UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Análisis del terrorismo: Predicción del perpetrador

Proyecto de Tesis II

Autor: Ingrid Ipanaqué Casquina

Asesor: Juan Carlos Espejo Delzo

2021

Resumen

El desafío más importante para la humanidad en todo el mundo son los ataques terroristas. Anticiparse al grupo terrorista, que es responsable de los resultados y actividades, utilizando información histórica es una tarea difícil debido a la escasez de datos terroristas detallados. Sin embargo, el campo de la investigación sobre el terrorismo es un campo activo dedicado al desarrollo, la evaluación de nuevos modelos, herramientas y técnicas de software para ayudar a combatirlo. Por esas razones, el presente proyecto tiene como objetivo desarrollar cuatro modelos predictivos, para comprender el comportamiento de los grupos terroristas, basados en aprendizaje automático y el uso de la técnica de análisis factorial datos mixtos sobre el conjunto de datos mundial sobre el terrorismo. La predicción se basa en obtener el nombre de los grupos atacantes o autores de una actividad terrorista, de esta manera poder prevenir este tipo de ataques. Los modelos utilizados son las Máquinas de vectores de soporte, Bosques aleatorios, Árboles de decisión y Máquina de aumento de gradiente. De los cuales, el modelo basado en árboles, Bosques aleatorios, demuestra predecir correctamente el nombre del grupo perpetrador obteniendo un rendimiento de $84\,\%$, $83\,\%$, $84\,\%$ y $82\,\%$ en las métricas: accuracy, precisión, recall y f1-score respectivamente. Se concluye que el rendimiento del modelo mejora favorablemente aplicando una técnica adecuada para la elección de los atributos. Además, se observa que el uso de modelos basados en árboles es adecuado para procesar la base de datos mundial sobre terrorismo.

PALABRAS CLAVE: GTD, Terrorismo, Máquinas de vectores de soporte, Bosques aleatorios, Árboles de decisión, Máquina de aumento de gradiente.

Índice general

Índice de figuras

Índice de cuadros

Índice de Acrónimos

IA	Inteligencia artificial
GTD	Global Terrorism Database
START	Study of Terrorism And Responses to Terrorism
TGPM	Terrorist Group Prediction Model for Counter Terrorism
FAMD	Factor Analysis of Mixed Data
MVS	Máquinas de Vectores de Soporte
SVR	Support Vector Regression
SVC	Support Vector Classifier
GBM	Gradient Boosting Machine
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 automático o más conocido en inglés como machine learning. El aprendizaje automático se enfoca en gestionar datos para transformarlos en información útil y generar conocimiento sin necesidad de supervisión humana. Además, obtiene tasas de éxito elevadas en distintos campos de estudio, es por esto que se aplica 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 para estudiar e identificar los posibles autores de una actividad terrorista. Debido a esto y a lo expuesto anteriormente que resulta interesante la aplicación de aprendizaje automático a ésta área, ya que está a disposición una gran cantidad de datos etiquetados.

Desde un inicio, este proyecto de investigación estuvo fomentado por el deseo de entender el comportamiento de los distintos modelos de aprendizaje automático, y compararlos entre sí, para la predicción del autor de una actividad terrorista. Si bien se sabe que el aprendizaje automático es un tema de investigación que se usa comunmente, no existen muchos artículos o trabajos de investigación que utilicen este tipo de modelos para entender el comportamiento de los grupos terroristas y así poder predecir su comportamiento. Además, hasta la fecha existen distintas librerías disponibles para los leenguajes de programación más usados, que nos ayudan a la implementación de diversos modelos de aprendizaje automático, además de contar con documentación, y material académico para realizar una correcta implementación de los modelos propuestos.

1.2. Objetivos

El objetivo de este proyecto de investigación es el de implementar cuatro modelos para la predicción del autor de una actividad terrorista mediante el uso de aprendizaje automático. Estos son: Máquinas de vectores de soporte, Árboles de decisión, Bosques aleatorios y Máquina de aumento de gradiente.

Específicamente, los objetivos de este trabajo con respecto al sistema son:

- Entender el funcionamiento de los modelos de máquinas de vectores de soporte, Árboles de decisión, Bosques aleatorios y Máquina de aumento de gradiente.

- Implementar los cuatro modelos predictivos mencionados anteriormente.
- Mejorar los resultados esperados aplicando distintas técnicas en el preprocesamiento de datos, y diferentes atributos de entrada mediante el análisis factorial datos mixtos.
- Evaluar el rendimiento de los modelos implementados.

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

- Obtener el conocimiento necesario para implementar los procesos involucrados en aprendizaje automático, empleando las herramientas de programación disponibles.
- Adquirir la capacidad de interpretar resultados erróneos producidos por los modelos luego del proceso de entrenamiento y validación.
- Obtener la capacidad de utilizar modelos de aprendizaje automático 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 Máquinas de Vectores de Soporte, Árboles de Decisión y Máquina de aumento de gradiente.

- **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 Máquinas de Vectores de Soporte, Árboles de Decisión, Bosques Aleatorios y Máquina de aumento de gradiente. 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 presentan los conceptos previos, los trabajos relacionados y conclusiones. Inicialmente, se presentan los conceptos previos utilizados en los trabajos relacionados y en este, para el mejor entendimiento y desarrollo de la solución planteada. En los trabajos relacionados, se presentan los trabajos que ayudaron a desarrollar el presente trabajo. Por último, en las conclusiones de este capítulo se presenta un consenso donde se indica el rumbo de este trabajo con respecto a los trabajos relacionados.

2.1. Conceptos previos

2.1.1. Aprendizaje automático

El objetivo principal del aprendizaje automático es estudiar, diseñar y mejorar modelos matemáticos que se puedan entrenar, una vez o continuamente, con datos relacionados con el contexto, para inferir el futuro y tomar decisiones sin completar conocimiento de todos los elementos que influyen. En otras palabras, un agente adopta un enfoque de aprendizaje estadístico, tratando de determinar las distribuciones de probabilidad correctas y utilizarlos para calcular la acción que es más probable que tenga éxito y con el menor error.

A continuación se explicarán algunos algoritmos de aprendizaje automático usados en el desarrollo del presente proyecto.

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.

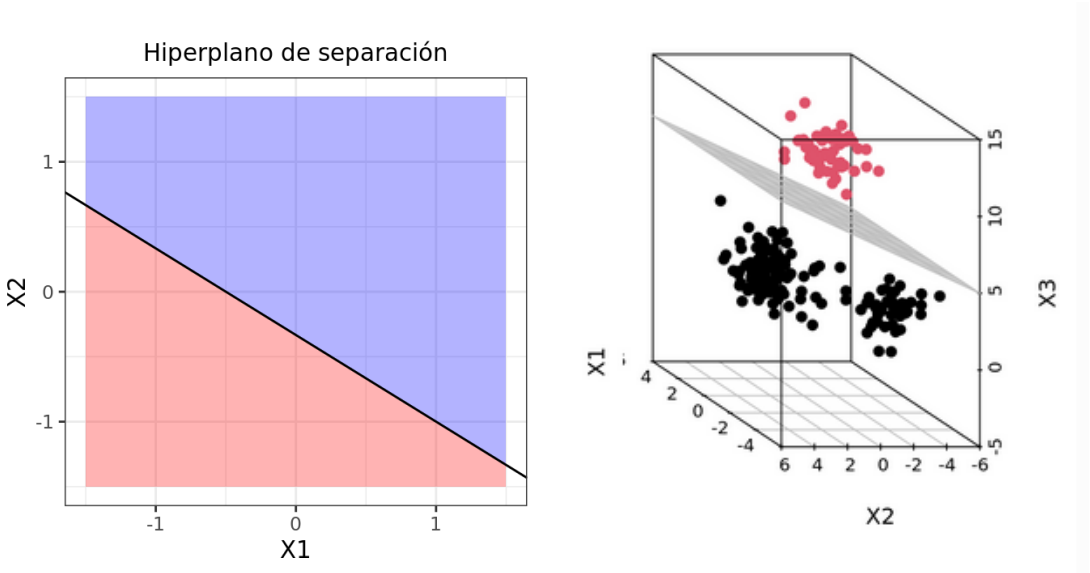
Fue introducido por primera vez por Cortes und Vapnik [1995] para problemas de clasificación, llamado clasificador de vectores de soporte. Posteriormente, Vapnik u. a. [1996] desarrolló el algoritmo SVM para los problemas de regresión, llamado regresor de vectores de soporte.

Las MVS, incluido el clasificador de vectores de soporte (SVC) y el regresor de vectores de soporte (SVR), se encuentran entre los métodos más robustos y precisos en todos los algoritmos de minería de datos conocidos. Las SVM, que fueron desarrolladas originalmente por Vapnik en la década de 1990, tienen una sólida base teórica arraigada en la teoría del aprendizaje estadístico y, a menudo, son insensibles al número de dimensiones. Últimamente, las SVM se han desarrollado a un ritmo rápido tanto en la teoría como en la práctica.

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 (2.1a) 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 (2.1b) muestra el hiperplano en un espacio tridimensional.



(A) 2D (B) 3D

FIGURA 2.1: Hiperplano de separación
Fuente: https://www.cienciadedatos.net

Aunque el tiempo de entrenamiento de incluso las SVM más rápidas puede ser extremadamente lento, son muy precisos debido a su capacidad para modelar límites de decisión no lineales complejos. Son mucho menos propensos al sobreajuste que otros métodos. Los vectores de soporte encontrados también proporcionan una descripción compacta del modelo aprendido.

Árboles de Decisión

Los árboles de decisión son un método de aprendizaje supervisado no paramétrico que se utiliza para clasificación y regresión. El objetivo es crear un

modelo que prediga el valor de una variable objetivo mediante el aprendizaje de reglas de decisión simples inferidas de las características de los datos.

Los árboles de decisión pueden funcionar de manera eficiente con conjuntos de datos no normalizados porque su estructura interna no está influenciada por los valores asumidos por cada característica. [Bonaccorso, 2017]

Un árbol de decisión se dibuja al revés con su raíz en la parte superior. En la figura (2.2) el texto en negrita representa una condición (/internal node), en base a lo cuál el árbol se divide en ramas (/edges). Al final de la rama que ya no se divide, se encuentra la decisión (/leaf), en este caso, si el pasajero falleció o sobrevivió, representado en texto rojo o verde respectivamente.

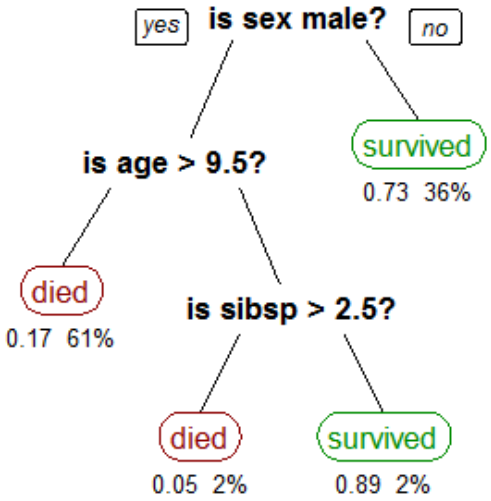


FIGURA 2.2: Ejemplo de un Árbol de decisión
Fuente: https://towardsdatascience.com

Esta metodología se conoce más comúnmente como árbol de decisión de aprendizaje a partir de datos y el árbol mostrado en la figura (2.2) se denomina árbol de clasificación, ya que el objetivo es clasificar al pasajero como sobreviviente o muerto.

Por lo tanto, usamos árboles de decisión para modelar el conjunto de datos presentado en el presente proyecto. Adicionalmente se realizan pruebas con el algoritmo C4.5.

Algoritmo C4.5 , es el sucesor de ID3. ID3(Iterative Dichotomiser 3), fue desarrollado por Quinlan [1993].

C4.5 convierte los árboles entrenados (es decir, la salida del algoritmo ID3) en conjuntos de reglas si-entonces. Luego, se evalúa la precisión de cada regla para determinar el orden en el que deben aplicarse.

Day	Outlook	Temperature	Humidity	Windy	Play Golf?
1	Sunny	85	85	False	No
2	Sunny	80	90	True	No
3	Overcast	83	78	False	Yes
4	Rainy	70	96	False	Yes
5	Rainy	68	80	False	Yes
6	Rainy	65	70	True	No
7	Overcast	64	65	True	Yes
8	Sunny	72	95	False	No
9	Sunny	69	70	False	Yes
10	Rainy	75	80	False	Yes
11	Sunny	75	70	True	Yes
12	Overcast	72	90	True	Yes
13	Overcast	81	75	False	Yes
14	Rainy	71	80	True	No

FIGURA 2.3: Ejemplo de entrada de conjunto de datos en C4.5
Fuente: Xindong Wu [2009]

Vemos cómo se induce el árbol de la figura 2.4 a partir de la figura 2.3. El primer atributo elegido para una prueba de decisión es $Outlook$. Para entender el ¿por qué?, se debe calcular la entropía de la variable de clase (¿PlayGolf?). Esta variable toma dos valores con probabilidad 9/14 (para "Sí") y 5/14 (para "No"). La entropía de una variable aleatoria de clase que toma valores de c con probabilidades $p_1, p_2, ..., p_c$ viene dado por la ecuación (2.1).

$$E(p_i) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{2.1}$$

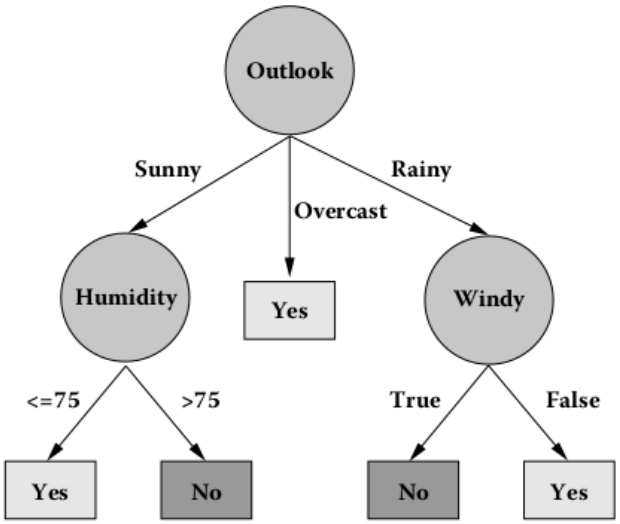


FIGURA 2.4: Árbol de decisión inducido por C4.5 para el conjunto de datos de la figura (2.3)
Fuente: Xindong Wu [2009]

La entropía de $PlayGolf?$ usando la ecuación (2.1) resulta 0.940. Consideramos cada atributo a su vez para evaluar la mejora en la entropía que proporciona. Para una variable aleatoria dada, digamos $Outlook$, la mejora en la entropía, representada como $Gain(Outlook)$, se calcula según la ecuación (2.2):

$$Gain(atributo) = E(\text{atributo en D}) - \sum_{v} \frac{|D_v|}{|D|} E(\text{atributo en D}) \tag{2.2}$$

En donde, v es el conjunto de posibles valores, D denota el conjunto de datos completos, D_v es el subconjunto de datos para el cual el atributo tiene ese valor y la notación $|\cdot|$ denota el tamaño de un conjunto de datos.

Este cálculo muestra que $Gain(Outlook) = 0.940 - 0.694 = 0.246$. Del mismo modo se calcula $Gain(Windy) = 0,940 - 0,892 = 0,048$ y a los demás atributos de forma sistemática demostrando que $Outlook$ es el mejor atributo para ramificar.

En Python(3.2.1) podemos encontrar la librería Chefboost (3.2.3), en donde, los árboles de decisión construidos se almacenan como declaraciones if de

Python en un directorio `outputs/rules`. En el gráfico (2.5) se muestra un ejemplo de reglas de decisión.

```
def findDecision(Outlook, Temperature, Humidity, Wind, Decision):
   if Outlook == 'Rain':
      if Wind == 'Weak':
         return 'Yes'
      elif Wind == 'Strong':
         return 'No'
      else:
         return 'No'
   elif Outlook == 'Sunny':
      if Humidity == 'High':
         return 'No'
      elif Humidity == 'Normal':
         return 'Yes'
      else:
         return 'Yes'
   elif Outlook == 'Overcast':
      return 'Yes'
   else:
      return 'Yes'
```

FIGURA 2.5: Ejemplo de reglas de decision del algoritmo C4.5
Fuente: https://github.com

Bosques Aleatorios

Como técnica útil para problemas de pronóstico, Breiman [2001] introdujo el algoritmo Bosques Aleatorios.

Un bosque aleatorio es un conjunto de árboles de decisión construidos sobre muestras aleatorias con una política diferente para dividir un nodo: en lugar de buscar la mejor opción, en dicho modelo, se usa un subconjunto aleatorio de características (para cada árbol), tratando de encuentre el umbral que mejor separe los datos. (2.6)

Como resultado, habrá muchos árboles entrenados de una manera más débil y cada uno de ellos producirá una predicción diferente.

Hay dos formas de interpretar estos resultados; el enfoque más común se basa en un voto mayoritario (la clase más votada se considerará correcta). Sin embargo, scikit-learn (3.2.3) implementa un algoritmo basado en promediar los resultados, lo que produce predicciones muy precisas. Incluso si son teóricamente diferentes, el promedio probabilístico de un bosque aleatorio

entrenado no puede ser muy diferente de la mayoría de las predicciones (de lo contrario, debería haber diferentes puntos estables); por lo tanto, los dos métodos a menudo conducen a resultados comparables. [Bonaccorso, 2017]

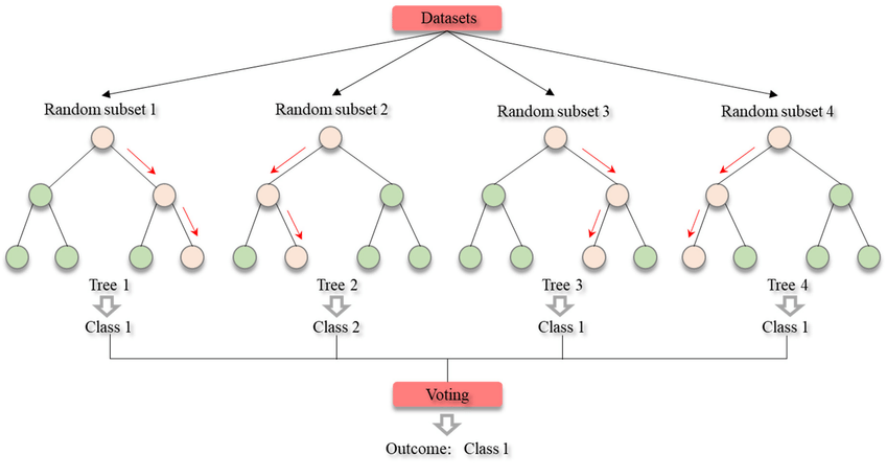


FIGURA 2.6: Flujo de los bosques aleatorios
Fuente: https://researchgate.com

Gradient Boosting Machines

Las Máquinas de Aumento de Gradiente o Gradient Boosting Machines es una técnica potente de aprendizaje automático, las cuales han demostrado un éxito considerable en una amplia gama de aplicaciones. Son personalizables para distintas necesidades particulares de la aplicación, como aprender con respecto a diferentes funciones de pérdida.

Boosting es un método que convierte a los *learners* o aprendices débiles en aprendices fuertes. En este método, cada nuevo árbol se ajusta a una versión modificada del conjunto de datos original. Es decir que las máquinas de aumento de gradiente entrenan de manera gradual, aditiva y secuencial. Ya que se construyen de manera secuencial, cada uno de los modelos posteriores intenta reducir el error del modelo anterior. Esto se logra construyendo el nuevo modelo sobre errores o residuales de las predicciones anteriores.

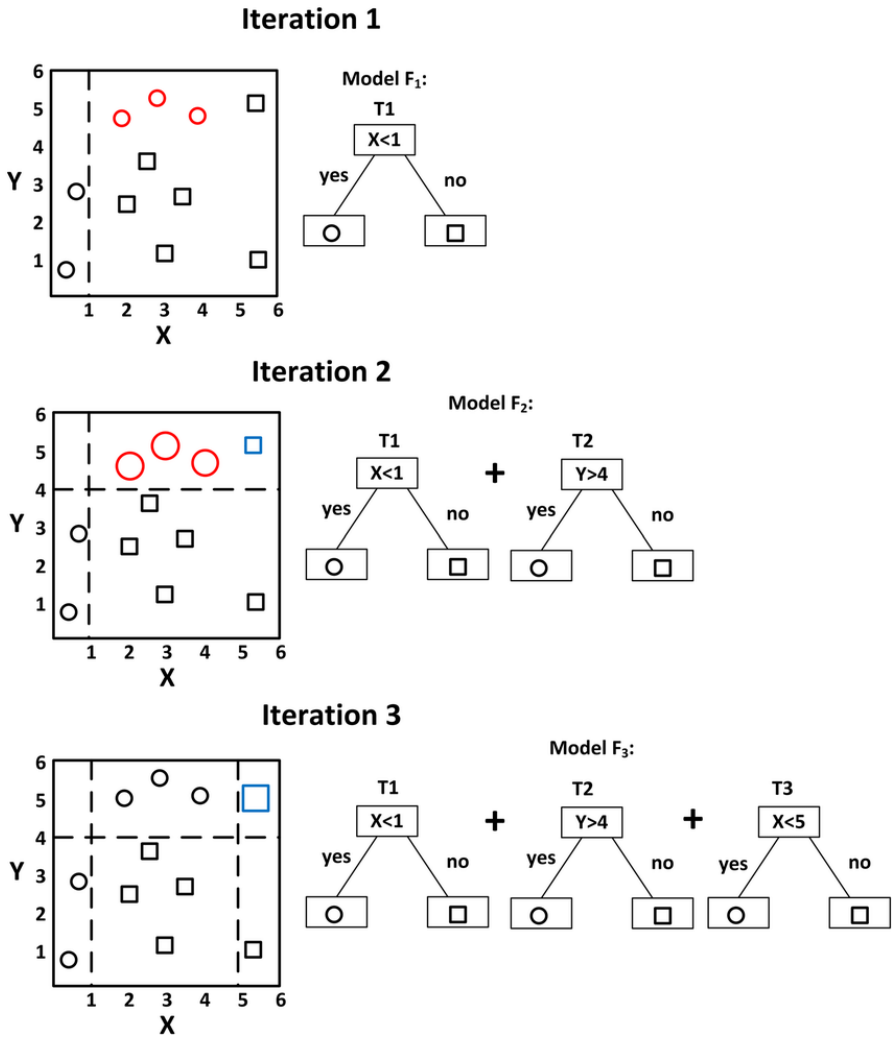


FIGURA 2.7: Representación simple de visualización de las Máquinas de Aumento de Gradiente.
Fuente: https://researchgate.com

Este modelo se basa en la intuición de que el mejor modelo siguiente posible, cuando se combina con modelos anteriores, minimiza el error de predicción general. La idea clave es establecer los resultados objetivo para el próximo modelo con el fin de minimizar el error. El resultado objetivo para cada caso en los datos depende de cuánto impacta el cambio de predicción de ese caso en el error de predicción general.

Si un cambio pequeño en la predicción de un caso provoca una gran caída en el error, el resultado objetivo siguiente del caso es un valor alto y las predicciones del nuevo modelo que están cerca de sus objetivos reducirán el error.

Si un cambio pequeño en la predicción de un caso no provoca ningún cambio en el error, el resultado objetivo siguiente del caso es cero. Por lo tanto, cambiar esta predicción no reduce el error.

El nombre de Aumento de Gradiente surge porque los resultados objetivo para cada caso se establecen en función del gradiente del error con respecto a la predicción. Cada modelo nuevo da un paso en la predicción que minimiza el error, en el espacio de posibles predicciones para cada caso de entrenamiento.

2.1.2. 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 (2.3).

$$Accuracy = \frac{Cantidad\,de\,predicciones\,correctas}{Cantidad\,total\,de\,predicciones} \tag{2.3}$$

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 (2.4).

$$Precision = \frac{Verdadero\ Positivo}{Verdadero\ Positivo + Falso\ Positivo} \tag{2.4}$$

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 (2.5).

$$recall = \frac{Verdadero\ Positivo}{Verdadero\ Positivo + Falso\ Negativo} \tag{2.5}$$

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 (2.6).

$$F1\ Score = 2x\frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.6}$$

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. Terrorism Analytics, Learning to Predict the Perpetrator

Trabajo realizado y publicado por Talreja u. a. [2017]. En este trabajo se analiza el conjunto de datos del GTD y se predice el perpretador o autor de un ataque terrorista en India. Para este objetivo se utilizan técnicas de análisis de datos mixtos y algoritmos de aprendizaje automático como Árboles de Decisión, Bosques Aleatorios y Máquinas de Vectores de Soporte. En donde se obtiene un accuracy de $60\,\%$, $58.5\,\%$ y $73.2\,\%$ respectivamente.

2.2.2. TGPM: Terrorist Group Prediction Model for Counter Terrorism

Trabajo realizado y publicado por Sachan und Roy [2012]. En este trabajo se analiza el conjunto de datos del GTD y se predice el perpretador o autor de un ataque terrorista en India. Para este objetivo se construye un modelo de predicción de grupos terroristas (TGPM) para predecir el grupo terrorista involucrado en un ataque dado. Este modelo inicialmente aprende similitudes de incidentes terroristas de varios ataques terroristas para predecir el grupo responsable. En donde se obtiene un accuracy de $80.41\,\%$.

2.2.3. An Experimental Study of Classification Algorithms for Terrorism Prediction

Trabajo realizado y publicado por Tolan und Soliman [2015]. En este trabajo se analiza el conjunto de datos del GTD y se predice el perpretador o autor de un ataque terrorista en Egipto. Para este objetivo se utilizan técnicas de clasificación de minería de datos, tales como, Naïves Bayes, K-Nearest Neighbour (KNN), Tree Induction (C4.5), Iterative Dichotomiser (ID3), y

Support Vector Machine (SVM). En donde se obtiene un accuracy de $69\,\%$, $72.5\,\%$, $56.3\,\%$, $21.1\,\%$ y $71.8\,\%$ respectivamente.

2.3. Conclusiones

Conforme a lo presentado anteriormente, tanto en avances como en investigaciones previas, se puede observar que el tema del terrorismo es estudiado con distintos enfoques, ya sea con distintos modelos de aprendizaje automático u orientado a países específicos. Además podemos concluir que el campo de investigación de *Aprendizaje automático* ha sido aplicado exitosamente en el área del terrorismo.

En este caso, el presente proyecto se basará en el enfoque planteado por el artículo [Talreja u. a., 2017], usando la técnica de análisis de datos mixtos para la selección de atributos, pero expandiendo el enfoque a nivel mundial y comparando los resultados con distintos modelos de aprendizaje automático.

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]

Scikit-learn

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

XGBoost

XGBoost es una biblioteca de software de código abierto para C ++, Java, Python, R, Julia, Perl y Scala. Es una biblioteca optimizada de aumento de gradiente distribuida diseñada para ser altamente eficiente, flexible y portátil. Implementa algoritmos de aprendizaje automático bajo el marco Gradient Boosting. [Chen und Guestrin, 2016]

Chefboost

Chefboost es un marco de árbol de decisión ligero de aumento de gradiente, bosque aleatorio y adaboost que incluye ID3, C4.5, CART, CHAID y algoritmos de árbol de regresión regulares con soporte de características categóricas. [Serengil, 2019]

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 la secuencia seguida durante la investigación. Primero se detallan los atributos utilizados del conjunto de datos GTD y los pasos seguidos durante el preprocesamiento de los datos, en la sección *Preprocesamiento de datos*. En las secciones siguientes se detallan los modelos de aprendizaje automático utilizados. Estos son: Máquinas de vectores de soporte, Árboles de decisión, Bosques aleatorios y por último, Gradient boosting machines.

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 cuentan con $105,203$ instancias y se utilizan 17 atributos para los modelos de aprendizaje automático.

Debido a la alta dimensión de los datos, se utiliza el Análisis Factorial de Datos Mixtos (FAMD) en el conjunto de datos para encontrar los atributos que contribuyen más a predecir a los perpetradores. Gracias a FAMD se utilizan

16 atributos de entrada de los 135 brindados por el conjunto de datos. 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.
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	Country	El país o locación en donde ocurrió el incidente.
6	Region	El código de región basado en 12 regiones.
7	Latitude	La latitud de la ciudad en la que ocurrió el evento.
8	Longitude	La longitud de la ciudad en la que ocurrió el evento.
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	Attacktype1	El método general de ataque y la amplia clase de tácticas utilizadas.
15	Targtype1	El tipo general de objetivo/víctima. Consta de 22 categorías.
16	Weaptype1	Éxito de un ataque terrorista.
17	Gname	El nombre del grupo que llevó a cabo el ataque.

TABLA 4.1: Atributos del GTD para los modelos de aprendizaje automático.

En la tabla (4.2) se muestra un registro típico de un incidente terrorista registrado en el conjunto de datos.

iyear	2015
imonth	1 (Enero)
iday	1
extended	0
country	113 (Libya)
region	10 (Middle East & North Africa)
latitude	32.069287
longitude	20.151145
vicinity	0
crit1	1
crit1	1
crit3	1 (Outside international humanitarian law)
doubtterr	0
attacktype1	3 (Bombing/Explosion)
targtype1	17 (Terrorists/Non-State Militias)
weaptype1	6 (Explosives)
gname	Shura Council of Benghazi Revolutionaries

TABLA 4.2: Registro típico en el conjunto de datos GTD.

Atributos de entrada y salida

En la tabla (4.1) se muestran todos los atributos usados del conjunto de datos GTD, estos se despliegan en atributos de entrada y salida para los distintos modelos.

El atributo de salida para los distintos modelos es la columna etiquetada como $gname$. Este campo contiene el nombre del grupo que llevó a cabo el ataque. Si no hay información disponible sobre el grupo perpetrador, este campo se codifica como "Desconocido". El campo de $gname$ cuenta con $3,616$ nombres de grupos terroristas, entre los cuales tenemos:

- Taliban
- Islamic State of Iraq and the Levant (ISIL
- Shining Path (SL)
- Al-Shabaab
- 23rd of September Communist League
- Black Nationalists
- Farabundo Marti National Liberation Front (FMLN)
- etc.

Los atributos restantes son los atributos de entrada para los modelos.

Teniendo más claro la distribución de los datos a lo largo del conjunto de datos, 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 $80\,\%$ para el entrenamiento y $20\,\%$ 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. Limpieza de datos

El presente proyecto está orientado a la predicción del grupo terrorista o autor de un ataque, es por esta razón que analizamos la información contenida en el campo $gname$ del conjunto de datos GTD. En la tabla (4.3) podemos observar algunos de los nombres de los grupos terroristas con su respectivo porcentaje de presencia en el conjunto de datos.

Grupo terrorista	Ataques (%)
Unknown	45.053
Taliban	4.558
Islamic State of Iraq and the Levant (ISIL)	3.334
Otros	<3 %

TABLA 4.3: Nombres de los grupos terroristas con su respectivo porcentaje

De la tabla (4.3) se observa que el $45\,\%$ de los registros están etiquedatados como $Unknown$ o $Desconocido$. Debido a que se desea predecir el grupo terrorista los registros desconocidos son eliminados del conjunto de datos.

4.1.3. Codificación de etiquetas

En el conjunto de datos de GTD, existen algunas características en formato de texto, como el nombre del grupo. No es factible procesar este tipo de datos en los modelos de Máquinas de Vectores de Soporte y Bosques aleatorios, 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]

4.1.4. 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 (4.1.4) se puede observar cuáles son las columnas que tienen mayor cantidad de datos faltantes.

De la gráfica mostrada se puede observar que las columnas que contienen data faltante son: $latitude$, $longitude$ con una cantidad considerable de datos nulos. Se pueden utilizar diferentes técnicas de interpolación para completar este tipo de datos.

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, 2020, cap. 8]

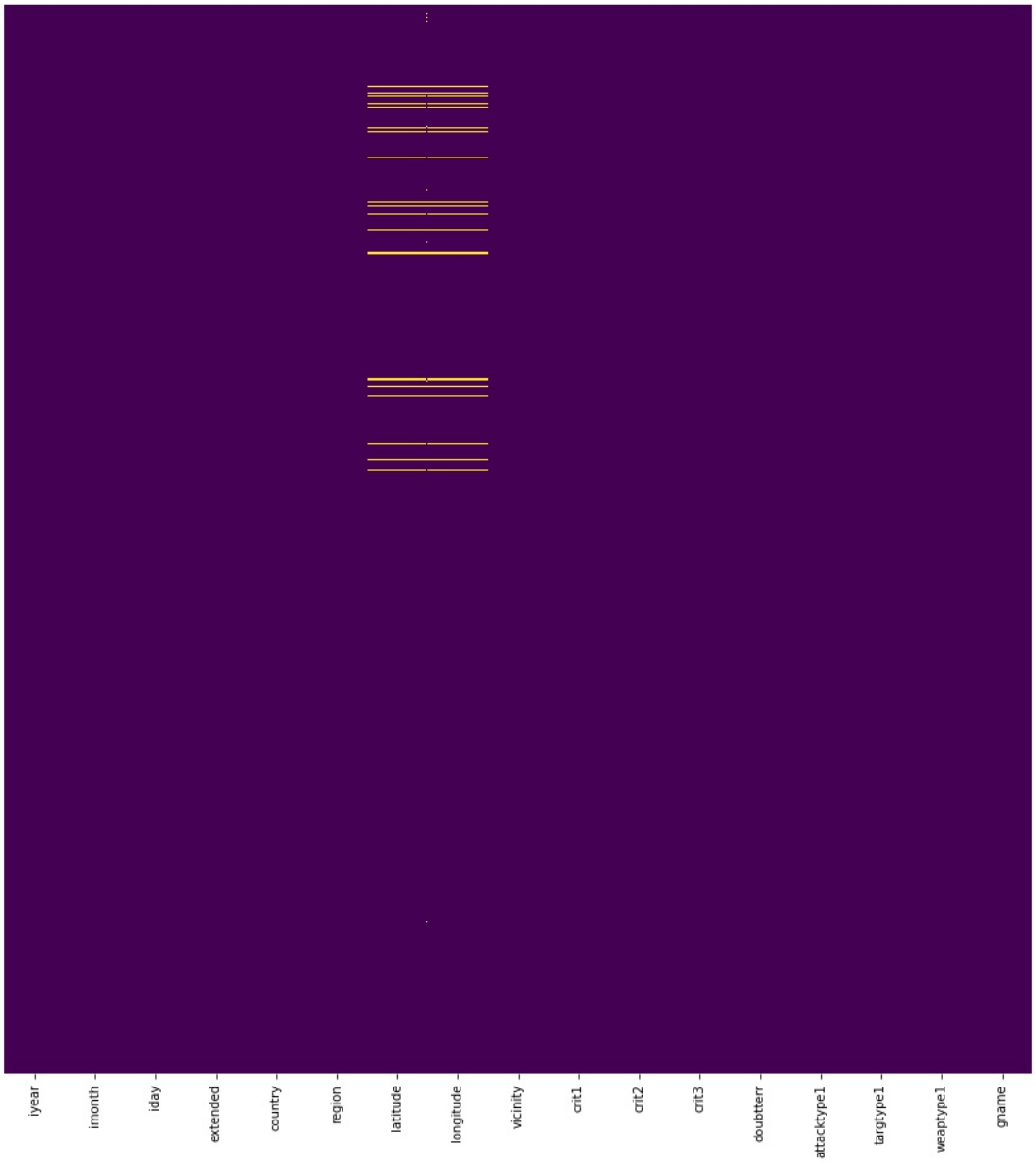


FIGURA 4.1: Gráfico informativo de datos faltantes
Fuente: Elaboracion propia

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.

4.2. Máquinas de Vectores de Soporte

Las máquinas de Vectores de Soporte funciona muy bien en problemas de clasificación, especialmente cuando el conjunto de datos no contiene mucho ruido y es equilibrado. Por esto, analizamos los registros en el atributo de salida $gname$, y se observa que existen nombres de grupos terroristas con menos de 10 registros en el conjunto de datos. Estos registros generan ruido en el modelo y se opta por eliminarlos.

Dado que el conjunto de datos resultante contiene características independientes y un conjunto de entrenamiento grande, se realiza el entrenamiento del modelo con kernel lineal. Esto ayuda a evitar el sobreajuste.

Se entrena el clasificador de Máquinas de Vectores de Soporte con los 16 atributos de entrada descritos en (4.1.1), usando la librería SVC de Scikit-learn (3.2.3) implementada en Python (3.2.1). Dados los datos de entrenamiento etiquetados, el algoritmo genera un hiperplano óptimo que categoriza los registros.

4.3. Árboles de Decisión

Debido a que en el conjunto de datos GTD existen datos con relaciones complejas y no lineales entre los parámetros de entrada y el parámetro de

salida o la predicción, se utiliza un modelo basado en árboles de decisión para modelar los datos y predecir el grupo perpetrador de cada ataque.

El árbol de decisión se centra en las relaciones entre varios eventos o atributos y, por lo tanto, replica el curso natural de los eventos.

Se entrena el clasificador de árbol de decisión con los 16 atributos de entrada descritos en (4.1.1), usando la librería DecisionTreeClassifier implementada en Python (3.2.1). En este algoritmo, los datos se dividen en subgrupos de forma recursiva, se selecciona un atributo y se formula una prueba lógica sobre este atributo. Luego, dependiendo del resultado de la prueba, el arból se ramifica. El subconjunto de datos que satifacen dicho resultado se mueve al nodo hijo correspondiente y las pruebas se ejecutan recursivamente en el nodo hijo.

4.4. Bosques Aleatorios

El algoritmo de Bosques Aleatorios es uno de los mejores algoritmos de clasificación que es capaz de clasificar grandes cantidades de datos con precisión.

Para cada uno de los árboles, el modelo implementa el método de bagging para generar un conjunto de datos de entrenamiento aleatorio. Las características de división también son seleccionadas de forma semi-aleatoria. Se proporciona un subconjunto aleatorio de una proporción especificada del espacio de posibles características de división. El pseudocódigo de Bosques Aleatorios se muestra en (1).

To generate c classifiers:

for *i=1 to c* **do**
    Randomly sample the training data D with replacement to produce D_i
    Create a root node, N_i contining D_i
    Call BuildTree(N_i)
end

BuildTree(N**):**

if *N no contains instances of only one class* **then**
    Randomly select x % of the possible splitting features in N
    Select features F with the highest information gain to split on
    Create f child nodes of $N, N_1, ..., N_f$, where F has f possible values ($F_1, ..., F_f$)
    for *i=1 to f* **do**
        Set the contents of N_i to D_i, where D_i is all instances in N that match F_i
        Call BuildTree(N_i)
    end
end

Algoritmo 1: Bosques Aleatorios. Fuente: Guoa u. a. [2019]

Se entrena el modelo de bosques aleatorios con los 16 atributos de entrada descritos en (4.1.1), usando la librería RandomForestClassifier de Scikit-learn (3.2.3) implementada en Python (3.2.1), utilizando el criterio $gini$ para la información de $Gain$ (2.2).

4.5. Gradient Boosting Machines

El algoritmo de Gradient Boosting Machines o Máquinas de Aumento de Gradiente es atractivo ya que puede dar predicciones o una buena clasificación

de datos a partir de una gran cantidad de datos de entrada como características, a su vez, permite relaciones flexibles y no lineales. Otros métodos son menos flexibles. El pseudocódigo de este modelo se muestra en (2).

Initialized all predictions to the sample log-odds:
$f_i^{(0)} = log\frac{\hat{p}}{1-\hat{p}}$
for $j \in 1...M$ **do**
  Compute the residual (i.e. gradient) $z_i = y_i$ - $\hat{p}_i$
  Randomly sample the training data
  Train a tree model on the random subset using the residuals as the outcome
  Compute the terminal node estimates of the Pearson residuals: $r_i = \frac{1/n\sum_i^n (y_i - \hat{p}_i)}{1/n\sum_i^n \hat{p}_i(1-\hat{p}_i)}$
  Update the current model using $f_i = f_i + \lambda f_i^{(j)}$
end

Algoritmo 2: Aumento de gradiente simple para clasificación. Fuente: Max Kuhn [2014]

Se entrena el modelo de bosques aleatorios con los 16 atributos de entrada descritos en (4.1.1), usando la librería XGBClassifier de XGBoost (3.2.3) implementada en Python (3.2.1).

Los valores de los hiperparámetros utilizados están detallados en la tabla (4.4).

Hiperparámetro	Valor
Número de estimadores	16
Tasa de aprendizaje	0.001

TABLA 4.4: Hiperparámetros de entrenamiento

Capítulo 5

Resultados y Discusiones

En este capítulo se muestran los resultados obtenidos a partir de las técnicas mencionadas en el capítulo anterior, tanto los gráficos generados como el resultado predictivo de los modelos de aprendizaje automático, tales como Máquinas de vectores de soporte, Árboles de Decisión, Bosques aleatorios y Máquinas de aumento de gradiente, y un análisis e interpretación de estos resultados considerando las métricas descritas en el estado del arte. En la subsección *Pre-procesamiento de datos*, se muestra el conjunto de datos con el que trabajan los modelos de aprendizaje automático. En la subsecciones siguientes, se muestran los resultados obtenidos por los modelos antes mencionados, En la subsección *Comparación de los modelos* se realiza una comparación de los resultados obtenidos. Finalmente en *Discusiones*, se discute e interpreta los resultados obtenidos.

5.1. Resultados

En esta sección se muestran los resultados obtenidos de los modelos desarrollados. Se presenta primero los resultados producidos a partir del procesamiento de datos. Luego, a partir del conjunto de datos depurado, se evalúa el rendimiento de los modelos de MVS, Árboles de Deción, Bosques Aleatorios y Máquinas de aumento de gradiente, mostrando los resultados para cada modelo.

5.1.1. Pre-procesamiento de datos

Mediante el pre-procesamiento de datos descrito en la sección (4.1), obtenemos como resultado una nueva dimensión para el conjuntos de datos GTD ofrecido por [START]. Esta nueva dimensión es detallada en la tabla (5.1).

Conjunto de datos	Cantidad de atributos	Cantidad de instancias
GTD	135	201,183
GTD procesado	17	105,203

TABLA 5.1: Dimensiones del conjunto de datos

En la figura (5.1) podemos observar una muestra del conjunto de datos resultante del pre-procesamiento de datos.

iyear	imonth	iday	extended	country	region	latitude	longitude	vicinity	crit1	crit2	crit3	doubtterr	attacktype1	targtype1	weaptype1	gname
1970	0	0	0	130	1	19.371887	-99.086624	0	1	1	1	0	6	7	13	23rd of September Communist League
1970	1	1	0	217	1	37.005105	-89.176269	0	1	1	1	0	2	3	5	Black Nationalists
1970	1	2	0	218	3	-34.891151	-56.187214	0	1	1	1	0	1	3	5	Tupamaros (Uruguay)
1970	1	1	0	217	1	43.468500	-89.744299	0	1	1	0	1	3	4	6	Weather Underground, Weathermen
1970	1	6	0	217	1	39.758968	-104.876305	0	1	1	1	1	7	4	8	Left-Wing Militants

FIGURA 5.1: Muestra del conjunto de datos no normalizado
Fuente: Elaboracion propia

5.1.2. Máquinas de Vectores de Soporte

Luego del entrenamiento y validación del modelo de MVS, se obtiene un porcentaje final para el accuracy de $65.6\,\%$. Además, para la precisión, recall y f1-score también obtuvo un porcentaje final de $65.6\,\%$.

Support Vector Machines muestra funcionar bien en problemas de clasificación, especialmente cuando el conjunto de datos sin ruido. Además, evita el sobreajuste de datos.

5.1.3. Árboles de decisión

Los Árboles de decisión son algoritmos naturalmente explicables e interpretables. Además, también podemos encontrar los valores de importancia de las características para comprender cómo funciona el modelo.

La importancia de las características se refiere a una clase de técnicas para asignar puntuaciones a las características de entrada a un modelo predictivo que indica la importancia relativa de cada característica al realizar una predicción. Para el modelo de Árboles de decisión se calcula la importancia de las carácterísticas y esta es mostrada en la figura (5.2).

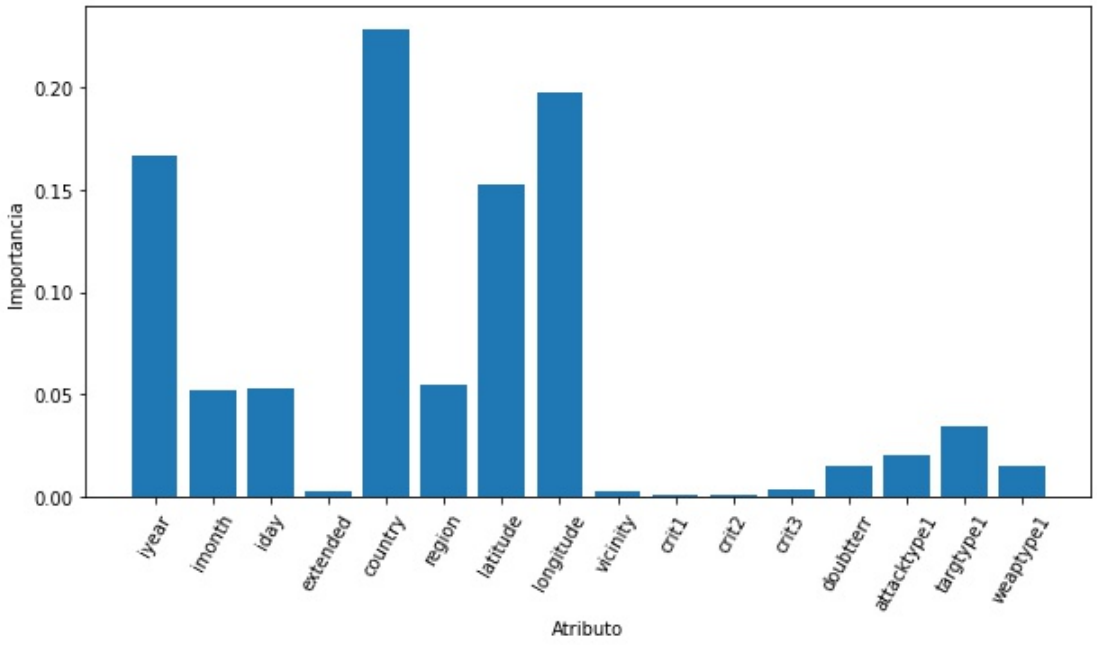


FIGURA 5.2: Importancia de las características del modelo de Árboles de decisión
Fuente: Elaboracion propia

Luego del entrenamiento y validación del modelo de Árbol de decisión, se obtiene un accuracy de $79.9\,\%$. Además, obtiene una precisión de $80.4\,\%$, un porcentaje de $79.9\,\%$ para el recall y $79.8\,\%$ para el f1-score. Superando de esta manera a los resultados mostrados por el modelo de MVS.

5.1.4. Bosques Aleatorios

Para el modelo de Bosques aleatorios se calcula la importancia de las carácterísticas o atributos de entrada al modelo y esta es mostrada en la figura (5.3).

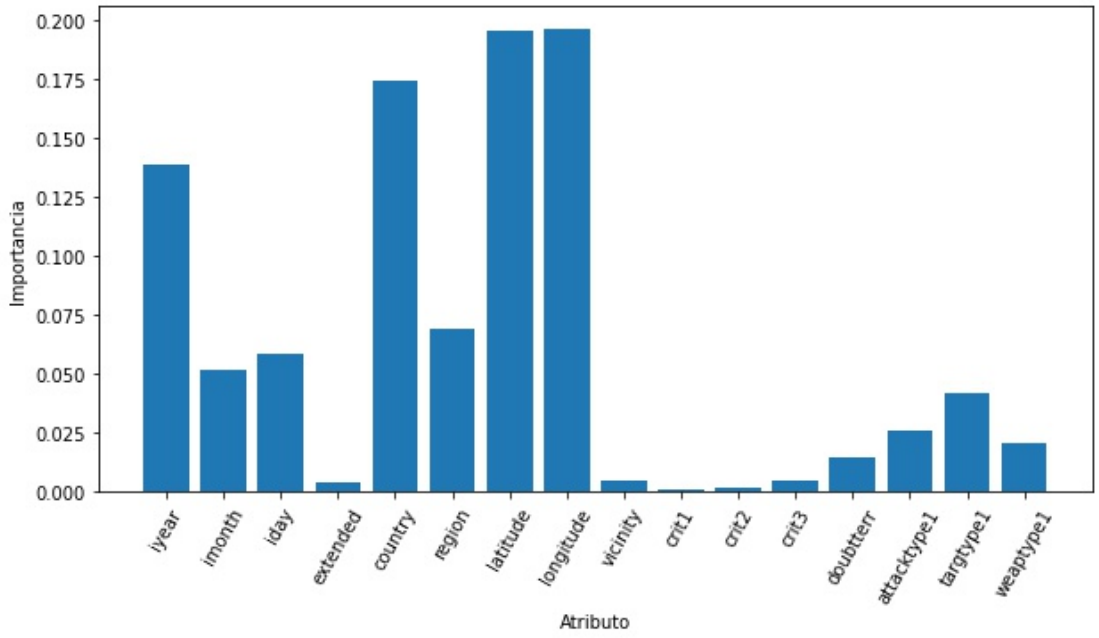


FIGURA 5.3: Importancia de las características del modelo de Bosques aleatorios
Fuente: Elaboracion propia

Luego del entrenamiento y validación del modelo de Bosques aleatorios, se obtiene un accuracy de $84\,\%$. Además, obtiene una precisión de $83\,\%$, un porcentaje de $84\,\%$ para el recall y $82.8\,\%$ para el f1-score. Estos resultados superan al modelo de Árboles de decisión y al modelo de MVS.

5.1.5. Gradient Boosting Machines

Luego del entrenamiento y validación del modelo de GBM, se obtiene un porcentaje final para el accuracy de $71.1\,\%$. Además obtiene una precisión de $60.9\,\%$, un porcentaje de $71.1\,\%$ para el recall y $64.6\,\%$ para el f1-score. Algunos de estos resultados superan el modelo de MVS pero no logra mejorar los resultados obtenidos por los modelos de árboles de decisión y bosques aleatorios.

5.1.6. Comparación de los modelos

En esta sección, se comparan los modelos aprendizaje automático desarrollados anteriormente como Máquinas de vectores de soporte, Árboles de decisión, Bosques aleatorios y Gradient boosting machine. Se usa las métricas de accuracy, precisión, recall y F1-Score, descritas en la sección (2.1.2). La comparación de estos resultados se visualiza en la figura (5.4).

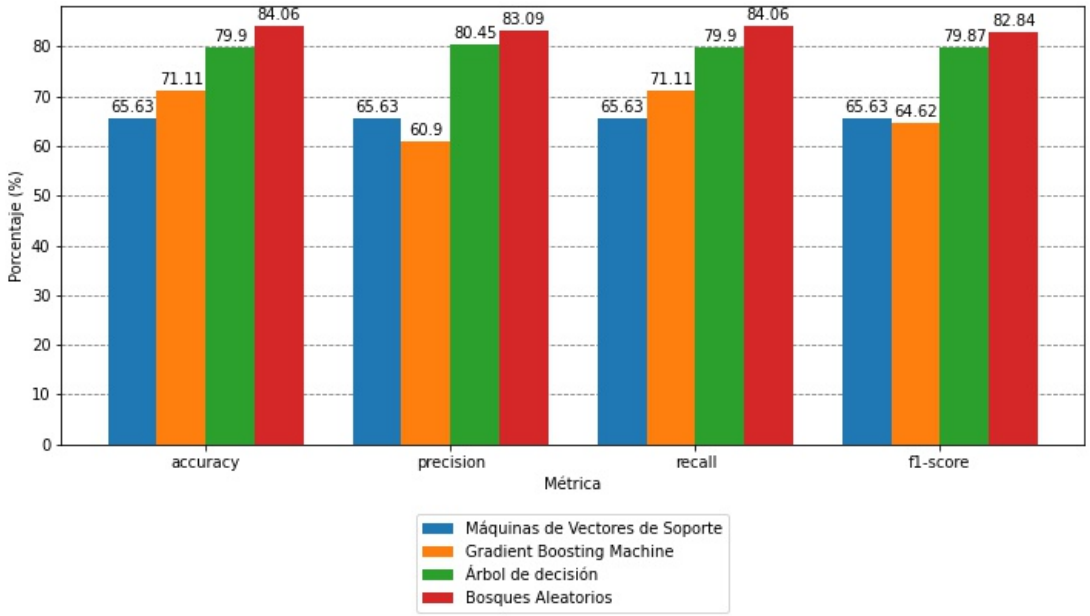


FIGURA 5.4: Comparación de las métricas resultantes
Fuente: Elaboracion propia

Los resultados mostrados en la figura comparativa (5.4) se despliegan en la tabla (5.2).

Algoritmo	Accuracy (%)	Precisión (%)	Recall (%)	F1-Score (%)
MVS	65.63	65.63	65.63	65.63
GBM	71.11	60.90	71.11	64.62
Árbol de decisión	79.9	80.45	79.9	79.87
Bosques aleatorios	84.06	83.09	84.06	82.84

TABLA 5.2: Comparación de los resultados para los distintos modelos

La tabla (5.2) muestra que el modelo de Bosques aleatorios tiene los mejores resultados a comparación de los demás modelos de aprendizaje automático. Esto se puede corroborar visualmente en la figura (5.4). Este modelo obtuvo un accuracy de $84\,\%$ con resultados similares en las demás métricas en una versión acotada del conjunto de datos GTD, probandose con incidentes registrados hasta este año de descarga. Las posteriores versiones del GTD podrían incluir registros de nuevos grupos terroristas.

5.2. Discusiones

En esta sección se evalúa los resultados obtenidos y problemas durante el desarrollo del presente proyecto. Primero se comenta el rendimiento de los modelos de aprendizaje automático comparándolos entre sí. Luego se comenta una breve comparación con los resultados vistos en la sección de Trabajos previos (2.2).

5.2.1. Rendimiento de los modelos

Al observar el rendimiento de los modelos de aprendizaje automático, se encuentra que los resultados obtenidos para los modelos de árboles, tales como árboles de decisión y bosques aleatorios, superan el $79\,\%$ logrando alcanzar un accuracy y recall de $84.06\,\%$ para el caso de Bosques Aleatorios, $83.09\,\%$ y $82.84\,\%$ para la precisión y f1-score respectivamente. Mientras que el modelo de Gradient Boosting Machine dificilmente supera el porcentaje de $71\,\%$ para el accuracy y recall, obteniendo el porcentaje más bajo de la tabla (5.2) para la precisión con $60.9\,\%$ seguido por $64.62\,\%$ para el f1-score. Adicionalmente se observa que el modelo de Máquinas de Vectores de soporte obtiene el puntaje más bajo para el accuracy, precisión, recall y f1-score con $65.63\,\%$ para cada uno.

Los modelos basados en árboles son algoritmos de aprendizaje rápido y el conjunto de datos trabajado logra superar $100k$ registros entonces se puede

decir que es un conjunto de datos enorme, por lo que es aconsejable utilizar este tipo de algoritmos para el conjunto de datos del terrorismo.

Este resultado difiere de los resultados que se muestran en los artículos referenciados en los trabajos previos (2.2.1) y (2.2.3).

5.2.2. Comparación con otros modelos

En el artículo [Tolan und Soliman, 2015] se muestran varios modelos de aprendizaje automático, entre ellos está el algoritmo de árboles C4.5 y las máquinas de vectores de soporte. Este artículo está enfocado en los ataques ocurridos en Egipto y se usan registros del conjunto de datos GTD hasta el año 2013.

Por otro lado, el artículo [Talreja u. a., 2017] muestra los algoritmos de C4.5, bosques aleatorios y máquinas de vectores de soporte, enfocados en los ataques ocurridos en India usando registros del conjunto de datos GTD hasta el año 2015. Según el artículo, siguiendo el método de Análisis de factores mixtos, los atributos óptimos son:

- iyear
- attacktype1
- targtype1
- targsubtype1
- weaptype1
- latitude
- longitude
- natlty
- property
- int_any

- multiple
- crit3

Siguiendo esta recomendación, el cálculo de la importancia de las características para el modelo de C4.5 se muestra en la figura (5.5).

feature	importance
property	0.1787
weaptype1	0.1614
multiple	0.1540
targtype1	0.1482
targsubtype1	0.1467
attacktype1	0.1323
crit3	0.0380
latitude	0.0157
longitude	0.0155
natlty1	0.0115
iyear	0.0000
INT_ANY	-0.0020

FIGURA 5.5: Importancia de las características C4.5
Fuente: Elaboracion propia

En la figura (5.5) se observa que el atributo *iyear* obtiene una importancia de 0 y el atributo *int_any* una importancia negativa de -0.0020. En este caso, no existe problema ya que en el presente proyecto se utilizan diferentes atributos los cuales se mencionan en la tabla (4.1). Cabe mencionar que el accuracy obtenido al replicar el modelo de árboles de decisión con el algoritmo C4.5 (2.1.1), de la librería Chefboost (3.2.3), es menor al porcentaje de $30\,\%$ con el enfoque presentado en la actual investigación.

En los dos artículos mencionados anteriormente los resultados son favorables para el modelo de máquinas de vectores de soporte obteniendo un accuracy de $71.8\,\%$ y $73.2\,\%$ respectivamente. Mientras que en el presente proyecto este modelo obtiene un accuracy de $65.63\,\%$ y es el modelo con menor rendimiento. Esto se debe a que este proyecto se enfoca en el análisis global de

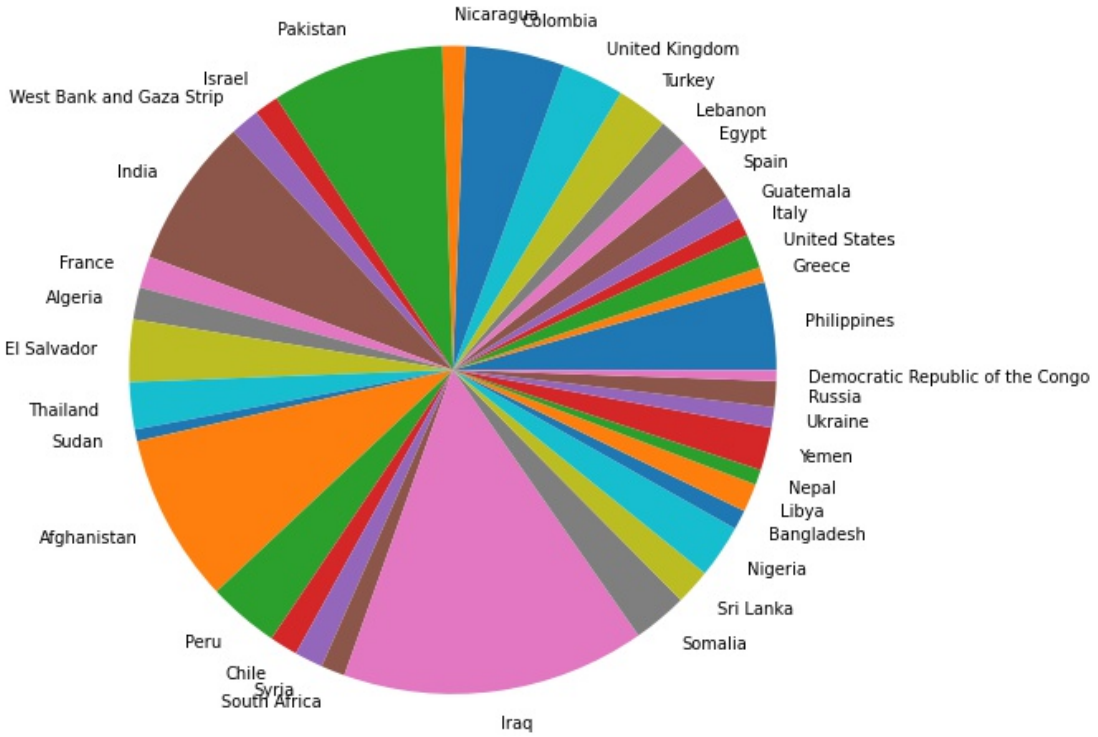


FIGURA 5.6: Distribución de datos por países
Fuente: Elaboracion propia

los ataques terroristas y no se enfoca en una región en específico, a pesar de que existe una gran concentración de datos en la región de la India, como lo muestra la imagen (5.6). Adicionalmente, la cantidad de datos procesados es mayor ya que se utilizan los datos registrados hasta el presente año. Por último, se hace uso de diferentes atributos del conjunto de datos GTD.

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 un factor del terrorismo que pueda ayudar a relizar predicciones de futuras actividades terroristas. Se ha identificado un factor importante de predecir, este es el nombre del grupo que realiza el ataque terrorista.

Se desarrollaron cuatro modelos basados en aprendizaje automático para predecir el nombre del grupo autor de una actividad terrorista. Los modelos son: Máquinas de vectores de Soporte, Árboles de decisión, Bosques aleatorios y Máquina de aumento de gradiente. Los resultados demuestran que el modelo basado en Bosques aleatorios obtiene el mejor rendimiento respecto a los demás, alcanzando un accuracy de más del $84\,\%$.

Las técnicas de aprendizaje automático pueden abarcas diversas áreas de aplicación, en este caso puede ayudar a los organismos encargados a tener mayor comprensión de los ataques terroristas y poder prevenirlos.

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

- El análisis de datos y el aprendizaje automático ofrecen un enfoque prometedor para ayudar a los investigadores a determinar rápidamente el autor más probable de un ataque terrorista.
- A comparación de otros modelos, como Máquinas de vectores de soporte, los modelos de Árboles de decisión no se vieron muy influenciados por *outliers*.
- En comparación con los análisis y resultados de los trabajos previos, el modelo desarrollado de Bosques aleatorios y el preprocesamientos de datos seguido durante el proceso de investigación demostraron ser los más adecuados para la predicción del grupo terrorista usando 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 Árboles es adecuado, y su rendimiento mejora, para procesar macrodatos.

6.2. Trabajo a futuro

Para posibles trabajos a futuro, se propone mejorar el rendimiento del modelo de Bosques aleatorios mediante técnicas para lidear con clases desbalanceadas en el conjunto de datos de entrada, para el modelo presentado en este proyecto no se aplica ninguna técnica de balanceo de datos por lo que se invita al lector a descubrir cómo trabajar con datos desbalanceados.

De igual manera, se propone probar otros métodos, como los clasificadores de conjuntos (ensemble classifiers) y el aprendizaje profundo para mejorar aún más la precisión de la predicción de grupos terroristas. Lo que permitiría a las agencias de investigación reducir posibilidades y actuar rápidamente para atrapar a los verdaderos perpetradores.

Finalmente, los métodos desarrollados pueden ser aplicados en otros campos con el objetivo de ampliar este campo y generar nuevas soluciones.

Bibliografía

[Bonaccorso 2017] BONACCORSO, Giuseppe: *Machine Learning Algorithms*. Packt, 2017

[Breiman 2001] BREIMAN, L.: Random Forests. (2001)

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

[Chen und Guestrin 2016] CHEN, Tianqi ; GUESTRIN, Carlos: XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA : ACM, 2016 (KDD '16), S. 785–794. – URL `http://doi.acm.org/10.1145/2939672.2939785`. – ISBN 978-1-4503-4232-2

[Cortes und Vapnik 1995] CORTES, C. ; VAPNIK, V.: Support-vector networks. (1995)

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

[Guoa u. a. 2019] GUOA, Hongquan ; NGUYENB, Hoang ; VUC, Diep-Anh ; BUI, Xuan-Nam: Forecasting mining capital cost for open-pit mining projects based on artificial neural network approach. (2019)

[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

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

[Max Kuhn 2014] MAX KUHN, Kjell J.: *Applied Predictive Modeling*. Springer, 2014

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

[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

[Quinlan 1993] QUINLAN, J. R.: Programs for Machine Learning. (1993)

[Sachan und Roy 2012] SACHAN, Abhishek ; ROY, Devshri: TGPM: Terrorist Group Prediction Model for Counter Terrorism. (2012)

[Serengil 2019] SERENGIL, Sefik I.: *chefboost*. 2019. – URL `https://github.com/serengil/chefboost`

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

[Talreja u. a. 2017] TALREJA, Disha ; NAGARAJ, Jeevan ; NJ, Varsha ; MAHESH, Kavi: Terrorism Analytics: Learning to Predict the Perpetrator. (2017)

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

[Tolan und Soliman 2015] TOLAN, Ghada M. ; SOLIMAN, Omar S.: An Experimental Study of Classification Algorithms for Terrorism Prediction. (2015)

[Vapnik u. a. 1996] VAPNIK, Vladimir ; GOLOWICH, Steven E. ; SMOLA, Alex: Support Vector Method for Function Approximation, Regression Estimation and Signal Processing. In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*, MIT Press, 1996, S. 281–287

[Xindong Wu 2009] XINDONG WU, Vipin K.: *The Top Ten Algorithms in Data Mining*. CRC Press, 2009