UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Predicción del comportamiento de actividades terroristas basado en Redes Neuronales Profundas

Proyecto de Tesis I

Autor: Ingrid Ipanaqué Casquina

Asesor: Juan Carlos Espejo Delzo

2020

Resumen

Debido a la creciente popularidad del aprendizaje profundo, el presente proyecto tiene como objetivo desarrollar un modelo predictivo para comprender el comportamiento de las actividades terroristas basados en redes neuronales profundas y el uso de una técnica sintética de sobremuestreo de minorías, SMOTE, para el conjunto de datos mundial sobre terrorismo . Estos comportamientos se basan en si un ataque terrorista será con fin suicida o no, si tendrá éxito, el tipo de arma que se utilizará, el tipo de ataque y la región que será atacada. El rendimiento de la red neuronal profunda es comparado con un modelo de redes neuronales y tres algoritmos tradicionales. La red neuronal cuenta con una capa, la red neuronal profunda cuenta con cinco capas y los algoritmos tradicionaes son: Regresión Logística, Máquinas de Vectores de Soporte y Naive Bayes. El rendimiento obtenido por la red neuronal profunda es más del $95\,\%$ en las métricas: accuracy, precision, recall y F1-Score, mientras que la red neuronal alcanza un accuracy de $85\,\%$ y los algoritmos tradicionales alcanzan el $79\,\%$. Se concluye que el rendimiento del modelo implementado mejora favorablemente aplicando una técnica adecuada para el tratamiento de datos desbalanceados. Además, se observa que el uso de aprendizaje profundo es adecuado para procesar macrodatos.

PALABRAS CLAVE: redes neuronales, redes neuronales profundas, SMOTE, terrorismo.

Índice general

Índice de figuras

Índice de Acrónimos

IA	Inteligencia artificial
ANN	Artificial Neural Network
DNN	Deep Neural Network
GTD	Global Terrorism Database
START	Study of Terrorism And Responses to Terrorism
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
BCE	Binary Cross Entropy
CCE	Categorical Cross-Entropy
Adam	Adaptive Moment Estimation
MVS	Máquinas de Vectores de Soporte
ETC	Etcétera

Agradecimientos

Agradezco a mis padres por la paciencia y el apoyo que me han brindado a lo largo de mi formación académica.
Agradezco a mi hermano por los consejos y por ser un ejemplo a seguir.
Agradezco a mi asesor por brindarme su apoyo y sus consejos durante la realización de este proyecto y demostrar ser un excelente maestro.

Capítulo 1

Introducción

En este capítulo se presenta la motivación, los objetivos y la estructura que sigue el presente proyecto. En la sección de motivación se presenta y explica los motivos que influenciaron a la elección del tema propuesto. La segunda sección está compuesta por el objetivo principal y los objetivos específicos planteados para la elaboración del presente proyecto. En la última sección se presenta la estructura del seminario, en donde se muestra una breve explicación del propósito de cada capítulo.

1.1. Motivación

La inteligencia artificial ha evolucionado rápidamente durante los últimos años, ha pasado de ser un concepto de investigación a convertirse en material de apoyo para distintas áreas ya que tiene fines y aplicaciones claras. Uno de los campos o enfoque que comprende la inteligencia artificial es el aprendizaje profundo o más conocido en inglés como deep learning. El aprendizaje profundo se enfoca en gestionar datos para transformarlos en información útil y generar conocimiento sin necesidad de supervisión humana. Además, obtiene tasas de éxito elevadas a comparación de otros métodos de aprendizaje, gracias a esto puede aplicarse con éxito a grandes volúmenes de datos para la aplicación de conocimiento y realizar predicciones a partir de él.

Históricamente, una de las amenazas más importantes para la civilización es el terrorismo, ya que ha afectado la calidad de vida de las personas hasta la actualidad en todo el mundo. El terrorismo se ha estudiado durante décadas

para comprender los factores que lo causan y también comprender los efectos sociales y económicos que genera.

El estudio del terrorismo es un campo que no es ajeno a la aplicación de inteligencia artificial, hasta la fecha se han implementado algoritmos de aprendizaje automático y modelos de redes neuronales para estudiar los diferentes factores del terrorismo. Debido a esto y a lo expuesto anteriormente que resulta interesante la aplicación del aprendizaje profundo a ésta área, ya que está a disposición una gran cantidad de datos etiquetados.

Desde un inicio, este proyecto estuvo fomentado por el deseo de entender el comportamiento de un modelo basado en aprendizaje profundo, siendo específicos un modelo de redes neuronales profundas aplicado a la predicción de ciertos factores de ataques terroristas. Si bien se sabe que las redes neuronales profundas o el aprendizaje profundo en general es aún un tema de investigación, existen diversas investigaciones que aplican modelos en aprendizaje profundo en conjuntos de datos pequeños obteniendo tasas de éxito elevadas y fomentan el uso de estos modelos en conjuntos de datos más grandes. Además, hasta la fecha ya existen librerías disponibles para los lenguajes de programación más usado, que nos ayudan a la implementación de diversos modelos de aprendizaje profundo, además de contar con documentación, y material académico para realizar una correcta implementación del modelo propuesto.

1.2. Objetivos

El objetivo de este seminario es el de implementar un modelo para la predicción de cinco factores de actividades terroristas mediante el uso de Redes Neuronales Profundas.

Especificamente, los objetivos de este proyecto con respecto al sistema son:

- Entender el funcionamiento de las arquitecturas de red para aprendizaje profundo.

- Implementar un modelo de red neuronal profunda como modelo predictivo.
- Mejorar los resultados esperados por la arquitectura utilizada aplicando un preprocesamiento de datos con técnica sintética de sobremuestreo de minorías.
- Evaluar el rendimiento del modelo implementado.

Y los objetivos con respecto a las competencias académicas desplegadas en el trabajo son:

- Obtener el conocimiento necesario para implementar los procesos involucrados *deep learning*, empleando las herramientas de programación disponibles.
- Adquirir la capacidad de interpretar resultados erróneos producidos por el modelo luego del proceso de aprendizaje.
- Obtener la capacidad de utilizar redes neuronales profundas como medio para la solución de problemas que lo requieran.

1.3. Estructura del Seminario

- **Introducción:**
  En este capítulo se introduce al lector en el tema a tratar, comentando las motivaciones, los objetivos y la estructura del presente proyecto.
- **Estado del Arte:**
  En este capítulo se presenta el fundamento teórico sobre el cual se basa el trabajo para el mejor entendimiento del lector, además se muestran los trabajos e investigaciones relacionadas que fueron realizados anteriormente y que ayudaron a desarrollar el presente trabajo.
- **Recursos y Herramientas:**
  En este capítulo se detalla el hardware, software, dataset y librerias

empleadas en el desarrollo de la investigación, explicando su función y el motivo de su uso.

- **Metodología de Desarrollo:**
  En este capítulo se explica el procedimiento central, paso a paso, de la investigación. Primero se explica el preprocesamiento implementado al conjunto de datos. En la siguiente parte, se muestra el entrenamiento de los modelos de ANNs y DNNs. En esta sección primero se describe la arquitectura de los modelos de ANNs y DNNs implementados, y se finaliza explicando el proceso de aprendizaje de los modelos de ANNs y DNNs.

- **Resultados y Discusiones:**
  En este capítulo se presentan los resultados obtenidos a partir de la implementación de las técnicas mencionadas en el capítulo anterior, tanto para el modelo de red neuronal como para el modelo de red neuronal profunda. Además se realiza un análisis e interpretación de estos resultados.

- **Conclusiones y Trabajo a Futuro:**
  Finalmente, en este capítulo se detallan las conclusiones y el trabajo a futuro del proyecto. En primer lugar se discuten los resultados obtenidos y mostrados en el capítulo anterior. Adicionalmente se plantean ideas y posibles mejoras del modelo desarrollado con el fin de realizar su implementación a futuro.

Capítulo 2

Estado del Arte

En este capítulo se presenta el fundamento teórico de los tópicos involucrados y los trabajos previos que ayudaron al desarrollo del presente proyecto.

2.1. Conceptos previos

2.1.1. Redes Neuronales Artificiales

Las ANN (por sus siglas en inglés, Artificial Neural Network) están, en general, inspiradas en el comportamiento de las neuronas en las redes neuronales biológicas. Por lo que, las ANN se conforman por un conjunto de neuronas artificiales relacionadas a partir de conexiones ponderadas y está dividida en módulos. El modelo de una neurona se le llama habitualmente *nodo o unidad*, esta recibe un *input* desde otros nodos o desde una fuente externa de datos [S McCulloch und Pitts, 1990] y cada *intrada* tiene asociado un peso w. En la figura (2.1) se muestra el modelo simple de una neurona artificial.

En la imagen mostrada, se tiene una vista completa de una neurona, en donde los inputs x son multiplicados por un peso w, estos productos son sumados para después sumarles un término llamado *sesgo* y por último, esta sumatoria de productos es introducida en la función llamada *función de activación*, descrita en la sección (2.1.3), esta define la *salida* de cada nodo. Este

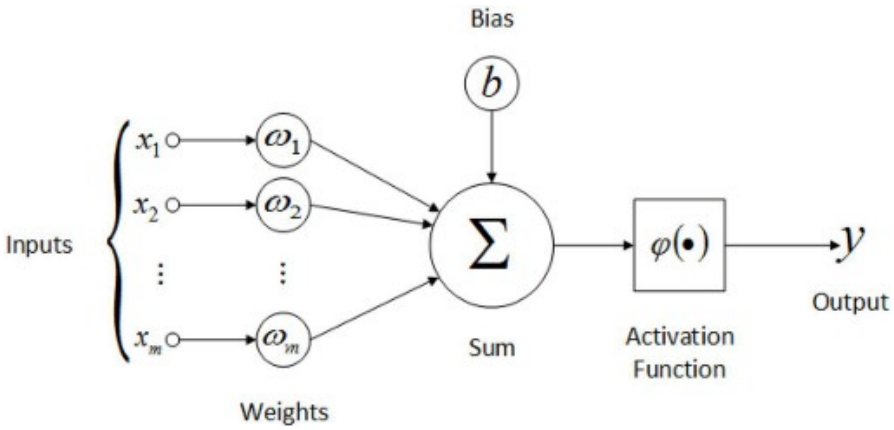


FIGURA 2.1: Representación de una neurona artificial
Fuente: https://medium.com

proceso se representa en la fómula (2.1).

$$salida = f(sesgo + (\sum_{i=1}^{N} w_i x_i)) \tag{2.1}$$

La arquitectura de una ANN está conformada por 3 módulos. El primer módulo es una *capa de entrada* (*Input layer*), puede tener *n* neuronas artificiales, donde *n* depende del número de variables de entrada. El segundo módulo está conformado por *capas ocultas* (*Hidden layers*), recibe como datos de entrada las salidas de la capa anterior. Por último tenemos el tercer módulo conformado por la *capa de salida* (*Output layer*), en donde la cantidad de nodos depende del tipo de problema que la ANN esté resolviendo.

Entrenar una red neuronal, es decir, aprender los valores de nuestros parámetros, pesos w_i y bias b, es la parte más genuina del aprendizaje y podemos ver este proceso de aprendizaje en una ANN como un proceso iterativo de ”ida y vuelta” por las capas de neuronas. El ”ir” es una propagación hacia adelante de la información y el ”retorno” es una propagación hacia atrás de la información, conocido en inglés como Forward-Propagation and Backward-Propagation, respectivamente.
Visualmente, se puede resumir el proceso de aprendizaje de una ANN con el esquema de las etapas en la figura (2.2).

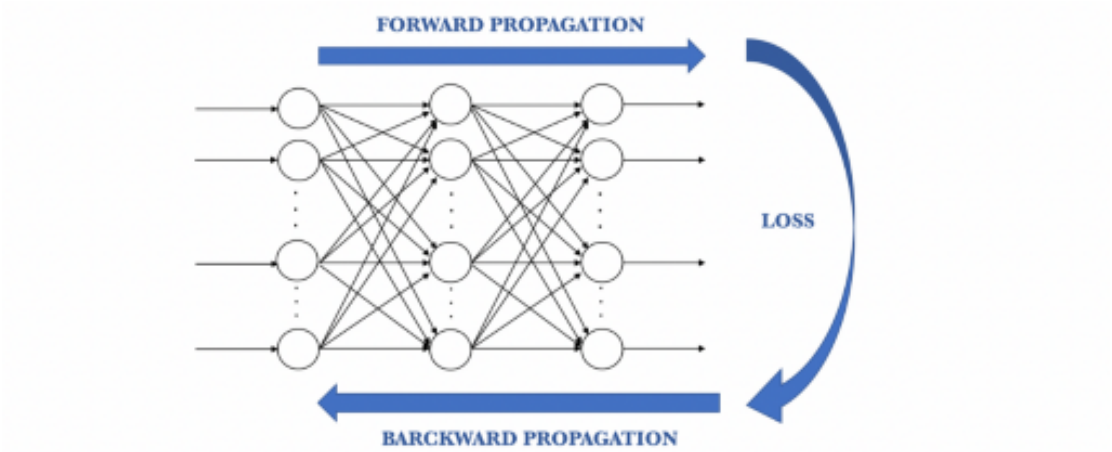


FIGURA 2.2: Proceso de aprendizaje de una ANN
Fuente: Elaboración propia

Propagación hace adelante (Forward-Propagation)

La primera fase, *Forward-Propagation*, ocurre cuando la red se expone a los datos de entrenamiento y estos cruzan toda la red neuronal para que se calculen sus predicciones.

Se usan las funciones de activación, descritas en la sección (2.1.3), para la activación de las neuraonas dada una entrada. Esta podría ser una fila de nuestro conjunto de datos de entrenamiento, como en el caso de la capa oculta. También pueden ser las salidas de cada neurona en la capa oculta, en el caso de la capa de salida. Luego se trabaja a través de cada capa de la ANN, calculando las salidas para cada neurona. Todas las salidas de una capa se convierten en entradas para las neuronas de la siguiente capa. Finalmente se alcanzará la capa final con un resultado de predicción.

Propagación hacia atrás (Backward-Propagation)

Propagación hacia atrás de errores o retropropagación es un método que altera los parámetros, tales como pesos w y sesgos $bias$, de la red neuronal.
Antes de realizar la propagación hacia atrás se utiliza una función de pérdida, descritas en la sección (2.1.4), para comparar y medir qué tan bueno o malo fue el resultado de predicción en relación con el resultado correcto. Una vez calculada la pérdida, esta información atraviesa la red neuronal en orden

inverso, desde la capa de salida hasta la capa de entrada. El algoritmo almacena las variables intermedias (derivadas parciales) necesarias al calcular el gradiente con respecto a algunos parámetros. Es decir, se toma la derivada de la pérdida para la capa de salida y la capa oculta, y los pesos se actualizan utilizando técnicas de optimización (2.1.5).

En resumen, *Backpropagation*, calcula y almacena secuencualmente los gradientes de variables intermedias y parámetros dentro de la red neuronal en orden inverso.

2.1.2. Redes Neuronales Profundas

En la década de 1980, se da inicio a la era cognitiva del aprendizaje profundo. La capacidad de crear redes neuronales profundas, o en inglés *Deep Neural Network (DNN)*, ha existido desde que Pitts [McCulloch und Pitts, 1943] introdujo el perceptrón multicapa. Sin embargo, no se pudo entrenar redes neuronales de manera eficaz hasta que Hinton [Rumelhart u. a., 1986] descubrió que la retropropagación servía para entrenar una red neuronal profunda. [Skansi, 2018]

Aunque los primeros enfoques publicados por Hinton y sus colaboradores se centran en el entrenamiento codicioso por capas y métodos no supervisados, el aprendizaje profundo moderno de última generación se centra en el entrenamiento de modelos de redes neuronales profundas utilizando el algoritmo de retropropagación.

El aprendizaje profundo o *Deep Learning* es un avance relativamente nuevo en la programación de redes neuronales y representa una forma de entrenar redes neuronales profundas. Esencialmente, cualquier red neuronal con más de dos capas es profunda. Esa arquitectura se puede visualizar en la figura (2.3).

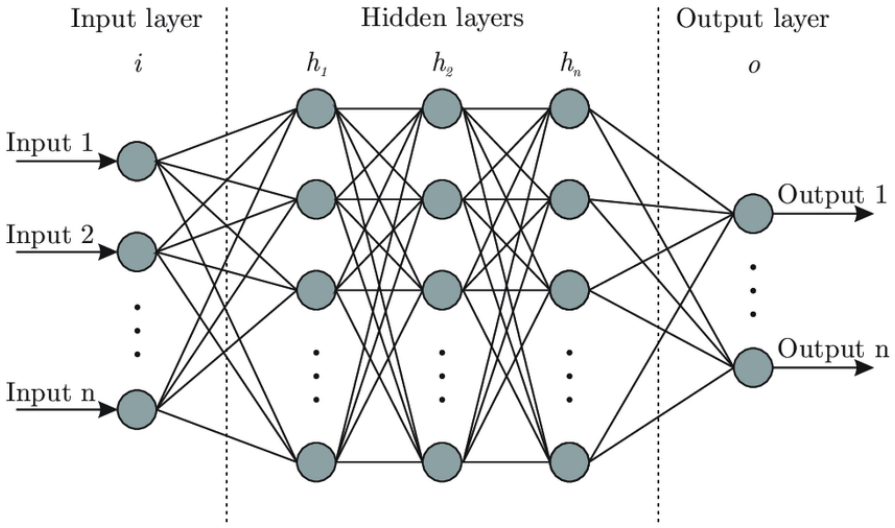


FIGURA 2.3: Arquitectura de una red neuronal profunda
Fuente: https://researchgate.com

2.1.3. Funciones de Activación

Las funciones de activación se utilizan para dar una *no linealidad* al modelo. y que la ANN sea capaz de resolver problemas más complejos [Villanueva, 2020]. Esto le da al modelo implementado el poder de ser más flexible al describir relaciones arbitrarias.

Estas funciones utilizan como entrada a la sumatoria de los pesos W_i multiplicado con las entradas x_i y adicionado el sesgo b [Kogan]. Esta entrada la definimos en la fórmula (2.2).

Dado $n, m \geq 1$, $Z : \mathbb{R}^n \to \mathbb{R}^m$.

$$Z(x_i) = b + \sum_{i=1}^{N} W_i x_i \tag{2.2}$$

Donde, $W_i \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$.

A continuación, se mostrará algunas funciones de activación:

Función Sigmoidea

La *función sigmoidea* [Cybenko, 1989] es la más popular y se define como la ecuación mostrada en la imagen (2.4), siendo e la constante exponencial. Como

se muestra en la figura, es derivable. Mientras z se aleja a la derecha de 0, la función se acerca a 1 y por el contrario, si se aleja a la izquierda, la función se acerca a 0.

Dado $z \in \mathbb{R}$, $f : \mathbb{R} \to [0, 1]$, $f(z) = \frac{1}{1+e^{-z}}$

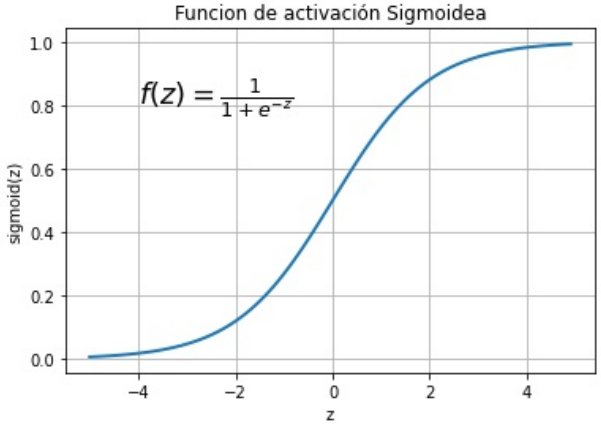


FIGURA 2.4: Función de activación Sigmoide
Fuente: Elaboración propia

Función ReLU

La mayoría de las redes neuronales actuales usan la función de activación llamada *Rectified Linear Unit* o ReLU [Nair und Hinton, 2010] ya que facilita el proceso de aprendizaje en la retropropagación. La cual se define como la ecuación mostrada en la figura (2.5). Se permite el paso de todos los valores positivos sin ser cambiados, y a todos los valores negativos se les asigna 0.

Dado $z \in \mathbb{R}$, $f : \mathbb{R} \to \mathbb{R}$, $f(z) = max(0, z)$

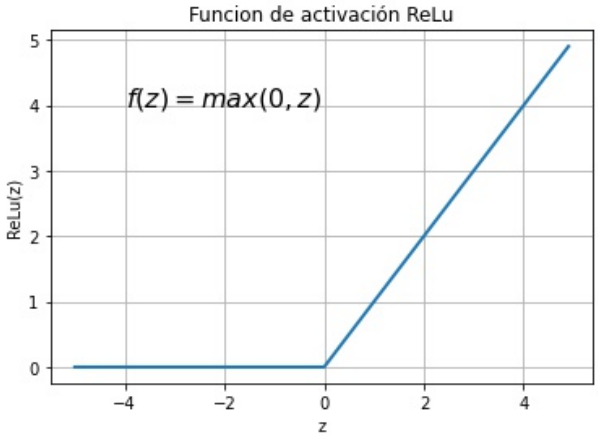


FIGURA 2.5: Función de activación ReLu
Fuente: Elaboración propia

Función Softmax

La *función softmax* [Duan u. a., 2003] resalta los valores más grandes y suprime los valores que están significativamente por debajo del valor máximo, aunque esto no es cierto para los valores pequeños. Normaliza las salidas para que sumen 1 para que puedan tratarse directamente como probabilidades sobre la salida.

Sea $\sigma : \mathbb{R}^K \to [0,1]^K$ y $j = 1, \ldots, K$ se define la función softmax en (2.3).

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.3}$$

2.1.4. Funciones de Pérdida

La función de pérdida, también denominada función objetivo, en esencia nos dice lo lejos que está en un momento dado, lo que la red nos ofrece como salida y el resultado que nosotros consideramos que es el correcto o deseado. El valor de la función de pérdida será luego un dato de entrada en el algoritmo de aprendizaje.

Binary Cross-Entropy

Binary Cross-Entropy (BCE) es una función de pérdida que se utiliza en tareas de clasificación binaria. Estas son tareas que responden a una pregunta con solo dos opciones, *sí* o *no*. La función de BCE evalúa la probabilidad de que la salida de un modelo corresponda a una clase determinada. Esta función se define en (2.4).

$$J(y_i, \hat{y}_i) = -\frac{1}{N}\sum_{i=0}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{2.4}$$

En donde, $\hat{y}_i$ es el i-ésimo valor escalar en la salida del modelo, y_i es el valor objetivo correspondiente (valor esperado) y N es el número de valores escalares en la salida del modelo.

Categorical Cross-Entropy

Categorical Cross-Entropy (CCE) es una función de pérdida que se utiliza en tareas de clasificación de clases múltiples. Estas son tareas en las que una predicción sólo puede pertenecer a una de las muchas categorías posibles, y el modelo debe decidir cuál. Esta función se define en (2.5).

$$J(y_i, \hat{y}_i) = -\sum_{i=0}^{N} y_i log(\hat{y}_i) \tag{2.5}$$

Donde, $\hat{y}_i$ es el i-ésimo valor escalar en la salida del modelo, y_i es el valor objetivo correspondiente, y N es el número de valores escalares en la salida del modelo.

2.1.5. Algoritmos de Optimización

Los optimizadores son algoritmos o métodos que se utilizan para cambiar los atributos de su red neuronal, como los pesos y la tasa de aprendizaje, con el fin de reducir las pérdidas.

Los optimizadores que utilicemos definirán cómo se cambiarán los parámetros mencionados de la ANN. Los algoritmos de optimización se

encargan de reducir las pérdidas y proporcionar los resultados más precisos posibles.

Descenso de gradiente

El descenso de gradiente [Ruder, 2016] es un algoritmo de optimización de primer orden que depende de la derivada de primer orden de una función de pérdida.

Calcula de qué manera deben modificarse los pesos para que la función pueda alcanzar un mínimo. Mediante backpropagation, la pérdida se transfiere de una capa a otra y los parámetros se modifican en función de las pérdidas para que la pérdida se pueda minimizar.
El algoritmo de descenso de gradiente se calcula como se muestra en (2.6).

$$W^{i+1} = W^i - \alpha \frac{dL}{dw} \tag{2.6}$$

Donde, L es la función de pérdida, W^i son los pesos y α es el ratio de aprendizaje, también llamado *learning rate* o velocidad de actualización.

Estimación del momento adaptativo

Conocido en inglés como Adaptive Moment Estimation (Adam) [Kingma und Ba, 2015], es un algoritmo de optimización de la tasa de aprendizaje adaptativo que se diseñó específicamente para entrenar redes neuronales profundas.

Se puede ver a Adam como una variante de la combinación de RMSProp [Tieleman und Hinton, 2012] y momentum [Polyak, 1964] con algunas distinciones importantes. En primer lugar, en Adam, el impulso se incorpora directamente como una estimación del momento de primer orden del gradiente.En segundo lugar, Adam incluye correcciones de sesgo a las estimaciones de los momentos de primer orden y los momentos de segundo orden para dar cuenta de su inicialización en el origen. En general, se considera

que Adam es bastante robusto a la elección de hiperparámetros, aunque la velocidad de aprendizaje a veces debe cambiarse del valor predeterminado sugerido. [Ian Goodfellow, 2016]

Adam calcula las tasas de aprendizaje individuales para diferentes parámetros. Usa estimaciones del primer y segundo momento del gradiente para adaptar la tasa de aprendizaje para cada peso de la red neuronal.

Primero se actualiza la estimación sesgada del primer y segundo momento en (2.7) y (2.8) respectivamente.

$$s \leftarrow \beta_1 s + (1 - \beta_1) g \tag{2.7}$$

$$r \leftarrow \beta_2 r + (1 - \beta_2) g^2 \tag{2.8}$$

En donde, β_1 y β_2 son hiperparámetros introducidos del algoritmo que por defecto tienen los valores de 0.9 y 0.999 respectivamente y g es la gradiente. Se define el primer y segundo momento en (2.9) y (2.10) respectivamente.

$$\hat{s} = \frac{s}{1 - \beta_1^t} \tag{2.9}$$

$$\hat{r} = \frac{r}{1 - \beta_2^t} \tag{2.10}$$

En donde t es un paso de tiempo que se inicializa en 0 y se actualiza mediante iteraciones. La regla de actualización para Adam se muestra en (2.11).

$$\theta^{i+1} = \theta^i - \alpha \frac{\hat{s}}{\sqrt{\hat{r}} + \epsilon} \tag{2.11}$$

En donde, α es la velocidad de aprendizaje cuyo valor predeterminado es 0.001 y ϵ es una constante cuyo valor es 10^{-8}.

2.1.6. Métricas de evaluación

La exactitud o accuracy de un modelo es muy interpretable, fácil de identificar y buena cuando hay clases equilibradas, de lo contrario puede ser engañosa, es por esta razón que se tienen diferentes métricas para evaluar un modelo de manera eficaz. Para las tareas de clasificación, las métricas de evaluación comunes son el accuracy, precision, recall y F1-score. [Matthew Moocarme, 2020]

Accuracy

Accuracy se refiere a cuán cerca del valor real se encuentra el valor medido o predicho y está relacionada con el sesgo de la estimación. Es quizá la métrica más simple que uno pueda imaginar, y se define como el número de predicciones correctas dividido por el número total de predicciones. Esta métrica se calcula mediante la fórmula mostrada en (2.12).

$$Accuracy = \frac{Cantidad\ de\ predicciones\ correctas}{Cantidad\ total\ de\ predicciones} \tag{2.12}$$

Precisión

Hay muchos casos en los que el *accuracy* no es un buen indicador del rendimiento de un modelo. Uno de estos escenarios es cuando la distribución de clases está desequilibrada (una clase es más frecuente que otras). En este caso, incluso si predice todas las muestras como la clase más frecuente, obtendría una alta tasa de accuracy, lo que no tendría ningún sentido. Por lo tanto, también debemos analizar las métricas de rendimiento específicas de la clase, la precisión es una de esas métricas.

La *Precisión* se refiere a la dispersión del los valores obtenidos de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión y se calcula mediante la fórmula (2.13).

$$Precision = \frac{Verdadero\ Positivo}{Verdadero\ Positivo + Falso\ Positivo} \tag{2.13}$$

Recall

Recall es otra métrica importante que se define como la fracción de muestras de una clase que el modelo predice correctamente. Más formalmente se muestra en la fórmula (2.14).

$$recall = \frac{Verdadero\ Positivo}{Verdadero\ Positivo + Falso\ Negativo} \tag{2.14}$$

F1 Score

Según la aplicación, es posible que desee dar mayor prioridad a el *Recall* o la *Precisión*. Pero hay muchas aplicaciones en las que estos son importantes. Por lo tanto, es natural pensar en una forma de combinar estos dos en una sola métrica. Una métrica popular que los combina se llama *F1-score*, que es la media armónica de *Precisión* y *Recall* definida como se muestra en la ecuación (2.15).

$$F1\ Score = 2x\frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.15}$$

2.1.7. Matriz de Confusión

La *Matriz de Confusión* no es una métrica pero es importante conocer. Uno de los conceptos clave en el rendimiento de la clasificación es la matriz de confusión (también conocida como matriz de error), que es una visualización tabular de las predicciones del modelo frente a las etiquetas de verdad fundamental. Cada fila de la matriz de confusión representa las instancias en una clase predicha y cada columna representa las instancias en una clase real, tal y como lo muestra la figura (2.6).

		Valores predichos	
		Negativo	**Positivo**
Valores actuales	**Negativo**	Verdadero negativo (VN)	Falso positivo (FP)
	Positivo	Falso negativo (FN)	Verdadero positivo (VP)

FIGURA 2.6: Representación básica de la matriz de confusión
Fuente: Elaboración propia

Donde,

- **Verdadero negativo (VN)**: Recuento de resultados que originalmente fueron negativos y se predijeron como negativos.
- **Falso positivo (FP)**: Recuento de resultados que originalmente fueron negativos pero se predijeron positivos.
- **Falso negativo (FN)**: Recuento de resultados que originalmente fueron positivos pero se predijeron negativos.
- **Verdadero positivo (VP)**: Recuento de resultados que originalmente fueron positivos y se predijeron como positivos.

2.1.8. Métricas calculadas a partir de la matriz de confusión

Curva ROC

Otra forma importante de evaluar un modelo de clasificación es utilizando una curva ROC. Una *curva ROC* es un gráfico entre la tasa de verdaderos positivos (recall) y la tasa de falsos positivos (1 - especificidad). La especificidad está definida como el número de predicciones negativas dividido por el número total de negativos reales.

En la Figura 2.7 se muestra un ejemplo de curva ROC.

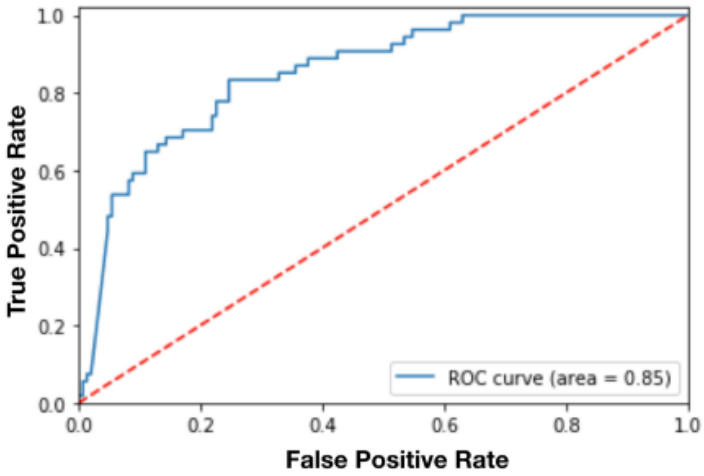


FIGURA 2.7: Un ejemplo de la curva ROC
Fuente: https://towardsdatascience.com

Área bajo la curva

El área bajo la curva, o por sus siglas en inglés Area Under Curve - AUC, es un valor numérico que representa el área bajo una curva ROC. El gráfico (2.7) nos muestra un ejemplo de AUC. No existe una regla estándar para la puntuación AUC, pero en la tabla (2.1) se muestran los valores generalmente aceptables y cómo se relacionan con la calidad del modelo.

Puntuación AUC	Calidad del modelo
0.9 a 1	Excelente
0.8 a 0.9	Bueno
0.7 a 0.8	Justo
0.6 a 0.7	Pobre
0.5 a 0.6	Falla

TABLA 2.1: Puntaje AUC general.
Fuente: ver pag. 205 de Matthew Moocarme [2020]

2.1.9. Algoritmos tradicionales de aprendizaje automático

Regresión Logística

La *Regresión logística* es un algoritmo de clasificación. Se considera como un modelo de regresión en estadística pero se comenzó a usar como un clasificador.

La regresión logística se utiliza principalmente en la actualidad por dos razones. Primero, da una interpretación de la importancia relativa de las características, lo cual es bueno tener si deseamos construir una intuición sobre un conjunto de datos dado. La segunda razón, que es mucho más relevante, es que la regresión logística es en realidad una red neuronal de una neurona. Al comprender la regresión logística, estamos dando un primer e importante paso hacia las redes neuronales y el aprendizaje profundo. Dado que la regresión logística es un algoritmo de aprendizaje supervisado, debemos tener los valores objetivo para el entrenamiento incluidos en los vectores de fila para el conjunto de entrenamiento. [Skansi, 2018]

Se puede ver una representación esquemática de la regresión logística en la figura (2.8).

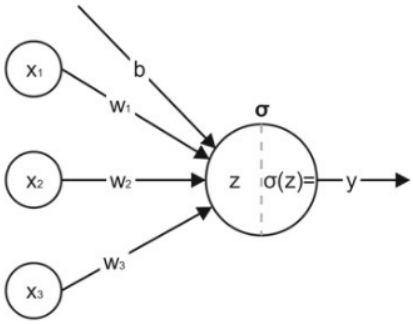


FIGURA 2.8: Vista esquemática de regresión logística
Fuente: ver pág. 62 de Introduction to Deep Learning [Skansi, 2018]

La regresión logística se divide en la ecuación mostrada en (2.2), el cálculo de la suma ponderada, y la función Sigmoidea mostrada en (2.1.3). De la unión

de estos dos, obtenemos la ecuación (2.16).

$$y = \sum_{i>0}^{N} \sigma(b + w_i x_i) \tag{2.16}$$

Para este cálculo se necesita los datos de entrada x_i, los pesos w_i y el sesgo b. El objetivo de la regresión logística es aprender vector de pesos óptimo y bias también para lograr una buena clasificación. El único aprendizaje en regresión logística es encontrar un buen conjunto de pesos.

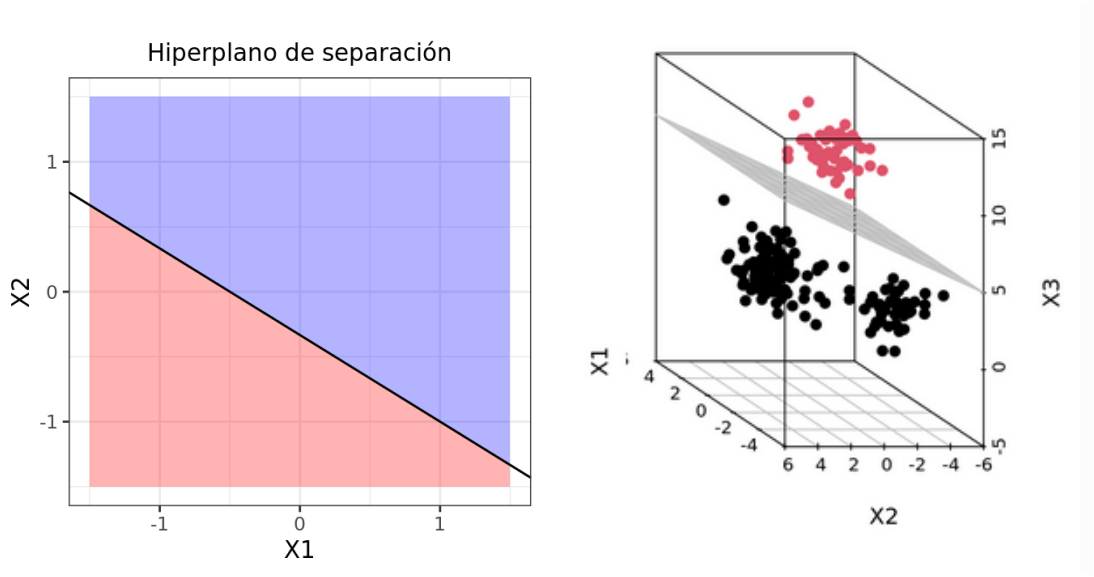
Máquinas de Vectores de Soporte

Máquinas de Vectores de Soporte, abreviado como MVS, es utilizada comúnmente ya que produce una precisión significativa con menos potencia de cálculo. MVS se puede utilizar para tareas de regresión y clasificación, pero se usa ampliamente en los objetivos de clasificación.

El objetivo del algoritmo es encontrar un hiperplano en un espacio N-dimensional, N es el número de características, que clasifique claramente los puntos de datos. Para separar las dos clases de puntos de datos, hay muchos hiperplanos posibles que podrían elegirse. El objetivo es encontrar un plano que tenga el margen máximo, es decir, la distancia máxima entre puntos de datos de ambas clases. Maximizar la distancia del margen proporciona cierto refuerzo para que los puntos de datos futuros puedan clasificarse con mayor precisión.

En un espacio de dos dimensiones, el hiperplano es un subespacio de 1 dimensión, es decir, una recta. En un espacio tridimensional, un hiperplano es un subespacio de dos dimensiones, un plano convencional. Para dimensiones $p > 3$ el concepto de subespacio se mantiene con $p - 1$ dimensiones. La figura (2.9a) muestra el hiperplano de un espacio bidimensional. Se consigue buenos resultados cuando el límite de separación entre clases es aproximadamente lineal. Si no lo es, su capacidad decae drásticamente. Una estrategia para enfrentarse a este tipo de problemas consiste en expandir las dimensiones del espacio original. El hecho de que los grupos no sean

linealmente separables en el espacio original no significa que no lo sean en un espacio de mayores dimensiones. La figura (2.9b) muestra el hiperplano en un espacio tridimensional.



(A) 2D (B) 3D

FIGURA 2.9: Hiperplano de separación
Fuente: https://www.cienciadedatos.net

Naïve Bayes

Naïve Bayes es un algoritmo simple, pero sigue siendo muy útil para grandes conjuntos de datos. Se basa en una técnica de clasificación estadística llamada *Teorema de Bayes*.

Optando por una visión probabilística del aprendizaje automático y afirmamos que todos los algoritmos de aprendizaje automático en realidad aprenden solo $P(y|x)$, podríamos decir que Naïve Bayes es el algoritmo de aprendizaje automático más simple, ya que solo tiene las necesidades básicas para hacer que el "cambio" de $P(f|t)$ a $P(t|f)$ funcione. En este algoritmo se asume que las variables predictoras son independientes entre sí. Es decir, que la presentcia de cierta carácterística o atributo en un conjunto de datos no está relacionado con la presencia de otra característica. [Skansi, 2018]

La simplicidad de este algoritmo se debe a que se proporciona una forma de calcular la probabilidad "posterior" de que ocurra cierto evento A, dadas

algunas probabilidades de eventos "anteriores". La probabilidad se define en (2.17).

$$P(A|R) = \frac{P(R|A)P(A)}{P(R)} \tag{2.17}$$

En donde:

- $P(A)$: Probabilidad de A.
- $P(R|A)$: Probabilidad de que se de R dado A.
- $P(R)$: Probabilidad de R.
- $P(A|R)$: Probabilidad poterior de que se de A dado R.

2.2. Trabajos Previos

El tema del terrorismo se ha estudiado durante varios años para poder desarrollar distintos mecanismos que ayuden a reducir la probabilidad de una actividad terrorista y el uso de inteligencia artificial no es ajeno a este tema. Es por esto que a continuación se presentan los trabajos previos que ayudaron y encaminaron el desarrollo de este proyecto.

2.2.1. Comparison of Machine Learning Approaches in the Prediction of Terrorist Attacks

Trabajo realizado y publicado por Agarwal u. a. [2019]. En este trabajo se analiza el conjunto de datos del GTD y se predicen los factores que podrían ser causa del aumento del terrorismo. Para este objetivo se utilizan técnicas de minería de datos y algoritmos de aprendizaje automático como Máquinas de Vectores de Soporte, Naive Bayes, Árbol de decisión, Bosques Aleatorios o Random Forest y Regresión Logística. En donde se obtiene un accuracy de $73\,\%$, $85\,\%$, $89\,\%$, $89\,\%$ y $92\,\%$ respectivamente.

2.2.2. N-hidden layer artificial neural network architecture computer code: geophysical application example

Trabajo realizado y publicado por Ogunbo u. a. [2020]. En este trabajo se investiga el comportamiento de las ANN de N-capas ocultas orientados a problemas geofísicos. Los resultados de este artículo demuestran que las ANN con 2 a 4 capas ocultas tienen una menor accuracy comparado con ANN que cuentan con más capas ocultas, ya que estas obtienen un accuracy mayor a $97\,\%$ mientras que las ANN con 2 capas ocultas se obtiene un error del $20\,\%$. Estas métricas se obtienen prediciendo la litología y la identificación de fluidos en la industria del petróleo y el gas mediante la relación de Poisson.

2.2.3. An Enhanced Deep Neural Network for Predicting Workplace Absenteeism

Trabajo realizado y publicado por Shah u. a. [2020]. En este trabajo se investiga el problema del absentismo en el ambiente laboral, por lo que proponen un modelo de DNN para predecir el comportamiento de los empleados hacia la puntualidad. A su vez utilizan la herramienta SMOTE para el preprocesamiento de los datos. El rendimiento del modelo propuesto alcanza el $90.6\,\%$ y se propone usar el modelo propuesto en conjuntos de datos más extensos.

2.3. Conclusiones

Conforme a lo presentado anteriormente, avances e investigaciones previas, se puede concluir que el campo de investigación de *Deep Learning* ha sido aplicado en distintas áreas y es capaz de adquirir flexibilidad, independientemente al tipo de problema en el que se haya desarrollado alcanzando una tasa de éxito alta.

En este caso, el presente proyecto se basará en el enfoque planteado por el artículo [Shah u. a., 2020], para el preprocesamiento de datos y modificando el

modelo de redes neuronales profundas utilizando los conclusiones obtenidas en [Ogunbo u. a., 2020].

Capítulo 3

Recursos y Herramientas

En este capítulo se presentan y detallan las herramientas, el conjunto de datos y las bibliotecas utilizadas para el desarrollo de la investigación del presente proyecto.

3.1. Hardware

3.1.1. Asus K451L

El equipo principal con el que se contó fue una portatil de marca Asus, usada principalmente para el entrenamiento de las redes y el proceso detallado en el capítulo (4). Sus especificaciones se muestran en la tabla (3.1).

Arquitectura del procesador	Intel i5-4200U
Procesador gráfico	Intel HD Graphics 4400
Frecuencia de procesamiento	1.60GHz
Núcleos de procesamiento	4
Memoria principal	8 GB
Sistema Operativo	Linux Ubuntu 18.04.5 LTS x86_64

TABLA 3.1: Especificaciones de Asus K451L.

3.2. Software

A continuación se presentan los detalles del lenguaje de programación y librería utilizados en la implementación de la red, su entrenamiento y validación

3.2.1. Python

Python [Foundation] es un lenguaje de programación de alto nivel y es considerada como una de las mejores opciones para programación científica. Es elegida por esas razones, además de la gran variedad de librerías especializadas en deep learning con las que cuenta.

3.2.2. Jupyter Notebook

Jupyter Notebook [jupyter notebook] es una herramienta basada en navegador para la creación interactiva de documentos que combina texto explicativo, matemáticas, cálculos y su salida de medios enriquecidos.

3.2.3. Bibliotecas

Las bibliotecas usadas para el desarrollo de este proyecto son:

Numpy

NumPy es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas. [Harris u. a., 2020]

Pandas

Pandas es una de las bibliotecas de Python más utilizadas en la ciencia de datos. Proporciona estructuras y herramientas de análisis de datos de alto rendimiento y fáciles de usar. [pandas development team, 2020]

Matplotlib

Matplotlib es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. [Hunter, 2007]

Keras

Keras es una biblioteca de Python de código abierto, potente y fácil de usar para desarrollar y evaluar modelos de aprendizaje profundo. Incluye las eficientes bibliotecas de cálculo numérico Theano y TensorFlow y le permite definir y entrenar modelos de redes neuronales. [Chollet u. a., 2015]

Scikit-learn

Scikit-learn es una biblioteca de software de aprendizaje automático para el lenguaje de programación Python. [Pedregosa u. a., 2011]

3.3. Conjunto de datos

El presente proyecto desarrolló modelos de IA, por lo que la elección de los datos era uno de los factores más importantes. Se hizo uso de la fuente desarrollada por el Consorcio Nacional para el Estudio del Terrorismo y las Respuestas al Terrorismo (START).

3.3.1. Global Terrorism Database (GTD)

GTD es la base de datos sin clasificar más completa de ataques terroristas del mundo. El GTD es elaborado por un equipo dedicado de investigadores y personal técnico. [START]

El GTD es de código abierto que proporciona información sobre ataques terroristas nacionales e internacionales en todo el mundo desde 1970 y a la fecha incluye más de $190,000$ eventos. Para cada evento, se encuentra disponible una amplia gama de información, incluida la fecha y el lugar del incidente, las

armas utilizadas, la naturaleza del objetivo, el número de víctimas y, cuando sea identificable, el grupo o individuo responsable.

Capítulo 4

Metodología de Desarrollo

En el presente capítulo se detallará los pasos seguidos durante la investigación, considerando el preprocesamiento de datos y el entrenamiento de la red neuronal y red neuronal profunda.

4.1. Preprocesamiento de datos

Debido a la gran cantidad de datos, el preprocesamiento de datos suele ser el primer y más importante paso para corregir distintas deficiencias que se puedan encontrar. Resolver un problema de aprendizaje automático constiste en optimizar una función matemática. Para esto, se debe trabajar con datos numéricos que nos permitan ser usados en contextos matemáticos.

4.1.1. Selección de los atributos

Para el desarrollo de este trabajo se contaron con $181,857$ instancias y se utilizaron 34 atributos de los 135 brindados por el dataset. La descripción de cada uno de atributos están mostrados en la tabla (4.1).

N°	Atributo	Descripción
1	iyear	El año en el que ocurrió el incidente.
2	imonth	El valor numérico del mes en el que ocurrió el incidente.
3	iday	El valor númerico del día en el que ocurrió el incidente.

N°	Atributo	Descripción
4	Extended	Si 1=”yes”, la duración del incidente se extendió más de 24 horas; si 0=”no” la duración del incidente fue menor.
5	Provstate	Nombre de la región administrativa subnacional de primer orden en la que ocurre el evento.
6	Latitude	La latitud de la ciudad en la que ocurrió el evento.
7	Longitude	La longitud de la ciudad en la que ocurrió el evento.
8	Specificity	Identifica la resolución geoespacial de los campos de latitud y longitud.
9	Vicinity	La región en una ubicación cercana.
10	Crit1	Objetivo político, económico, religioso o social; 1=”yes”, 0=”no”.
11	Crit2	Intención de coaccionar, intimidar o publicitar a un público más amplio, 2=”sí”, 0=”no”.
12	Crit3	Fuera del derecho internacional humanitario, 1=”yes”, 0=”no”.
13	Doubtterr	Puede haber cierta incertidumbre sobre si un incidente cumple con todos los criterios de inclusión. 1=”yes” hay una duda, 0=”no” esencialmente no hay duda.
14	Multiple	1=”yes”, el ataque es parte de un incidente múltiple. 0=”no”, el ataque no es parte de un incidente múltiple.
15	Natlty1	La nacionalidad del objetivo que fue atacado.
16	Propextent	Si ¿Daños a la propiedad? es ”yes”, una categoría describe el alcance del daño a la propiedad.
17	Ishostkid	Registra si las víctimas fueron tomadas como rehenes. ”-9”=”desconocido”.
18	Ransom	El incidente implicó una demanda de rescate monetario.
19	Country	El país o locación en donde ocurrió el incidente.
20	City	Nombre de la ciudad, aldea o pueblo en el que ocurrió el incidente.
21	Gname	El nombre del grupo que llevó a cabo el ataque.

N°	Atributo	Descripción
22	Individual	El ataque fue llevado a cabo por una persona o varias que no se sabe que estén afiliadas a un grupo u organización.
23	Nkillus	El número de ciudadanos estadounidenses que murieron como resultado del incidente.
24	Nkillter	Limitada a sólo las muertes de perpetradores.
25	Nwound	Registra el número de lesiones no mortales confirmadas tanto de los perpetradores como de las víctimas.
26	Nwoundus	Registra el número de lesiones no mortales confirmadas de ciudadanos estadounidenses, tanto de los perpetradores como de las víctimas.
27	Nwoundte	Las convenciones siguen el campo ”Número de víctimas mortales”.
28	Property	Aparece ”yes” si hay evidencia de daños a la propiedad por el incidente.
29	Targtype1	El tipo general de objetivo/víctima. Consta de 22 categorías.
30	Suicide	“Sí” en aquellos casos en los que exista evidencia de que el autor no tuvo la intención de escapar vivo del ataque.
31	Success	Éxito de un ataque terrorista.
32	Weaptype1	Éxito de un ataque terrorista.
33	Region	El código de región basado en 12 regiones.
34	Attacktype1	El método general de ataque y la amplia clase de tácticas utilizadas.

TABLA 4.1: Tabla de atributos del GTD.

Atributos de entrada y salida

Primero se trabaja con el dataset completo, tanto para la Codificación de etiquetas en la sección (4.1.2) como para el Manejo de datos faltantes en la sección (4.1.3), luego de eso se divide el dataset en 5, en los cuales sólo

se diferencian en la salida, y se trabaja con cada uno de estos de manera independiente.

Se cuentan con 34 atributos en el dataset descrito anteriormente, de las cuales 33 son nuestras características de entrada y 1 será la salida para los distintos modelos de ANN y DNN. Las salidas de estos modelos son:

- **Suicidio:** Este campo indica si la actividad terrorista será un suicidio o no. Esta clasificación es binaria, donde 1=”sí” y 0=”no”.
- **Éxito:** Este campo indica si la actividad terrorista tendrá éxito o no. Esta clasificación es binaria, donde 1=”sí” y 0=”no”.
- **Tipo de arma:** Este campo nos indica la clasificación del tipo general de armas utilizadas en la actividad terrorista. Esta clasificación es multiclase. Las clases consideradas son las siguientes:
  1. Biológico
  2. químico
  3. Radiológico
  4. En blanco
  5. Armas de fuego
  6. Explosivos
  7. Armas falsas
  8. Incendiario
  9. Cuerpo a cuerpo
  10. Vehículo
  11. Equipo de sabotaje
  12. Otro
  13. Desconocido

- **Región:** Este campo clasifica la región que será blanco de la actividad terrorista. Esta clasificación es multiclase. Las clases consideradas son las siguientes:

  1. Norteamérica
  2. América Central y el Caribe
  3. Sudamerica
  4. Este de Asia
  5. El sudeste de Asia
  6. Asia del Sur
  7. Asia Central
  8. Europa Oriental
  9. Europa del Este
  10. Oriente Medio y África del Norte
  11. Africa Sub-sahariana
  12. Australasia y Oceanía

- **Tipo de ataque:** Este campo clasifica el tipo de ataque realizado como actividad terrorista. Esta clasificación es multiclase. Las clases consideradas son las siguientes:

  1. Asesinato
  2. Asalto armado
  3. Bombardeo / explosión
  4. Secuestro
  5. Toma de rehenes (incidente de la barricada)
  6. Toma de rehenes (secuestro)
  7. Ataque a instalaciones / infraestructura
  8. Asaltos sin armas

9. Desconocido

Teniendo más claro la distribución de atributos para cada predicción, se debe realizar una partición de las muestras, dado que se necesita un grupo de muestras para el entrenamiento y otro para la validación.

La proporción de las particiones es del $90\,\%$ para el entrenamiento y $10\,\%$ para la validación. Las muestras de entrenamiento están etiquetadas como *X_train* y *Y_train*, mientras que para la validación están dados por *X_test* y *Y_test*.

4.1.2. Codificación de etiquetas

En el conjunto de datos de GTD, existen algunas características en formato de texto, como el nombre del grupo, país, entre otros. No es factible procesar este tipo de datos en un modelo de NNA o DNN, por lo que fue necesario codificarlas.

Actualmente se cuenta con diversas técnicas para este proceso, tales como: TFIDF. Word2Vec, GloVe, One hot encoding, etc. Para este proyecto caso se eligió la clase *LabelEncoder* de la biblioteca *sklearn*, la cual se utiliza para convertir datos no numéricos en datos numéricos y así poder trabajar con ellas.

La función *LabelEncoder* codifica etiquetas de una característica categórica en valores numéricos entre 0 y el número de clases -1. Una vez instanciado, el método fit lo entrena (creando el mapeado entre las etiquetas y los números) y el método transform transforma las etiquetas que se incluyan como argumento en los números correspondientes. El método $fit\_transform$ realiza ambas acciones simultáneamente. [Pedregosa u. a., 2011]

4.1.3. Manejo de datos faltantes

El GTD contiene valores faltantes, muchos conjuntos de datos del mundo real los continen también, pueden estar codificados como espacios en blanco, NaN u otro marcador de posición.

Una estrategia para utilizar datasets incompletos es descartar filas y/o columnas que contengan valores faltantes. Sin embargo, se perderían datos que pueden ser valiosos. Una mejor estrategia y la usada en el presente proyecto es inferir los datos de la parte conocida del dataset. En la figura (4.1.3) se puede observar cuáles son las columnas que tienen mayor cantidad de datos faltantes.

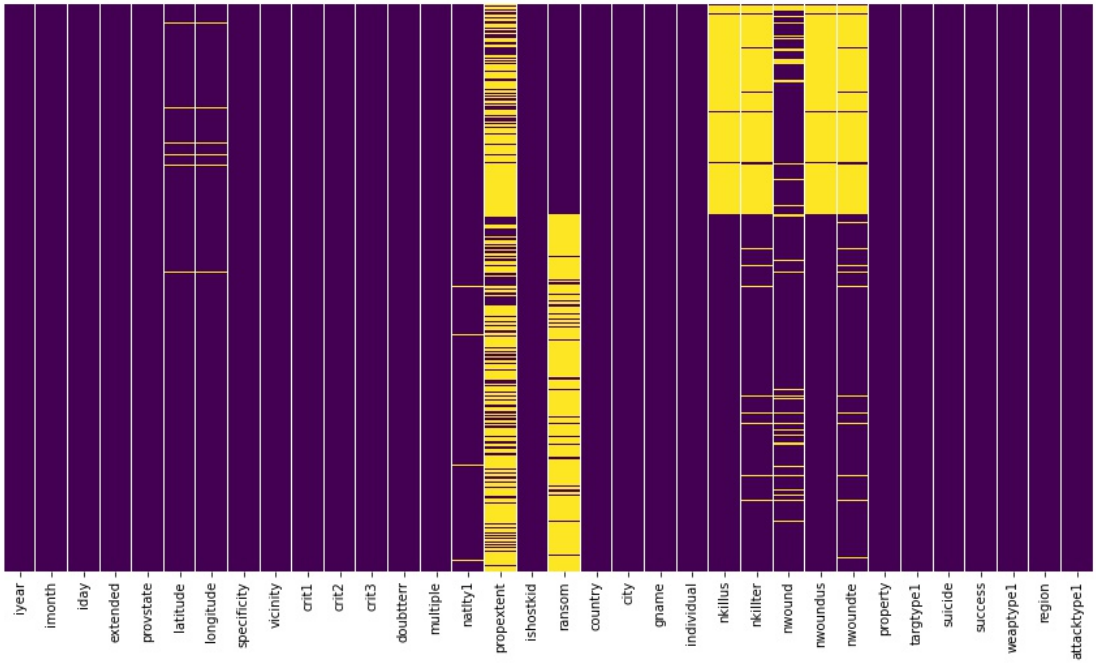


FIGURA 4.1: Gráfico informativo de datos faltantes
Fuente: Elaboracion propia

De la gráfica mostrada se puede observar que la mayor cantidad de datos faltantes se encuentran en las columnas: $propextent$ y $ransom$, a esto le sigue las columnas $nkillus$, $nkillter$, $nwoundus$ y $nwoundte$, con una gran cantidad de datos faltantes. Se pueden utilizar diferentes técnicas de interpolación para completar los datos faltantes. En esta investigación, *SimpleImputer* de la biblioteca *sklearn* se utiliza para completar los datos faltantes, reemplazando estos datos por la *media* a lo largo de cada columna. [Brownlee, 2020a, cap. 8]

4.1.4. Tratamiento de clases desbalanceadas

Durante el análisis del GTD, se observa la existencia de data no balanceada. Los datos desbalanceados se refiere a problemas de clasificación en donde las

clases no son representadas de manera equitativa.

Es común trabajar con conjuntos de datos de clasificación que no tengan exactamente el mismo número de instancias en cada clase. Pero cuando hay un desequilibrio de clases significante como $4:1$, por ejemplo, esto causaría que el entrenamiento de ANN y DNN sea sesgado. [Brownlee, 2020b]

Existen varias técnicas para lidear con estos problemas, entre ellas está el generar muestras sintéticas, de esta manera mantendremos los datos de forma equilibrada. Una forma de generar muestras sintéticas es muestrear aleatoriamente los atributos de las clases minoritarias. El enfoque más utilizado para sintetizar nuevos ejemplos se denomina Técnica de sobremuestreo de minorías sintéticas, o SMOTE para abreviar [Chawla u. a., 2002]. Esta herramienta es proporcionada por la biblioteca *imbalanced-learn*.

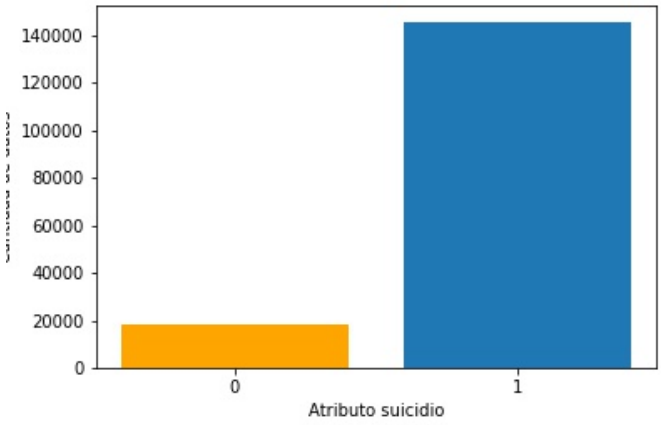
En la figura (4.2) se muestra un ejemplo de balanceo de datos para el caso de predicción del éxito.

4.1.5. Normalización

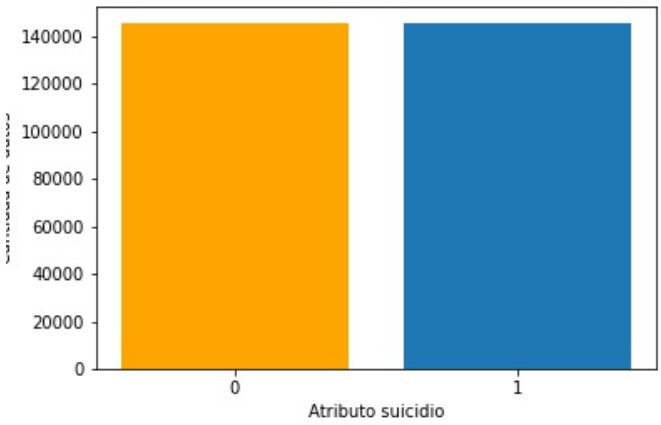
El conjunto de datos hasta el momento contiene columnas que tienen valores como 0 y 1, pero también otras tienen valores en cientos o miles. En esta situación, el algoritmo de aprendizaje dificilmente aprenderá un patrón y converjerá a un mínimo global. Por lo que es antes de que los datos sean procesados serán normalizados, en el rango de 0 a 1 o -1. Para este caso, se usará *MinMaxScalar* de la biblioteca *sklearn*. En donde, para cada valor le resta el promedio de todos los valores y lo divide por la desviación estándar, para convertir los datos en el rango de -1 a 1. La fórmula de estandarización se expresa en la ecuación (4.1).

$$Z_i = \frac{X_i - \bar{X}}{s} \tag{4.1}$$

Donde X_i son todas las muestras para una característica dada, X es el promedio de todas las muestras por característica y s es la desviación estándar.



(A) Datos antes del balanceo.



(B) Datos después del balanceo.

FIGURA 4.2: Ejemplo del balanceo de clases con SMOTE.
Fuente: Elaboración propia

4.2. Entrenamiento del modelo

Como lo mencionamos en la sección (4.1.1), contamos con 5 conjuntos de datos, de los cuales 2 son tareas de clasificación binaria, y los 3 restantes son tareas de clasificaciónń multiclase. Para estas 5 predicciones, se realizaron dos modelos, uno basado en ANN y otro basado en DNN. A continuación se detallará la arquitectura de las redes usadas y el procedimiento empleado.

4.2.1. Arquitectura del modelo de ANN

El modelo de ANN tiene como entrada los atributos mostrados en la tabla (4.1), para todos los casos el input tiene una dimensión de 33. Seguido de una capa oculta que cuenta con 10 nodos. Finalmente la capa de salida que cuenta con un solo nodo para el caso de una clasificación binaria. Para las tareas de clasificación multiclase, la capa de salida contará con una cantidad de nodos igual a la cantidad de clases del atributo de salida. Esto se visualiza mejor en la tabla (4.2).

Predicción	Clasificación	Nodos
Suicidio	Binaria	1
Éxito	Binaria	1
Tipo de arma	Multiclase	12*
Region	Multiclase	12
Tipo de ataque	Multiclase	9

TABLA 4.2: Tabla con la cantidad de nodos en la capa de salida.

**En el caso del tipo de arma, no existen datos de la clase 4.*

La arquitectura del modelo de ANN se puede observar en la figura (4.3).

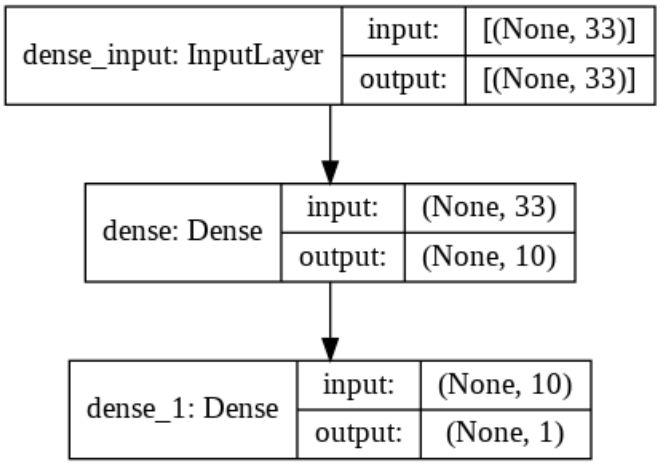


FIGURA 4.3: Arquitectura de la red neuronal.
Fuente: Elaboración propia.

4.2.2. Proceso de aprendizaje del modelo de ANN

Luego del pre-procesamiento de datos explicado anteriormente, se almacenarán los datos en una matriz. Considerando las proporciones de las particiones, la cual fue detallada en la sección (4.1.1) y las dimensiones de cada conjunto de datos mostrado en (5.1), se tendrán los datos almacenados en X_train, cuya dimensión es de $cantidad\ instancias\ x\ 33$, y en Y_train, cuya dimensión es de $1\ x\ 33$. De esta manera se da inicio a la fase de *Forward Propagation* (2.1.1).

Comenzando el proceso, se inicializa una matriz de pesos con el mismo tamaño que los atributos de entrada. En este caso contamos con 10 nodos en la siguiente capa, entonces la dimensión de la matriz de pesos W_1 es de $33x10$. También proporcionamos un sesgo, representado por b_1. Los pesos y el sesgo son inicializados de manera aleatoria usando *Glorot Uniform initializer*. Teniendo los datos de entrada, los pesos y el bias, se calcula el input de la capa oculta mediante la fórmula (2.2), en donde obtenemos $Z^{[1]}$. Luego de la suma ponderada se pasa a la función de activación ReLu (2.1.3), dando una salida $A^{[1]}$. Para la capa de salida, se multiplica el output de la capa oculta con la matriz de pesos W_2 y se proporciona un bias b_2, la dimensión de W_2 es 10 x nodos, en donde la cantidad de nodos de la capa de salida se observa en la tabla (4.2). Se calcula el input de la capa de salida de la misma manera descrita anteriormente, mediante la suma ponderada mostrada en (2.2), obteniendo $Z^{[2]}$. Y usando la función de activación Sigmoidea (2.1.3) en las predicciones binarias, en los casos de clasificación multiclase se usó la función de activación Softmax (2.1.3), dando la salida $A^{[2]}$.

Continuando con el proceso de aprendizaje, calculamos la pérdida comparando los valores predichos con los valores reales usando la función de pérdida Binary Cross-Entropy (2.1.4) o Categorical Cross-Entropy (2.1.4), según como corresponda. Esto asociación se puede observar en la tabla (4.2).

Durante el proceso de *Back-Propagation* (2.1.1), se toma la derivada de la

pérdida para la capa de salida y la capa oculta. Los pesos W_1 y W_2 se actualizan utilizando la técnica de optimización Adam, explicada en la sección (2.1.5), implementado en Keras (3.2.3).

Los valores de los hiperparámetros utilizados están detallados en la tabla (4.3).

Hiperparámetro	Valor
Épocas	500
Tamaño de batch	128
Tasa de aprendizaje	0.001

TABLA 4.3: Hiperparámetros de entrenamiento

El proceso de aprendizaje de la ANN continúa hasta que la cantidad de épocas fijada sea completada. Los atributos de la ANN se muestra en la figura (4.4).

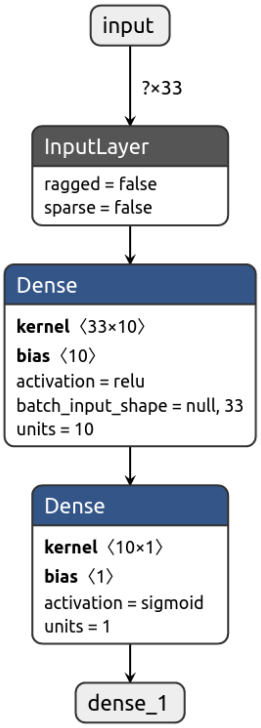


FIGURA 4.4: Atributos de la red neuronal de clasificación binaria.
Fuente: Elaboración propia

4.2.3. Arquitectura del modelo de DNN

El modelo de DNN tiene como entrada los atributos mostrados en la tabla (4.1), para todos los casos el input tiene una dimensión de 33, de la misma forma que el modelo de ANN, pero estas difieren en el siguiente módulo. Este modelo cuenta con 5 capas ocultas. La primera capa cuenta con 100 nodos, la segunda oculta cuenta con 50 nodos, continúa la tercera capa con 30 nodos, luego la cuarta capa con 10 nodos y finalmente la última capa oculta cuenta con 5 nodos. El último módulo, la capa de salida cuenta con las mismas condiciones que el modelo de ANN mostrados en la tabla (4.2).

La arquitectura del modelo de DNN se puede observar en la figura (4.5).

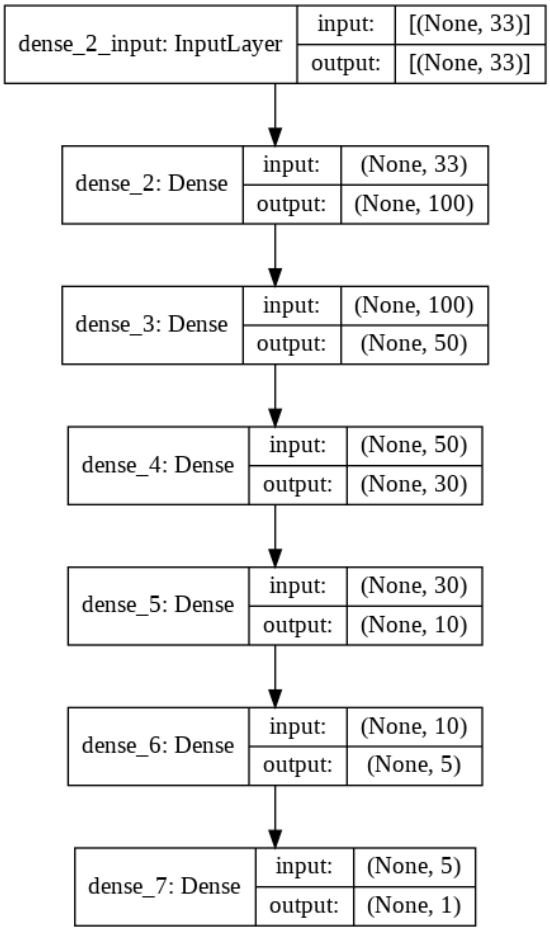


FIGURA 4.5: Arquitectura de la red neuronal profunda.
Fuente: Elaboración propia.

4.2.4. Proceso de aprendizaje del modelo de DNN

El proceso y cálculo de la propagación hacia adelante (2.1.1) en DNN es el mismo que en ANN. Manteniéndose el input para la capa de entrada, X_train e Y_train, cuya dimensión es $cantidad\,instancias\,x\,33$ y $1\,x\,33$ respectivamente. Inicializamos los pesos W_i y bias b_i mediante *Glorot Uniform Initializer*, pero las dimensiones de estas varían según la cantidad de nodos de cada capa oculta. Por ejemplo, W_1 tendrá una dimensión de 33x100, ya que la primera capa oculta cuenta con 100 nodos. Con los datos de entrada, los pesos y bias, el input de la capa oculta será presentada por Z_i, y calculada mediante la fórmula (2.2). Este proceso se repetirá para todas las capas ocultas, usando la función de activación ReLu (2.1.3) para cada capa.

En la última capa, la función Binary Cross-Entropy (2.1.4) o Categorical Cross-Entropy (2.1.4) se utiliza para calcular la pérdida. En el caso de la clasificación binaria se utiliza la función de activación Sigmoidea (2.1.3) y, en el caso de la clasificación multiclase se utiliza Softmax (2.1.3), dando la salida $A^{[i]}$. Luego, durante la retropropagación (2.1.1), la derivada de la función de pérdida con respecto a pesos y bias se toma en cada capa. Los pesos y los bias, W_i y b_i, se actualizan mediante el algoritmo de optimización Adam (2.1.5). Los valores de los hiperparámetros utilizados están detallados en la tabla (4.3).

El proceso de aprendizaje de la DNN continúa hasta que la cantidad de épocas fijada sea completada. Los atributos de la DNN se muestra en la figura (4.6).

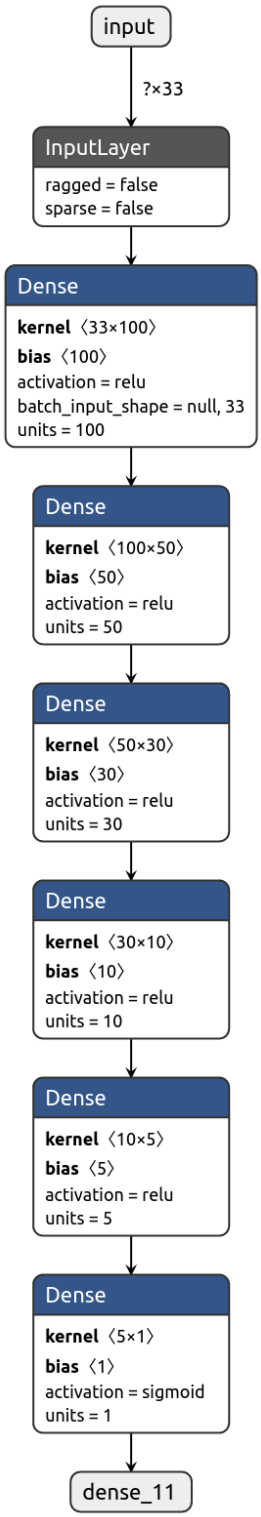


FIGURA 4.6: Atributos de la red neuronal profunda de clasificación binaria.
Fuente: Elaboración propia

Capítulo 5

Resultados y Discusiones

En este capítulo se muestran los resultados de la investigación, tanto los gráficos generados como el resultado predictivo de la red neuronal y red neuronal profunda realizada y un análisis e interpretación de estos resultados, considerando las métricas descritas en el estado del arte. Adicionalmente, se compara los modelos de ANN y DNN con algunos algoritmos tradicionales de aprendizaje, tales como Regresión logística, Maquinas de vectores de soporte y Naive Bayes .

5.1. Resultados

En esta sección se muestran los resultados obtenidos del sistema desarrollado. Se presenta primero los resultados producidos a partir del procesamiento de datos con la técnica sintética de sobremuestreo de minorías. Luego, a partir de los nuevos conjuntos de datos, se evalúa el rendimiento de la DNN, mostrando los resultados para cada conjunto de datos.

5.1.1. Técnica sintética de sobremuestreo de minorías

Mediante el preprocesamiento de datos descrito en la sección (4.1), obtenemos como resultado nuevas dimensiones para los conjuntos de datos. Estos son detallados en la tabla (5.1). La cantidad de atributos no ha sido afectada.

Predicción	Cantidad de atributos	Cantidad de instancias
Suicidio	33	333,572
Éxito	33	309,326
Tipo de arma	33	1,016,526
Region	33	563,394
Tipo de ataque	33	733,443

TABLA 5.1: Tabla de dimensiones del conjunto de entrenamiento.

5.1.2. Rendimiento del ANN

Accuracy durante el entrenamiento y validación

En la figura (5.1), se muestra la gráfica de la relación entre el accuracy respecto a las épocas calculado por el modelo de ANN. Se muestra en la figura (5.1a) para la predicción de suicidio, en la figura (5.1b) para la predicción de éxito, en la figura (5.1c) para la predicción del tipo de arma, en la figura (5.1d) para la predicción de la región y en la figura (5.1e) para la predicción de el tipo de ataque.

La curva de color azul representa la evolución del accuracy a lo largo de las épocas en el proceso de entrenamiento y la curva de color naranja representa la evolución del accuracy a lo largo de las épocas en el proceso de validación. Durante el entrenamiento se tuvo un porcentaje final de $97\,\%$, $84\,\%$, $75\,\%$, $96\,\%$ y $80\,\%$ para la predicción del suicidio, éxito, tipo de arma, región y tipo de ataque respectivamente. Durante la validación se tuvo un porcentaje final de $97\,\%$, $83\,\%$, $75\,\%$, $93\,\%$ y $85\,\%$ para la predicción del suicidio, éxito, tipo de arma, región y tipo de ataque respectivamente.

5.1.3. Rendimiento de la DNN

Para poder conocer el rendimiento de la DNN, se basa en los indicadores, mostrados en la sección (2.1.6), y el *Loss*. Estos y la *Matriz de confusión* son necesarios para poder dar un correcto veredicto acerca del modelo

desarrollado. El *Loss* o pérdida es el indicador que penaliza cuando la predicción es incorrecta.

Accuracy durante el entrenamiento y validación

En la figura (5.2), se muestra la gráfica de la relación entre el accuracy respecto a las épocas o epochs de los conjuntos de datos de entrenamiento (*X_train*) y validación (*X_test*) calculada por el modelo de DNN.

Se muestra en la figura (5.2a) el entrenamiento para la predicción de suicidio, en la figura (5.2b) para la predicción de éxito, en la figura (5.2c) para la predicción del tipo de arma, en la figura (5.2d) para la predicción de la región y en la figura (5.2e) para la predicción de el tipo de ataque.

La curva de color azul representa la evolución del accuracy a lo largo de las épocas en el proceso de entrenamiento y la curva de color naranja representa la evolución del accuracy a lo largo de las épocas en el proceso de validación. Durante el entrenamiento se tuvo un porcentaje final de $99\,\%$, $93\,\%$, $94\,\%$, $98\,\%$ y $92\,\%$ para la predicción del suicidio, éxito, tipo de arma, región y tipo de ataque respectivamente. Durante la validación se tuvo un porcentaje final de $98\,\%$, $92\,\%$, $94\,\%$, $98\,\%$ y $92\,\%$ para la predicción del suicidio, éxito, tipo de arma, región y tipo de ataque respectivamente.

Loss durante el entrenamiento y validación

En la figura (5.3), se muestra la gráfica de la relación entre la pérdida o Loss respecto a las épocas o epochs de los conjuntos de datos de entrenamiento y validación del modelo de DNN.

Se muestra en la figura (5.3a) la pérdida durante la predicción de suicidio, en la figura (5.3b) la pérdida durante la predicción de éxito, en la figura (5.3c) la pérdida durante la predicción del tipo de arma, en la figura (5.3d) la pérdida durante la predicción de la región y en la figura (5.3e) la pérdida durante la predicción de el tipo de ataque.

La curva de color azul representa la evolución de la pérdida a lo largo de las épocas en el proceso de entrenamiento y la curva de color naranja

representa la evolución de la pérdida a lo largo de las épocas en el proceso de validación. Analizando las figuras, en los casos de clasificación binaria continúa decreciendo hasta la época 500, en cambio para los casos de multiclase decrece más lento luego de la época 100 o 150.

Matriz de confusión

Se muestran las matrices resultantes del modelo de DNN. En el caso de la clasificación binaria, suicidio y éxito, las tablas tienen una dimensión de 2x2 y son mostradas en las figuras (5.4a) y (5.4b) respectivamente. En el caso de clasificación multiclase, tipo de arma, región y tipo de ataque, las tablas tienen una dimensión igual a la cantidad de clases de cada caso y estas se muestran en las figuras (5.4c), (5.4d) y (5.4e) respectivamente.

Una matriz de confusión con la mayor cantidad de datos en la diagonal demuestra una alta precisión del modelo propuesto. Como se muestran en las figuras, las matrices cumplen la concentración de datos en la diagonal, por lo que se demuestra que el modelo DNN es eficiente al realizar predicciones.

Curva ROC

La curva ROC muestra el desempeño del modelo de clasificación en los umbrales de clasificación. El ROC calculado por DNN para las predicciones de suicidio y éxito se muestran en la figura (5.5). Y las predicciones para los casos de predicción multiclase como tipo de arma, región y tipo de ataque se muestra en la figura (5.6). El ROC muestra que DNN puede realizar una clasificación con un accuracy superior a $89\,\%$.

5.1.4. Comparación en las distintas métricas

La comparación de las métricas, descritas en la sección (2.1.6), entre el modelo de ANN y DNN se muestra en la figura (5.7). De estas figuras podemos observar que el modelo DNN obtiene un rendimiento mínimo del $90\,\%$ y

máximo de $99\,\%$ en accuracy, precisión, recall y F1-Score. Mientras que el modelo de ANN tiene un rendimiento del $64\,\%$ y máximo de $97\,\%$.

De estos resultados podemos observar que a medida que aumente el número de capas, la red alcanza un mejor rendimiento y podrá realizar predicciones más eficientes.

5.1.5. Comparación con algoritmos tradicionales

En esta sección, el rendimiento de los modelos basados en ANN y DNN se comparan con algunos algoritmos tradicionales de aprendizaje automático como: Regresión Logística, SVM y Naïve Bayes. Se usa las métricas de accuracy, precisión, recall y F1-Score promedio. Cuyos resultados se pueden visualizar en la tabla (5.2).

Algoritmo	Accuracy (%)	Precisión promedio (%)	Recall promedio (%)	F1-Score promedio (%)
Naive Bayes	68.5	67.8	64.8	64.2
Región Logística	77.4	75.0	81.6	77.1
MVS	79.9	75.6	79.8	76.7
ANN	87.0	90.8	82.5	86.0
DNN	95.3	95.0	95.0	95.3

TABLA 5.2: Comparación del rendimiento de ANN y DNN con algoritmos tradicionales de aprendizaje automático, como, Naive Bayes, Regresión Logística y MVS

5.2. Discusiones

En esta sección se evalúa los resultados obtenidos y problemas. Primero se comenta el rendimiento de la red neuronal profunda en comparación de la red neuronal. Luego se comentan los resultados obtenidos en comparación

con lo algoritmos tradicionales. Por último, se mencionan otros hiperparámetro modificados para mejorar el rendimiento de la red neuronal profunda.

5.2.1. Rendimiento de los modelos

Al observar el rendimiento de la DNN en comparación con la ANN para el caso de predicción del Suicidio, se encuentra que el accuracy obtenido por la ANN (5.2a) es cercano al obtenido por la DNN (5.1a), además la ANN no lográ mejorar luego de las 100 épocas.

Para el caso de predicción del Éxito, el entrenamiento de la ANN (5.1b) contiene ruido en la validación a comparación de la DNN (5.2b), además esta alcanza mayor accuracy.

Para el caso de predicción del Tipo de arma, el modelo ANN (5.1c) mantiene un accuracy de $75\,\%$ luego de 150 épocas, es decir, no es capaz de realizar ninguna mejora. Mientras que el modelo de DNN (5.2c) supera el $92\,\%$ luego de 150 epochs y luego continúa elevandose ligeramente. En este caso es más notorio la mejora del rendimiento de la DNN.

Para el caso de predicción de la Región, el modelo ANN (5.1d) logra superar un accuracy de $95\,\%$ desde el inicio pero no se observa una mejora hasta la época 250. Mientras que el modelo de DNN (5.2d) supera el accuracy de $98\,\%$.

Para el caso de predicción del Tipo de ataque, el modelo ANN (5.1e) tiene dos saltos, primero se eleva de $75\,\%$ a $79\,\%$ en la época $180 - 200$ y luego se eleva nuevamente en la época 460 superando el $80\,\%$. Mientras que el accuracy para el modelo DNN (5.2e) supera el $91\,\%$ luego de 100 épocas y se incrementa ligeramente a través de las épocas.

Luego del análisis de el rendimiento de la DNN, se demuestra que a medida que aumenta el número de capas, la red es capaz de aprender una no linealidad más compleja en macrodatos y, por lo tanto, puede realizar predicciones de manera más eficiente.

5.2.2. Algoritmos tradicionales

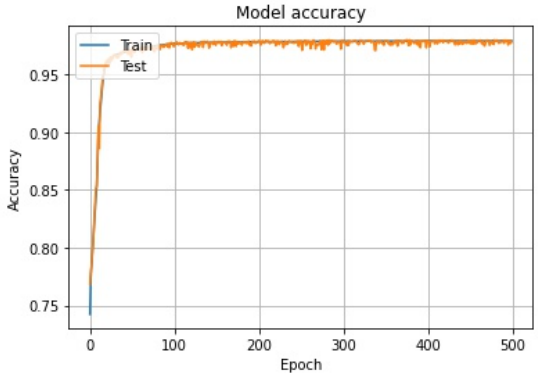
Los algoritmos tradicionales de aprendizaje automático mostrados en la tabla (5.2) e incluso el modelo de NN de una sola capa, no capturan el patrón en el conjunto de datos y no logran superar un rendimiento mayor a $87\,\%$. En cambio, el modelo de DNN logra un rendimiento promedio de $95\,\%$ en todas las métricas de evaluación.

Los resultados muestran que el modelo de DNN es el más adecuado para el conjunto de datos GTD, siendo este un ejemplo de macrodatos, podemos decir que el rendimiento mejora con un modelo de redes neuronales profundas.

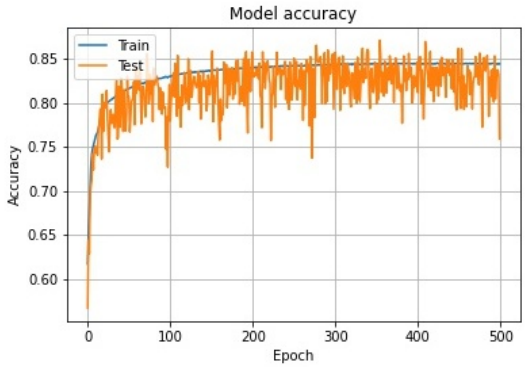
5.2.3. Hiperparámetros de DNN

Los hiperparámetros de un modelo de red neuronal y red neuronal profunda son importantes para obtener el mejor resultado posible. El primer parámetro a analizar son las *épocas*, el cual cumple la función de una iteración. En un inicio, se consideró igualarlo a 250 ya que en los casos de predicción multiclase no se observan una gran mejora a excepción del caso de Tipo de arma, aunque la mejora es leve para los demás, pero para el caso de predicción binaria del Éxito, el rendimiento mejora mientras las épocas aumentan, igualándose finalmente este parámetro a 500.

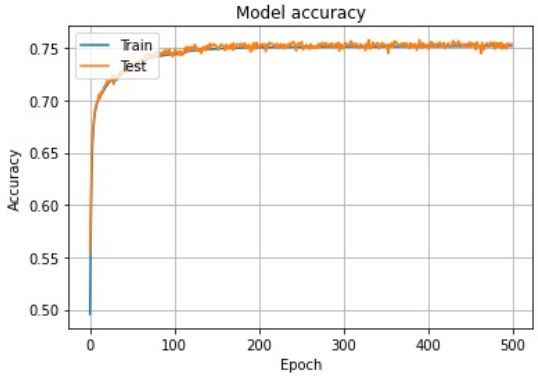
El segundo hiperparámetro a analizar es el *learning rate* o tasa de aprendizaje. Se sabe que, a medida que se reduce la tasa de aprendizaje, el entrenamiento logra ser más preciso pero resulta ser más lento. Mientras que si este es aumentado, el aprendizaje se vuelve rápido pero 'brusco'. Luego de realizarse pruebas con ciertas variaciones de la tasa de aprendizaje, y con fines de mejorar el rendimiento del modelo, se iguala a 0.001.



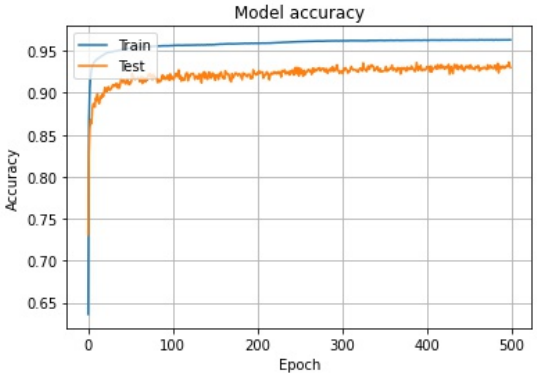
(A) ANN para la predicción del Suicidio.



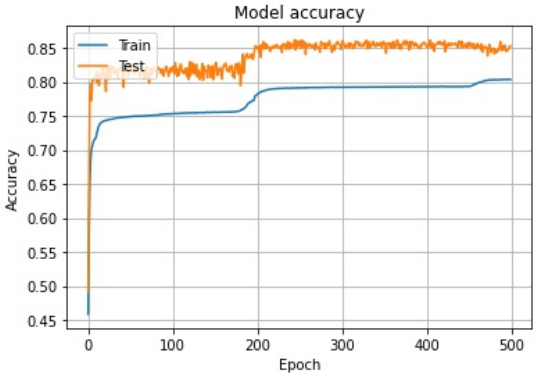
(B) ANN para la predicción de Éxito



(C) ANN para la predicción del Tipo de arma

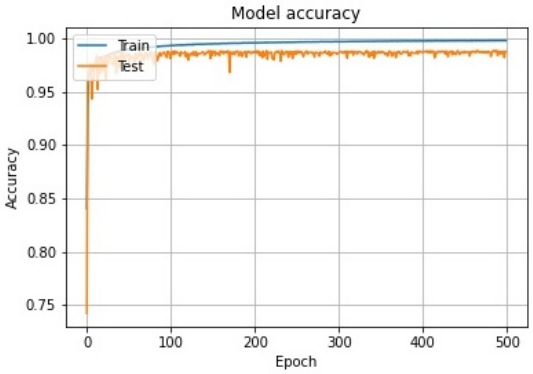


(D) ANN para la predicción de la Región

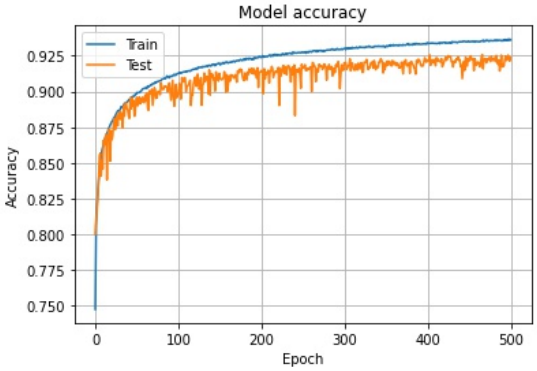


(E) ANN para la predicción de la Tipo de ataque

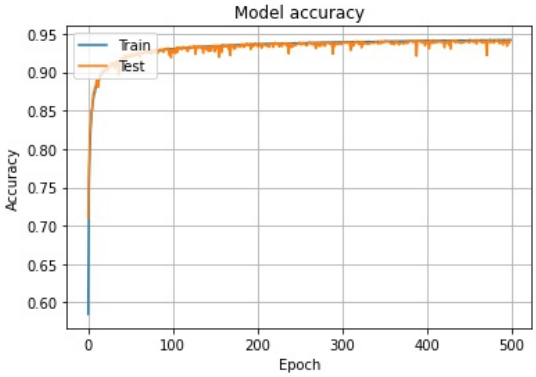
FIGURA 5.1: Gráfica de *Accuracy* vs. *Epochs* del modelo ANN.
Fuente: Elaboración propia



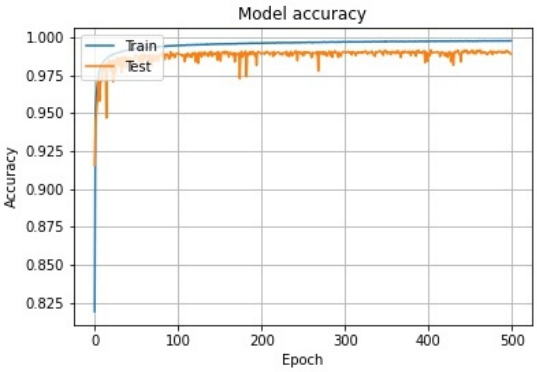
(A) DNN para la predicción del Suicidio.



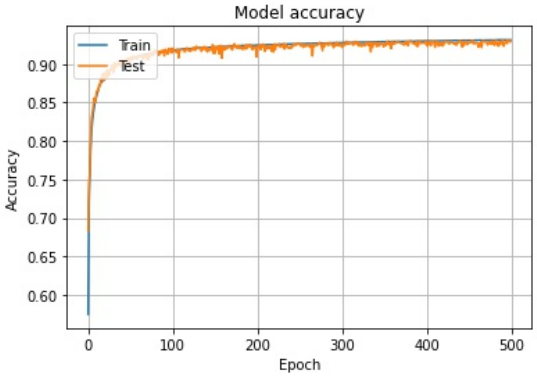
(B) DNN para la predicción de Éxito



(C) DNN para la predicción del Tipo de arma

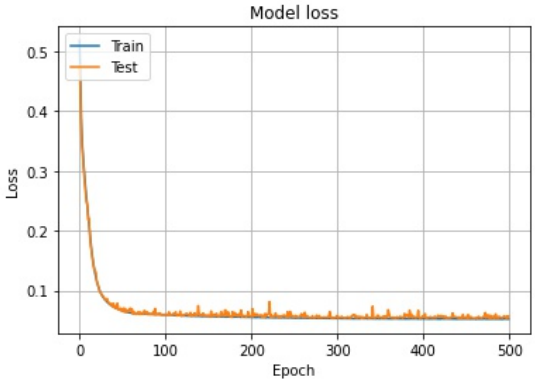


(D) DNN para la predicción de la Región

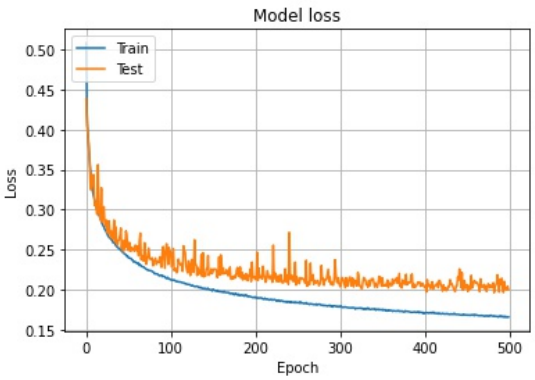


(E) DNN para la predicción de la Tipo de ataque

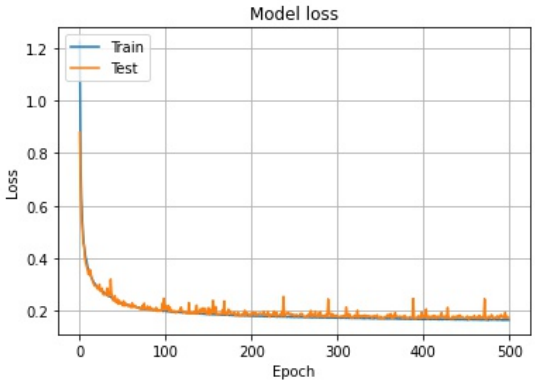
FIGURA 5.2: Gráfica de *Accuracy* vs. *Epochs* del modelo DNN.
Fuente: Elaboración propia



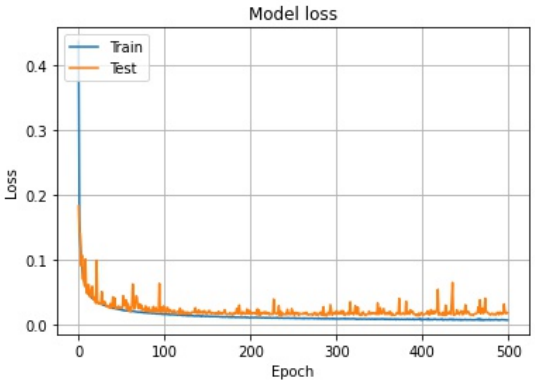
(A) Pérdida de la DNN para la predicción del Suicidio



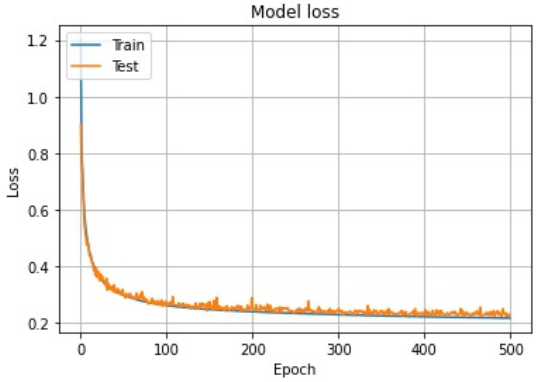
(B) Pérdida de la DNN para la predicción del Éxito



(C) Pérdida de la DNN para la predicción del Tipo de arma

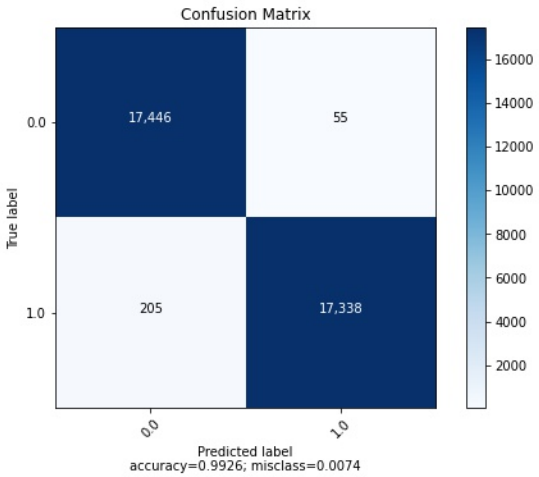


(D) Pérdida de la DNN para la predicción de la región

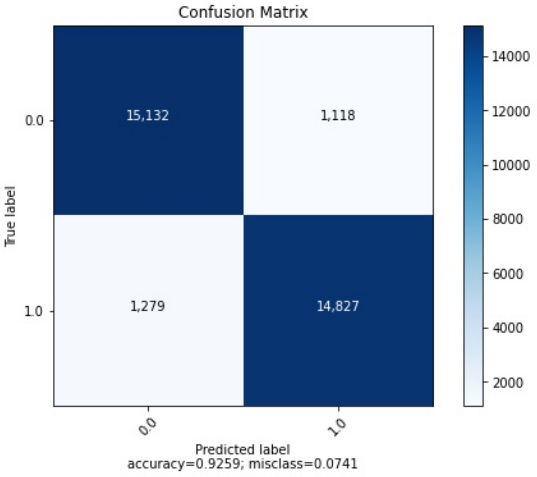


(E) Pérdida de la DNN para la predicción del Tipo de arma

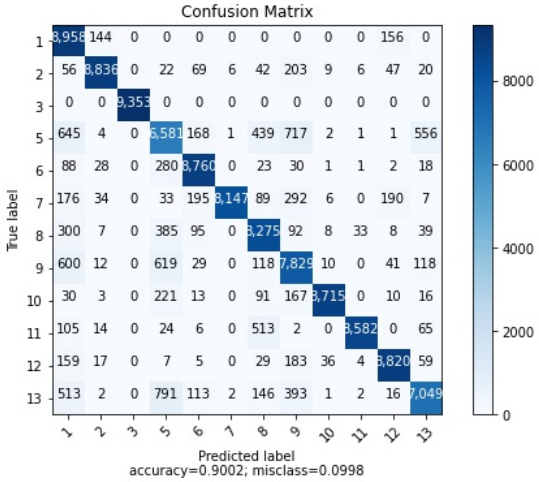
FIGURA 5.3: Gráfica de *Loss* vs. *Epochs* del modelo DNN.
Fuente: Elaboración propia



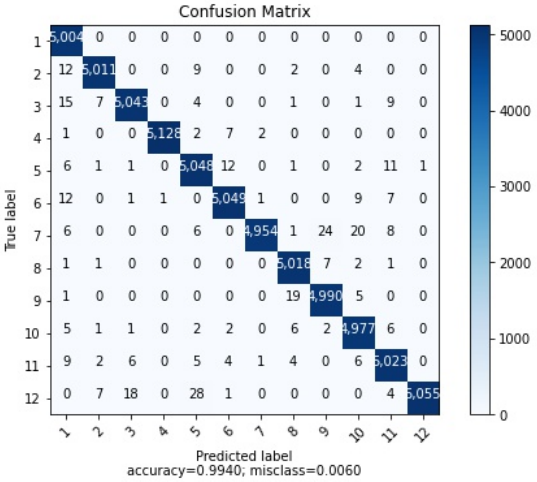
(A) Matriz de confusión del Suicidio



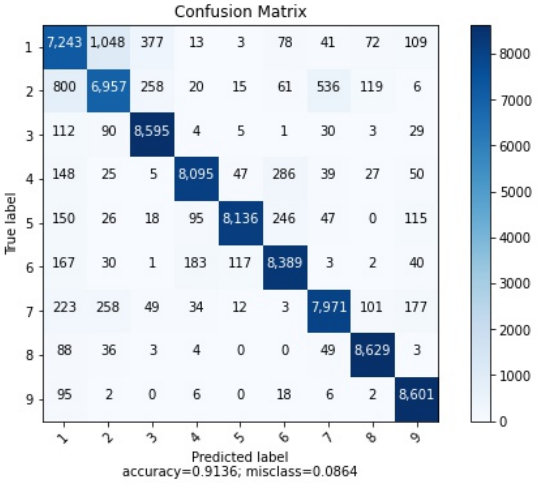
(B) Matriz de confusión del Éxito



(C) Matriz de confusión del Tipo de arma

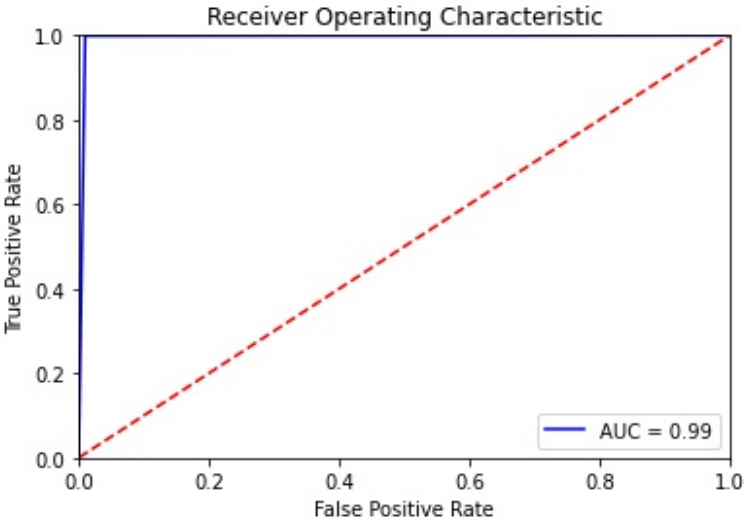


(D) Matriz de confusión de la Region

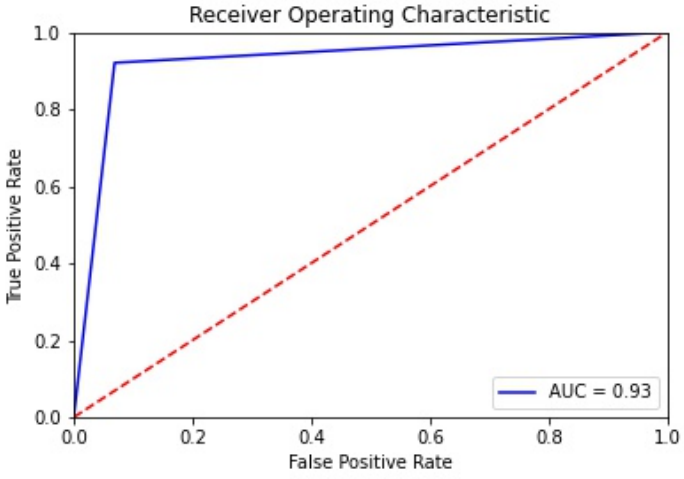


(E) Matriz de confusión del Tipo de ataque

FIGURA 5.4: Matriz de confusión del modelo DNN.
Fuente: Elaboración propia

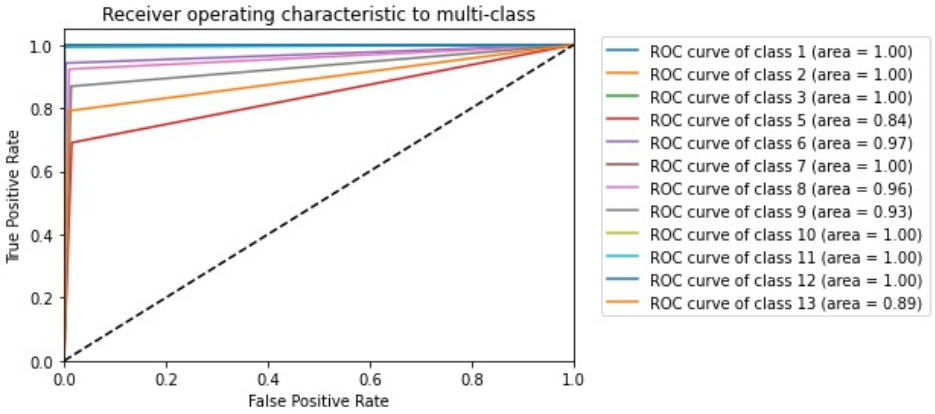


(A) Curva ROC de DNN para la predicción del Suicidio

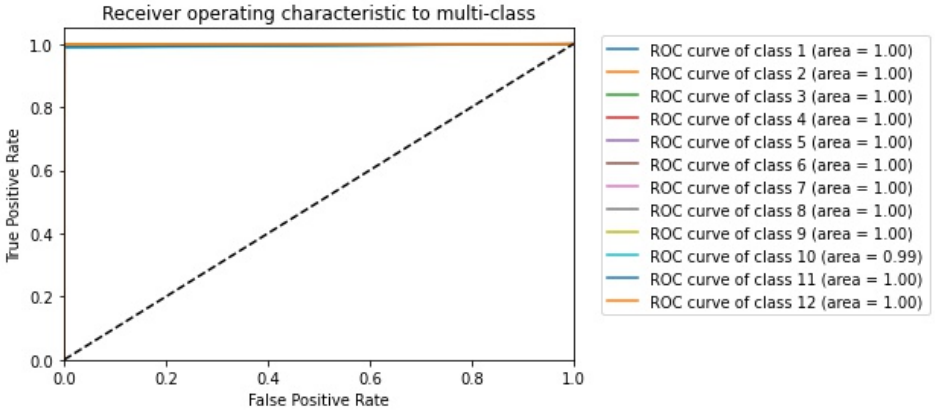


(B) Curva ROC de DNN para la predicción del Éxito

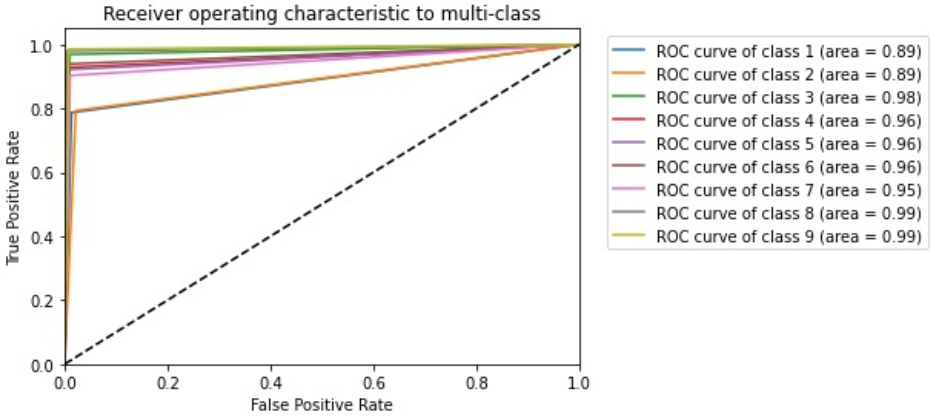
FIGURA 5.5: Curva ROC calculada por DNN en la predicción de Suicidio y Éxito.
Fuente: Elaboración propia



(A) Curva ROC de DNN para la predicción del Tipo de arma

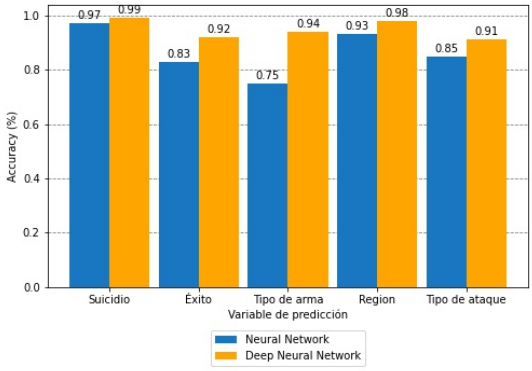


(B) Curva ROC de DNN para la predicción de la Region

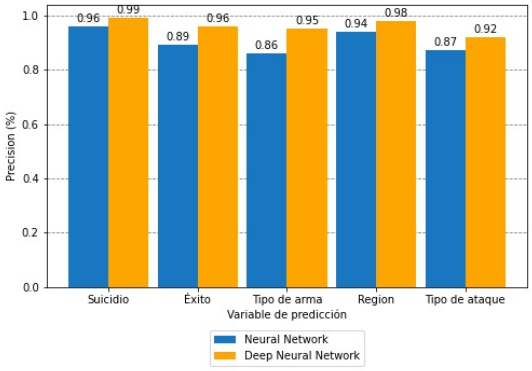


(C) Curva ROC de DNN para la predicción del Tipo de ataque

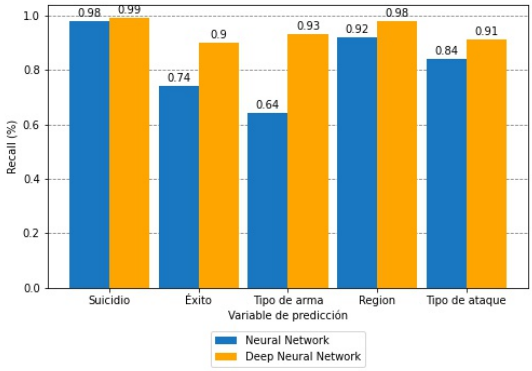
FIGURA 5.6: Curva ROC calculada por DNN en la predicción del Tipo de arma, Región y Tipo de ataque.
Fuente: Elaboración propia



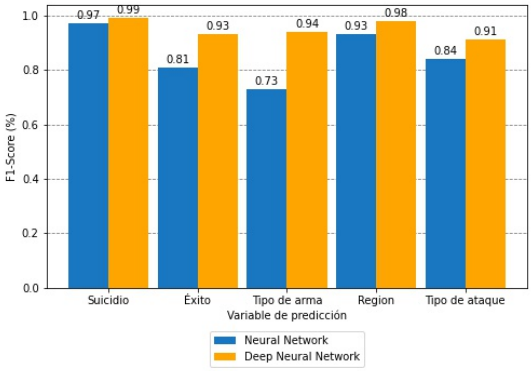
(A) Accuracy obtenido para ANN y DNN



(B) Precision obtenida para ANN y DNN



(C) Recall obtenido para ANN y DNN



(D) F1-Score obtenido para ANN y DNN

FIGURA 5.7: Comparación de las métricas: Accuracy, Precision, Recall y F1-Score para la red neuronal y red neuronal profunda
Fuente: Elaboración propia

Capítulo 6

Conclusiones y Trabajo a Futuro

En este último capítulo se describen las conclusiones y el trabajo a futuro. En las conclusiones se presenta lo que se logró realizar y los resultados obtenidos como consecuencia. En trabajo a futuro se plantea ideas y posibles mejoras del modelo desarrollado para su futura implementación, con el fin de ampliar el conocimiento en este campo.

6.1. Conclusiones

En el presente proyecto se ha investigado soluciones basadas en Inteligencia Artificial para comprender los diversos factores del terrorismo que puedan ayudar a realizar predicciones de futuras actividades terroristas. Se ha identificado cinco factores importantes de predecir, estos son si el tipo de ataque es suicida o no, si el ataque tiene éxito o no, qué tipo de arma se puede usar, qué región puede ser atacada y qué tipo de arma se utilizará.

Se desarrolló un modelo basado en Redes Neuronales y otro basado en Redes Neuronales Profundas para predecir los cincos factores descritos anteriormente pero los resultados demostraron que el modelo de Redes Neuronales Profundas es más preciso, alcanzando un accuracy de más del $95\,\%$.

Las técnicas de aprendizaje profundo pueden abarcas diversas áreas de aplicación, en este caso puede ayudar a los organismos encargados a tener mayor comprensión de los factores del terrorismo y poder prevenirlos.

De lo explicado hasta este punto, se presentan las conclusiones a partir de los resultados obtenidos en el presente proyecto:

- El uso de Aprendizaje profundo es conveniente para desarrollar un modelo de Red Neuronal Profunda para la predicción de cinco factores de las actividades terroristas.

- El Conjunto de Datos Mundial sobre el Terrorismo contiene clases desbalanceadas por lo que es favorable y mejora el rendimiento de la Red Neuronal Profunda utilizar la técnica de sobremuestreo, como parte del pre-procesamiento de datos de entrada.

- En comparación con los métodos tradicionales de aprendizaje automático y el modelo de Redes Neuronales, el modelo desarrollado de Redes Neuronales resulta ser el más adecuado para el Conjunto de Datos Mundial sobre el Terrorismo.

- El Conjunto de Datos Mundial sobre el Terrorismo es un ejemplo para macrodatos, por esta razón se demuestra que el uso de Redes Neuronales Profundas es adecuado, y su rendimiento mejora, para procesar macrodatos.

6.2. Trabajo a futuro

Para posibles trabajos a futuro, se propone el reto de optimizar el modelo de predicción, mediante la evaluación y análisis de algoritmos de selección de características, tal y como se propone en el artículo Jundong Li u. a. [2017], de esta forma igualar o mejorar el rendimiento de la red neuronal profunda desarrollado en el presente proyecto de tesis.

De igual manera, se propone mejorar el aprendizaje de la red neuronal profunda mediante distintas técnicas para lidear con clases desbalanceadas en el conjunto de datos de entrada, para el modelo presentado en este proyecto se aplica la técnica sintética de sobremuestreo de minorías pero se invita al lector a descubrir distintas técnicas.

Finalmente, los métodos desarrollados pueden ser aplicados en otros campos con el objetivo de ampliar el campo del aprendizaje profundo.

Bibliografía

[Agarwal u. a. 2019] AGARWAL, P. ; SHARMA, M. ; CHANDRA, S.: Comparison of Machine Learning Approaches in the Prediction of Terrorist Attacks. (2019)

[Brownlee 2020a] BROWNLEE, Jason: Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python. (2020)

[Brownlee 2020b] BROWNLEE, Jason: Imbalanced Classification with Python: Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Larning. (2020)

[Chawla u. a. 2002] CHAWLA, Nitesh ; BOWYER, Kevin ; HALL, Lawrence ; KEGELMEYER, W.: SMOTE: Synthetic Minority Over-sampling Technique. In: *J. Artif. Intell. Res. (JAIR)* 16 (2002), 06, S. 321–357

[Chollet u. a. 2015] CHOLLET, François u. a.: *Keras*. `https://keras.io`. 2015

[Cybenko 1989] CYBENKO, G.: Approximation by superpositions of a sigmoidal function. In: *Mathematics of Control, Signals and Systems* 2 (1989), S. 303–314

[Duan u. a. 2003] DUAN, K. ; KEERTHI, S. ; CHU, W. ; SHEVADE, S. ; POO, A.: Multi-category Classification by Soft-Max Combination of Binary Classifiers. In: *Multiple Classifier Systems*, 2003

[Foundation] FOUNDATION, Python S.: *Python 3.6.9 documentation*. – URL `https://docs.python.org/release/3.6.9/`

[Harris u. a. 2020] HARRIS, Charles R. ; MILLMAN, K. J. ; WALT, St'efan J. van der ; GOMMERS, Ralf ; VIRTANEN, Pauli ; COURNAPEAU, David ;

WIESER, Eric ; TAYLOR, Julian ; BERG, Sebastian ; SMITH, Nathaniel J. ; KERN, Robert ; PICUS, Matti ; HOYER, Stephan ; KERKWIJK, Marten H. van ; BRETT, Matthew ; HALDANE, Allan ; R'IO, Jaime F. del ; WIEBE, Mark ; PETERSON, Pearu ; G'ERARD-MARCHANT, Pierre ; SHEPPARD, Kevin ; REDDY, Tyler ; WECKESSER, Warren ; ABBASI, Hameer ; GOHLKE, Christoph ; OLIPHANT, Travis E.: Array programming with NumPy. In: *Nature* 585 (2020), September, Nr. 7825, S. 357–362. – URL `https://doi.org/10.1038/s41586-020-2649-2`

[Hunter 2007] HUNTER, J. D.: Matplotlib: A 2D graphics environment. In: *Computing in Science & Engineering* 9 (2007), Nr. 3, S. 90–95

[Ian Goodfellow 2016] IAN GOODFELLOW, Aaron C.: *Deep Learning*. MIT Press, 2016

[Jundong Li u. a. 2017] JUNDONG LI, Kewei C. ; WANG, Suhang ; MORSTATTER, Fred ; TREVINO, Robert P. ; TANG, Jiliang ; LIU, Huan: Feature Selection: A Data Perspective. (2017)

[Kingma und Ba 2015] KINGMA, Diederik P. ; BA, Jimmy: Adam: A Method for Stochastic Optimization. In: *CoRR* abs/1412.6980 (2015)

[Kogan] KOGAN, Gene: *Redes neuronales*. – URL `https://ml4a.github.io/ml4a/es/neural_networks/`

[Matthew Moocarme 2020] MATTHEW MOOCARME, Ritesh B.: *The Deep Learning with Keras Workshop*. Packt, 2020

[McCulloch und Pitts 1943] MCCULLOCH, W. S. ; PITTS, W.: A Logical Calculus of the Idea Immanent in Nervous Activity. In: *Bulletin of Mathematical Biophysics* 5 (1943), S. 115–133

[Nair und Hinton 2010] NAIR, V. ; HINTON, Geoffrey E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: *ICML*, 2010

[jupyter notebook] NOTEBOOK, The jupyter: *Jupyter Notebook 6.1.4 documentation*. – URL `https://jupyter-notebook.readthedocs.io/en/stable/notebook.html`

[Ogunbo u. a. 2020] OGUNBO, J. N. ; ALAGBE, O. A. ; OLADAPO, M. I. ; SHIN, C.: N-hidden layer artificial neural network architecture computer code: geophysical application example. (2020)

[Pedregosa u. a. 2011] PEDREGOSA, F. ; VAROQUAUX, G. ; GRAMFORT, A. ; MICHEL, V. ; THIRION, B. ; GRISEL, O. ; BLONDEL, M. ; PRETTENHOFER, P. ; WEISS, R. ; DUBOURG, V. ; VANDERPLAS, J. ; PASSOS, A. ; COURNAPEAU, D. ; BRUCHER, M. ; PERROT, M. ; DUCHESNAY, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[Polyak 1964] POLYAK, Boris T.: Some methods of speeding up the convergence of iteration methods. In: *Ussr computational mathematics and mathematical physics* 4 (1964), Nr. 5, S. 1–17

[Ruder 2016] RUDER, Sebastian: An overview of gradient descent optimization algorithms. In: *ArXiv* abs/1609.04747 (2016)

[Rumelhart u. a. 1986] RUMELHART, D. ; HINTON, Geoffrey E. ; WILLIAMS, R. J.: Learning internal representations by error propagation, 1986

[S McCulloch und Pitts 1990] S MCCULLOCH, W ; PITTS, W: A logical calculus of the ideas immanent in nervous activity. 1943. In: *Bulletin of mathematical biology* 52 (1990), 02, S. 99–115; discussion 73

[Shah u. a. 2020] SHAH, Syed Atif A. ; UDDIN, Irfan ; AZIZ, Furqan ; AHMAD, Shafiq ; AL-KHASAWNEH, Mahmoud A. ; SHARAF, Mohamed: An Enhanced Deep Neural Network for Predicting Workplace Absenteeism. (2020)

[Skansi 2018] SKANSI, Sandro: *Introduction to Deep Learning*. Springer, 2018

[START] START: *Global Terrorism Database*. – URL `https://start.umd.edu/gtd/`

[pandas development team 2020] TEAM, The pandas development: *pandas-dev/pandas: Pandas*. Februar 2020. – URL https://doi.org/10.5281/zenodo.3509134

[Tieleman und Hinton 2012] TIELEMAN, Tijmen ; HINTON, Geoffrey: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In: *COURSERA: Neural networks for machine learning* 4 (2012), Nr. 2, S. 26–31

[Villanueva 2020] VILLANUEVA, José D.: *Redes neuronales - Introduccion*. 2020. – URL https://www.iartificial.net/