Clasificador del Bosón de Higgs utilizando Naive Bayes, Árbol de Decisión y Deep Learning

Sebastian Peralta
Ciencias de la Computación
Universidad Peruana de Ciencias Aplicadas
u201816030@upc.edu.pe

Francesco Bassino
Ciencias de la Computación
Universidad Peruana de Ciencias Aplicadas
u201816649@upc.edu.pe

***Abstract*—El propósito de este paper para el curso de Data Mining es poder clasificar de manera correcta las partículas del Bosón de Higgs. Para esto utilizamos 3 diferentes frentes para poder luego comparar sus métodos y resultados (Acurracy) y determinar cual es el mejor para este caso. A lo largo del trabajo se mencionó sobre que es el Bosón de Higgs (y su relevancia en la ciencia), así también sobre el análisis exploratorio de los datos previo al entrenamiento para así entender de mejor manera los datos expuestos en el dataset. Se realizaron tres diferentes entrenamientos en los que se obtuvo los siguientes resultados: Naive Bayes obtuvo un accuracy de: 66,8%, Árbol de Decisión obtuvo 76,1% y por último el modelo creado desde cero de Deep Learning obtuvo un máximo de 85% siendo el mejor para la clasificación del Bosón de Higgs.**

I. Introducción

Para poder desarrollar este trabajo, primero es importante conocer y entender lo mejor posible que es un Bosón de Higgs, como funciona, que aporta y sobre todo, como es posible detectarlo. El ultimo es fundamental para nuestro trabajo, ya que de esa manera podremos interpretar mejor los datos de entrada y podremos comprobar de manera efectiva las clasificaciones que nos arrojen los tres diferentes modelos con los cuales vamos a predecir (que se hablarán mas a detalle en la siguiente sección del paper).

Un Bosón de Higgs [1] es un tipo de partícula elemental que se cree tiene un papel fundamental en el mecanismo por el cual se origina la masa en todo el universo. Todavía no se puede saber a ciencia cierta si existe o no. Sin embargo, uno de los principales objetivos del LHC (Gran Colisionador de Hadrones) es poder confirmar o refutar dicha existencia. Este no es un trabajo fácil, ya que el Bosón de Higgs no se puede detectar directamente ya que una vez se crea, prácticamente de manera instantánea se desintegra, creando así otras partículas elementales. Por consiguiente, solo se puede detectar sus ”huellas” las cuales son esas partículas elementales creadas. Se puede apreciar en la Figura 1 como es que el LHC trabaja. Se puede ver las partículas elementales creadas por el Bosón se esparcen por todo el generador.

Con toda esta información, en este trabajo se pretende predecir todos los parámetros que debe tener una colisión para producir un Bosón de Higgs utilizando tres tipos de técnicas. La primera es el algoritmo de Naive Bayes, luego el Árbol de Decisión y por ultimo una Red Neuronal utilizando Deep Learning. Luego compararemos los resultados obtenidos por cada modelo, así como su tiempo de respuesta para poder sustentar cual sería la mejor opción.

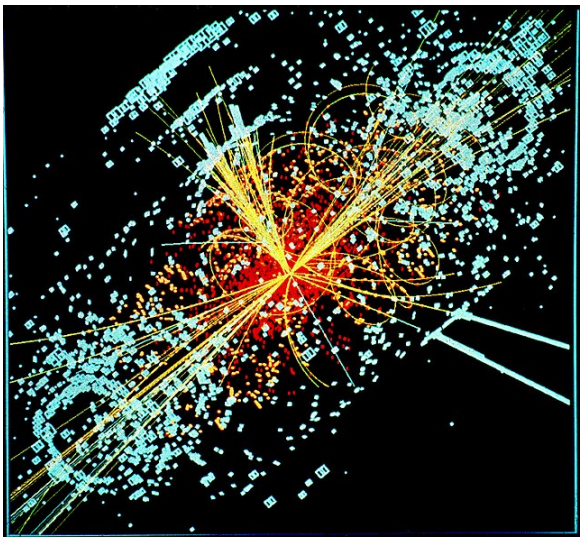

Fig. 1. Imagen representativa del Bosón de Higgs dentro de un colisionador de partículas

II. Marco Teórico

Pese a que se comparará tres diferentes modelos, el mas importante es la Red Neuronal, por lo que se abordará con mas profundidad, dando mas detalle del modelo y sus resultados.

Para desarrollar este trabajo se optará por usar el lenguaje de programación python ya que nos brinda la posibilidad de hacer el análisis exploratorio de los datos, el preprocesamiento y el modelado todo de manera sencilla y eficiente. Asimismo, el trabajo se desarrollará en Google Colab la cual es una herramienta que permite poder ejecutar lineas de código por separado, además de proporcionar un link en donde cualquiera que entre puede ver todo el código desde cualquier navegador. Además, se utilizarán librerías para el análisis y manipulación de datos como:

- Pandas
- Matplotlib
- Numpy

En cuanto a los modelos, se utilizaron las librerias de:

- Sklearn: para modelos de Naive Bayes y Decision Tree.

- Tensorflow: para implementar red neuronal desde cero (DNN).

A continuación se hablará mas sobre los tres metododos de clasificación:

A. *Naive Bayes*

Naive Bayes es, como dice Mosquera. R (2017) [2], un clasificador probabilístico simple con fuerte suposición de independencia. A menudo proporciona una mejor precisión de clasificación en conjuntos de datos en tiempo real que cualquier otro clasificador. También requiere una pequeña cantidad de datos de entrenamiento. El clasificador Naive-Bayes aprende de los datos de entrenamiento y luego predice la clase de la instancia de prueba con la mayor probabilidad posterior. La formula de Naive Bayes es la siguiente:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

Donde:

- $P(c|x)$: Probabilidad que c ocurra dado que x ya ocurrió
- $P(x|c)$: Probabilidad que x ocurra dado que c ya ocurrió
- $P(c)$: Probabilidad que c ocurra
- $P(x)$: Probabilidad que x ocurra

B. *Decision Tree*

Los Árboles de Decisión son [3] modelos predictivos formados por reglas binarias (si/no) con las que se consigue repartir las observaciones en función de sus atributos y predecir así el valor de la respuesta. Este método para la clasificación es uno de los más populares debido a su versatilidad y sus buenos resultados. Algunas ventajas sobre este modelo son: Fáciles de interpretar aun cuando las relaciones entre predictores son complejas, no se necesita ningún tipo de distribución específica, no se ven influenciados por outliers, entre otras ventajas. Sin embargo, también cuenta con algunas desventajas, algunas son: Son sensibles a datos de entrenamiento desbalanceados, no son capaces de extrapolar fuera del rango de los predictores, entre otras.

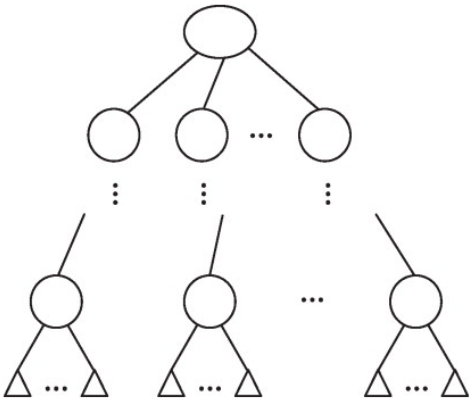

Fig. 2. Imagen representativa un Árbol de decisión

En la Figura 2 se puede ver como se representaría de manera gráfica un Árbol de Decisión. El cual de acuerdo a un criterio preestablecido se dirige a un camino u otro, y así sucesivamente hasta poder encontrar una respuesta al final del árbol.

C. *Red Neuronal*

El objetivo principal de una red neuronal es [4] tratar de encontrar la relación entre las características de un conjunto de datos, y consiste en un conjunto de algoritmos que imitan el trabajo del cerebro humano. Una ”neurona” en una red neuronal es una función matemática que recoge y clasifica la información según una arquitectura específica.
Estas clasificaciones pueden ser de diferentes tipos como: binarias, multi-class o multi-label. En este caso, nosotros usaremos una clasificación binaria, ya que queremos hallar si con determinados parámetros existe o no un Bosón de Higgs.

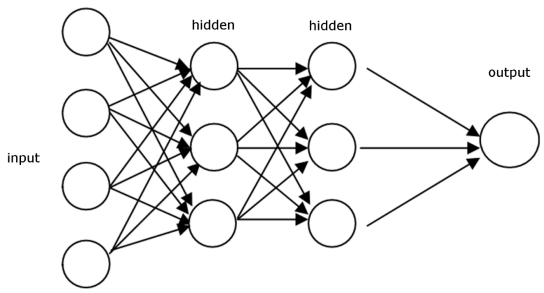

Fig. 3. Representación de Red Neuronal

La Figura 3 indica una red neuronal con dos capas ocultas. Además se puede ver que hay cuatro datos de entrada y solo uno de salida, siendo una red neuronal con clasificación binaria.
Asimismo, un factor muy importante son las funciones de activación que tomarán cada una de estas neuronas. Las funciones de activación [5] se utilizan para determinar la salida de la red neuronal como ”si” o ”no”. Asigna los valores resultantes entre 0 y 1 o -1 y 1, etc. (dependiendo de la función). Existen diferentes funciones para diferentes propósitos y complejidades. Algunas de las funciones de activación más usadas son: Linear, No-Lineal, Tan, Sigmoid, ReLu, entre otras.
En este caso nosotros utilizaremos dos funciones de transferencia en cuatro capas ocultas: tres capas usando $ReLu$ y una usando $Sigmoid$ 4. Esta configuración de de red neuronal es bastante común, pero es un buen comienzo ya que no queremos sobre-complicar el modelo ya que eso significaría mas cara computacional y mas tiempo.

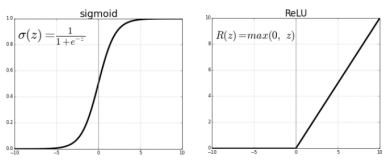

Fig. 4. Sigmoid v/s ReLu

III. Análisis y Exploración de los Datos

Los datos usados para este trabajo fueron sacados de: https://www.kaggle.com/competitions/higgs-boson/overview

A partir de la sección 3 hacia adelante, el paper estará basado en el Google Colab realizado por nosotros, en donde entramos mas a profundidad y estará todo el código junto con los modelos: https://drive.google.com/file/d/1e5h0HEKb-3SqZUWf4gYg_vH7B5sDKO84/view?usp=sharing

Antes de poder realizar cualquier tipo de modelado y entrenamiento es indispensable poder entender los datos que se usarán. Por eso realizar un análisis y preprocesamiento de los datos siempre será el primer paso ante cualquier proyecto de Deep Learning o Data Mining.
Al descargar el dataset $taining.csv$ para el entrenamiento y $test.csv$ para el testeo, podemos apreciar que se cuenta con diversas columnas que ayudarán al modelo a clasificar. Una de las columnas mas importantes es $Label$ ya que esa será la columna que nos diga si es o no un Bosón de Higgs, siendo s como un "sí".
Para efectos visuales se han mostrado las columnas mas relevantes:

	EventId	DER$_MMC$	:	:	Weight	Label
0	100000	138.47	:	:	0.002653311	s
1	100001	160.937	:	:	2.233584487	b
2	100002	-999	:	:	2.347388944	b
3	100003	143.905	:	:	5.446378212	b
4	100004	175.864	:	:	6.245332687	b

Sabiendo que la columna que nos dirá nuestra respuesta, es importante poder ver de forma gráfica cuantas tiene el valor de s y b ya que en el dataset hay 25000 filas 5 Se puede ver

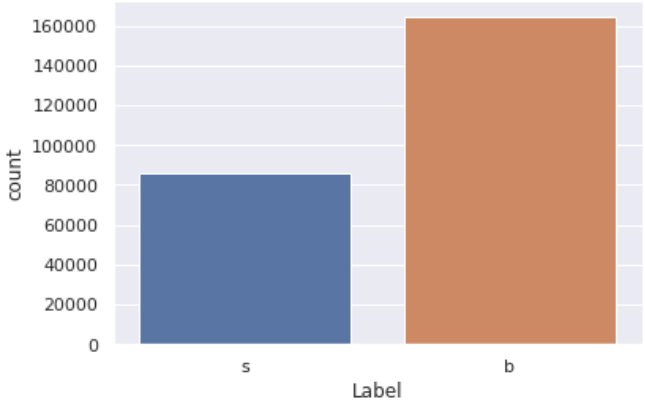

Fig. 5. s v/s b

como hay el doble de negativos que de positivos en el dataset $training.csv$ el cual nos servirá para entrenar los diferentes modelos.
Luego hacemos el preprocesamiento respectivo, el cual consiste en cambiar los valores de $Label$ por un números en vez de letras, ya que será mas fácil realizar el entrenamiento de los diferentes modelos. Asimismo, eliminaremos la columna $Weight$ ya que no se encuentra dentro del dataset $test.csv$ el cual como se mencionó nos servirá para poder validar la clasificación del modelo.

IV. Modelado

Una vez hecho el preprocesamiento podemos pasar a los clasificadores mencionados al inicio de trabajo. Para los dos primeros, como se comentó se usará sklearn ya que tiene los modelos listos para ser usados. Antes de ello, tenemos que dividir nuestra data entre validación y entrenamiento de manera aleatoria, para que los resultados sean lo mas fidedignos posibles.

A. *Naive Bayes*

Para utilizar el modelo de Naive Bayes lo único que se tiene que hacer es llamar a la función $GuassianNB()$ y luego lo entrenamos con la función$fit(trainX, trainY)$. Por ultimo para poder predecir utilizamos $predict()$ como se puede ver en el código:

```
print("Naive Bayes")
nBayes = GaussianNB()
nBayes = nBayes.fit(trainX, trainY)
accuracy = nBayes.score(valX, valY)
print ("Accuracy:", accuracy)
results = nBayes.predict(test.head(20))
```

B. *Decision Tree*

Al ser de la misma librería y ser un modelo relativamente fácil de utilizar sigue los mismos principios. La diferencia es que se llama a la función $DecisionTreeClassifier()$ para crear el modelo como se puede ver en el siguiente código:

```
print("Arbol de Decision")
decTree = DecisionTreeClassifier()
decTree = decTree.fit(trainX, trainY)
accuracy = decTree.score(valX, valY)
print ("Accuracy:", accuracy)
results = decTree.predict(test.head(20))
```

C. *Red Neuronal*

La red neuronal es un poco mas compleja que los otros dos modelos, por eso hay mas pasos previos antes de poder entrenar la data. Antes de comenzar tenemos que importar $tensorflow$ y $StandardScaler$. Una vez hecho eso normalizamos la data para que pueda ser ingresada a la RN. Una vez hecho esto podemos crear nuestra red.
Como se mencionó en la sección del Marco Teórico usaremos cuatro capas, siendo tres de ReLu y una de sigmoid como se podrá apreciar en el código:

```
model = tf.keras.Sequential([
tf.keras.layers.Dense(128,
            activation='relu',
input_shape=(trainX.shape[1],)),
tf.keras.layers.Dense(256,
                activation='relu'),
tf.keras.layers.Dense(256,
                activation='relu'),
tf.keras.layers.Dense(1,
                activation='sigmoid')
    ])
```

En el código anterior se definieron cuatro *DenseLayers* siendo el primer número la cantidad de unidades o "neuronas" en cada una de esas capas. Luego esta el tipo de activación que tendrá cada capa.Al final nuestra red neuronal se vería como la Figura 6:

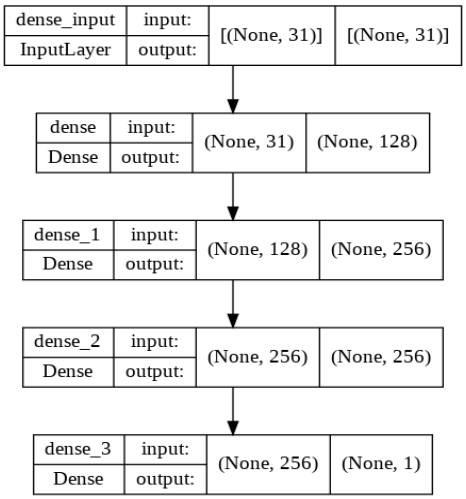

Fig. 6. Representación gráfica de la Red Neuronal creada

Luego se entrenará por 100 épocas para poder ver los resultados:

```
print("Comenzando entrenamiento...")
historial = model.fit(trainX_scaled,
    trainY, epochs=100, verbose=True)
print("Modelo entrenado!")
```

Con ese código obtendremos no solo el entrenamiento sino que información relevante sobre el proceso como: Su perdida, su accuracy, recall, cuantas épocas va y cuantas iteraciones por época. Toda esta información es importante a la hora de decidir si queremos mejorar u optimizar el modelo.

V. Análisis de Resultados

A continuación se realizará el análisis de los resultados mostrados por cada uno de los modelos:

A. *Acccuracy de los Modelos*

Luego de realizar los tres modelos y obtener su precisión los resultados son los siguientes:

- Para Naive Bayes su Accuracy llego a 0.66796 lo cual es 66,8%
- Para Árbol de Decisión su Accuracy llego a 0.76106 lo cual es 76,1%
- Para la Red Neuronal su Accuracy llego a 85%

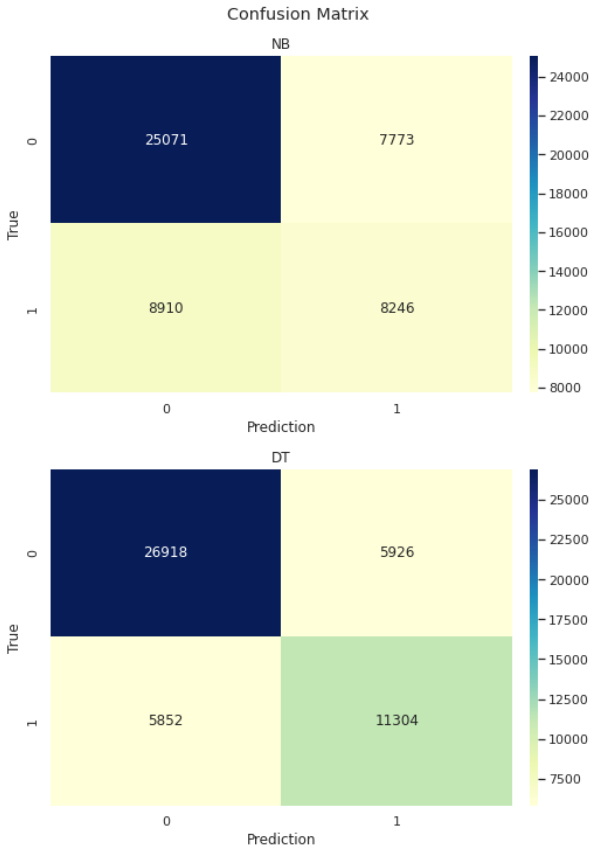

Fig. 7. Naive Bayes v/s Desicion Tree

En el gráfico 7 se puede comparar la matriz de confusión entre Naive Bayes y Árbol de Decisión. En el eje x tenemos los valores que los modelos han predicho, mientras que en el eje y los valores reales. Podemos ver que ambos modelos predijeron correctamente cuando NO era Bosón de Higgs. Sin embargo se puede ver como Naive Bayes tiene un bajo numero de verdaderos positivos, lo cual explica porqué tuvo un accuracy global de casi 10% menos que DT. Asimismo, el DT si pudo clasificar de manera satisfactoria los positivos verdaderos aunque no como los falsos verdaderos.

En cuanto a la red neuronal se obtuvo lo siguiente:
Como se puede ver en el gráfico 8 se obtuvo un accuracy de 85% de manera casi consistente. El Loss [6] se mantiene siempre bajo, este mide la diferencia entre el resutlado esperado vs el obtenido, lo que significa que entre mas bajo menos perdida. Por ultimo el Precision [7] esta un poco por debajo del Accuracy, este se calcula como la relación entre el número de muestras Positivas clasificadas correctamente y el número total de muestras clasificadas como Positivas (ya sea correcta o incorrectamente). Este puede estar por debajo debido al numero de épocas con el que se realizo el modelo,

ya que son pocas.

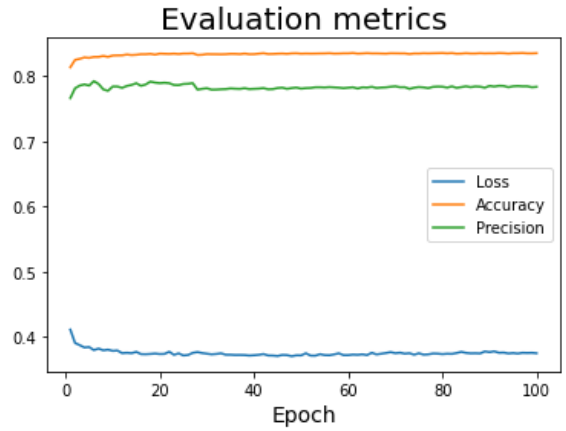

Fig. 8. Accuracy v/s Loss v/s Precisión

En la figura 9 se puede ver porqué es que la Red Neuronal obtuvo mas accuracy que los demás modelos y es porque la cantidad de verdaderos positivos y verdaderos negativos que clasifica correctamente son superiores. Se puede ver como clasificó mas de 1000 mas que DT y más de 4000 que NB de positivos y mas de 1000 y 2000 negativos. Además, obtuvo menos errores en los falsos positivos y falsos negativos con respecto a los otros dos modelos.

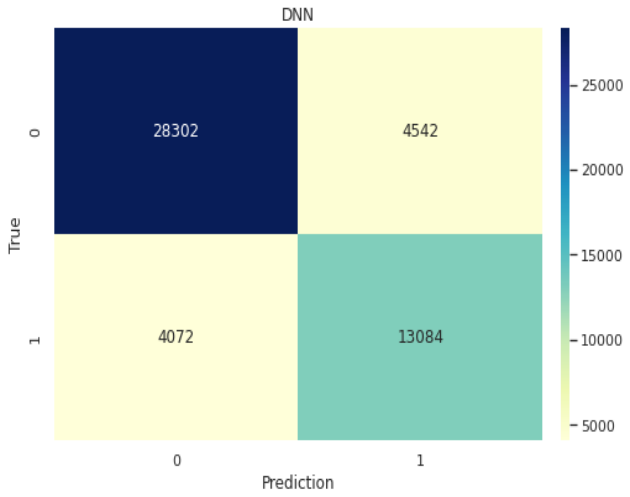

Fig. 9. Matriz de Confusión de Red Neuronal

B. *Comparación de Acccuracy de los Modelos*

Se puede observar que los resultados obtenidos tienen sentido, ya que el modelo menos complejo de los tres obtuvo el menor accuracy, mientras que la red neuronal, siendo el mas avanzado obtuvo un 85% siendo un modelo apto para uso mas avanzado.

	Naive Bayes	**Árbol de Decisión**	**Red Nuronal**
Accuracy	66,8%	76,1%	85%
True Pos	8246	11304	13084
True Neg	25071	26918	28302

Hay que tener en cuenta que un buen modelo esta al rededor de 85% y 95%, ya que mas que eso estaríamos hablando de un posible overfitting (cuando el modelo no aprende, sino memoriza la información) y menos de eso podría ser un posible underfitting (cuando el modelo no llega a aprender la información y la clasifica mal).

C. *Posibles Mejoras a la Red Neuronal*

Estos son algunas de las posibles mejoras que se pueden realizar al modelo para así tener un mejor resultado:

- Ajuste de hiperparámetros
- Mejorar las funciones de activación
- Correcta elección de la tasa de aprendizaje
- Numero de épocas correcto

Con estos cuatro puntos, se podría mejorar de manera substancial el accuracy del modelo. El primero habla sobre los parámetros iniciales que se le pueden dar al modelo. Nosotros solo definimos dos, la función de activación y el numero de neuronas, sin embargo se pueden inicializar mas valores para tener un mejor resultado

Como se mencionó anteriormente hay diversas funciones de activación, por lo que poner las adecuadas en las capas correctas puede haces mucha mas diferencia. Se puede hacer una comparación entre las diferentes funciones para poder determinar cual es la que tiene mejor relación con los datos.

La tasa de aprendizaje es a la velocidad que esta aprende. Es muy importante señalar la tasa correcta, ya que si se pone muy baja puede que nunca llegue a converger al mínimo global ya que terminaría antes de que llegue, de lo contrario, si es muy alta, el modelo estaría saltando con pasos muy grandes, haciéndose igual de difícil encontrar el mínimo global.

Por ultimo, las épocas determinan cuanto es que se va a ejecutar el modelo. Poner muy pocas significa que nuestro red aprenderá poco. De lo contrario si se pone muchas, nuestro modelo podría quedarse estancado en un accuracy y se estaría ejecutando de más, gastando recursos y tiempo o podría hacer que la red tenga overfitting.

VI. CONCLUSIÓN

En conclusión, podemos decir que se cumplió con el objetivo del trabajo el cual era poder realizar un modelo que pudiera clasificar el Bosón de Higgs. Para eso se utilizaron tres diferentes modelos, haciendo énfasis en la Red Neuronal, ya que esta era la que prometía tener la mejor precisión además que era parte de requisito del trabajo. Con los tres modelos se obtuvieron resultados buenos, sin embargo como se estimó, la red neuronal ganó sin necesitar de una configuración compleja, demostrando su potencial.

Asimismo se investigó de manera detallada para poder brindar no solo un marco teórico completo, sino además con un análisis detallado de los datos así también las posibles mejoras que se le puede realizar al modelo para poder así

elevar su predicción.

Este trabajo nos ha sido de gran ayuda por dos factores principales: El primero es poder entender el Bosón de Higgs, un tema verdaderamente interesante de la física de partículas. El segundo, un conocimiento mas profundo del Deep Learning y las redes neuronales, las cuales hoy en día son sumamente importantes convirtiéndose en el futuro de la tecnología y del mundo.

REFERENCES

1 "Bosón de Higgs: qué es y por qué es tan importante," //www.lavozdegalicia.es/noticia/sociedad/2012/07/04/boson-higgs-importante/00031341426666016949655.htm, jul. 2012, [Online; accessed 2-May-2022].

2 R, M., O, C., and Parra, L., "Máquinas de soporte vectorial, clasificador naïve bayes y algoritmos genéticos para la predicción de riesgos psicosociales en docentes de colegios públicos colombianos," in *Información tecnológica*, vol. 29, dic. 2018.

3 Amat Rodrigo, J., "Árboles de decisión con python: regresión y clasificación," https://www.cienciadedatos.net/documentos/py07_arboles_decision_python.html, oct. 2020, [Online; accessed 2-May-2022].

4 Hajiyeva, A., "Neural network for classification with tensorflow," https://www.analyticsvidhya.com/blog/2021/11/neural-network-for-classification-with-tensorflow/, nov. 2012, [Online; accessed 2-May-2022].

5 Sharma, A., "Activation functions in neural networks," https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, sep. 2017, [Online; accessed 2-May-2022].

6 Seb, "An introduction to neural network loss functions," https://programmathically.com/an-introduction-to-neural-network-loss-functions/#:~:text=The%20loss%20function%20in%20a,all%20losses%20constitutes%20the%20cost., sep. 2021, [Online; accessed 2-May-2022].

7 Fawzy, A., "Evaluating deep learning models: The confusion matrix, accuracy, precision, and recall," https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/, 2020, [Online; accessed 2-May-2022].