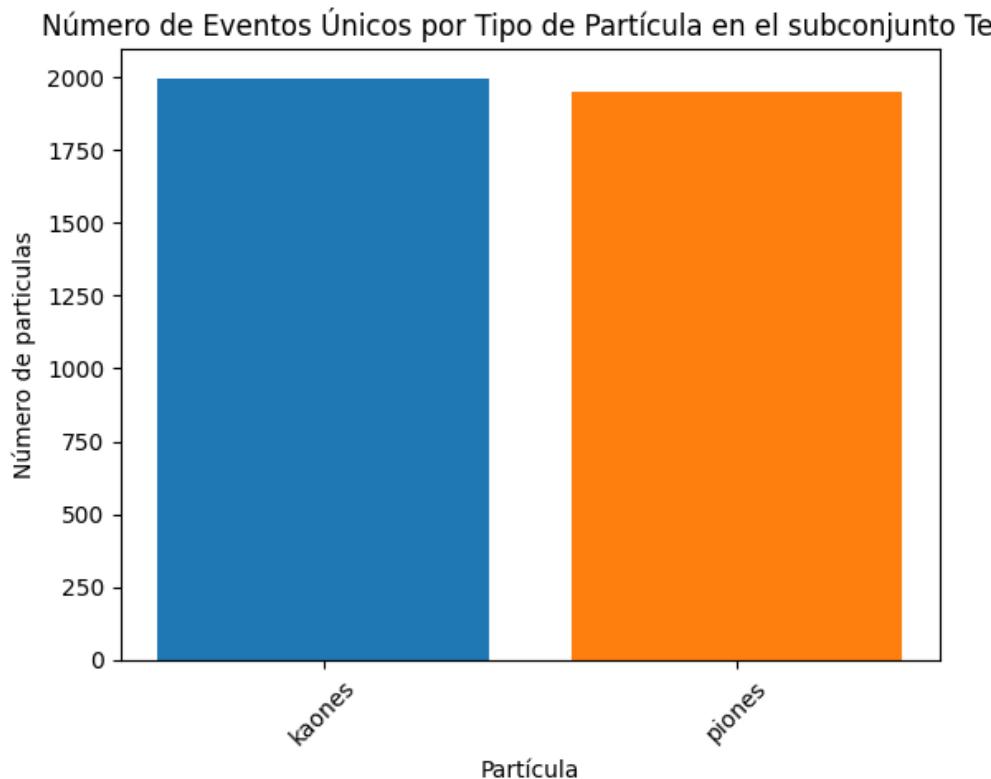


Figura 15: Número de eventos de cada partícula en el conjunto de prueba

El conjunto de entrenamiento (figura 13) 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 14) es esencia 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 15) 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 eventos	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, si no también para asegurar que cumpla con los estándares requeridos para su aplicación práctica, evitando problemas en decisiones basadas en sus predicciones [17]

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. [34]

En el contexto de nuestro problema, al realizar la clasificación binaria entre las dos partículas se ha asegurado que la distribución del número de

eventos 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 eventos complejos, utilizando técnicas de padding para manejar eventos de tamaño variable. Paralelamente, se explorará un enfoque de Machine Learning que agrupa "hits" por evento en una estructura predefinida, aunque este método puede implicar una pérdida de información cuando los eventos 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 eventos completos en lugar de hits individuales. Este método enfrentará desafíos como el ajuste de tamaños variables de eventos 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 evaluará 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 eventos. Como hemos comentado, no todos los eventos tienen el mismo número de hits. Por tanto habrá que elegir un N que representará el número de hits por evento. En el caso de que el N sea mayor que el número de hits por eventos, nos quedaremos con los todos los hits del evento y rellenaremos con 0 todos los huecos hasta completar y si N es menor que el número de eventos 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 eventos y se obtendrán los resultados de la clasificación entre los dos tipos de partículas (kaones o piones) y la energía verdadera que presenta ese evento. Estos eventos 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, esta 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 16: Estructura de datos para el problema de clasificación con modelos tradicionales de Aprendizaje Automático

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

6.1.2. Elección del número de hits por evento

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

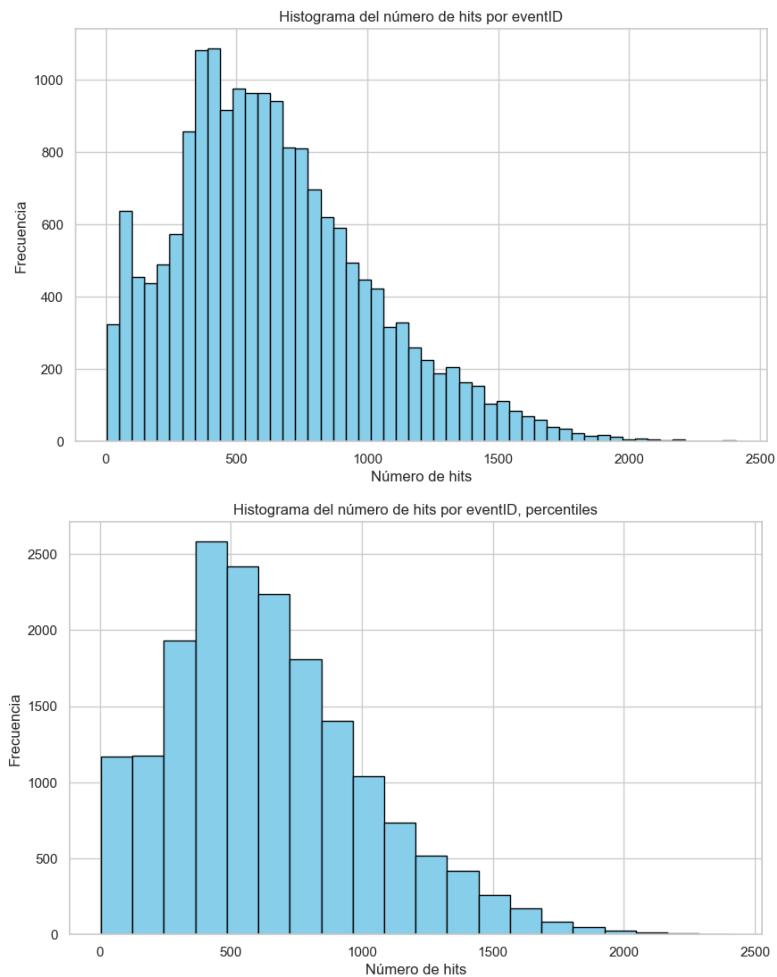


Figura 17: Distribución del número de hits por eventos

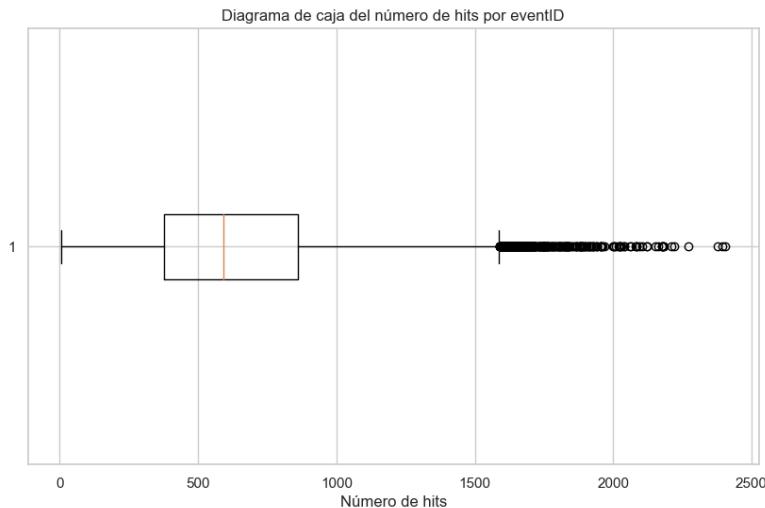


Figura 18: Diagrama de caja de la distribución del número de hits por evento

Observando los histogramas 17, podemos ver que la mayoría de los eventos tienen un número de hits que caen dentro de una amplia gama. El primer histograma de 17 muestra una distribución con picos aproximados entre 500 y 1000 hits por evento. El segundo histograma 17 recalca que hay una alta concentración de eventos 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 17 sugieren que el rango de 400 a 800 hits por evento 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 18 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 [DIAGRAMA DE CAJA] 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 selecciono un rango de hits por eventos 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 17 y 18. 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 Tradiciones de Aprendizaje Automático

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

- **Regresión Logística:** [14] La Regresión Logística es el conjunto de modelos estadísticos utilizados cuando se desea conocer la 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** [27] 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:** [20] 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. Estas máquinas 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:** [32] 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:** [18] 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 con

conjuntos de datos grandes. Se utiliza en sistemas de recomendación, reconocimiento de patrones, entre otros.

- **XGBoost:** [11] 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:** [22] 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

6.1.4. 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 16. 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 Evento:** Se agrupan los datos ordenados por *eventID*, lo que permite manejar los datos de cada evento individualmente

en los pasos siguientes.

3. **Filtrado por selección de *PDGcode*:** Dentro de cada grupo de eventos, 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 eventos 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 evento.
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 virtual 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 evento, 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.

6.1.5. Resultados del entrenamiento

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

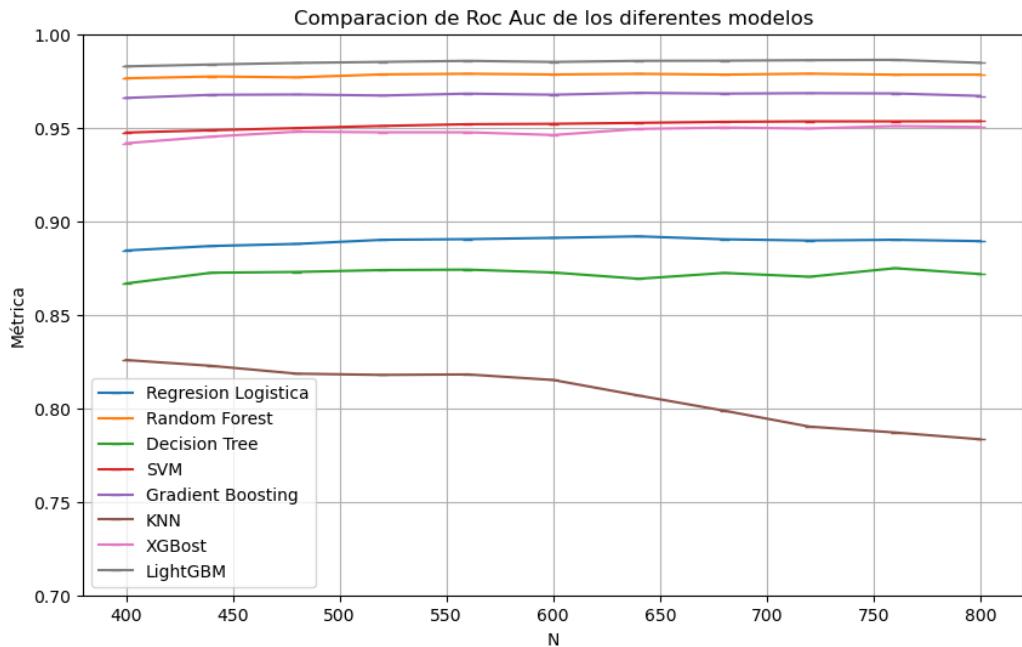


Figura 19: Comparativa de accuracy de todos los modelos tradicionales de Aprendizaje Automático

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	0.9361	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	0.8938	800	0.8807	400	0.0130
K-NN	0.7228	400	0.6574	800	0.0653
XGBoost	0.9508	760	0.9417	400	0.0091
LightGBM	0.9510	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:

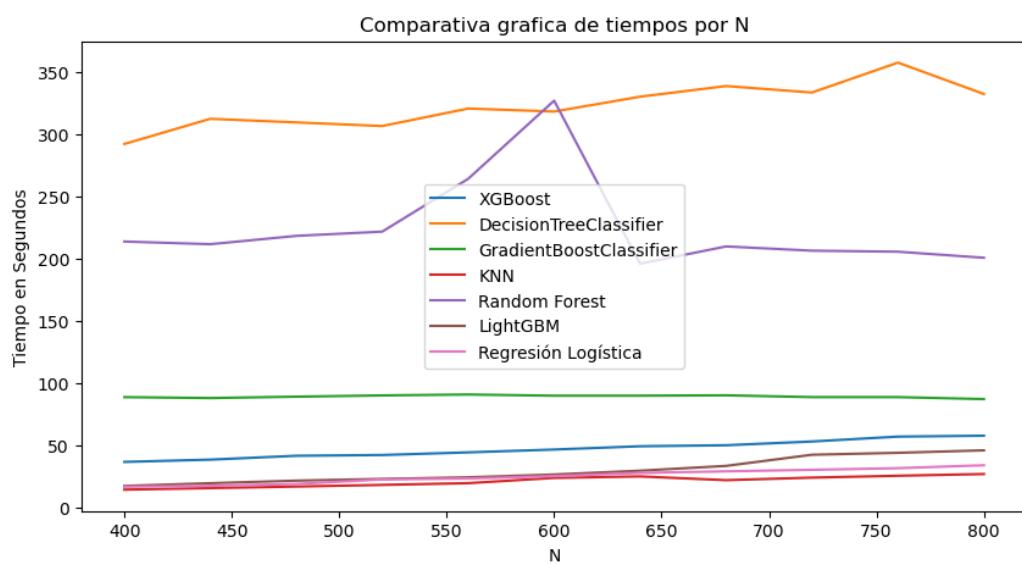


Figura 20: Comparativa de los tiempos de entrenamiento de los modelos de Aprendizaje Automático

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

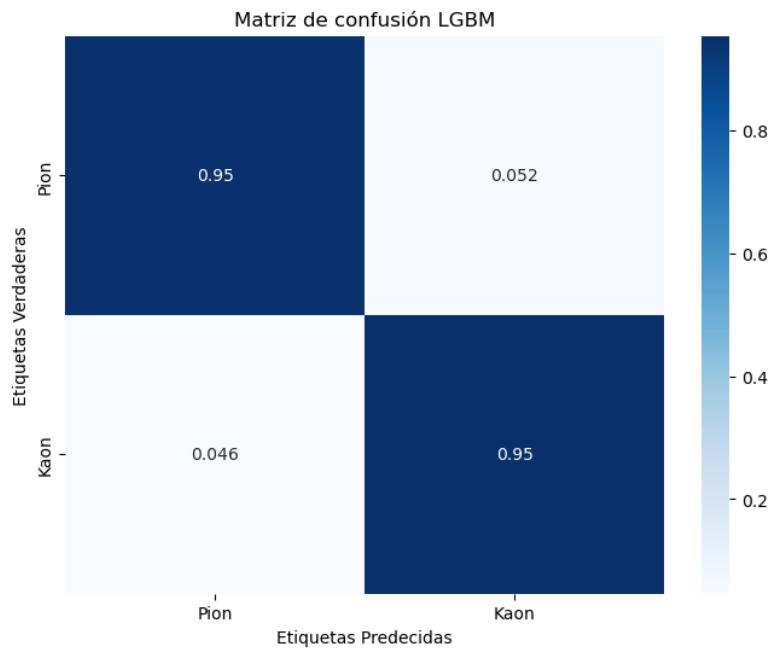


Figura 21: Matriz de confusión del entrenamiento de LightGBM

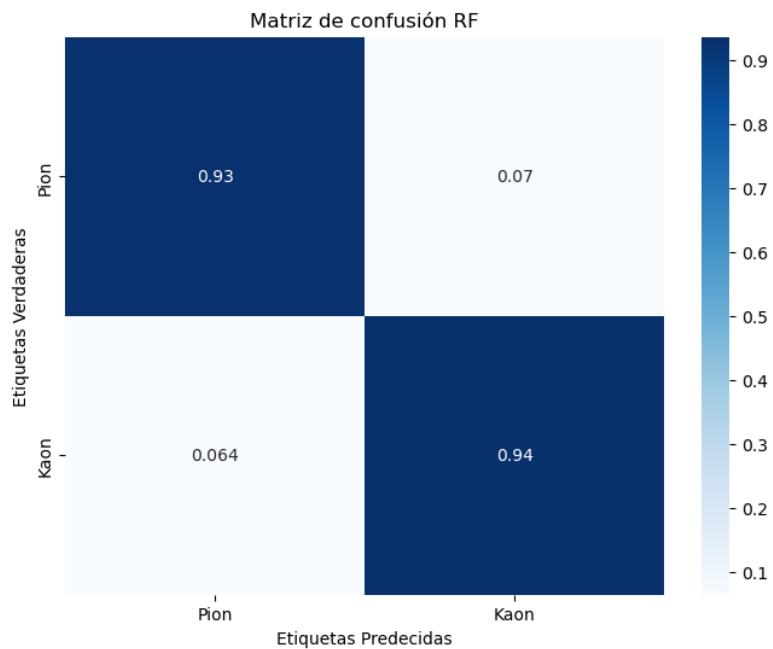


Figura 22: Matriz de confusión del entrenamiento de Random Forest

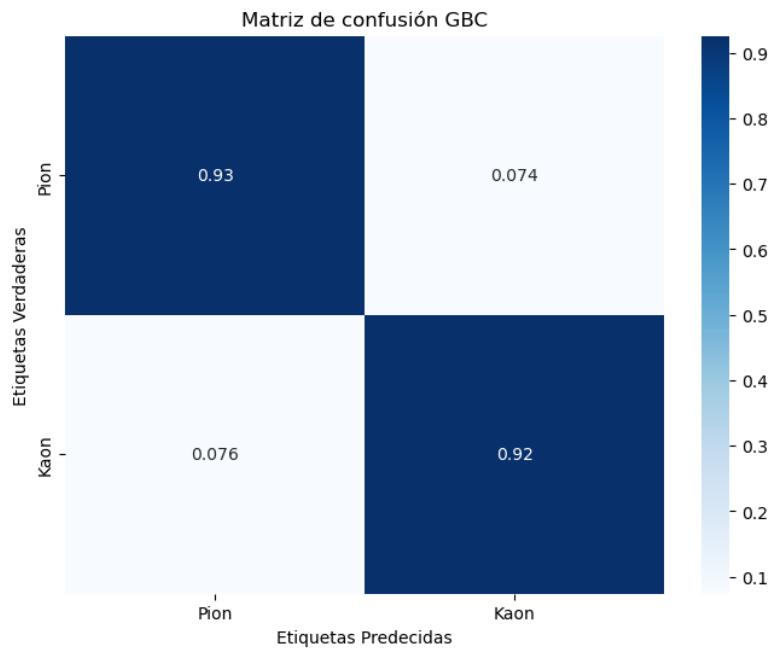


Figura 23: Matriz de confusión del entrenamiento de Gradient Boosting

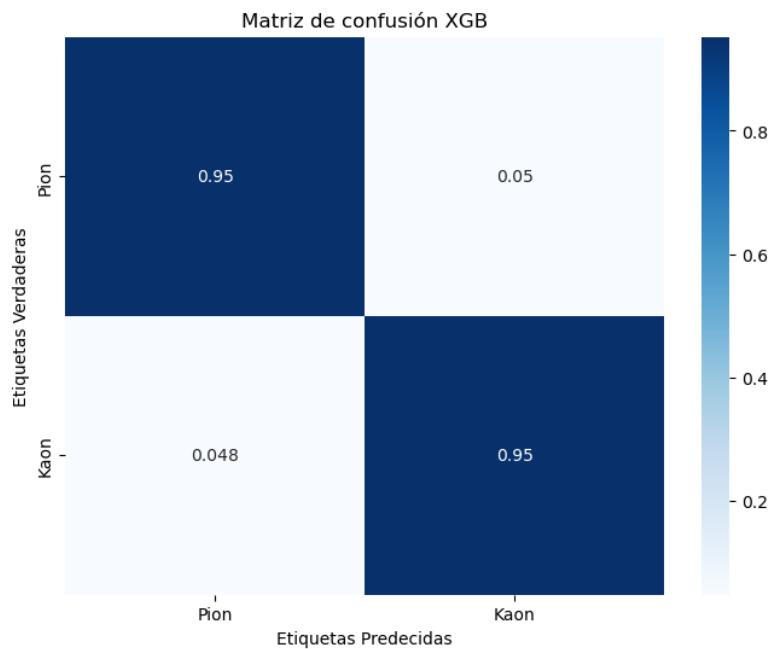


Figura 24: Matriz de confusión del entrenamiento de XGBoost

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

6.1.6. Entrenamiento por Validación Cruzada

La Validación Cruzada [30] 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 comparar 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:

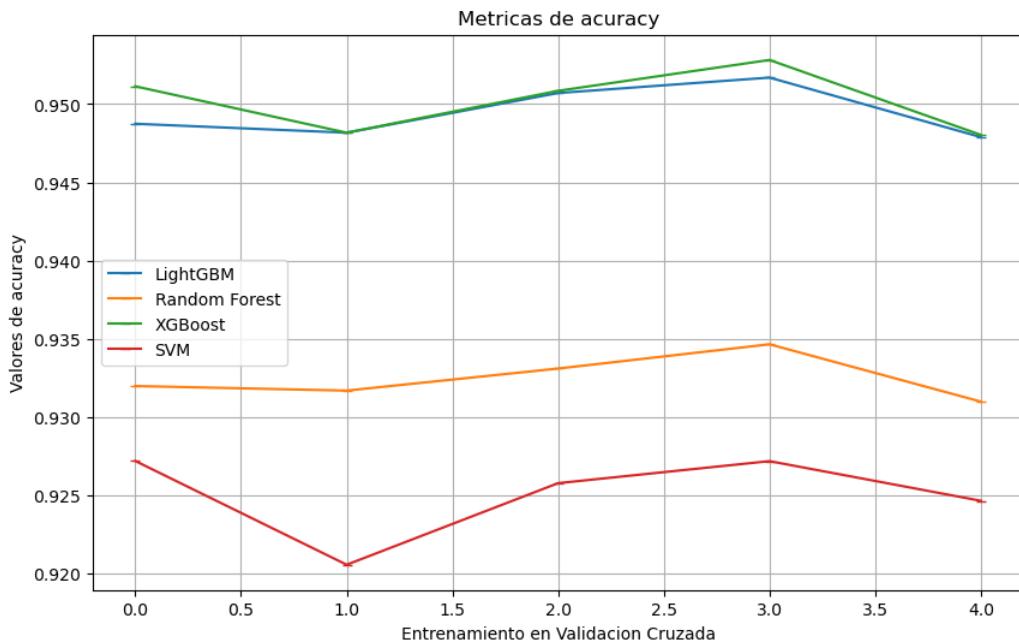


Figura 25: Comparativa de accuracy en el entrenamiento de los modelos de Aprendizaje Automático por Validación Cruzada

Estos son los resultados numéricos en la tabla resumen:

Modelo	Iteración 1	Iteración 2	Iteración 3	Iteración 4	Iteración 5
Random Forest	0.93199099	0.93169976	0.93310801	0.93465709	0.93099563
Gradient Boosting	0.9272036	0.92057457	0.9257851	0.92719335	0.9246585
LightGBM	0.94874683	0.94817631	0.95071117	0.95169694	0.94789466
XGBoost	0.95114052	0.94817631	0.95085199	0.95282355	0.94803549

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

Según la imagen 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.

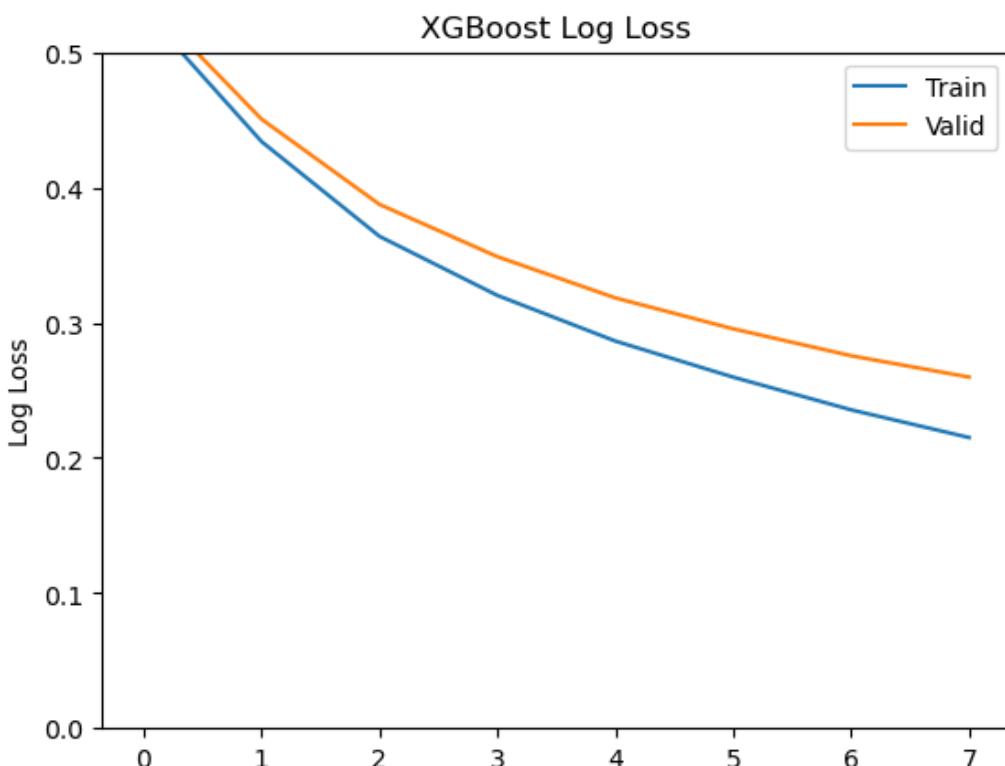


Figura 26: Función de perdida del entrenamiento de XGBoost

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 clasifica 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 evento. Por consiguiente, adaptaremos la función de filtrado de datos para que incluyan como nuevo output, la energía de entrada correspondiente a cada evento, 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 bajas	Accuracy medias	Accuracy altas
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

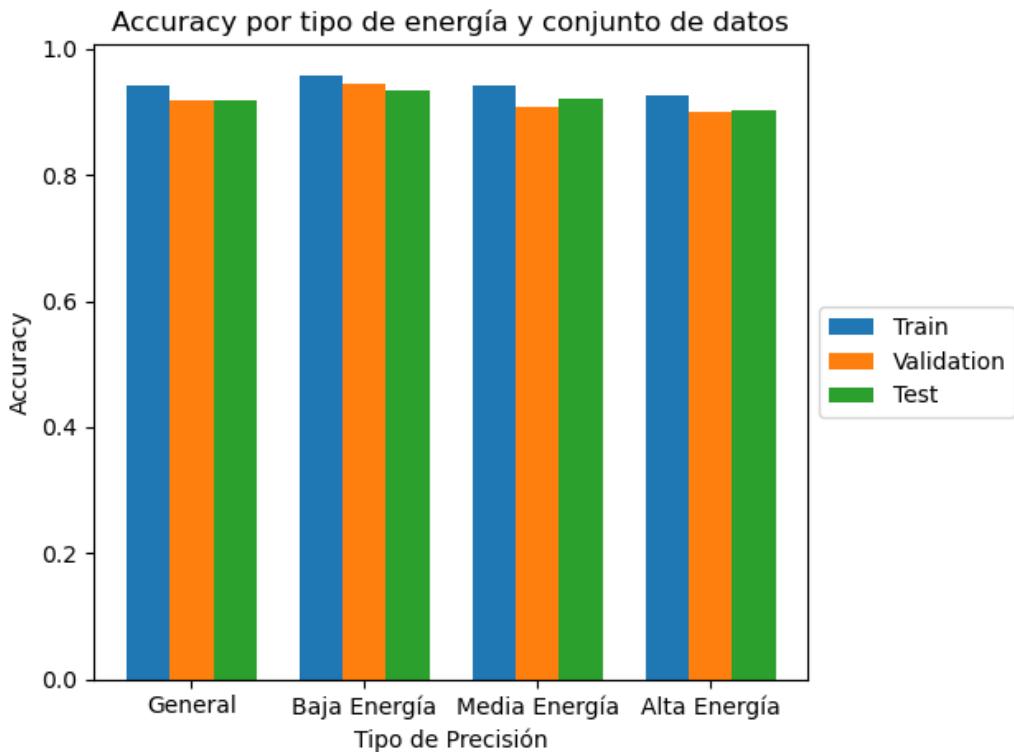


Figura 27: Resultados en términos de Accuracy de todos los conjuntos de entrenamiento

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.

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.

6.2. 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 evento con el mayor número de hits en el conjunto de entrenamiento. Este criterio garantiza que si en futuros conjuntos de datos se identifica un evento con una cantidad superior de hits, se prioriza 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.

6.2.1. Adaptación del conjunto de datos para el entrenamiento

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

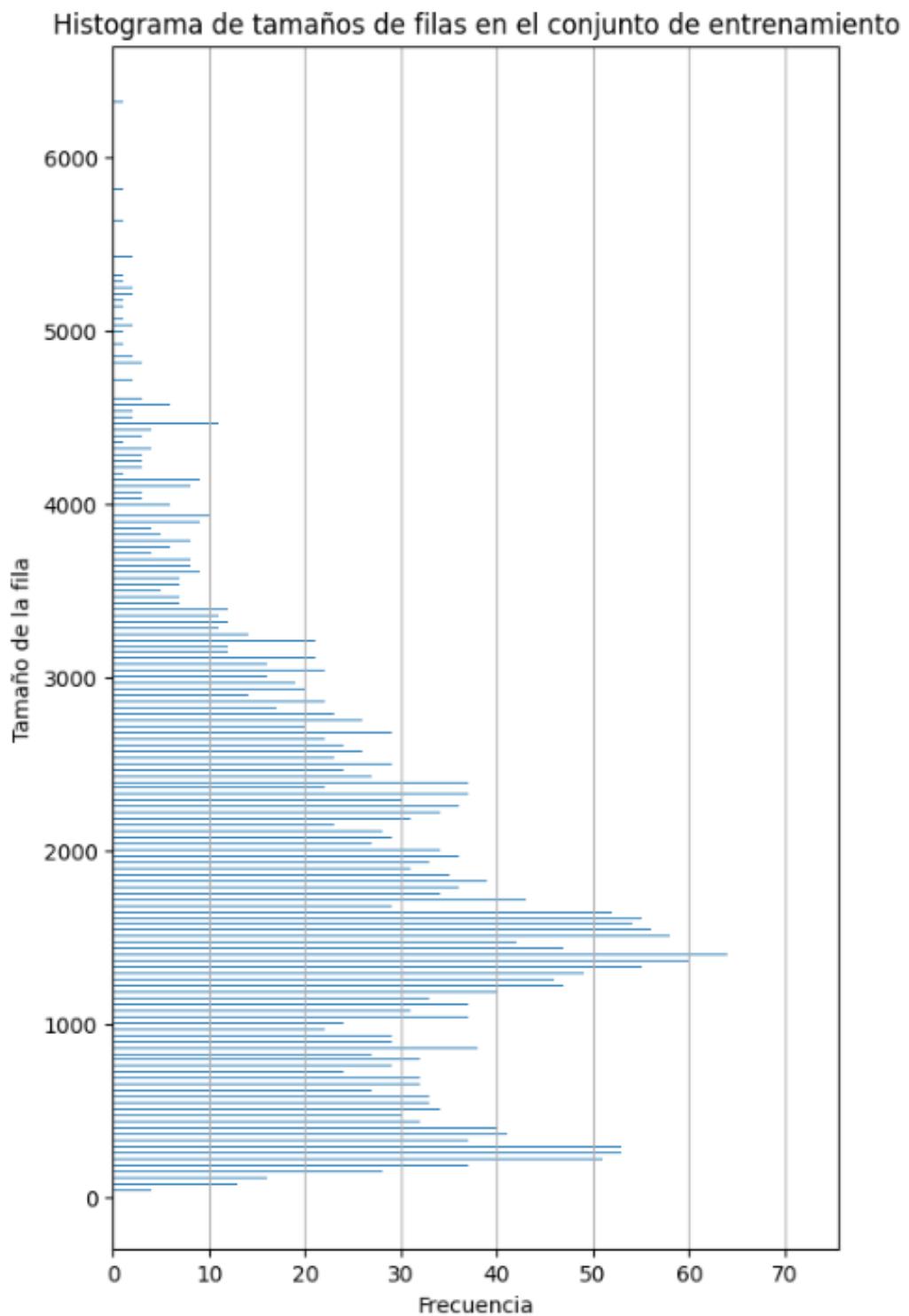


Figura 28: Frecuencia de la longitud de las cadenas de hits por eventos

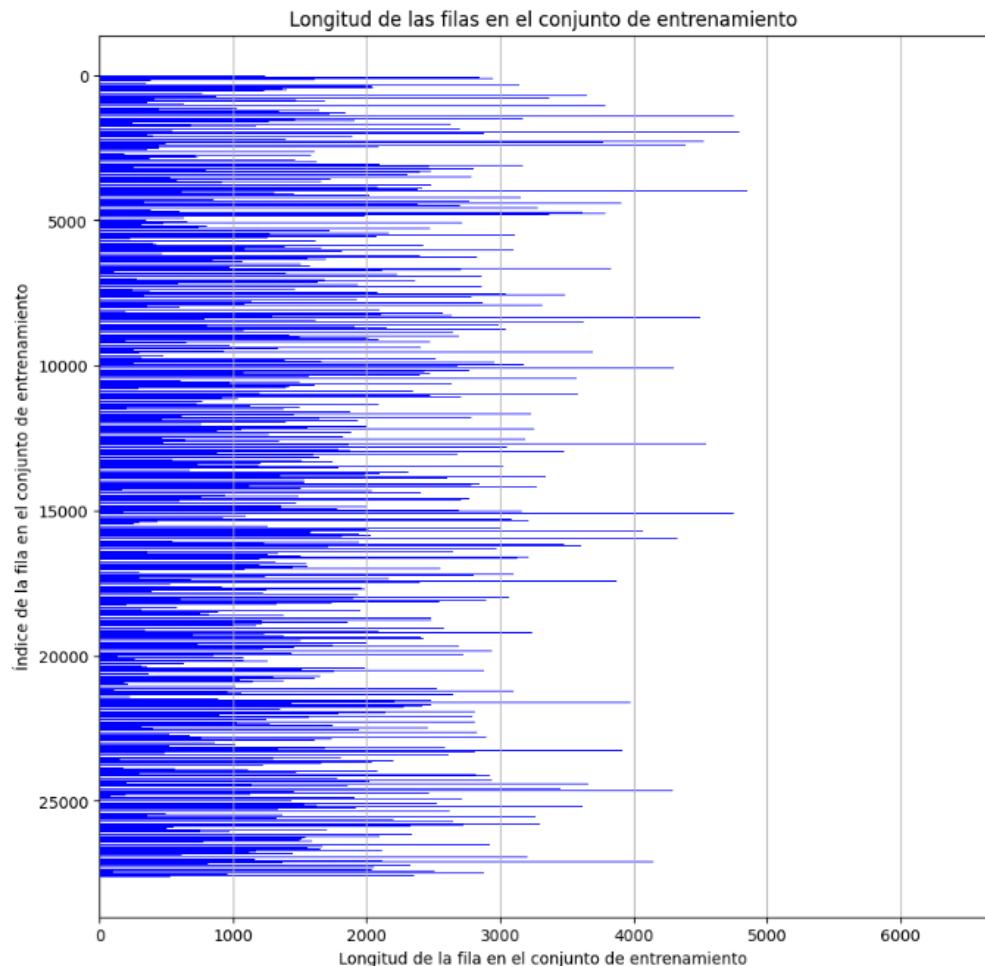


Figura 29: Longitud de las cadenas de hits por eventos

La cantidad de hits del evento 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 30 y completamos con un padding de ceros hasta completar cada fila, como podemos ver en la siguiente imagen.

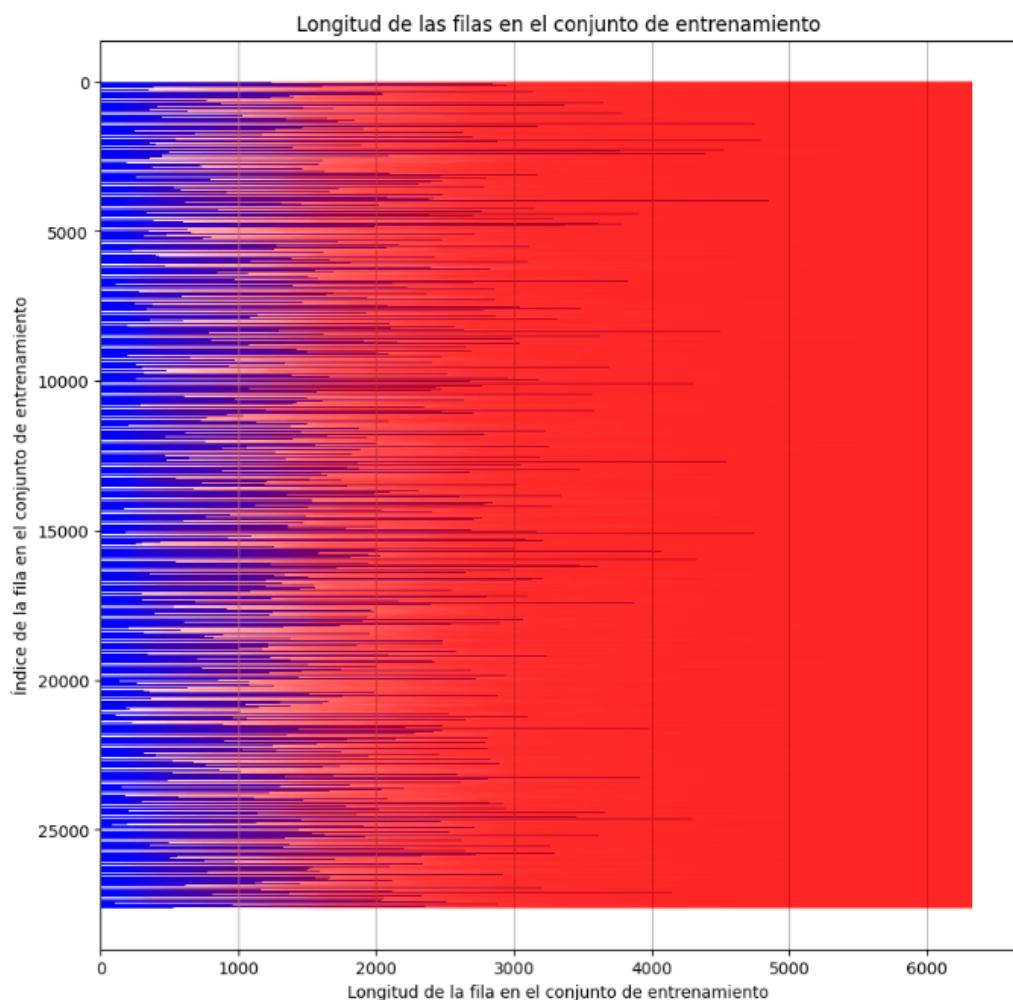


Figura 30: Longitud de las cadenas de hits por eventos

6.2.2. 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 Convolutacional

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 usadase en los distintos modelo son las siguientes:

- **Convoluciones:** Las capas de convolución en el análisis de series temporales y señales unidimensionales, aplican una operación de convolución a los 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. [23]
- **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 $f(x) = \max(0, x)$. [24]
- **Global Average Pooling:** Es una técnica de reducción de dimensiones aplicada 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. [26]

- **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. [24]
- **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.[15]
- **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]

6.2.3. 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 30. 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. Despues 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 a 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 la bibliotecas de TensorFlow a la hora de compilar el modelo. [21]

- **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.

6.2.4. Resultados de entrenamiento

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

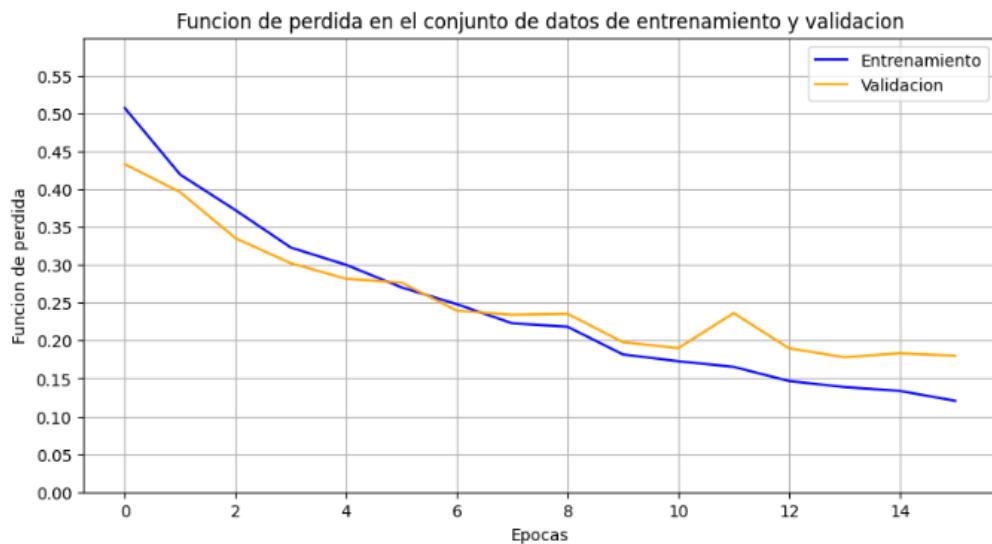


Figura 31: Comportamiento de la función de perdida entrenamiento arquitectura CNN

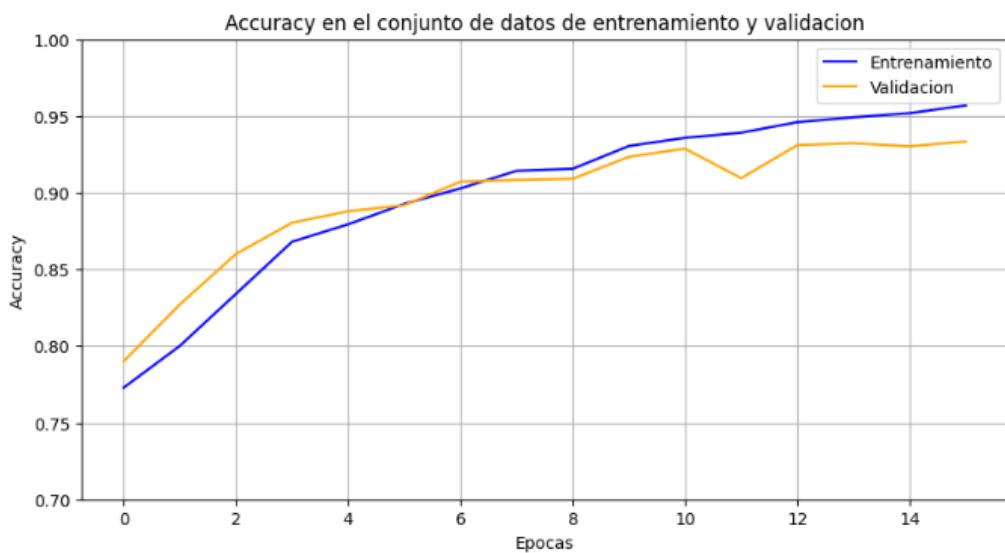


Figura 32: Comportamiento del accuracy en el entrenamiento arquitectura CNN

Según la figura 31 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 32 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 sido 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.

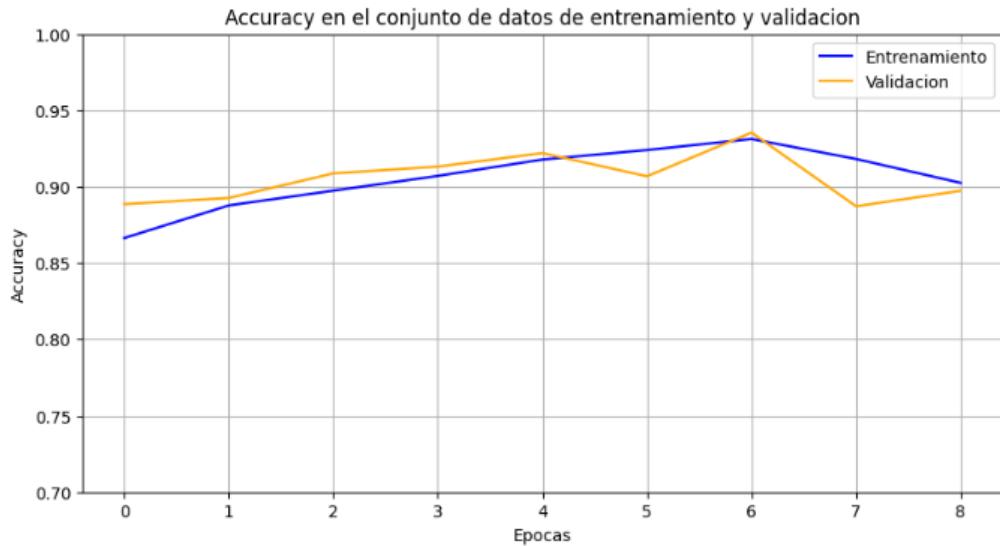


Figura 33: Comportamiento de la función de perdida entrenamiento arquitectura LSTM

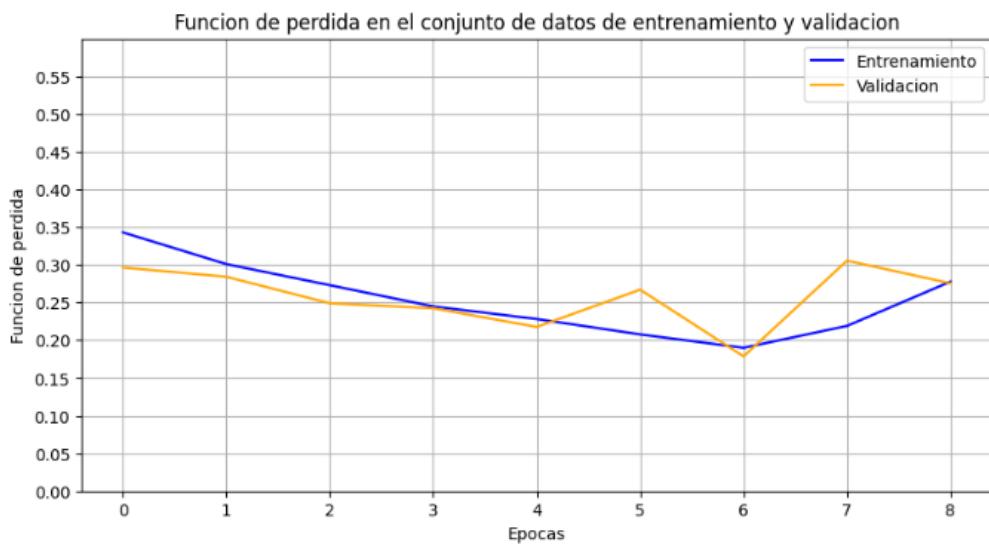


Figura 34: Comportamiento del accuracy en el entrenamiento arquitectura LSTM

Como podemos ver en la gráfica de la función de pérdida 33 podemos que 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 34, la precisión con el conjunto de datos de entrenamiento aumenta de manera constante desde aproximadamente 0.85 hasta 0.90 en sus primeras 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:

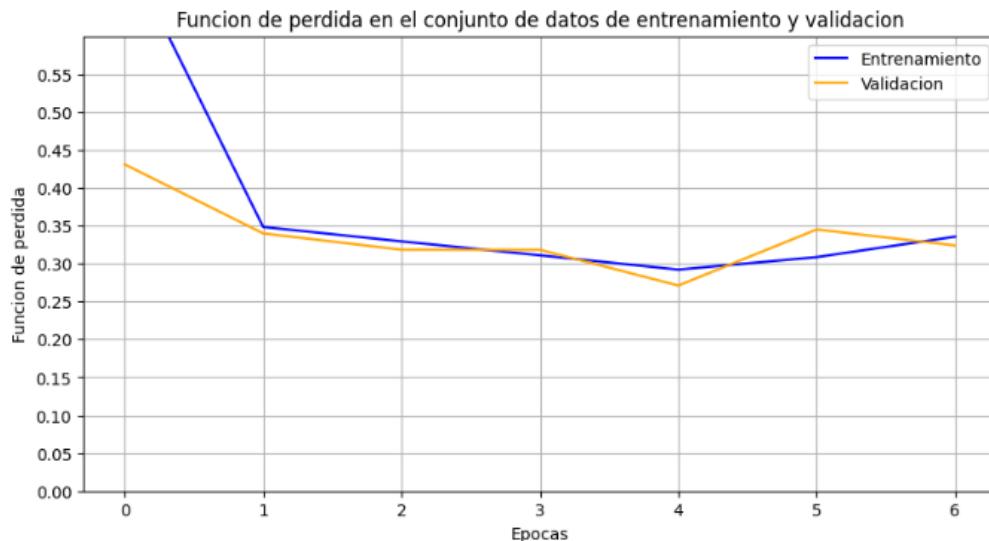


Figura 35: Comportamiento de la función de perdida entrenamiento arquitectura GRU

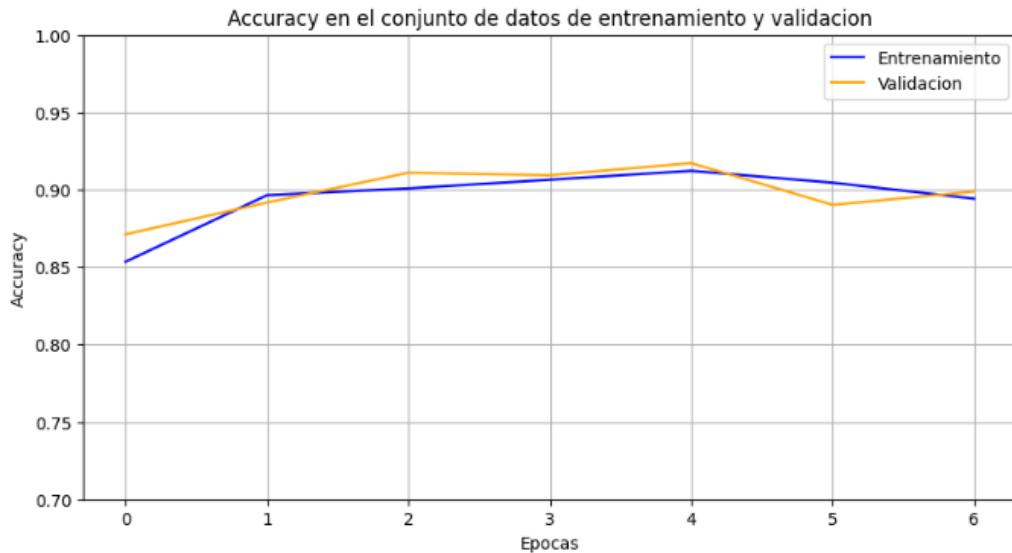


Figura 36: Comportamiento del accuracy en el entrenamiento arquitectura GRU

Según la gráfica de la función de pérdida 35 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 continua 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 perdida 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 36 vemos como 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.

6.2.5. Comparación de entrenamientos

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

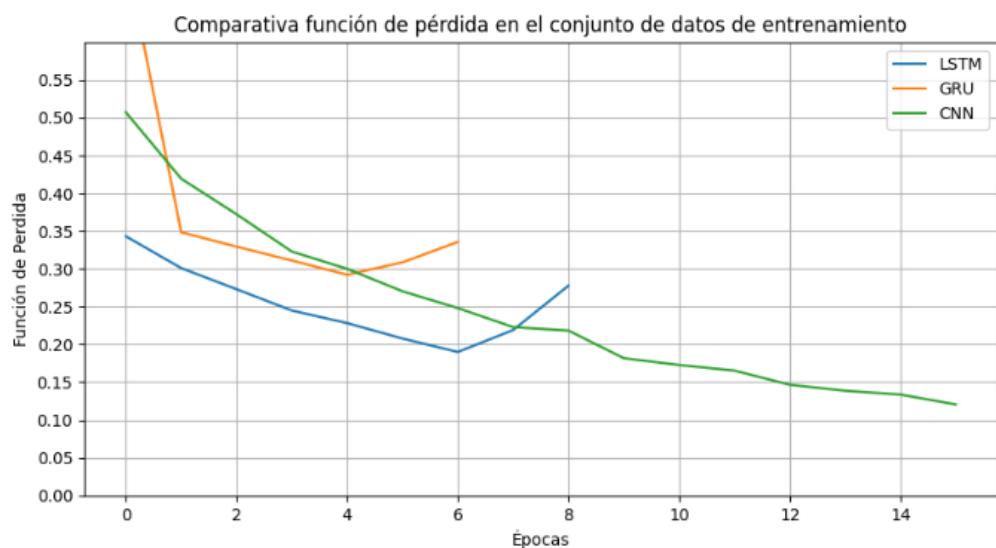


Figura 37: Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de entrenamiento

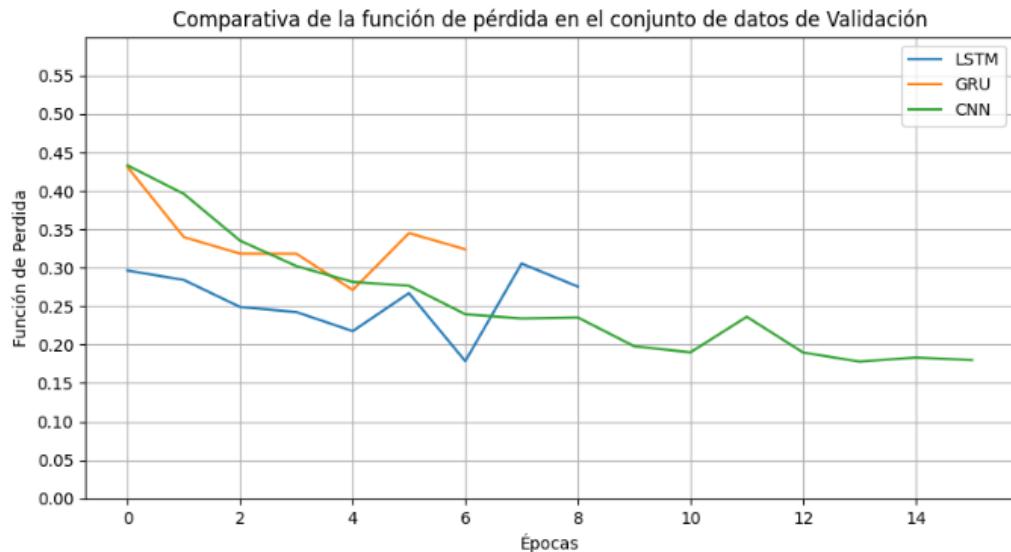


Figura 38: Comparación del entrenamiento de las Redes Neuronales: función de perdida en el conjunto de validación

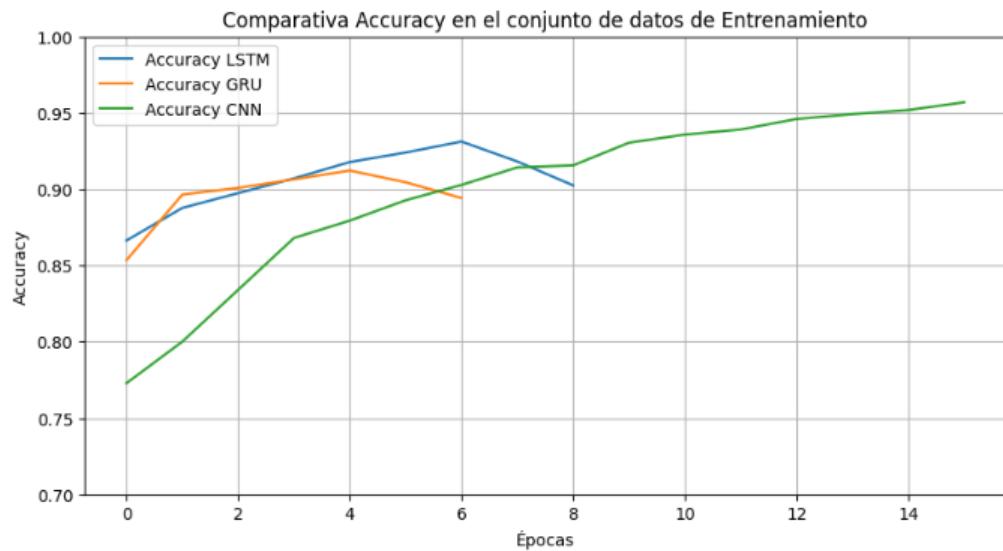


Figura 39: Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de entrenamiento

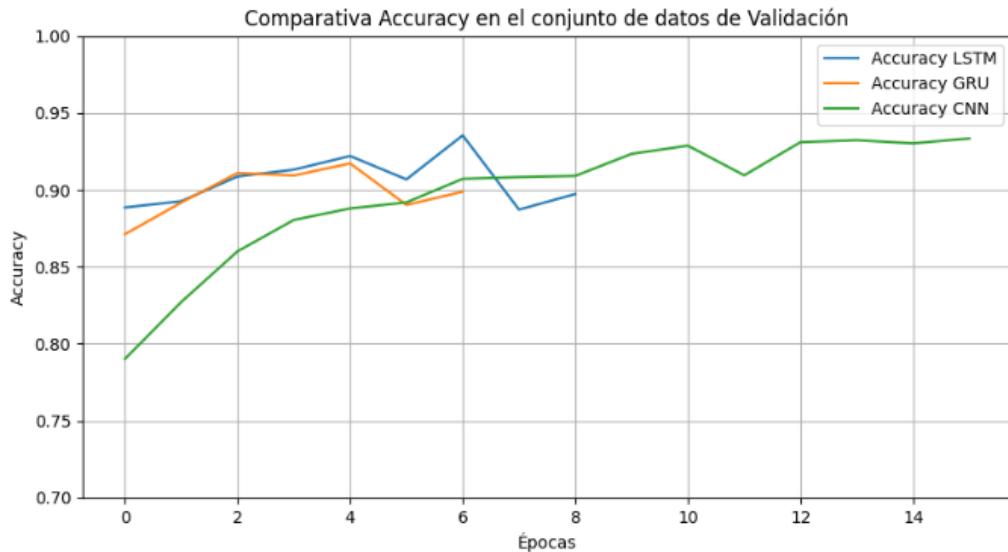


Figura 40: Comparación del entrenamiento de las Redes Neuronales: accuracy en el conjunto de validación

En cuanto al accuracy en el conjunto de datos de validación, vemos como 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 como

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.

La función de pérdida en el conjunto de datos de entrenamiento muestra como la red convolucional disminuye consistentemente a lo largo de las épocas, mostrando una tendencia a la baja y estabilizando en niveles bajos hacia las épocas finales. La arquitectura compuesta por las capas LSTM disminuye de manera constante, pero muestra más fluctuaciones que las red convolucionales. La arquitectura compuesta por las capas GRU disminuye de forma significativa, pero luego se estabiliza y muestra 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.

6.2.6. 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.

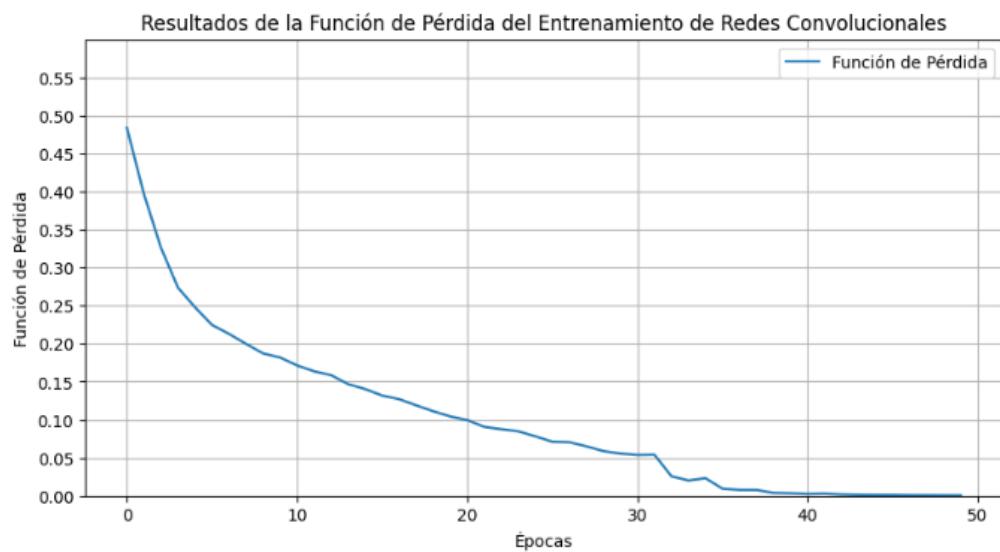


Figura 41: Función de pérdida del entrenamiento del mejor modelo (CNN)

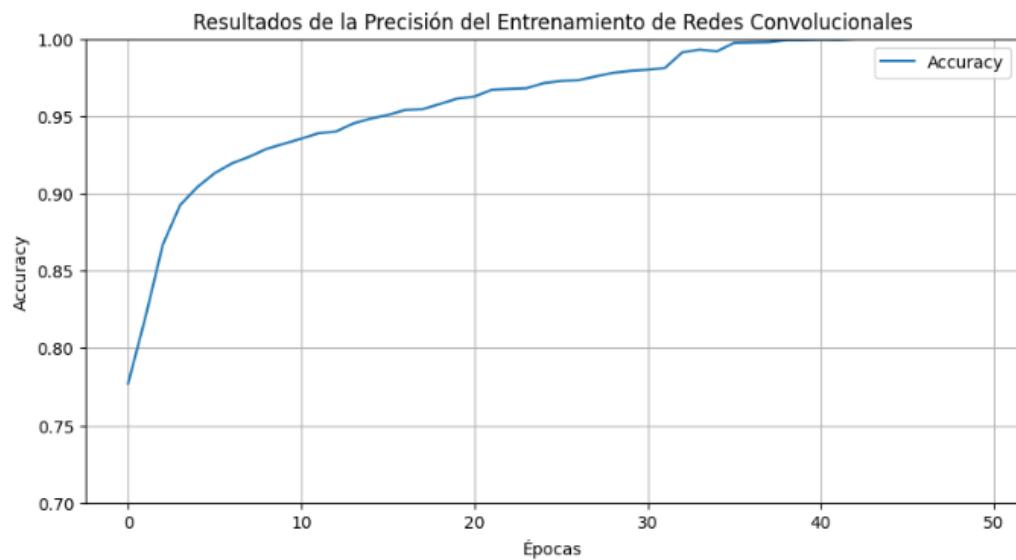


Figura 42: Accuracy del entrenamiento del mejor modelo (CNN)

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 41, 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 42 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.

6.2.7. 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

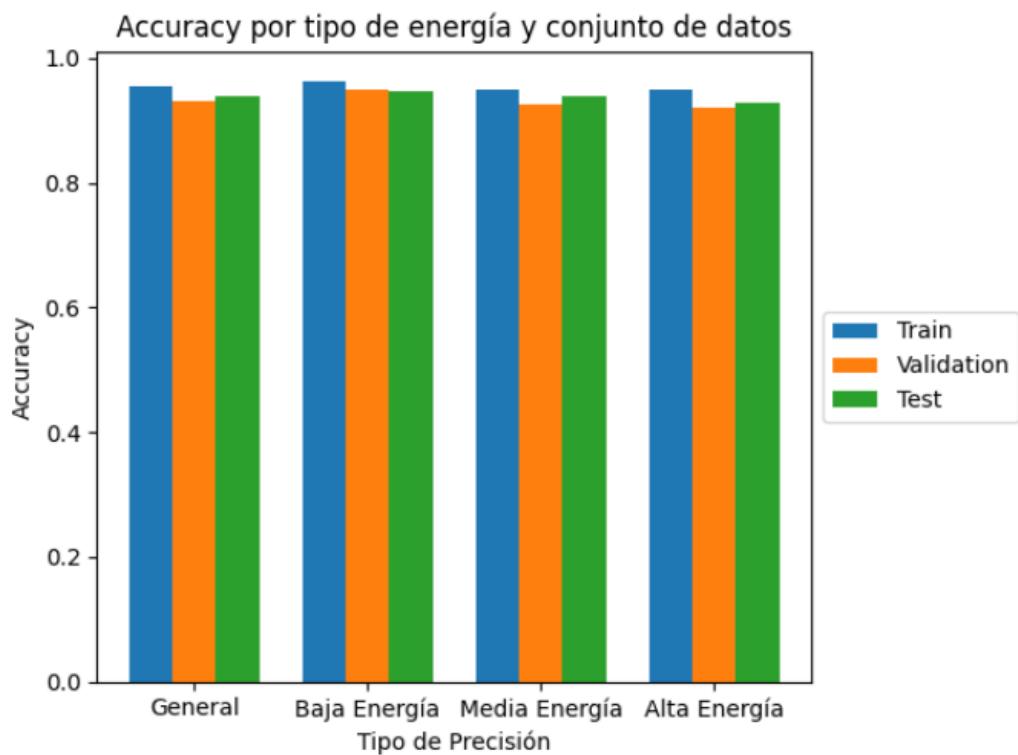


Figura 43: Resultados de accuracy en los distintos tipos de energías

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 evento el valor correspondiente a la variable *trueE*. Este valor, como se dijo en el apartado de clasificación es el mismo para un mismo evento. 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 16. 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 16, probaremos con esta misma estructura pero al principio de cada 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:

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*

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

Figura 44: Estructura de datos para el problema de regresión con modelos tradiciones de Aprendizaje Automático

7.2. Elección del número de hits por evento

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 eventos 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 17 y 18.

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. [19]

- **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 (α), 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. [16]
- **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 coeficiente, Lasso penaliza la suma de los valores absolutos de los coeficientes. Lasso elimina aquellas variables que no contribuyen al modelo. [36]
- **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. [10]
- **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]

Dentro de la elección de todos los algoritmos anteriores 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 correcto, si no también configurarlo a través de los hiperparámetros correctamente.

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. [28]

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 formula es la siguiente: $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$ [31]
- **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:

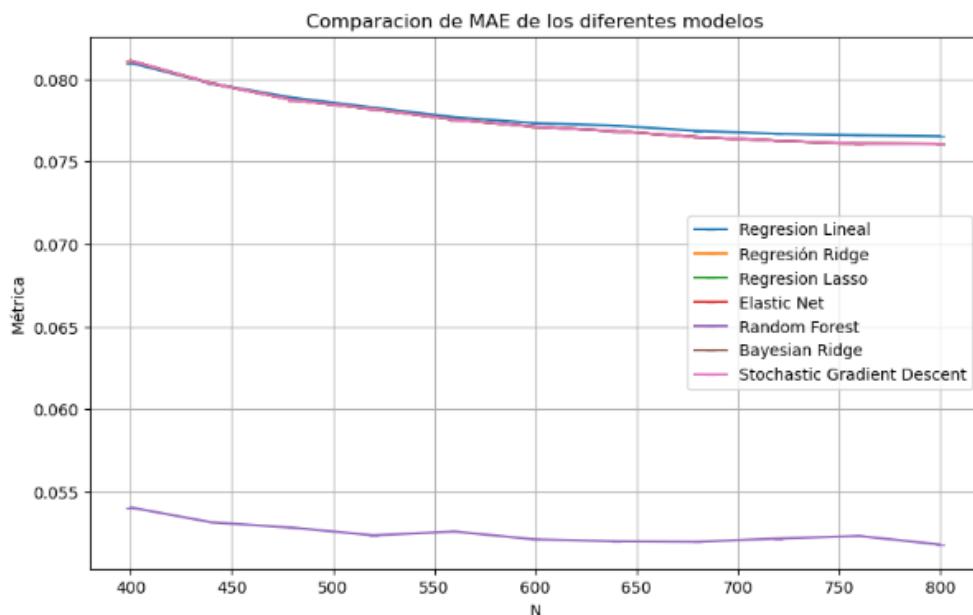


Figura 45: Error Absoluto Medio

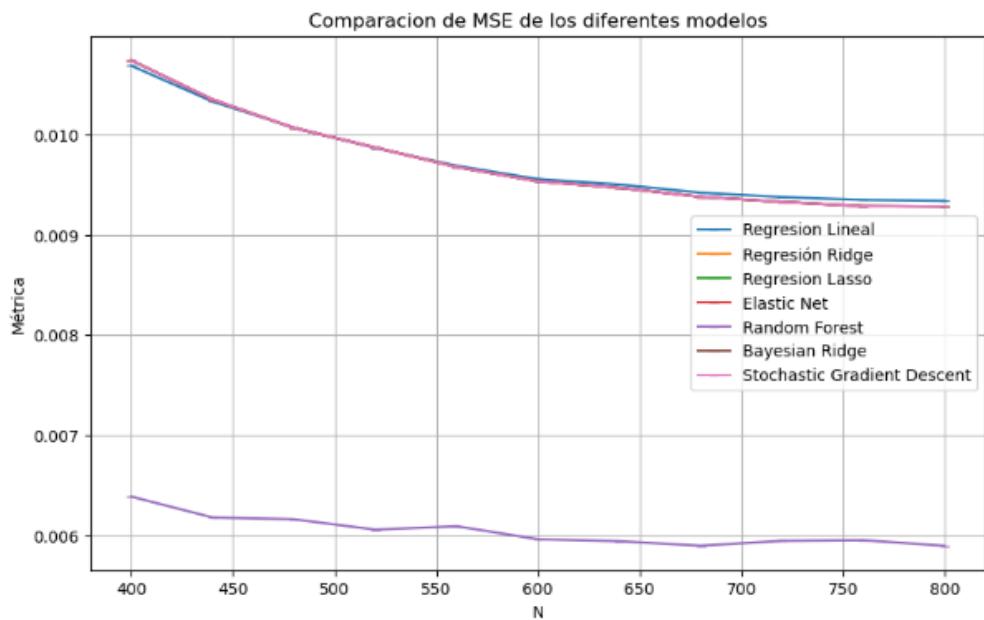


Figura 46: Error Cuadrático Medio

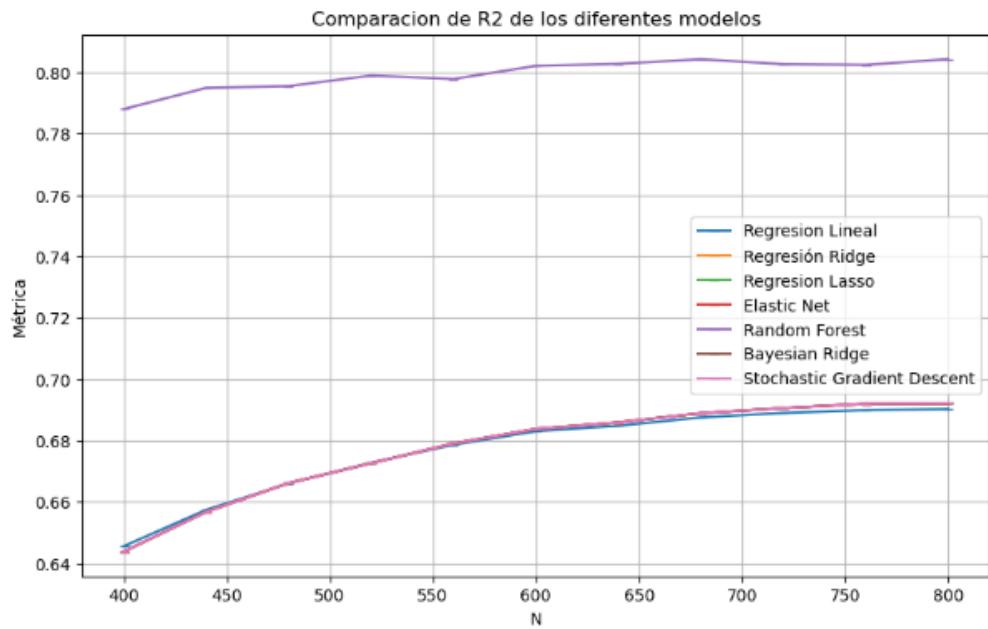


Figura 47: Coeficiente de Determinación

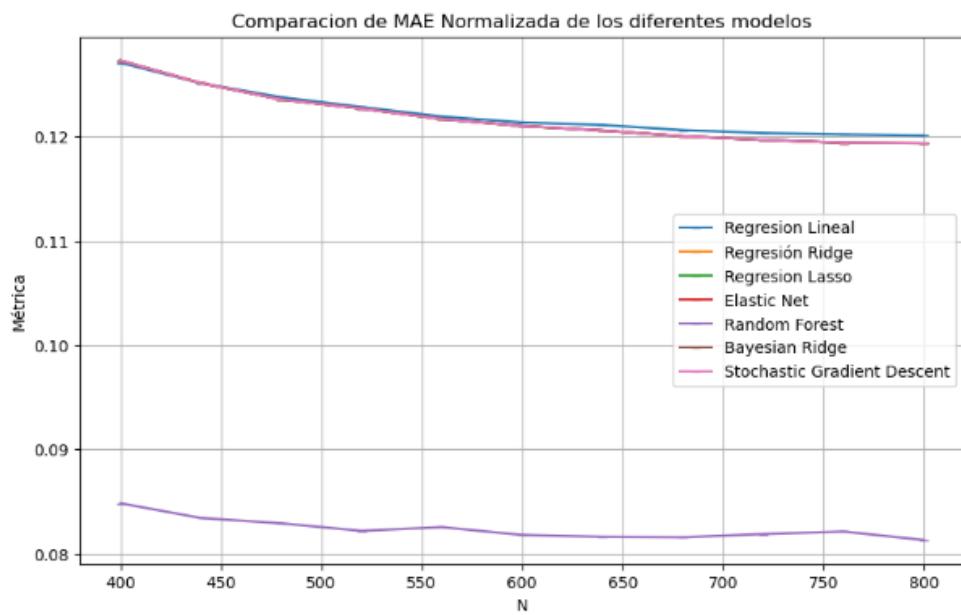


Figura 48: Error Absoluto Medio Normalizado

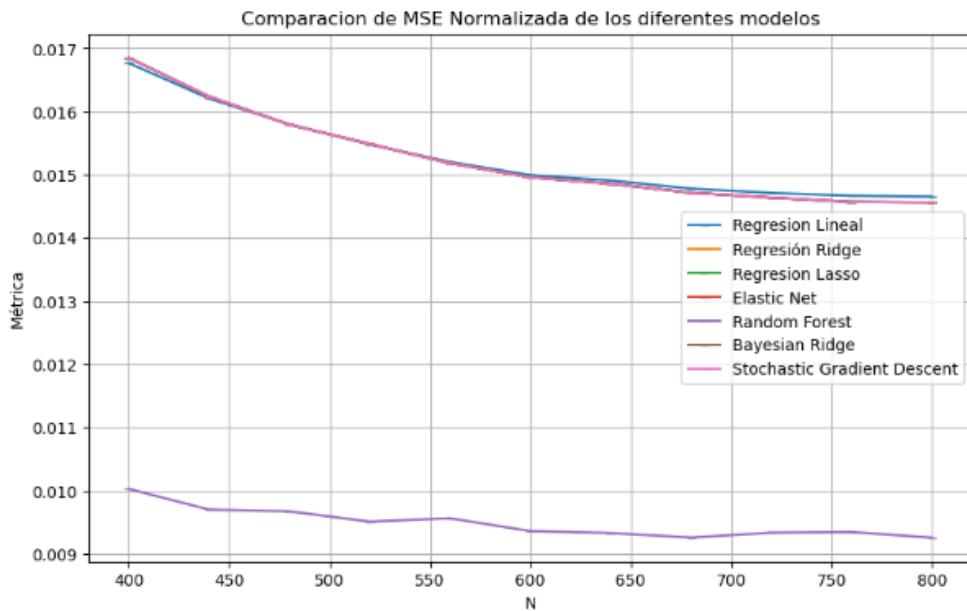


Figura 49: Error Cuadrático Medio Normalizado

El MAE y su versión normalizada (figuras 45 y 48) 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 46 y 49) 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 47) 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:

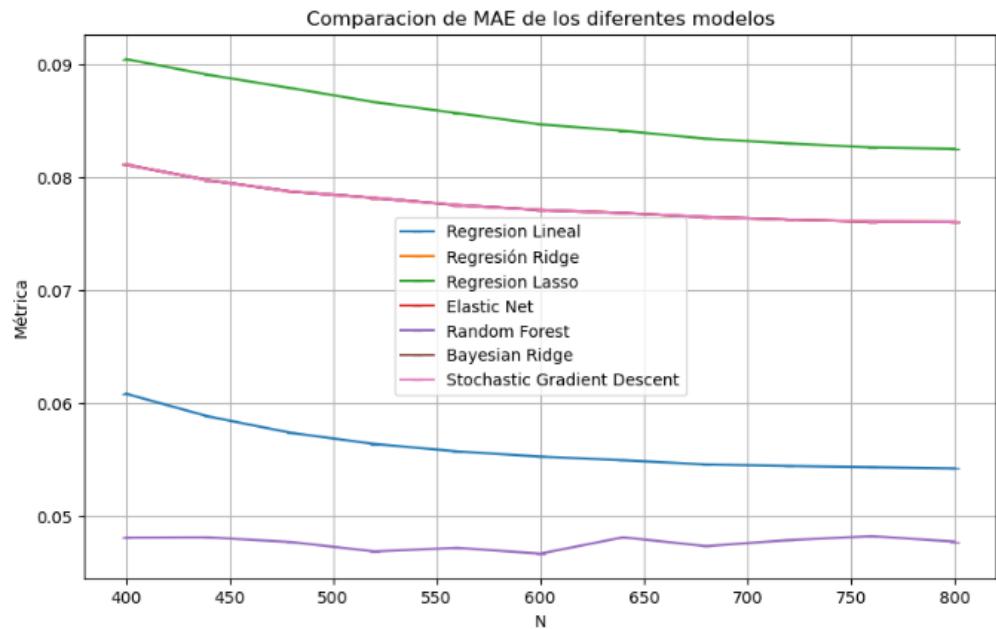


Figura 50: Error Absoluto Medio

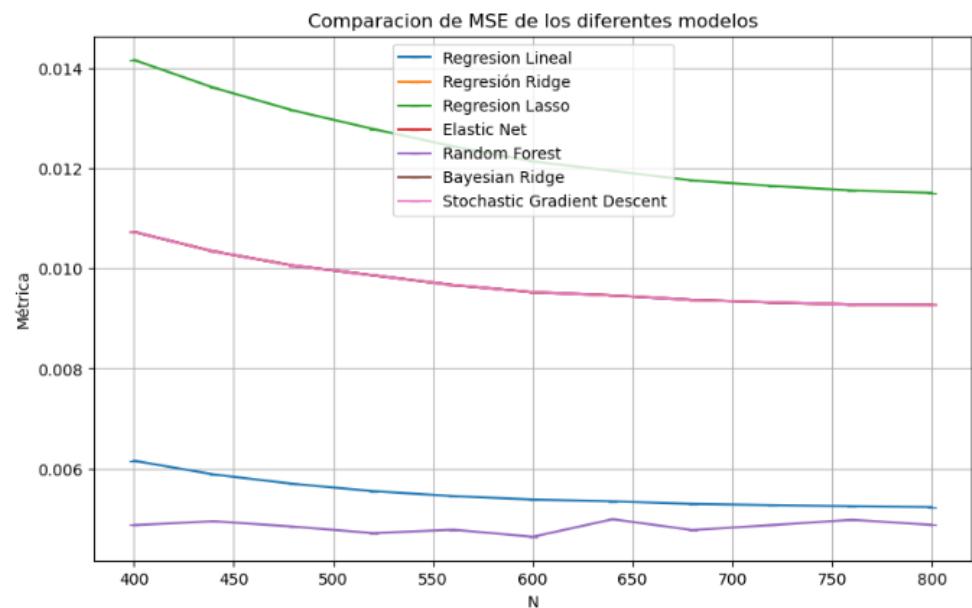


Figura 51: Error Cuadrático Medio

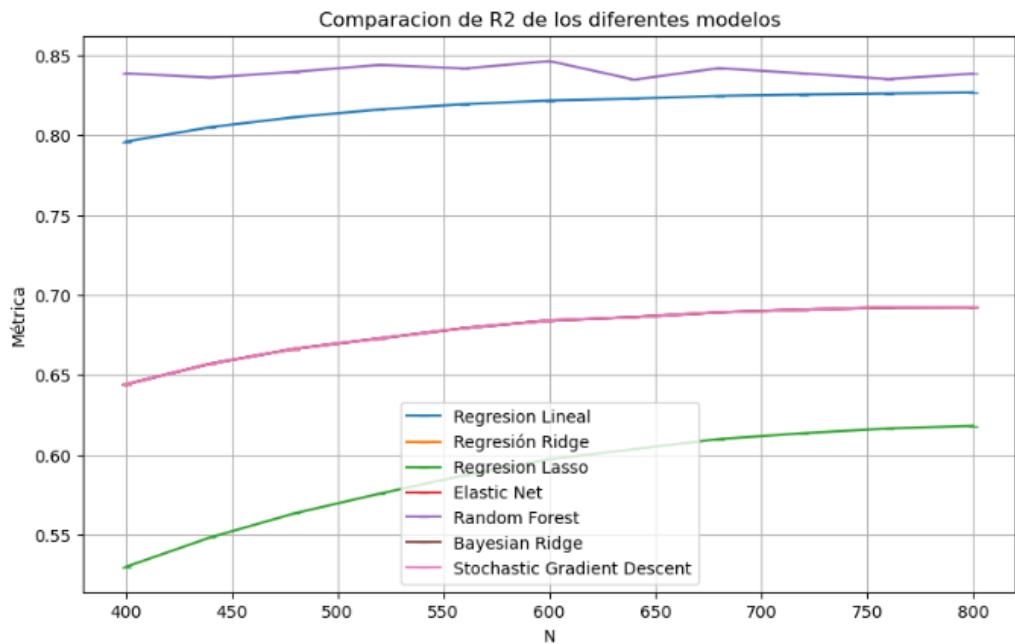


Figura 52: Coeficiente de Determinación

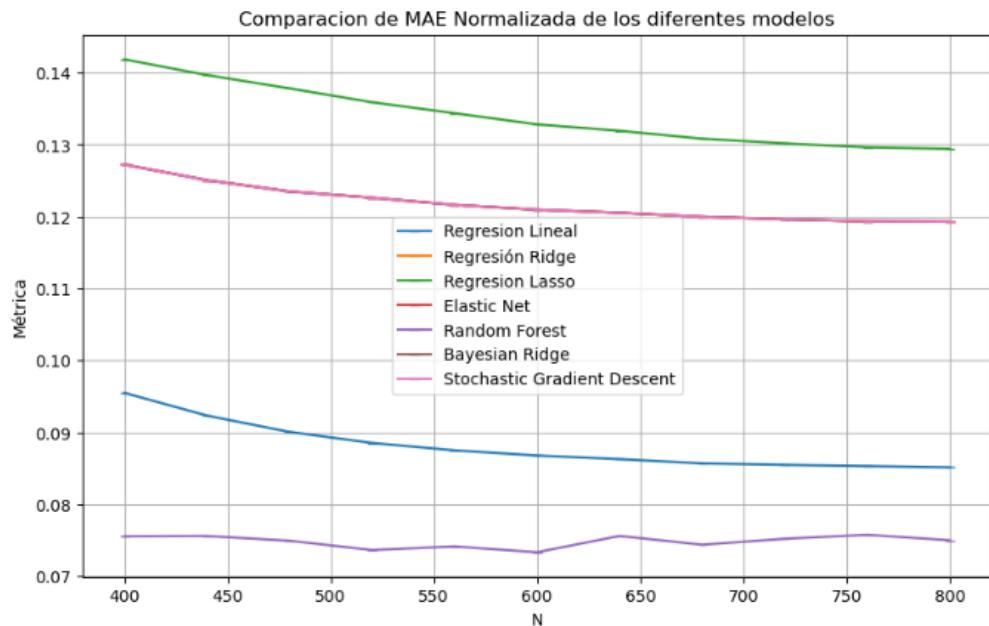


Figura 53: Error Absoluto Medio Normalizado

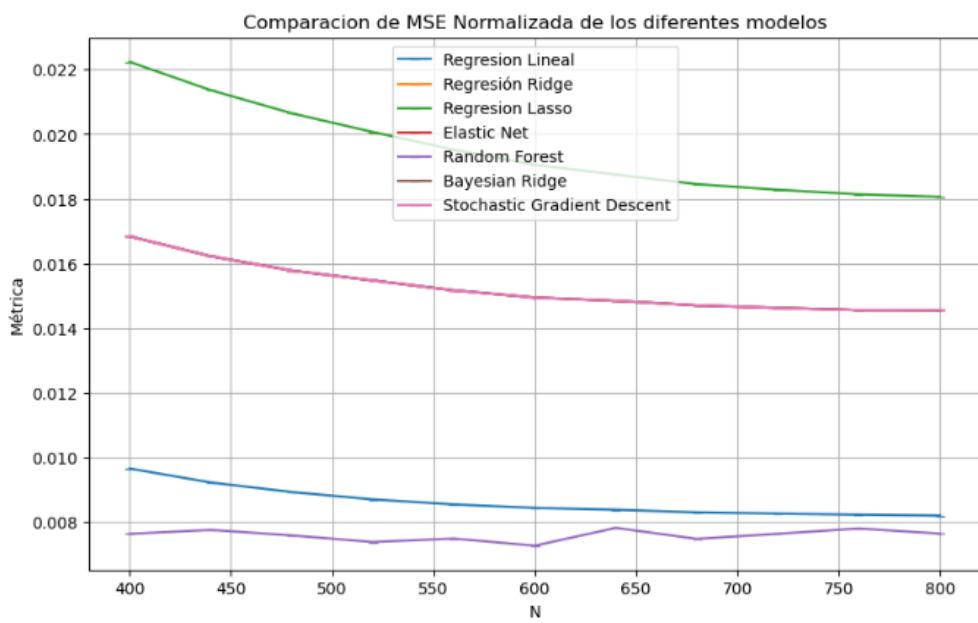


Figura 54: Error Cuadrático Medio Normalizado

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 50 y 53), 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 51 y 54), 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 modelos presentan un rendimiento inferior.

En cuanto a R^2 (figura 52), 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:

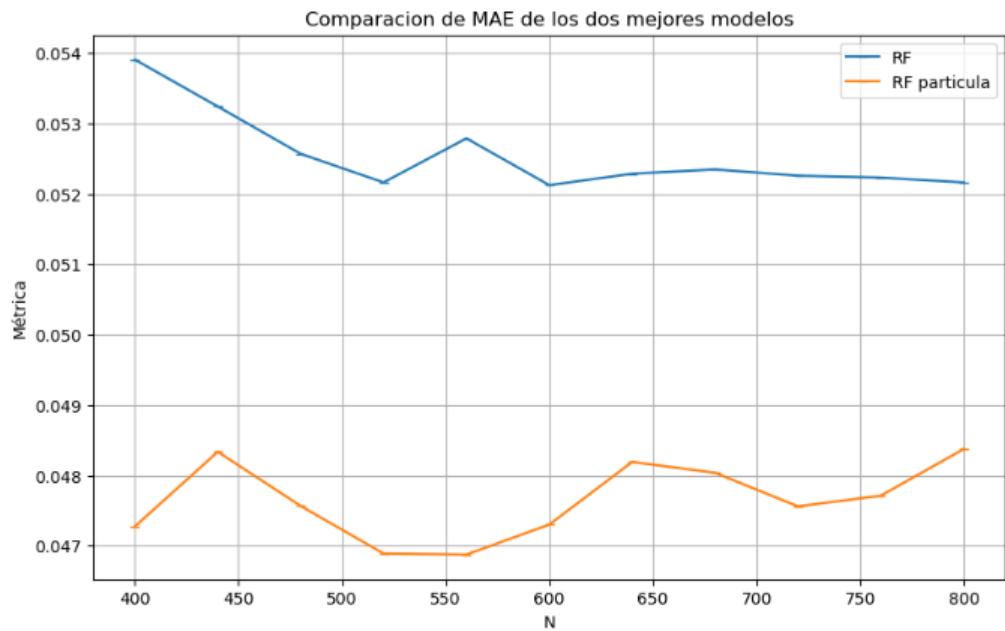


Figura 55: Comparativa para Random Forest del Error Absoluto Medio

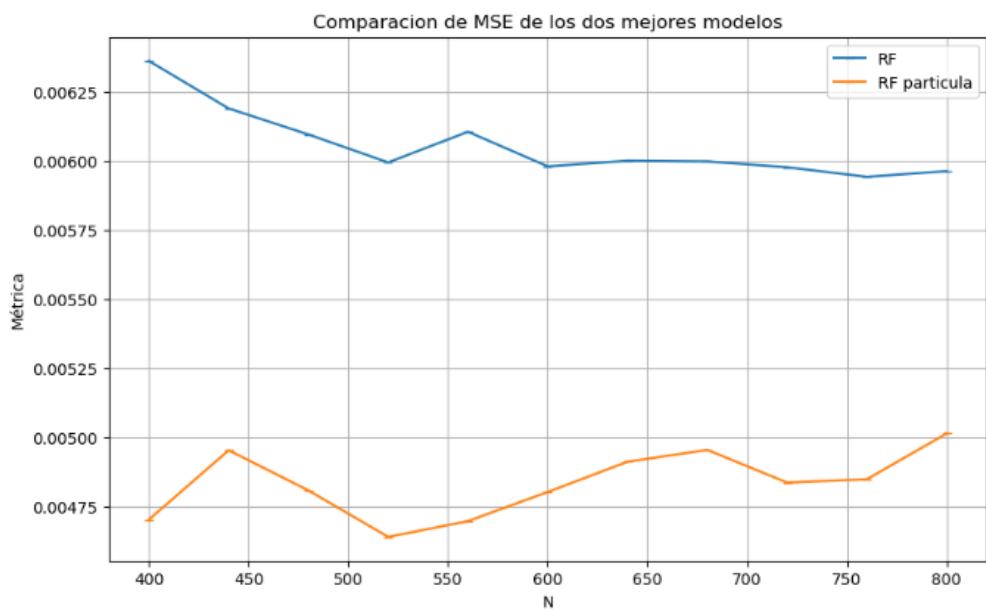


Figura 56: Comparativa para Random Forest del Error Cuadrático Medio

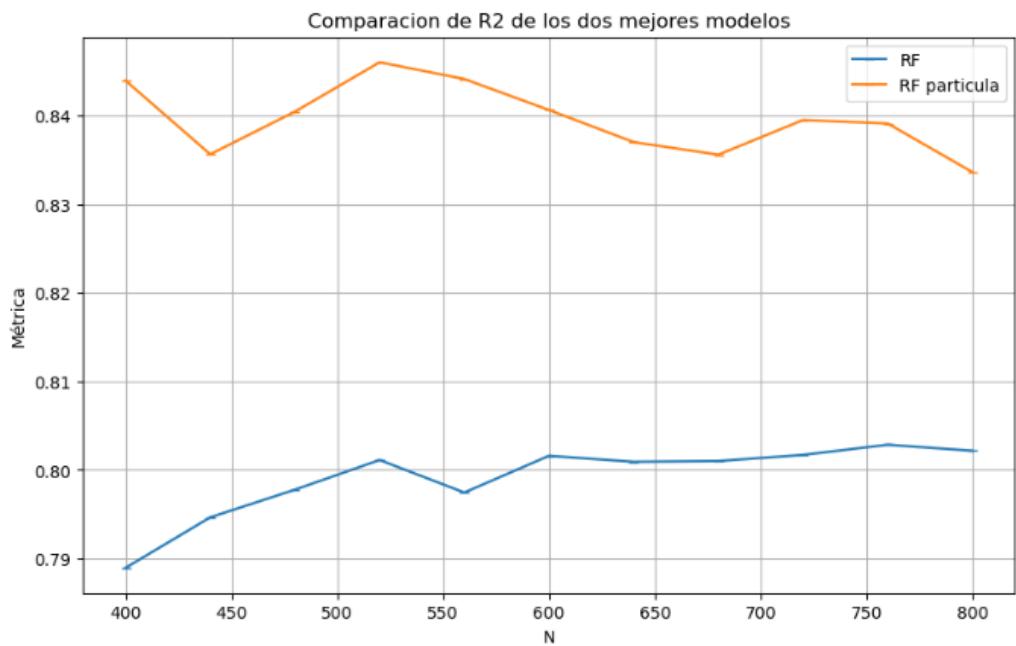


Figura 57: Comparativa para Random Forest del Coeficiente de Determinación

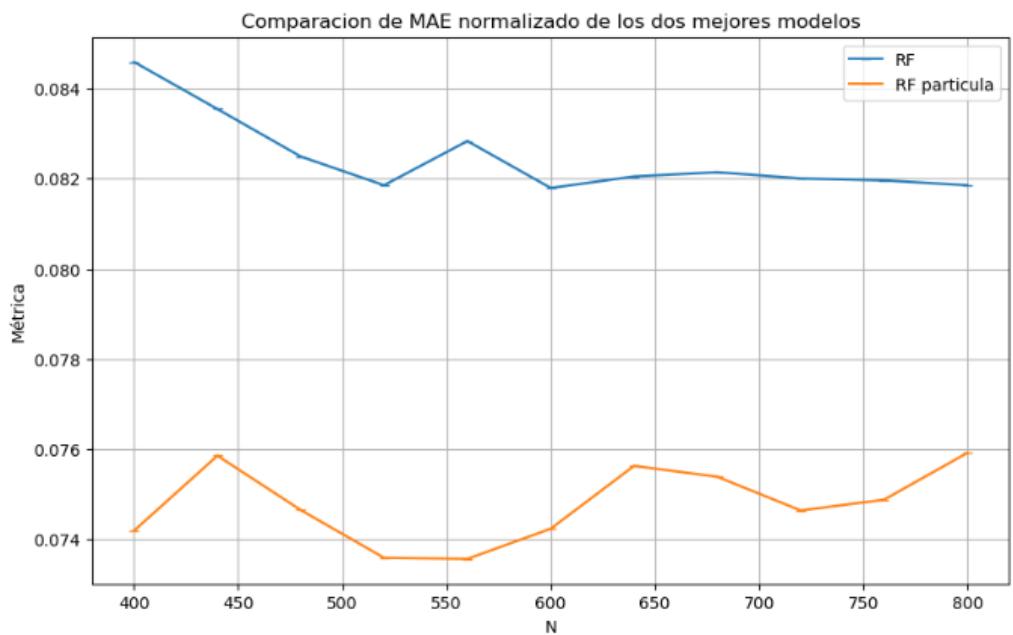


Figura 58: Comparativa para Random Forest del Error Absoluto Medio Normalizado

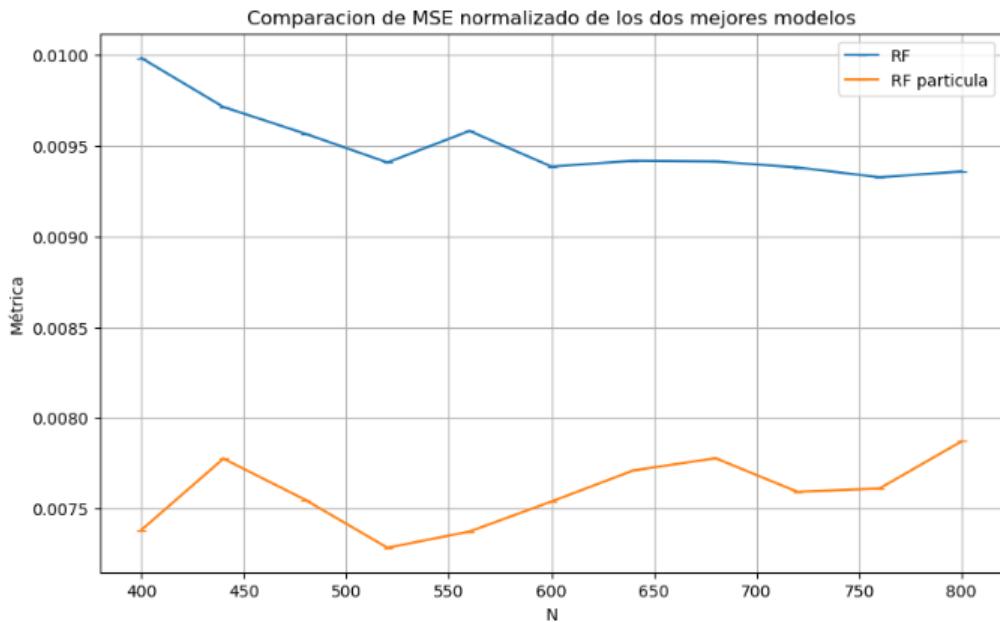


Figura 59: Comparativa para Random Forest del Error Cuadrático Medio Normalizado

Comparando el MSE normalizado (figura 59) 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 56) 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 55) 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 58), 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 57), 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 explíca 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. 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).

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

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 ?? y en la tabla 8 podemos ver los resultados obtenidos por el mejor modelos de clasificación dentro de todos los modelos de Aprendizaje Automático tradicionales. En la figura 43 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 [29] 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*.

Y para ejecutar la API localmente, se utilizará *Uvicorn* [38]. 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 script 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 XGBoost, 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.

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 60: Coste maquina virtual con GPU

Según la figura 60 el coste son 56 dolares al mes, para la realización de este TFG, con el plazo de un mes seria suficiente para realizar todas las pruebas oportunas. Estos 56 dolares 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	x €	x €
8	Horas Bruno	x €	x €
300	Horas Rubén	x €	x €
1	Maquina con GPU	51.77 €	51.77 €
1	Equipo con CPU	479.00€	479.00€
Total			x €

Cuadro 16: Horas de trabajo de los tutores

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 y redes neuronales. A lo largo del desarrollo del Trabajo de 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, superan significativamente en términos de precisión y capacidad de generalización. Esto se debe a su habilidad para capturar características complejas y patrones intrincados en 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 de regresión eficiente y accesible. La implementación de scripts en Python para la 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.

Referencias

- [1] Juan Alvear Alonso-Villaverde. *Métodos de entrenamiento de redes neuronales basados en inyección de ruido*. Trabajo Fin de Grado, Universidad Autónoma de Madrid. 2020. URL: https://repositorio.uam.es/bitstream/handle/10486/688275/alvear_alonso_villaverde_juan_tfg.pdf?sequence=1&isAllowed=y.
- [2] Jolice P. van den Berg et al. “Lower Transplacental Antibody Transport for Measles, Mumps, Rubella and Varicella Zoster in Very Preterm Infants”. En: *PLOS ONE* 9.4 (abr. de 2014), e94714. DOI: 10.1371/journal.pone.0094714. URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0094714>.
- [3] Ricardo Borja-Robalino, Antonio Monleón-Getino y José Rodellar. “Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning”. En: *ResearchGate* (2020). [Online; accessed 23-May-2024], págs. 172-18. URL: https://www.researchgate.net/publication/339943922_Estandarizacion_de_Metricas_de_Rendimiento_para_Clasificadores_Machine_y_Deep_Learning.
- [4] Léon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. En: *Proceedings of COMPSTAT’2010*. Paris, France: Physica-Verlag HD, 2010, págs. 177-186. URL: <https://leon.bottou.org/publications/pdf/compstat-2010.pdf>.
- [5] Leo Breiman. “Random Forests”. En: *Machine Learning* 45 (2001). Ed. por Robert E. Schapire, págs. 5-32. URL: <https://link.springer.com/article/10.1023/A:1010933404324>.
- [6] CERN. *Geant4 - A Toolkit for the Simulation of the Passage of Particles Through Matter*. [Online; accessed 23-May-2024]. 2024. URL: <https://geant4.web.cern.ch/>.
- [7] T. Chai y R. R. Draxler. “Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? – Arguments Against Avoiding RMSE in the Literature”. En: *Geoscientific Model Development* 7 (2014), págs. 1247-1250. DOI: 10.5194/gmd-7-1247-2014. URL: <https://gmd.copernicus.org/articles/7/1247/2014/gmd-7-1247-2014.pdf>.
- [8] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. En: *arXiv preprint arXiv:1406.1078* (2014). URL: <https://arxiv.org/abs/1406.1078>.

- [9] DUNE Collaboration. *Deep Underground Neutrino Experiment*. [Online; accessed 23-May-2024]. 2024. URL: <https://www.dunescience.org/>.
- [10] Achmad Efendi. “A Simulation Study on Bayesian Ridge Regression Models for Several Collinearity Levels”. En: *AIP Conference Proceedings*. Vol. 1913. 1. Dic. de 2017, pág. 020031. DOI: 10.1063/1.5016665. URL: https://www.researchgate.net/publication/321640059_A_simulation_study_on_Bayesian_Ridge_regression_models_for_several_collinearity_levels.
- [11] Esri. *Documentación XGBoost*. [Online; accessed 23-May-2024]. 2024. URL: <https://pro.arcgis.com/es/pro-app/latest/tool-reference/geoai/how-xgboost-works.htm>.
- [12] Fermilab. *Detectors and Computing - How It Works*. [Online; accessed 23-May-2024]. 2024. URL: <https://lbnf-dune.fnal.gov/how-it-works/detectors-and-computing/>.
- [13] Fermilab News. *LBNF/DUNE Brochure (Spanish)*. [Online; accessed 23-May-2024]. 2024. URL: <https://news.fnal.gov/wp-content/uploads/lbnf-dune-brochure-spanish.pdf>.
- [14] María Elvira Ferre Jaén. *FEIR 45: Regresión logística*. Apuntes del curso FEIR3, curso 2014/15 actualizados. Última actualización: jueves 04 abril 2019, 23:30:47. Abr. de 2019. URL: <https://gauss.inf.um.es/feir/45/>.
- [15] Sepp Hochreiter y Jürgen Schmidhuber. “Long Short-Term Memory”. En: *Neural Computation* 9.8 (1997). [Online; accessed 23-May-2024], págs. 1735-1780. URL: <https://www.bioinf.jku.at/publications/older/2604.pdf>.
- [16] Arthur E. Hoerl y Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems”. En: *Technometrics* 12.1 (feb. de 1970), págs. 55-67. URL: <https://homepages.math.uic.edu/~lreyzin/papers/ridge.pdf>.
- [17] M. Hossin y M.N. Sulaiman. “A Review on Evaluation Metrics for Data Classification Evaluations”. En: *International Journal of Data Mining & Knowledge Management Process (IJDkp)* 5.2 (mar. de 2015). [Online; accessed 23-May-2024], págs. 1-11. DOI: 10.5121/ijdkp.2015.5201. URL: <https://www.airconline.com/ijdkp/V5N2/5215ijdkp01.pdf>.
- [18] IBM. ¿Qué es el algoritmo de k vecinos más cercanos? [Online; accessed 23-May-2024]. 2024. URL: <https://www.ibm.com/es-es/topics/knn>.

- [19] IBM. *¿Qué es la regresión lineal?* [Online; accessed 23-May-2024]. 2024. URL: <https://www.ibm.com/es-es/topics/linear-regression>.
- [20] IBM. *What are Support Vector Machines (SVMs)?* [Online; accessed 23-May-2024]. Dic. de 2023. URL: <https://www.ibm.com/topics/support-vector-machine>.
- [21] Shruti Jadon. “A Survey of Loss Functions for Semantic Segmentation”. En: *arXiv preprint arXiv:2006.14822* (2020). URL: <https://arxiv.labs.arxiv.org/html/2006.14822>.
- [22] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. En: *Advances in Neural Information Processing Systems* 30 (2017). Published by Microsoft Research and Peking University. URL: https://papers.nips.cc/paper_files/paper/2017/file/f67130c606d8093d4fa45e49a25a8af0-Paper.pdf.
- [23] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. En: *arXiv preprint arXiv:1408.5882* (2014). Submitted on 25 Aug 2014 (v1), last revised 3 Sep 2014 (this version, v2). URL: <https://arxiv.org/abs/1408.5882>.
- [24] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. En: (2012), págs. 1097-1105. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
- [25] Lightly AI. *Train Test Split in Deep Learning*. [Online; accessed 23-May-2024]. 2024. URL: <https://www.lightly.ai/post/train-test-split-in-deep-learning>.
- [26] Min Lin, Qiang Chen y Shuicheng Yan. “Network In Network”. En: *arXiv preprint arXiv:1312.4400* (2013). URL: <https://arxiv.org/abs/1312.4400>.
- [27] Alexey Natekin y Alois Knoll. “Gradient Boosting Machines, A Tutorial”. En: *Frontiers in Neurorobotics* 7 (dic. de 2013), pág. 21. DOI: 10.3389/fnbot.2013.00021. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/>.
- [28] Optuna. *Optuna: A hyperparameter optimization framework*. [Online; accessed 23-May-2024]. 2024. URL: <https://optuna.org/>.
- [29] Sebastián Ramírez. *FastAPI: The modern web framework for your APIs*. [Online; accessed 23-May-2024]. 2024. URL: <https://fastapi.tiangolo.com/>.

- [30] Payam Refaeilzadeh, Lei Tang y Huan Liu. “Cross-Validation”. En: (2009). Arizona State University, [Online; accessed 23-May-2024]. URL: <http://leitang.net/papers/ency-cross-validation.pdf>.
- [31] Scott M. Robeson y Cort J. Willmott. “Decomposition of the Mean Absolute Error (MAE) into Systematic and Unsystematic Components”. En: *PLOS ONE* (feb. de 2023), e0279774. DOI: 10.1371/journal.pone.0279774. URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0279774>.
- [32] Lior Rokach y Oded Maimon. “Decision Trees”. En: (2024). [Online; accessed 23-May-2024]. URL: https://www.researchgate.net/publication/225237661_Decision_Trees.
- [33] SAS Institute Inc. *SAS Visual Analytics 8.3: Working with Report Objects*. [Online; accessed 23-May-2024]. 2024. URL: https://documentation.sas.com/doc/es/vacdc/v_025/vaobj/p02eoijulgpgow6n1aqa0q8cowydp.htm.
- [34] Doug Steen. “Understanding the ROC Curve and AUC: These binary classification performance measures go hand-in-hand — let’s explore”. En: (sep. de 2020). Published in Towards Data Science, [Online; accessed 23-May-2024]. URL: <https://towardsdatascience.com/understanding-the-roc-curve-and-auc-dd4f9a192ecb>.
- [35] T. Tao et al. “Statistical Properties of the Number of Muons in Extensive Air Showers and Its Relation to Primary Particle Mass”. En: *Physical Review D* 82.3 (2010).
- [36] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso”. En: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), págs. 267-288. URL: [https://webdoc.agsci.colostate.edu/koontz/arec-econ535/papers/Tibshirani%20\(JRSS-B%201996\).pdf](https://webdoc.agsci.colostate.edu/koontz/arec-econ535/papers/Tibshirani%20(JRSS-B%201996).pdf).
- [37] Universitat de València. *Material didáctico sobre Multicolinealidad*. [Online; accessed 23-May-2024]. 2024. URL: <https://www.uv.es/uriel/material/multicolinealidad3.pdf>.
- [38] Uvicorn Developers. *Uvicorn: The lightning-fast ASGI server implementation, using ‘uvloop’ and ‘httptools’*. [Online; accessed 23-May-2024]. 2024. URL: <https://www.uvicorn.org/>.