ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

SEGURIDAD PARA REDES IOT USANDO MACHINE LEARNING

ANÁLISIS Y COMPARACIÓN DE DISTINTOS ALGORITMOS DE MACHINE LEARNING PARA LA DETECCIÓN DE INTRUSIÓN EN REDES IOT

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

JOSE JEFFERSON CUEVA PRIETO

jose.cueva01@epn.edu.ec

DIRECTOR: DR. TARQUINO SÁNCHEZ

tarquino.sanchez@epn.edu.ec

DMQ, SEPTIEMBRE 2023

CERTIFICACIONES

Yo, JOSE JEFFERSON CUEVA PRIETO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

JOSE JEFFERSON CUEVA PRIETO

Certifico que el presente trabajo de integración curricular fue desarrollado por JOSE JEFFERSON CUEVA PRIETO, bajo mi supervisión.

DR. TARQUINO FABÍAN SÁNCHEZ ALMEIDA

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JOSE JEFFERSON CUEVA PRIETO

DR. TARQUINO FABÍAN SÁNCHEZ ALMEIDA

DEDICATORIA

Dedico este Trabajo de Integración Curricular especialmente a mi madre Rosa Prieto, a mi padre José Cueva, a mi hermano Cristian Carrión y querida esposa Diana Diaz, quienes me dieron los ánimos y han estado conmigo en los momentos tristes y alegres de mi vida.

AGRADECIMIENTO

Agradezco a Dios, mis padres y hermano, quienes me brindan siempre su apoyo, tanto moral como económico, ayudándome a cumplir mis objetivos. También agradezco al Dr. Tarquino Sánchez por sus conocimientos y persistencia en ayudarme a cumplir con el objetivo de este presente proyecto. Agradezco a la Ing. Ana Rodríguez por sus enseñanzas y su material de apoyo académico, que me ayudaron a entender el contenido desarrollado.

ÍNDICE DE CONTENIDO

CERTIFICACIONES..... I

DECLARACIÓN DE AUTORÍA.....II

DEDICATORIA.....III

AGRADECIMIENTO IV

ÍNDICE DE CONTENIDO..... V

RESUMEN VII

ABSTRACT VIII

1 INTRODUCCIÓN

1.1 Objetivo general 4

1.2 Objetivos específicos 4

1.3 Alcance 5

1.4 Descripcion metodologica aplicada en el proyecto 6

1.5 Marco teorico 7

1.5.1 Internet de las cosas 7

1.5.2 Preprocesamiento de datos 8

1.5.3 Etapas de la preparación de los datos 8

1.5.4 Conjunto de Datos 9

1.5.5 Machine Learning 9

1.5.6 Algoritmos para clasificación 9

1.5.7 Métricas de desempeño 12

1.6 Analisis de requerimientos 13

1.6.1 Requerimientos de hardware 14

1.6.2 Requerimientos de software 15

2 METODOLOGIA 15

2.1 Analisis de datos 16

2.1.1 Fuente de los datos 16

2.1.2 Descripcion del conjunto de datos 20

2.1.3 Preprocesamiento del conjunto de datos 22

2.1.4 Balanceo del conjunto de datos 23

2.1.5 Estructuración del conjunto de datos para entrenamiento y prueba.. 25

2.2 Análisis de datos utilizando algoritmos de Machine Learning 30
2.3 Análisis comparativo entre los modelos de Machine Learning 30

3 RESULTADOS,CONCLUSIONES Y RECOMENDACIONES 51
3.1 Resultados 51
3.2 Conclusiones 56
3.3 Recomendaciones 57
4 REFERENCIAS BIBLIOGRÁFICAS 57

RESUMEN

Los dispositivos IoT adquieren una gran cantidad de información personal, por lo cual dicha información debe resguardarse de posibles amenazas. La seguridad de los dispositivos IoT puede comprometerse debido a que poseen un bajo poder computacional. Esta seguridad puede lograrse procesando la información a través de algoritmos de Machine Learning. El análisis comprende desde el preprocesamiento de la información y prosigue con el entrenamiento y la evaluación, con el propósito de cumplir con un modelo de Machine Learning capaz de identificar tramas buenas y maliciosas sin sacrificar los recursos computacionales de los dispositivos IoT.

En este proyecto, se utilizará un conjunto de datos que contiene información de las tramas de la red. Estas tramas, en su mayoría, son buenas y cierta cantidad son maliciosas. Se depurarán, equilibrarán, normalizarán y reducirá la dimensionalidad de la información, con el fin de obtener un conjunto de datos eficiente para entrenar los algoritmos de Machine Learning.

Los algoritmos de Machine Learning ya entrenados serán evaluados con métricas de desempeño y consumo de recursos. Estas métricas permitirán determinar cuál modelo de Machine Learning resulta factible para sistemas IoT.

PALABRAS CLAVE: Machine Learning, clasificación, métricas, conjunto de datos, entrenamiento, evaluación.

ABSTRACT

IoT devices acquire a large amount of personal information, so this information must be protected from potential threats. The security of IoT devices can be compromised because of their low computational power. This security can be achieved by processing the information through Machine Learning algorithms. The analysis starts from the preprocessing of the information, followed by training and evaluation to meet a Machine Learning model capable of identifying good and malicious frames without sacrificing the computational resources of IoT devices.

In this project, a dataset containing information from network frames will be used. These frames, for the most part, are good and a certain amount are malicious. The information will be debugged, balanced, normalized and dimensionality reduced to obtain an efficient dataset to train Machine Learning algorithms.

The trained Machine Learning models will be evaluated with performance and resource consumption metrics. These metrics will allow determining which Machine Learning model is feasible for IoT systems.

KEYWORDS: Machine Learning, classification, metrics, data set, training, evaluation.

1 INTRODUCCIÓN

Los dispositivos IoT proveen servicios para automatizar o controlar ciertos campos de la industria como: salud, entretenimiento, finanzas, educación, agrícola, etc.… [1], facilitando las conexiones de objetos físicos con internet e insertándonos en un nuevo paradigma de la red.

El número de dispositivos IoT está en un crecimiento exponencial. En el 2019 existía 7.6 billones, y se prevé que para 2030 existan 24.1 billones [2]. La cantidad de información obtenida de la capa de percepción de los dispositivos IoT es inmensa. Este crecimiento viene acompañado de diferentes desafíos en términos de seguridad, como ataques del tipo DoS, Backdoors, Exploits, Fuzzers, etc.

Los dispositivos IoT se caracterizan por poseer recursos limitados en memoria y capacidad de procesar información. Esto contribuye a la existencia de anomalías en las redes y a que su seguridad sea vulnerada.

El análisis y la evaluación pueden ayudar a identificar cuando una red de dispositivos IoT está siendo comprometida. Cada algoritmo de Machine Learning requiere de ciertos requisitos para cumplir con su objetivo. Estos requisitos pueden generar un consumo de recursos no apropiados para redes IoT. Las métricas de estos modelos de Machine Learning serán el pilar para identificar si es sostenible el modelo, cumpliendo con la mejor eficiencia sin perder su función principal de identificar tramas buenas o maliciosas.

El conjunto de datos University of New South Wales - Network Based 15 (UNSW-NB15) [3] contiene recopilaciones de trafico de red. Fue creado y diseñado para evaluar sistemas de seguridad como los modelos de Machine Learning. El tráfico de red está compuesto por tramas buenas y maliciosas, lo que le permite utilizarle en investigaciones para desarrollar soluciones de seguridad. El conjunto de datos entra a una sección de preprocesamiento para identificar las etiquetas relevantes que permitirán el entrenamiento y posterior evaluación de los algoritmos de Machine Learning.

1.1 Antecedentes

Los ciberataques a dispositivos IoT van en aumento debido a que cada vez son más las personas y las organizaciones que compran dispositivos "inteligentes". No todos los usuarios piensan que vale la pena protegerlos; los ciberdelincuentes aprovechan estas oportunidades con el fin de una explotación financiera de estos dispositivos. Por ejemplo, utilizan la red de dispositivos inteligentes infectados para llevar a consigo ataques de denegación de servicio (DoS).

La seguridad y la protección de los recursos son preocupaciones primordiales. Se necesitan soluciones de seguridad y privacidad diseñadas apropiadamente, considerando la capacidad limitada de los dispositivos IoT.

Este trabajo se centra en el entrenamiento y evaluación de algunos algoritmos de Machine Learning, como Regresión Lineal, Redes Neurales Convolucionales, k-Nearest Neighbors (KNN), Naive Bayes. La evaluación se basa en el uso de métricas de medición como precisión, exactitud, matriz de confusión, tiempo de ejecución, el consumo de memoria de acceso aleatorio (RAM) y unidad central de procesamiento (CPU). El objetivo es encontrar el algoritmo de menor carga computacional que cumpla eficientemente con su propósito primordial de identificar trazas maliciosas y normales. Por lo tanto, se busca lograr una clasificación eficiente de trazas buenas y maliciosas para la implementación en las redes IoT, teniendo en cuenta los requerimientos de seguridad tales como disponibilidad, integridad y comunicación segura.

1.2 Objetivo general

OG: Analizar y comparar algunos algoritmos de Machine Learning a través de sus métricas con el fin determinar cuál es el más adecuado para sistemas IoT.

1.3 Objetivos específicos

OE1: Integrar y ordenar el conjunto de datos a través de la búsqueda datos faltantes, atípicos, transformación de datos cualitativos para así lograr un conjunto de datos homogéneo.

OE2: Obtener un conjunto de datos reorganizado a través de técnicas de balanceo y normalización para mejorar la calidad y utilidad en el análisis subsiguiente.

OE3: Evaluar el desempeño de cada algoritmo de Machine Learning utilizando análisis de componentes principales.

OE4: Clasificar las trazas buenas y malas mediante el entrenamiento y prueba de los algoritmos de Machine Learning, logrando así identificar valores de las métricas de evaluación y medición de recursos utilizados.

1.4 Alcance

El presente estudio consiste en inspeccionar, reorganizar y categorizar el conjunto de datos obtenido de la Universidad de Nueva Gales del Sur, el mismo que se puede obtener del enlace https://research.unsw.edu.au/projects/unsw-nb15-dataset.

El conjunto de datos fue recolectado y creado por Dr. Nour Moustafa en el año 2015 en un periodo de 2 días con un tamaño total de 100GB utilizando la herramienta IXIA PerfectStorm que genera pruebas de concepto del mundo real con datos normales y comportamientos de ataques. El conjunto de datos contiene 2'540.044 registros con 49 características [4]. Este conjunto de datos contiene tráfico normal y malicioso como: Generic, Exploits, Fuzzers, Dos, Reconnaissance, Analysis, Backdoors, Shellcode y Worm. El generador de tráfico IXIA PerfectStorm fue configurado con 3 servidores: 1 servidor para generar tráfico normal y 2 para generar tráfico malicioso como se evidencia en el esquema de red presentado en la Figura 1.

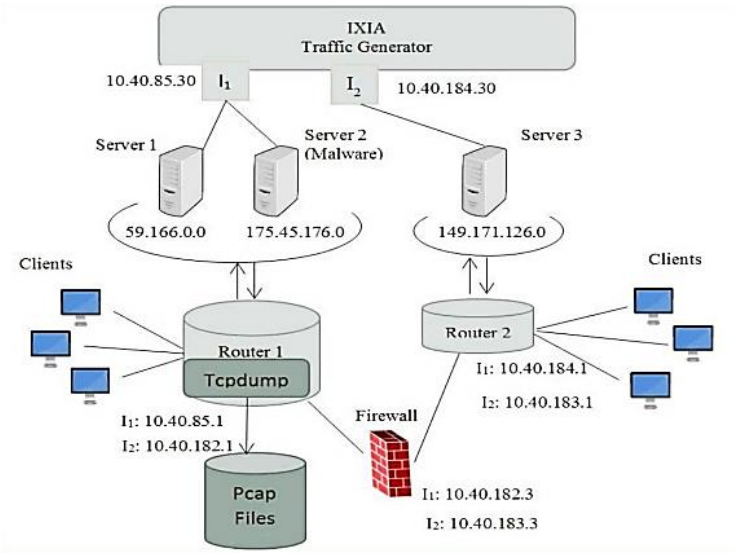

Figura 1. Esquema de red simulado [4]

Posteriormente, el conjunto de datos, una vez tratado e identificados los ataques más frecuentes se debe balancear y normalizar consecuentemente realizar un análisis de componentes principales para disminuir la cantidad de etiquetas originales del conjunto de datos y así podremos aplicar en el entrenamiento y evaluación de algunos tipos de algoritmos como: Regresión Lineal, Redes Neuronales Convolucionales, k-Nearest Neighbors KNN, Naive Bayes. Una vez evaluados los algoritmos, se realizará una comparación de métricas para buscar al algoritmo de menor carga computacional y que cumpla con su objetivo primordial de identificar trazas "maliciosas" y "normales".

Los algoritmos de Machine Learning de aprendizaje supervisado serán evaluados a través del uso de métricas como: la precisión, la exactitud, consumo de recursos computacionales. Se obtendrá la matriz de confusión tratando de mantener los menores valores de falsos positivos y falsos negativos, además de obtener los menores tiempos y

esfuerzos [1]. Al obtener un algoritmo eficiente se podrá aplicar a sistemas personales o empresariales.

1.5 Descripción metodológica aplicada en el proyecto

Cross-Industry Standard Process for Data Mining (CRISP-DM) es una metodología aplicada en distintos aspectos de la minería de datos se enfocada a la estructuración, organización de los procesos para el estudio de minería de los datos [5]. CRISP-DM está conformada por 6 fases presentadas en la Figura 2: 1) Comprensión general del proyecto, 2) Comprensión del conjunto de datos, 3) Preparación del conjunto de datos, 4) Modelado, 5) Evaluación y 6) Despliegue del proyecto. Cada fase incluye tareas específicas y entregables que ayudan a garantizar un proyecto de minería de datos exhaustivo y completo [5].

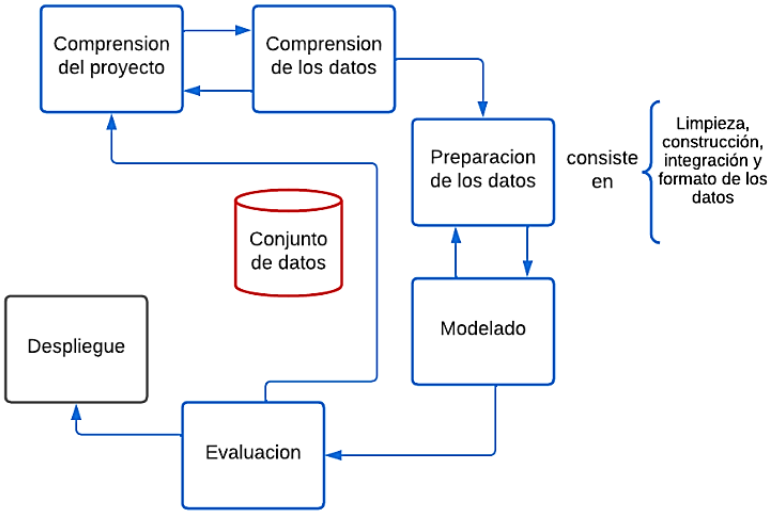

Figura 2. Diagrama de flujo CRISP-DM [5]

Para este proyecto de análisis y comparación de modelos de Machine Learning se enfocará en las primeras 5 etapas. Esta secuencia no es necesariamente estricta ya que en algunos casos se puede avanzar o retroceder hasta lograr cumplir con los objetivos [5].

1. **Comprensión del proyecto:** Dado que puede haber una variedad de problemas por resolver en el mundo de las redes IoT, en este punto es importante definir los problemas que surgen en el proyecto. La necesidad de encontrar algoritmos de Machine Learning eficaces que puedan satisfacer, junto a las redes de IoT que cumplan con la tarea principal de proteger la seguridad. Las redes IoT están formadas por dispositivos que tienen baja capacidad de procesamiento y su seguridad puede verse vulnerada.

2. **Comprensión de datos:** En esta etapa esta especificada la recopilación de datos o búsqueda de conjunto de datos con buena calidad de información y que puedan ser tratados en el contexto del Machine Learning.

3. **Preparación de los datos:** Con la construcción, limpieza, integración y formato de los tipos de datos se cumplirá con esta etapa, que requiere de cierto tiempo de análisis para tener depurado y organizado para la siguiente etapa del modelado.

4. **Modelado:** Al tener estructurado en un nuevo conjunto de datos, se procede aplicar análisis de componentes principales con la finalidad de reducir la dimensionalidad del conjunto de datos original eh implementar en algunos algoritmos de Machine Learning esto contribuirá a obtener resultados de los objetivos propuestos. Esta etapa es de múltiples repeticiones ya que se debe ajustar el modelo hasta obtener una coherencia en la muestra de las métricas.

5. **Evaluación:** Hasta esta etapa se tendrá la mayoría del proyecto. En esta etapa de evaluación y comparación de las métricas de los modelos de Machine Learning utilizados se tendrá de forma efectiva que algoritmos o algoritmo alguno sea eficiente en identificar las trazas buenas y maliciosas cumpliendo la menor carga computacional.

6. **Despliegue:** Esta etapa final, comprende en implementar para un entorno sea físico o simulado con fines educativos o comerciales. Para esta etapa está claramente descrita en el proyecto de titulación “Implementación de un sistema de detección de amenazas utilizando redes definidas por software” que corresponde al segundo componente.

1.6 MARCO TEORICO

1.6.1 Internet de las cosas

Internet de las Cosas(IoT), se define como a la interconexión de objetos tangibles como: aparatos individuales, automóviles, casas y edificios, que conectan a internet y que le es posible intercambiar y compartir su información. Estos poseen sensores y tecnologías integradas para capturar datos y transmitirlos a una red. En IoT se tiene entornos inteligentes, sistemas que pueden ser monitoreados y controlados de forma remota, esto permite realizar acciones en función de la recopilación de varios datos. Algunos ejemplos se tienen como los hogares inteligentes, automóviles y sistemas de la industria [6].

La arquitectura IoT está formada por 5 capas:

Capa de percepción: En un entorno físico tiene la función de recopilación de datos. De forma general, para esta capa las comunicaciones son de corto alcance [7].

Capa de dispositivo: Esta capa se encarga de la conectividad física y lógica entre el dispositivo IoT y la red. Puede incluir componentes como módems, controladores, etc [7].

Capa de conectividad: Esta capa se encarga de la comunicación entre IoT, la nube y el centro de datos. Puede incluir tecnologías como Wi-Fi, Bluetooth, Zigbee, etc [7].

Capa de plataforma: Esta capa se encarga de la organización y elaboración de datos que se recolectan de los dispositivos IoT [7].

Capa de aplicación: Proporciona una interfaz para interactuar con los datos recolectados por los dispositivos IoT [7].

1.6.2 Preprocesamiento de datos

Preprocesamiento de datos conlleva a un fin de adecuación para el modelado y análisis, esto se logra con el respectivo proceso de limpieza, transformación, integración y normalización de los datos. Los pasos necesarios de forma general implican la eliminación de datos irrelevantes o faltantes, revisión de tipos de datos y normalización de los datos obteniendo un formato adecuado para los algoritmos de Machine Learning. Además, con esto se logra reducir recursos y tiempo para entrenar e implementar datos de calidad en los modelos [8].

1.6.3 Etapas de la preparación de los datos [9]

Integración de datos: Obtener o crear alguna fuente con información relevante para luego integrar los datos desde las diferentes tablas y obtener un conjunto homogéneo de información.

Limpieza de datos: Consiste en la resolución de conflictos entre tipos de datos eliminación de valores atípicos, observación de presencia de ruido.

Transformación de datos: En los conjuntos de datos puede existir variedad de tipos de datos , estos deben ser transformados a una forma apropiada para la adaptación a los requerimientos base de algún algoritmo de Machine Learning. Estos conjuntos de datos son tratados con sumarizacion y operaciones de agregación.

Reducción de datos: Las características más pertinentes, los ejemplos representativos y la categorización de datos continuos son algunas estrategias utilizadas para reducir la cantidad de datos.

1.6.4 Conjunto de Datos

Se conoce como conjunto de datos a una colección de datos estructurados y organizados para aplicar el análisis y modelado. Pueden tener una gran gama de tipos de datos que incluye: datos de texto, numéricos, imágenes, audio y video. Estos datos son provenientes de variedades de fuentes, experimentos, encuestas, transacciones o simulaciones. El tamaño y la calidad del conjunto de datos afecta el rendimiento del modelo por esto se debe preprocesar cuidadosamente los datos antes de ser usados en alguno de los modelos del aprendizaje automático [10].

1.6.5 Machine Learning

Es un caso de inteligencia artificial que ayuda a diseñar aplicaciones con enfoque en la predicción de salidas sin programar condiciones explícitamente con resultados fijos. Los algoritmos de aprendizaje automático usan técnicas estadísticas para encontrar patrones en los datos, que luego se pueden usar para hacer predicciones o tomar medidas. Esto permite que la computadora mejore su rendimiento en una tarea específica con el tiempo [11].

Muchos factores afectan el éxito del aprendizaje automático en una tarea determinada. La fuente, estructuración, representación de los datos influyen en su calidad. Una fuente confiable, precisa ayuda a una correcta utilización de los algoritmos de Machine Learning además de la reducción de dimensionalidad de conjunto de datos permite que los algoritmos de aprendizaje operen de manera más rápida y efectiva [12].

1.6.6 Algoritmos para clasificación

Regresión Logística

El modelo de regresión logística (RL) es un modelo estadístico donde se evalúa relaciones entre una variable cuantitativa dependiente y variables cuantitativas independientes que puede ser más de dos variables [13].

En la regresión logística se desea probar a un conjunto de predictores de una variable dependiente clasificando a partir de variables independientes si sucede un evento o no sucede ese evento [13].

Este modelo se basa en clasificar si la salida pertenece o no a una clase, para esto se utiliza la función logística conocida comúnmente como función sigmoidea como la presentada en la ecuación 1.1 y se representa en una forma de S.

$$RL(Z) = \frac{1}{1+e^{-Z}} \tag{1.1}$$

La ecuación 1.2 representa a Z, esta es la entrada del modelo que se origina de las combinaciones lineales de las características con su respectivo peso influyendo más en algunos casos y menos en otros.

$$Z = w^T x = w_0 + w_1 x_1 + \cdots + w_m x_m \quad (1.2)$$

Redes Neurales Convolucionales

La red neuronal convolucional (CNN) es un enfoque de aprendizaje profundo que se usa ampliamente para resolver problemas complejos. Supera las limitaciones de la máquina tradicional. Las técnicas de aprendizaje profundo superan el problema de la selección de características al no requerir características preseleccionadas [14].

El modelo de aprendizaje profundo consiste en una colección de capas de procesamiento que pueden aprender varias características de los datos a través de múltiples niveles de abstracción. Múltiples niveles permiten que la red aprenda características distintas. El aprendizaje profundo surgió como un enfoque para lograr resultados prometedores en diversas aplicaciones. Hay diferentes arquitecturas de aprendizaje profundo, como redes de creencias profundas, redes neuronales de convolución, redes neuronales recurrentes, etc [14].

En la ecuación 1.3, se tiene a la ecuación del modelo general de una red neuronal de una sola capa, W representa al peso que relaciona o une con fuerza a las neuronas, X es un vector entrada y Y la salida [14].

$$F(X, W) = Y \quad (1.3)$$

Algunas de las capas comunes en una CNN son: [15]

La capa de entrada, que recibe valores de la entrada (por ejemplo, una imagen) y los convierte en una representación matemática que puede ser procesada por la red.

Las capas de convolución, que realizan una operación matemática llamada "convolución" en los datos de entrada. Esta operación permite a la red extraer características de los datos, como patrones y detalles visuales.

Las capas de agrupación (pooling), que reducen la dimensionalidad de los datos mediante la agrupación de los valores en una región determinada. Esto permite a la red mantener solo las características más importantes y eliminar el ruido.

Las capas completamente conectadas, que realizan una operación matemática llamada "producto punto" entre los datos de entrada y unos pasos aprendidos por la red.

Esto permite a la red hacer predicciones o tomar decisiones basadas en los datos de entrada.

La capa de salida está al final de la red, es una predicción o puede ser una decisión basada en los datos de entrada.

k-Nearest Neighbors KNN

KNN se refiere a un método estadístico no paramétrico en el aprendizaje supervisado que generalmente usa la distancia euclidiana. La distancia euclidiana en KNN determina el valor promedio del nodo desconocido que es k vecinos más cercanos como se visualiza en la Figura 3. Por ejemplo, si se pierde algún nodo, se puede anticipar a partir del valor promedio del vecino más cercano. Este valor no es exacto, pero ayuda a identificar el posible nodo faltante. El método KNN se utiliza en la detección de intrusos, detecciones de malware y detección de anomalías en IoT [16].

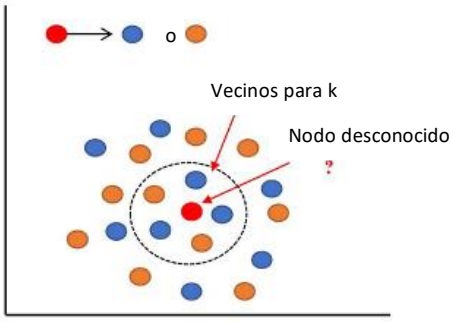

Figura 3: Ilustración del aprendizaje KNN [16]

Para utilizar el algoritmo KNN, se debe especificar el valor de k, que se considerarán al determinar la clase de una muestra. A menor valor de k, la clasificación será más sensible a ruido en los datos. Por otro lado, un valor más alto de k hará que la clasificación sea menos sensible al ruido, pero también puede aumentar la probabilidad de que la clase asignada sea menos precisa [16].

Naive Bayes

Este clasificador se basa en el teorema de Bayes, funciona con probabilidades condicionadas es decir que suceda un nuevo evento conociendo que se ha dado algo previamente. De esta manera se calcula una nueva probabilidad condicionada a través de la ecuación 1.4 [17].

$$X(h/E) = \frac{x(E|h)x(h)}{X(E)} \quad (1.4)$$

Donde:

- X(h) probabilidad de hipótesis.
- X(E) probabilidad de observación del conjunto de entrenamiento (E).
- X(E/h) probabilidad de observación de entrenamiento del conjunto E donde se visualiza h.
- X(h/E) probabilidad de h después de analizar el conjunto de entrenamiento E.

Métricas de desempeño

En el análisis de los algoritmos de Machine Learning se utiliza tasas de aciertos y tasa de fallos [18]. La matriz de confusión descrita en la Tabla 1, sirve para comparar las etiquetas reales y las pronosticadas, en este caso de estudio identificaría dos clases de tráfico normal y el y malicioso [18].

Tabla 1. Matriz confusión [18]

		Actual	
		Positivo	Negativo
Predicción	Positivo	TP	FP
	Negativo	FN	TN

TP (verdadero positivo) y TN (verdadero negativo) son términos que demuestran que las predicciones son correctas y que FN (Falso negativo) y FP (Falso Positivo) demuestran que las predicciones son incorrectas. Estos términos son importantes para las métricas [18].

Accuracy, medida que refleja el éxito general, es la tasa de aciertos clasificando instancias normales y maliciosas y se calcula con la ecuación 1.5 [18].

$$Exactitud = \frac{TN + TP}{TP + FP + TN + FN} \quad (1.5)$$

La precisión, nos indicara que tan lejos estamos del valor real, con esta métrica medimos la calidad del modelo en la tarea de clasificar correctamente las instancias maliciosas. Se calcula con la ecuación 1.6 [18].

$$Precision = \frac{TP}{TP + FP} \quad (1.6)$$

Análisis de componentes principales (PCA)

La finalidad de los componentes principales es la reducción a N variables de un espacio de observación con M variables originales, donde N<M. La reducción de la dimensión se logra a base de las combinaciones lineales de M. Las nuevas variables M extraen las características más representativas del espacio de observación [19].

1.7 Análisis de requerimientos

En este proyecto está constituido por dos componentes independientes como se observa en la Figura 4.

El primer componente, es encargado del análisis y comparación de algunos algoritmos de Machine Learning a través de sus métricas de evaluación, dichos algoritmos serán entrenados y puestos a prueba por un conjunto de datos de entrenamiento y prueba previamente preprocesados y balanceados.

El segundo componente, consiste en implementar un sistema de detección de amenazas usando el algoritmo de red neural convolucional sobre las redes definidas por software con esto permitirá detectar cuando la red IoT está siendo comprometida.

En ambos casos se requiere de infraestructura física como un computador y de un software para desarrollar en el lenguaje de programación de Python.

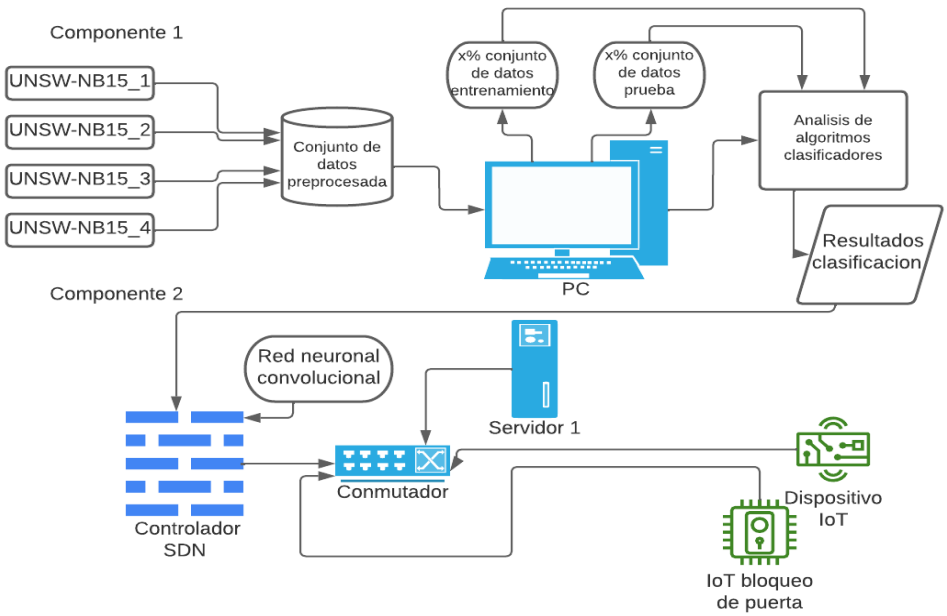

Figura 4. Componentes del proyecto seguridad para redes IoT. Fuente: El Autor

1.7.1 Requerimientos de hardware

Los equipos que se requieren son específicamente para almacenamiento de datos como: discos duros, Universal Serial Bus (USB) con valores superiores de almacenamiento a 1GB, mientras que para la parte de preprocesamiento y análisis de los diferentes algoritmos se requiere de equipos portátiles o de escritorio que tengan la capacidad suficiente para iniciar los programas respectivos similares como la Tabla 2.

Tabla 2 : Características del computador utilizado. Fuente: El Autor

Características de Google Colab	
Procesador	Intel(R) Xeon(R) CPU @ 2.20GHz
Tamaño del disco duro	107.7 GB
Memoria RAM	12.7 GB
Sistema operativo	Ubuntu
Características de la laptop	
Tipo de computador	Portátil
Sistema operativo	Windows 10 Pro
Procesador	I5 de tercera generación
Memoria RAM	8 GB
Modelo	Dell Inspiron 14z
Tamaño del disco duro	120 GB

1.7.2 Requerimientos de Software

Una parte esencial del proyecto son los programas que se involucran en el preprocesamiento y evaluación de los algoritmos de machine Learning. Para el preprocesamiento se requiere de algunos como *Excel, Rapid Miner, Python, Weka, SPSS* estos programas nos muestran características generales del conjunto de datos, que valores tienen más presencia, la existencia de valores nulos, filas y columnas vacías de forma que podemos identificar fácilmente la cantidad de información que se posee y su respectiva limpieza y reorganización.

Para el análisis se utilizará el lenguaje de programación de *Python* sobre el entorno desarrollo online de *Google Colab*.

2 METODOLOGIA

La metodología abordada en este proyecto se basa en los objetivos de la Tabla 3, los recursos y estructuración general del proyecto, la comprensión que se quiere para

abordar el uso de algoritmos de machine Learning para redes IoT, la selección de datos necesarios, el preprocesamiento de los datos, la selección de algoritmos, entrenamiento y evaluación de los algoritmos y finalmente la comparación de algoritmos.

Tabla 3. Etapas metodológicas con sus objetivos y resultados. Fuente: El Autor

Fases	**Objetivo**	**Resultado**
Comprensión del proyecto	OG	Tener entendido la forma a desarrollar el proyecto tomando en cuenta la capacidad del procesamiento de dispositivos IoT que es limitada.
Comprensión de datos	OE1	Conjunto de datos integro, sin valores atípicos y con formato de datos adecuado
Preparación de los datos	OE2:	Conjunto de datos estructurado en entrenamiento y prueba.
Modelado	OE4:	A partir de múltiples Iteraciones tener un criterio sobre el desempeño de cada modelo.
Evaluación	OE3:	Seleccionar uno o varios algoritmos de machine Learning que se visualice métricas eficientes.

2.1 Análisis de datos

2.1.1 Fuente de los datos

La fuente del conjunto de datos UNSW-NB15, fue extraída de paquetes de red sin procesar, creados por la herramienta IXIA PerfectStorm en el Cyber Range Lab de Australian Centre for Cyber Security (ACCS). Esta herramienta genera una combinación de actividades normales y comportamientos maliciosos de ataque similares a entornos reales [4].

La herramienta IXIA PerfectStorm puede generar tráfico desde 4 Gb/s hasta 80 Gb/s. Fue configurado por tres Servidores virtuales como se puede observar en la Figura 1. El S*ervidor 1* con dirección IP: 59.166.0.0 y el *Servidor 3* con dirección IP: 149.171.126.0, se utilizaron para generar tráfico de red normal y el *Servidor 2* con dirección IP: 175.45.176.0 para generar tráfico malicioso [4].

Estos servidores y el generador de tráfico estaban interconectados por dos interfaces virtuales: la Interfaz 1 con IP 10.40.85.30 y la Interfaz 2 con IP 10.40.184.30, con la finalidad de vincular la red pública y privada [4].

Los servidores se unieron a los hosts mediante dos enrutadores, cada uno de los cuales tenía dos interfaces para vincularse con clientes y capturar paquetes mientras se ejecuta la simulación. Las direcciones IP del Router 1 son IP1: 10.40.85.1 y IP2: 10.40.182.1 y las del *Router 2* son IP1: 10.40.184.1 y IP2: 10.40.183.1. Se conectó un cortafuegos a los dos enrutadores para filtrar el tráfico de emisión que muestra los eventos de seguridad en él [4].

La recopilación de paquetes sin procesar tuvo lugar durante 16 horas el 22 de enero de 2015 y otro segmento durante 15 horas el 17 de febrero de 2015 [4].

La herramienta Tcpdump utilizada para analizar tráfico de la red, capturo 100GB, se instaló en el *Router 1*, se extrajo los archivos pcap mientras la simulación se estaba ejecutando. Se transmitieron aproximadamente 280 Kbyte/s entre los nodos de la red durante los periodos de simulación [4].

El esquema para crear los conjuntos de datos en formato.csv se presenta en la Figura 5, donde los archivos .pcap son procesados por dos herramientas Argus y Bro-IDS obteniendo 2 conjuntos de datos con etiquetas que identifican características de las tramas estas etiquetas son comparadas entre sí para obtener un conjunto de datos unificado. En ciertas instancias se presentan características únicas de ataques sigilosos por lo que se utilizaron algunos algoritmos para agregar etiquetas calculadas y dando, así como resultado un conjunto de datos más entendible y manejable.

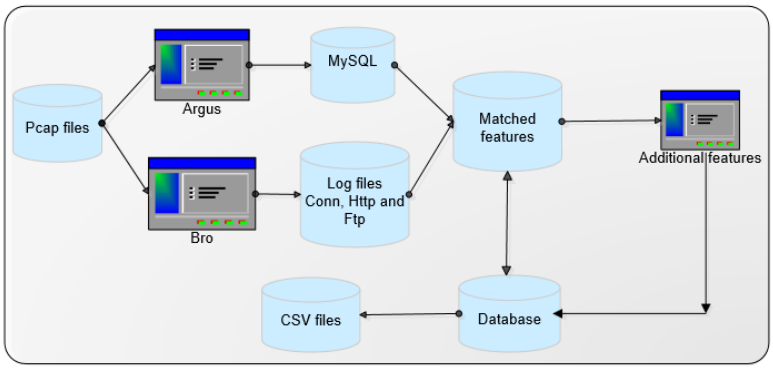

Figura 5. Esquema para la asignación de las etiquetas en el conjunto de datos UNSW-NB15 [4].

A la fecha de 14 de febrero 2023 se realizó una pequeña demostración de cómo obtener un conjunto de datos en formato .csv con etiquetas estructuradas a partir de paquetes de datos sin procesar como los archivos .pcap.

El inicio de esta demostración se dio desde la utilización de dos herramientas:

- Argus, herramienta de auditoría de código abierto desarrollada por Carter Bullard, se utiliza para analizar paquetes de red sin procesar y crear conjuntos de datos a partir de estos paquetes. [4]
- Zeek, antes llamada Bro-IDS es una herramienta de código abierto que analiza de forma eficiente el tráfico de la red con patrones anómalos y crea archivos de registro .log de algunos servicios como: CONN, DNS, FTP y HTTP. [20]. Estas herramientas utilizadas en el entorno de Ubuntu permiten seleccionar las etiquetas que pueden ser consideradas como información relevante para identificar tráfico normal o malicioso.

Para la instalación de Argus es sencilla, se requiere la ejecución de la *Instrucción 1*, en la terminal de *Ubuntu.* Ahora, como se puede ver en la Figura 5, para obtener un archivo legible .argus se ejecuta la *Instrucción 2* y finalmente para obtener un conjunto de datos en formato .csv se ejecuta la *Instrucción 3* donde se especifica al archivo .argus, las etiquetas seleccionadas y el nombre a guardar en archivo .csv obteniendo así un conjunto de datos con 30 etiquetas como en la Tabla 4.

```
root@debianlock:/home/debian/Descargas# argus -r 1.pcap -w 1.argus
```

Instrucción 1. Instrucciones para instalación de Argus

```
apt-get install argus-client y apt-get install argus-server
```

Instrucción 2. Para la conversión de archivo .pcap a. argus

```
root@debianlock:/home/debian/Descargas# ra -r 1.argus -s  stime, ltime, trans, sport,
dport, proto, spkts, dpkts, dmeansz, sloss, dloss, dttl, sload, dload,  sintpkt, dintpkt,
sjit, djit, swin, stcpb, ackdat, smeansz, sbytes, dbytes, rate, dtcpb, dwin, tcprtt,
synack, sttl, > argusv2.csv.
```

Instrucción 3. Selección de etiquetas a extraer del archivo .argus

Tabla 4. Muestra del conjunto de datos .csv generado por Argus. Fuente: El Autor

StartTime	LastTime	Sport	Proto	SrcPkts	DstPkts	SrcBytes	Rate	sTtl	dTtl	SrcLoad	DstLoad
0:49:33	0:49:33	0	0	0	0	0	0.000000	0	0	0	0.000000
19:23:28	19:23:28	22592	tcp	14	38	734	78.473373	62	252	8395.11*	503571.*
19:23:28	19:23:29	62762	tcp	8	16	364	14.170161	62	252	1572.27*	60929.2*
19:23:28	19:23:29	45235	tcp	12	12	628	13.677108	62	252	2740.17*	3358.62*
19:23:28	19:23:28	33159	tcp	4	6	172	74.087486	254	252	14158.9*	8495.36*
19:23:28	19:23:29	43722	tcp	10	6	534	39.417980	254	252	10112.0*	4709.13*
19:23:29	19:23:29	62994	tcp	10	6	534	33.373825	254	252	8561.49*	3987.05*
19:23:29	19:23:30	62708	tcp	10	8	534	26.683033	254	252	6039.78*	3892.58*
19:23:29	19:24:09	arp	2	56	92	0.075630	9.277307	0	0	39666.520000	39666.576000
19:23:30	19:23:31	63945	tcp	10	8	534	32.593025	254	252	7377.52*	4754.74*
19:23:30	19:23:32	40755	tcp	62	28	56329	42.520966	62	252	211825.*	8152.55*
19:23:31	19:23:31	63888	tcp	10	6	534	57.985134	254	252	14875.1*	6927.29*
19:23:31	19:23:31	60835	tcp	10	8	534	31.313030	254	252	7087.79*	4568.01*
19:23:31	19:23:32	52515	tcp	12	6	4142	55.764584	254	252	99641.4*	5878.24*

En la instalación de Zeek, se requiere ejecutar la *Instrucción 4.* Una vez instalado Zeek se ejecuta la siguiente *Instrucción 5*, con el nombre del archivo .pcap y obtendremos varios registros .log

```
echo 'deb http://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04/ /'
| sudo tee /etc/apt/sources.list.d/security:zeek.list
curl -fsSL
https://download.opensuse.org/repositories/security:zeek/xUbuntu_20.04/Release.ke
y | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/security_zeek.gpg > /dev/null
sudo apt update
```

Instrucción 4. Instrucciones para la instalación de Zeek. [20]

```
root@debianlock:/home/debian/Descargas# zeek -r 1. pcap
```

Instrucción 5. Extracción de archivos .log

Estos archivos .log se los puede transformar en .csv directamente ejecutando la *Instrucción 6*, en la terminal de Ubuntu, estas instrucciones permiten eliminar las primeras filas que solo tienen información de descripción, luego se reemplaza los saltos de línea por la (,).

```
root@debianlock:/home/debian/Descargas# sed -i '8d' conn.log
root@debianlock:/home/debian/Descargas# sed -i '1,6d' conn.log
root@debianlock:/home/debian/Descargas# sed -i 's/#fields\t//' conn.log
root@debianlock:/home/debian/Descargas# sed -i 's/\t/,/g' conn.log
```

Instrucción 6. Instrucciones para convertir un archivo .log a .csv

Una vez obtenido los archivos.csv de Argus y Zeek podemos hacer comparaciones entre las etiquetas de aquellos archivos. Una parte de suma importancia es tener identificado cada registro si fue una trama normal o maliciosa, para esta identificación se debe tener un informe con los eventos de seguridad que ocurren durante la simulación por la herramienta IXIA PerfectStorm a partir de aquel informe existe una Tabla 5, con etiquetas de relevancia para hacer comparación y obtener coincidencias con las etiquetas del conjunto de datos generado por Argus.

Para este caso solamente se trabajó con el conjunto de datos obtenido de la herramienta de Argus con 43.747 registros y además que cubría la mayoría de las etiquetas generadas por Zeek.

Tabla 5. Muestra de la tabla generada a partir de los informes de IXIA PerfectStorm. Fuente: El Autor

Start time	Last time	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port
1421927414	1421927416	Reconnaissance	HTTP	tcp	175451760	13284	14917112616	80
1421927415	1421927415	Exploits	Unix 'r' Service	udp	175451763	21223	14917112618	32780
1421927416	1421927416	Exploits	Browser	tcp	175451762	23357	14917112616	80
1421927417	1421927417	Exploits	Miscellaneous Batch	tcp	175451762	13792	14917112616	5555
1421927418	1421927418	Exploits	Cisco IOS	tcp	175451762	26939	14917112610	80
1421927419	1421927419	DoS	Miscellaneous	tcp	175451760	39500	14917112615	80
1421927419	1421927422	DoS	Miscellaneous	tcp	175451760	23910	14917112615	80
1421927420	1421927420	Generic	IXIA	tcp	175451760	29309	14917112614	3000
1421927421	1421927421	Exploits	Browser	tcp	175451760	61089	14917112618	80
1421927421	1421927422	Exploits	Browser	tcp	175451760	4159	14917112618	80
1421927422	1421927422	Exploits	SCADA	tcp	175451763	44762	14917112612	80
1421927423	1421927423	Exploits	Browser	tcp	175451762	43850	14917112613	80

Para la identificación de tramas que son buenas o malas, se realiza un procedimiento de comparación con los conjuntos de datos obtenidos de Argus y la tabla 5.

De esta manera se tendrá identificado cuando es una trama maligna con 1 y cuando es normal con 0. Esta columna que identifica al tráfico normal o maligno se la adjunta al conjunto de datos generado por Argus y así tendremos un conjunto de datos más completo donde se aplicara técnicas de balanceo y normalización obteniendo un conjunto de datos estructurado para entrenamiento y pruebas de Algoritmos de Machine Learning.

2.1.2 Descripción del conjunto de datos

UNSW-NB-15, esta divido en 4 archivos como se ve en la Figura 6, que consta de 700.000 registro en cada uno de los 3 primeros y 440.044 registros en el cuarto con un total de 2.540.044 de registros.

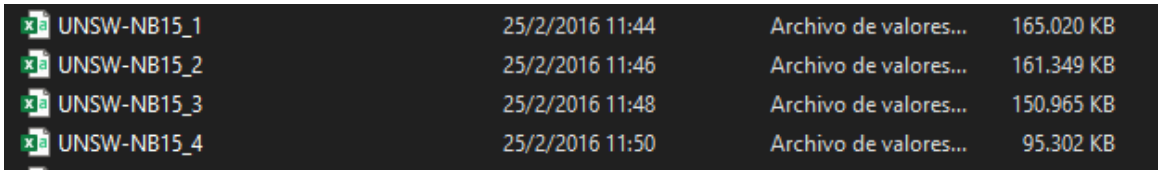

Figura 6. Archivos del conjunto de datos UNSW-NB15. Fuente: El Autor.

El conjunto de datos de nombre UNSW-NB-15, contine distintos tipos de ataques mostrados en la Tabla 6., está compuesto por las 35 primeras etiquetas generadas por las herramientas de Argus y Zeek mientras que 14 etiquetas finales fueron creadas mediante cálculos con el fin de detectar ataques sigilosos.

Tabla 6. Tipo de tráfico en UNSW-NB15. [4]

Tipo	Cantidad	Porcentaje [%]
Normal	2218764	87.35
Generic	215481	8.48
Backdoors	2329	0.09
DoS	16353	0.64
Reconnaissance	13987	0.55
Analysis	2677	0.11
Fuzzers	24246	0.95
Shellcode	1511	0.06
Worms	174	0.01
Exploits	44525	1.75
Total	2540047	

El conjunto de datos está conformado por 49 etiquetas y a través del programa WEKA se identificó que posee distintos tipos de datos como: enteros, float, tiempo, caracteres. Algunos registros contienen valores nulos, caracteres especiales. En resumen, datos cualitativos y cuantitativos.

A través del programa Rapid Miner, se determinó que etiquetas son cualitativas. En la Tabla 7, se puede visualizar las etiquetas cualitativas y el número de categorías que poseen. Además, se puede observar en la Tabla 8, ciertas inconsistencias como valores nulos.

Tabla 7. Numero de categorías que posee cada etiqueta cualitativa. Fuente: El Autor

Etiqueta	Nº Categorías
srcip(Fuente IP)	43
dstip(Destino IP)	47
proto(Protocolo)	135
state(Estado de la trama)	16
service(servicios)	13
attack_cat(Categoria del ataque)	13

Tabla 8. Presencia de valores nulos en algunas etiquetas. Fuente: El Autor

Etiqueta	Nº Nulos
dsport(Puerto destino)	304
sport(Puerto fuente)	8
ct_flw_http_mthd	1348143
is_ftp_login	1429877
attack_cat	2218760
ct_ftp_cmd	1429877

2.1.3 Preprocesamiento del conjunto de datos

El conjunto de datos a procesar posee algunas etiquetas con datos cualitativos y otras con datos cuantitativos, una gran variedad de los algoritmos de Machine Learning solo aceptan datos cuantitativos, por lo para este caso la conversión de los datos tiene una gran relevancia.

Como se mencionó en ítem “Descripción del conjunto de datos” , el conjunto de datos esta divido en 4 archivos, para analizar dicha información se procede a unir estos 4 archivos en uno, utilizando el Segmento de código 1, con el propósito de concentrar toda la información en un solo archivo para su procesamiento.

```
1. dfTotal= pd1.concat([dataf1,dataf2,dataf3,dataf4], axis=0)
```

Segmento de código 1. Unión de los 4 conjuntos de datos

Una vez identificado las etiquetas que poseen datos cualitativos, se utiliza el segmento de código 2, para poder categorizar los datos cualitativos empezando asignar a la primera

categoría el 0, segunda categoría el 1 y así hasta completar las categorías de datos que existan en la etiqueta.

```
1. dfTotal['srcip'] = pd.factorize(dfTotal['proto'])[0]
1. dfTotal['dstip'] = pd.factorize(dfTotal['state'])[0]
2. dfTotal['proto'] = pd.factorize(dfTotal['proto'])[0]
3. dfTotal['state'] = pd.factorize(dfTotal['state'])[0]
4. dfTotal['service'] = pd.factorize(dfTotal['service'])[0]
```

Segmento de código 2. Categorización de algunas etiquetas.

Para reemplazar los valores nulos se utiliza el siguiente Segmento de código 3 que consta en seleccionar las etiquetas que posean dichos valores y asignarles por el 0.

```
1. dfTotal['ct_flw_http_mthd']=dfTotal['ct_flw_http_mthd'].fillna(0)
2. dfTotal['sport']=dfTotal['sport'].fillna(0)
3. dfTotal['dsport']=dfTotal['dsport'].fillna(0)
4. dfTotal['attack_cat']=dfTotal['attack_cat'].fillna('-')
5. dfTotal['ct_ftp_cmd']=dfTotal['ct_ftp_cmd'].fillna(0)
6. dfTotal['is_ftp_login']=dfTotal['is_ftp_login'].fillna(0)
```

Segmento de código 3. Reemplazo de valores nulos por 0

En las etiquetas sport, dsport, ct_ftp_cmd son del tipo entero, pero poseen algunos valores del tipo strings para corregir esto utilizamos el siguiente Segmento de código 4, donde se reemplaza caracteres vacíos o desconocidos por 0 y finalmente para que no sean identificados como valores del tipo strings, se realiza la conversión a enteros.

```
1. dfTotal['ct_ftp_cmd'] = dfTotal['ct_ftp_cmd'].replace(' ', 0)
2. dfTotal['sport'] = dfTotal['sport'].replace('0x000b', 0)
3. dfTotal['dsport'] = dfTotal['dsport'].replace('0x000b', 0)
4. dfTotal['sport'] = dfTotal['sport'].replace('0x000c', 0)
5. dfTotal['dsport'] = dfTotal['dsport'].replace('0x000c', 0)
6. dfTotal['sport'] = dfTotal['sport'].replace('-', 0)
7. dfTotal['dsport'] = dfTotal['dsport'].replace('-', 0)
8. dfTotal['sport'] = dfTotal['sport'].astype(str).astype(int)
9. dfTotal['dsport'] = dfTotal['sport'].astype(str).astype(int)
10.dfTotal['ct_ftp_cmd'] = dfTotal['ct_ftp_cmd'].astype(str).astype(int)
```

Segmento de código 4. Reemplazo de strings por el valor de 0

2.1.4 Balanceo del conjunto de datos

Dentro del preprocesamiento de los datos está el balanceo como técnica para ajustar la distribución de clases, porque si los datos están desequilibrados, es decir si una clase tiene muchos más ejemplos que otra, puede existir problemas para entrenar el modelo. En el conjunto de datos usado existe una clase mayoritaria que son las tramas normales

mientras que una minoritaria de las tramas maliciosas esto provoca que el conjunto de datos se encuentre desbalanceado y tendríamos un resultado de clasificación con tendencia a las tramas normales.

Para tener una idea más clara de cómo está distribuido la salida 'Label' nos podemos ayudar del histograma presentado de la Figura 7.

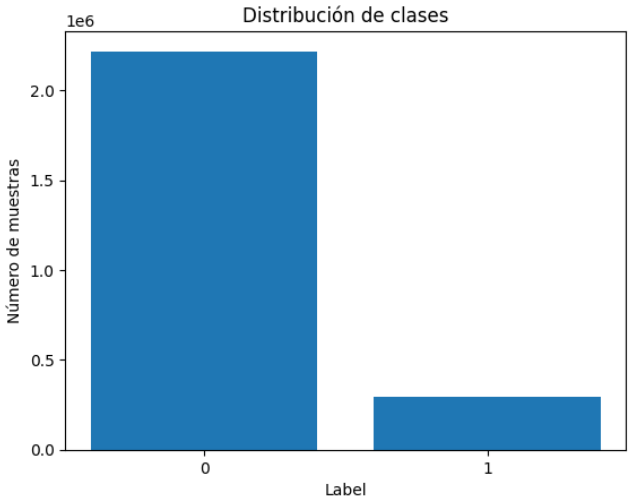

Figura 7. Histograma de clases de salida desbalanceado. Fuente: El Autor

En este histograma de la Figura 7, se evidencia la presencia mayoritaria cerca de 2,5 millones de filas de datos para la categoría 0 o conocido como tráfico normal mientras que para la categoría 1 se tiene solo alrededor de 260 mil filas de datos.

Existen varias técnicas para equilibrar un conjunto de datos. Este reúne las características para que se pueda implementar el submuestreo ya que posee gran cantidad de tramas normales y solo un pequeño porcentaje de tramas maliciosas [21] la forma que se puede implementar el submuestreo es de forma manual o utilizando las funciones de python 'Random Under-Sampler' o 'Near-Miss'.

Para lograr que el conjunto de datos desbalanceado este balanceado se aplicará la función 'Random Under-Sampler', esta permite que se reduzca la clase mayoritaria de forma aleatoria, es decir, se elimina cierta cantidad de registros que contengan la clase 0 de tramas normales hasta igualar a la clase 1 de las tramas maliciosas.

De esta manera nos queda un conjunto de datos con 642566 registros entre tramas buenas y malas. Este nuevo conjunto de datos equilibrado con la misma cantidad de tramas normales y tramas maliciosas se observa su distribución en el histograma de la Figura 8.

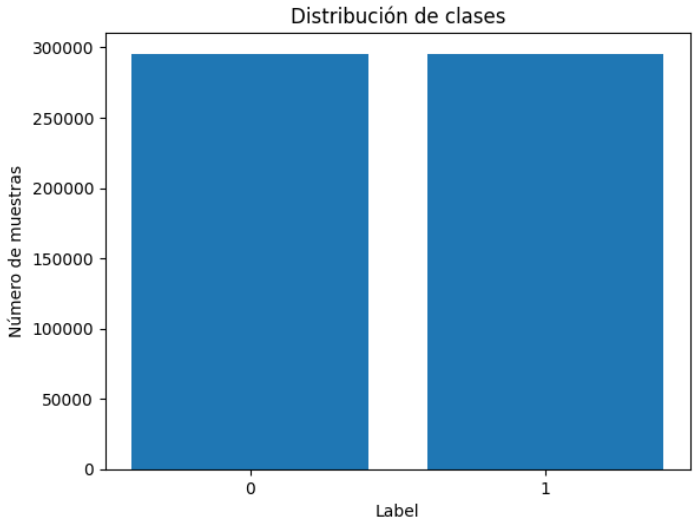

Figura 8. Histograma de conjunto de datos balanceado. Fuente: El Autor

2.1.5 Estructuración del conjunto de datos para entrenamiento y prueba

Dentro del estudio de las redes IoT se ha destacado que su poder computacional es limitado por lo cual es de suma importancia que el entrenamiento y prueba de los distintos algoritmos sea eficiente, donde se logre cumplir con métricas convincentes sin sacrificar el objetivo primordial detectar tramas maliciosas. Para lograr que se reduzca el esfuerzo que implica trabajar con la totalidad de registros y etiquetas se ha implementado algunos procedimientos como reducir aún más la cantidad de registros, seleccionando a los ataques más frecuentes y aplicar el análisis de componentes principales para reducir la cantidad de etiquetas.

Ataques más frecuentes en el conjunto de datos

En el análisis que se ha dado al conjunto de datos se puede observar que a pesar de balancear sigue conteniendo una gran cantidad de registros que pueden requerir mayor poder computacional en el entrenamiento y predicciones de los algoritmos de Machine Learning, por esto se pretende reducir aún más la cantidad de registros al trabajar con solo 4 categorías de ataques más frecuentes que se localizan en el conjunto de datos.

Inicialmente se debe recordar con la Tabla 9 que los datos están completamente cuantitativos, donde cada tipo de ataque es representado por un número entero.

Tabla 9. Categorización de los tipos de ataques. Fuente: El Autor

Categoría	Número
Exploits	1
Reconnaissance	2
DoS	3
Generic	4
Shellcode	5
Fuzzers	6
Worms	7
Backdoors	8
Analysis	9
Reconnaissance	10
Backdoor	11
Fuzzers	12
Shellcode	13

Ya conociendo el número de categoría que representa cada tipo de ataque podemos determinar cuáles son los más frecuentes, lo tenemos representado en la siguiente Figura 9. Con esta identificación se puede retirar del conjunto de datos a los otros 9 tipos de ataque. Y continuar el análisis de obtener un datashet eficiente.

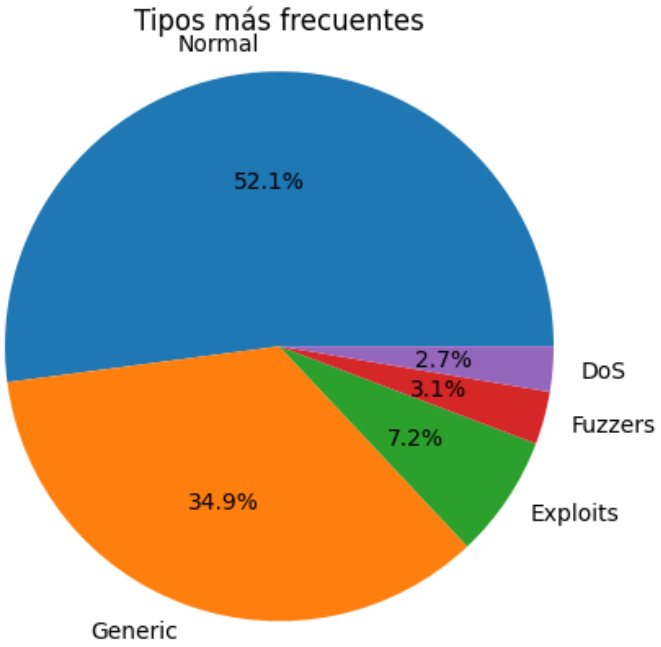

Figura 9. Tipos de ataques más frecuentes. Fuente: El Autor

Normalización del conjunto de datos

Algunos algoritmos son sensibles a valores atípicos que pueden entorpecer el entrenamiento y consecuentemente las predicciones, en la Figura 10 se puede observar cómo los valores promedios de unas 4 etiquetas están dispersos del promedio del resto de etiquetas. Se realizará la normalización de los datos utilizando el método de min-max expresado en la Ecuación 2.1 que consiste en ir restando el valor mínimo de cada valor y dividiendo el resultado por la diferencia entre el valor máximo y el mínimo del conjunto de datos.

$$Z_j^{normalizacion} = (Z_j - \min(Z)/(\max(Z) - \min(Z)) \qquad (2.1)$$

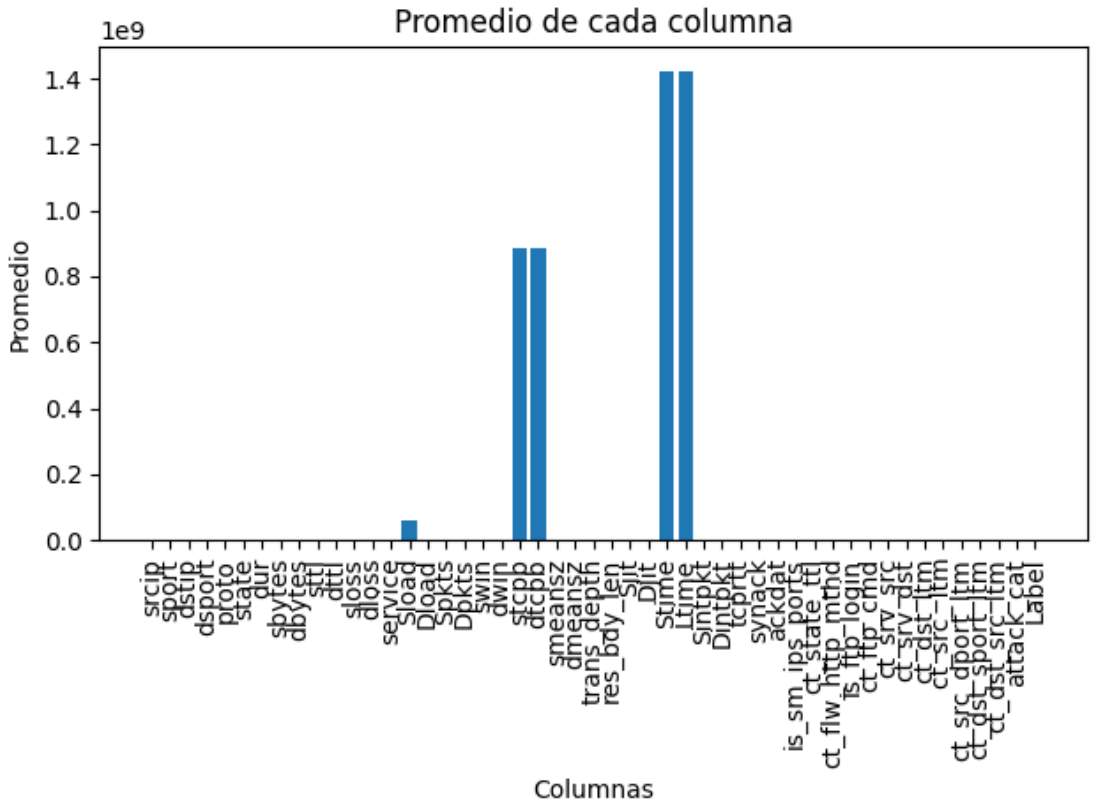

Figura 10. Histograma con promedios de las etiquetas sin normalizar. Fuente: El Autor

Hay que recalcar que solamente las etiquetas de entrada deben estar normalizadas en un rango que puede ser entre 0 o 1. La salida no debe ser normalizada ya que 'Label' tiene solo valores de 0 para tráfico normal y 1 para tráfico malicioso.

Luego de realizada la normalización de datos podemos revisar nuevamente el análisis de los promedios y en la Figura 11 tenemos que los promedios de las 4 etiquetas

mencionadas ya no están distanciados del resto. Además, se observa la normalización entre valores de 0 a 1

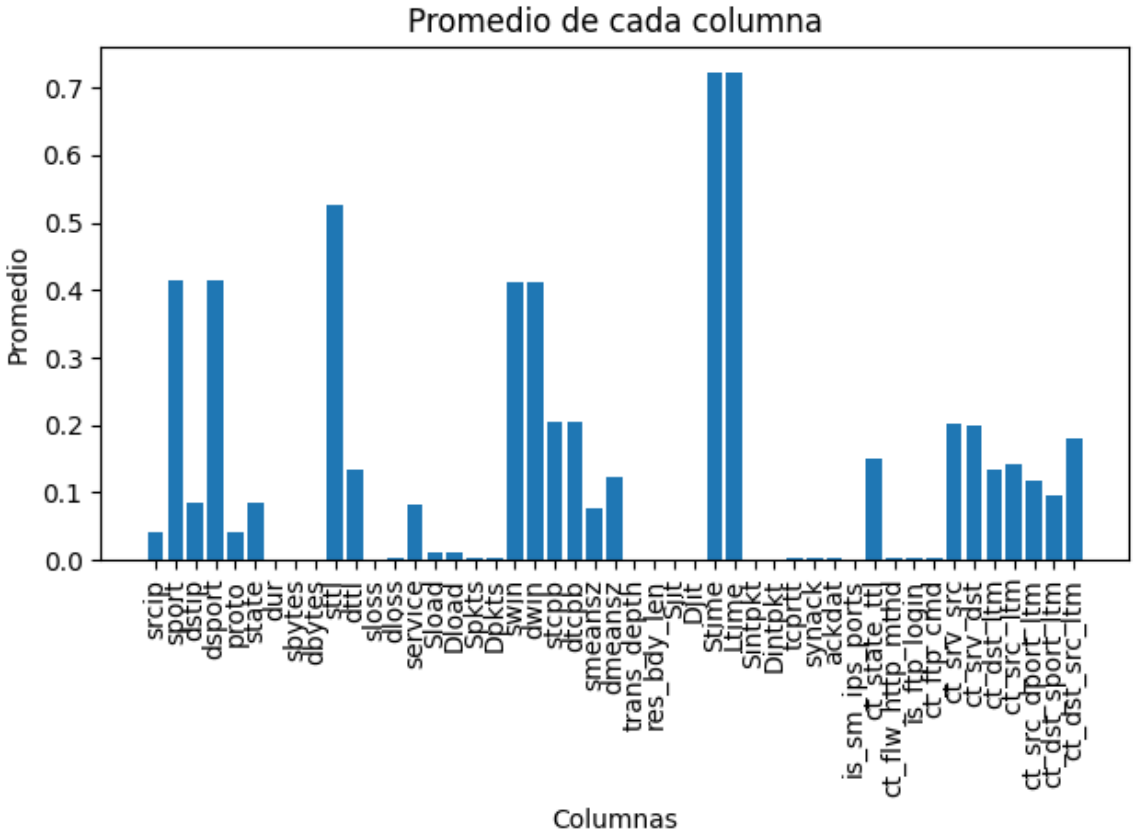

Figura 11. Histograma con promedios de las etiquetas ya normalizadas. Fuente: El Autor

Análisis Factorial

El conjunto de datos preprocesado hasta el momento contiene aun las 49 etiquetas del conjunto de datos original lo único que ha variado es la cantidad de registros o tramas.

En el análisis factorial como técnica nos permitirá reducir la dimensión o cantidad de etiquetas, estas nuevas etiquetas estarían representando las principales características del conjunto de datos. De esta manera lograremos que exista menos carga computacional al entrenamiento y prueba de los distintos algoritmos.

Con ayuda del programa IBM SPSS nos permitirá tener una idea más completa y general acerca de si es factible la reducción de dimensión, esto gracias al valor de KMO y Bartlett.

Káiser Meyer Olkin (KMO), permite determinar la adecuación de la data set dando valores entre 0 a 1 y así donde los valores más cercanos a 1 señala que es posible aplicar

reducción de dimensiones. En un rango entre 0.8 a 0.9 es bueno entre 0.7 y 0.8 aceptable [21].

Bartlett, evalúa una hipótesis nula Ho, si se consigue valores con nivel $p<0.05$ no se acepta la hipótesis nula y se considera adecuado para realizar la reducción de dimensiones [22].

En el análisis a través del programa SPSS Versión 29.0.1.0, se obtuvo algunos valores en los que se observa un valor de KMO = 0.856 y de Bartlett < 0.001, lo cual nos da un resultado favorable para proceder aplicar la reducción de dimensiones.

En el criterio para determinar cuántas componentes principales son factibles o reúnen la mayor cantidad de características del resto de variables, se lo realiza con el grafico de sedimentación que lo podemos observar en la Figura 12, este grafico nos muestra la varianza acumulada por los componentes principales, y justamente en las 10 componentes ya no hay más variaciones y se genera una línea de tendencia al resto de variables.

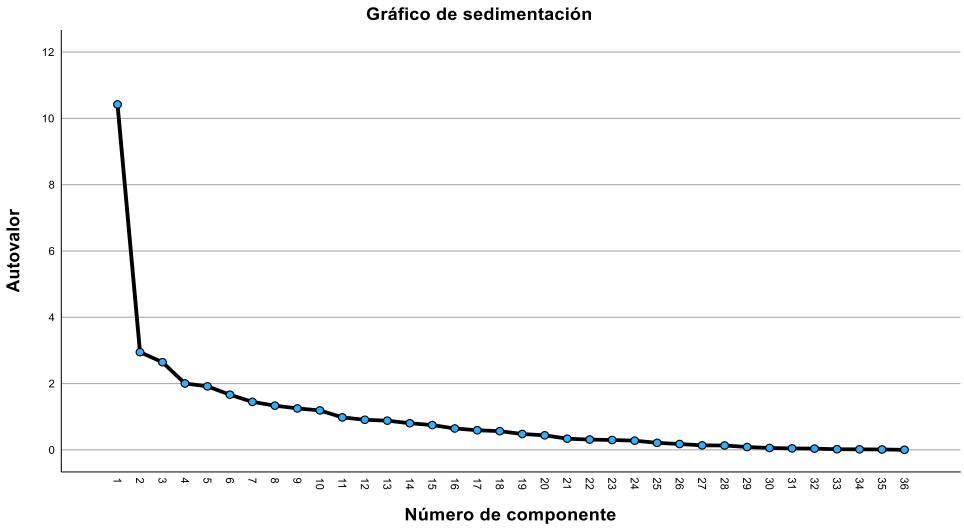

Figura 12. Gráfico de sedimentación Fuente: El Autor

Estos elementos se clasifican de acuerdo con la cantidad de variación en los datos originales que pueden explicar. El primer componente principal representa la mayor parte de la varianza, al que luego le sigue el segundo componente principal, que representa la segunda mayor cantidad de varianza, y así sucesivamente.

Para la aplicación de PCA se debe estandarizar el conjunto de datos con el Segmento de código 5, la estandarización consiste en calcular la media y la desviación estándar de cada etiqueta numérica en el conjunto de datos. Luego, se resta la media de cada etiqueta de cada uno de sus registros y se divide por su desviación estándar. Como resultado, cada variable tendrá una media de cero y una desviación estándar de uno.

```
1. from sklearn.preprocessing import StandardScaler
2. std=StandardScaler()
3. std_trans=std.fit_transform(dfGeneralN)
```

Segmento de código 5. Estandarización del conjunto de datos

Para obtener las 10 componentes principales representativas se utiliza el Segmento de código 6.

```
1. n_componentes=10
2. pca_v3=PCA(n_componentes)
3. pca_valor=pca_v3.fit_transform(std_trans)
4. pca_v3.components_
```

Segmento de código 6. Obtención de componentes principales.

Al obtener las 10 componentes principales no correlacionadas. Estas componentes agrupan las características del resto de componentes del conjunto de datos original. En la Tabla 10, se puede observar a cada componente principal puede contener registros positivos y negativos esto simplemente refleja cómo se distribuye la varianza en los datos originales a lo largo de esa componente particular. Una componente principal negativa indica que los valores de esa componente están en dirección opuesta a la dirección general de los datos en el espacio original.

Tabla 10. Muestra de los registros de las componentes principales. Fuente: El Autor.

	compt1	compt2	compt3	compt4	compt5	compt6	compt7	compt8	compt9	compt10
0	-3.184.84	0.623169	-0.79885	-0.07010	-0.21499	-0.39112	0.941481	-0.89382	0.76371	0.46755
1	4.775.61	2.850.36	-4.433.48	3.783.457	3.624.224	8.054.430	-6.020.45	-4.145.17	1.619.29	1.894.81
2	2.932.44	0.298149	-1.200.47	-0.28005	-0.21814	-0.90621	0.054247	0.248439	-0.31327	-0.1632
3	3.614.9	0.67790	-1.670.6	0.101901	0.070199	-0.41429	-0.50122	0.69461	-0.5006	-0.09732
4	2.829.59	1.111.02	0.14913	-3.142.91	-9.363.6	3.564.29	-0.69249	0.478342	0.520262	0.262352
616837	……	……	……	……	……	……	….	……	,,,,,,	……

Para la división del conjunto de datos en prueba y entrenamiento se lo realiza sencillamente con el programa WEKA versión 3.8.6 o directamente en Python con el siguiente Segmento de código 7. Teniendo así un 70% para el conjunto de entrenamiento y 30% al conjunto de prueba.

```
1. X_entren, X_prueb, y_entren, y_prueb = train_test_split(entrada, output_data,
   test_size=0.3, random_state=42)
```

Segmento de código 7. Separación en conjunto de entrenamiento y prueba

2.2 Análisis de datos utilizando algoritmos de Machine Learning

2.2.1 Condiciones iniciales para entrenamiento y prueba

Para iniciar con el entrenamiento y prueba de algunos de los modelos de machine Learning. Se tienen condiciones iniciales iguales en el conjunto de datos para todos estos algoritmos con el fin de evaluar al mismo nivel tanto en: etiquetas de entrada, normalización y cantidad de registros.

2.2.2 Estructuración de los algoritmos

Condiciones del conjunto de datos para Regresión Logística

Entre los requisitos para entrenamiento de regresión Logística es necesario que el conjunto de datos cumpla con algunos de estas condiciones:

- **Datos etiquetados:** Ya que es un algoritmo de aprendizaje supervisado requiere que tanto entradas como la salida estén con su respetivos nombres o etiquetas.
- **Variables independientes:** Asegurar que sean variables independientes y que sean influyentes en las variables de salida.
- **Linealidad:** La regresión logística aplica una relación lineal entre las etiquetas independientes y la probabilidad de la variable de salida.
- **Ausencia de multicolinealidad:** La multicolinealidad ocurre cuando hay una alta correlación entre las variables independientes.
- **Suficiente tamaño de muestra:** La regresión logística tiende a funcionar mejor con un tamaño de muestra más grande.

Bibliotecas y código de Python utilizado.

```
1. from sklearn.linear_model import LogisticRegression
2. from sklearn.model_selection import train_test_split
3. from sklearn.metrics import accuracy_score
```

Segmento de código 8. Bibliotecas para regresión logística

En el Segmento de código 8, podemos evidenciar las siguientes bibliotecas:

La biblioteca 1 es utilizada para la implementación de la regresión logística, la biblioteca 2 es para la división del conjunto de entrenamiento y prueba, la biblioteca 3 proporciona métricas de evaluación para medir el rendimiento del modelo.

Entre las principales funciones de la regresión logística tenemos:

```
1. model = LogisticRegression()
2. model.fit(X_train, y_train)
3. y_pred = model.predict(X_test)
```

Segmento de código 9. Estructuración de Regresión Logística

En el Segmento de código 9 , está conformado por la función de modelo para regresión logística la cual consiste en calcular la función sigmoide y así obtener probabilidades entre 0 y 1, calcula la función de costo donde evalúa a conjunto de entrenamiento, utiliza técnica de optimización como el descenso de gradiente. Y finalmente con la función de predicción donde evalúa el rendimiento con un conjunto de prueba , y se utilizan métricas de evaluación para medir su calidad.

Condiciones del conjunto de datos para Redes Neuronales Convolucionales

- **Tamaño adecuado:** El conjunto de entrenamiento debe tener un tamaño suficiente grande en cantidad de registros para que el modelo pueda aprender patrones y características representativas.
- **Normalización:** Se desea tener una media cercana a 0 y una desviación estándar cercana a 1.
- **Independencia:** El conjunto de entrenamiento y prueba debe ser independiente, sin superposición de muestras, para evitar sesgar la evaluación del modelo.
- **Equilibrio de clases:** Si hay clases desequilibradas, es importante que tanto el conjunto de datos para el entrenamiento como el conjunto de datos para las pruebas tengan una distribución similar de clases.

- **Aleatoriedad:** La selección de muestras para los conjuntos debe ser aleatoria para evitar sesgos en la selección.

Bibliotecas y código de Python utilizado

```
1. import tensorflow as tf
2. import numpy as np
3. from tensorflow.keras import layers
```

Segmento de código 10. Bibliotecas para redes neuronales convolucionales.

Entre las bibliotecas utilizadas del Segmento de código 10 esta:
numpy: Biblioteca que permite realizar operaciones matemáticas y numéricas en matrices
tensorflow: Permite construir y entrenar modelos de aprendizaje automático, incluidas redes neuronales.
tensorflow.keras: Proporciona una interfaz fácil de usar y modular para la construcción, compilación, entrenamiento y evaluación de modelos de redes neuronales.

```
1. # Definir la arquitectura de la CNN
2. model = tf.keras.Sequential([
3. layers.Reshape((10, 1), input_shape=(10,)),
4. layers.Conv1D(64, 3, activation='relu'),
5. layers.MaxPooling1D(2),
6. layers.Conv1D(128, 3, activation='relu'),
7. layers.MaxPooling1D(2),
8. layers.Flatten(),
9. layers.Dense(256, activation='relu'),
10. layers.Dropout(0.5),
11. layers.Dense(1, activation='sigmoid')
12. ])
```

Segmento de código 11. Estructura de redes neuronales convolucionales

La estructuración del código 11 de la red neuronal convolucional está en capas donde se describe:

Capa de Reshape: Esta capa cambia la forma de entrada de los datos de (10,) a (10, 1). Es necesario porque las capas convolucionales en Keras esperan una entrada en formato de imagen (ancho, alto, canales) y en este caso se utiliza una dimensión adicional para representar los canales.

Capa Conv1D: Esta capa convolucional tiene 64 filtros, cada uno de tamaño 3 y utiliza la función de activación ReLU.

Capa MaxPooling1D: Esta capa realiza un submuestreo en la salida de la capa convolucional, reduciendo la dimensionalidad del mapa de características mediante la selección del valor máximo en ventanas de tamaño 2.

Capa Conv1D: Otra capa convolucional con 128 filtros de tamaño 3 y función de activación ReLU.

Capa MaxPooling1D: Otra capa de submuestreo que reduce aún más la dimensionalidad del mapa de características.

Capa Flatten: Esta capa aplana el mapa de características en un vector de una sola dimensión que se prepara para una capa densa.

Capa Dense: Capa densa con 256 neuronas y función ReLU.

Capa Dropout: Esta capa realiza una regularización para evitar el sobreajuste, apagando aleatoriamente un porcentaje de las neuronas durante el entrenamiento (en este caso, el 50% de las neuronas se apagan).

Capa Dense: Capa para la salida con 1 neurona y función sigmoide. Produce la salida final de la red, que es un valor entre 0 y 1 para la clasificación binaria.

Condiciones del conjunto de datos para k-Nearest Neighbors

- **Datos numéricos:** KNN funciona mejor con datos numéricos en lugar de datos categóricos.
- **Datos normalizados:** Es recomendable normalizar los datos antes de aplicar KNN.
- **Datos sin valores faltantes:** KNN no maneja automáticamente valores faltantes en los datos.

Bibliotecas y código de Python utilizado

```
1. from sklearn.neighbors import KNeighborsClassifier
2. from sklearn.metrics import accuracy_score
```

Segmento de código 12. Bibliotecas para entrenamiento de KNN

Del Segmento de código 12 están las bibliotecas utilizadas como: sklearn.neighbors para implementar el algoritmo KNN y sklearn.metrics para evaluar el rendimiento del modelo.

```
#Definimos el número de vecinos
k=4
#Creamos el objeto KNN
knn=KNeighborsClassifier(n_neighbors=k)
knn
#Ajustar o entrenar
knn.fit(val_data,val_labels)
#Calcular o obtener la hipótesis o predicciones
predicciones=knn.predict(test_data)
```

Segmento de código 13. Estructuración del código de entrenamiento de KNN

En el Segmento de código 13, se puede destacar la selección k vecinos. Este aspecto es importante en el entrenamiento de KNN.

KNN no realiza un proceso de entrenamiento propiamente dicho, ya que no construye un modelo explícito a partir de los datos de entrenamiento. El algoritmo KNN almacena los datos de entrenamiento en memoria. Durante la etapa de entrenamiento, no se realiza ningún cálculo o procesamiento adicional, ya que KNN no "aprende" explícitamente a partir de los datos de entrenamiento.

KNN solamente predice calculando las distancias entre algún punto de prueba que se le asigna y un punto del conjunto de entrenamiento más cercano. A este cálculo de la distancia se llama distancia Euclidiana.

Condiciones del conjunto de datos para Naive Bayes

- **Tipo de datos:** Naive Bayes está diseñado para datos categóricos y continuos.
- **Datos redundantes:** Es fuerte frente a características redundantes.
- **Valores faltantes:** Naive Bayes puede manejar valores faltantes en los datos.
- **Validación cruzada:** Es de importancia evaluar a Naive Bayes con validación cruzada para obtener resultados confiables.
- **Conjunto de datos balanceado:** Naive Bayes requiere obligatoriamente que el conjunto de datos se encuentre balanceado de no ser así sus resultados presentarán sesgos a la clase mayoritaria.

Bibliotecas y código de Python utilizado

```
1. from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
2. from sklearn.model_selection import train_test_split
```

Segmento de código 14. Bibliotecas para la evaluación de Naive Bayes.

En el Segmento de código 14, se tiene las siguientes bibliotecas para Naive Bayes sklearn.naive_bayes: Se refiere a los tipos de clasificadores. GaussianNB: Es para datos continuos. sklearn.model_selection: Herramientas de evaluación y manejos del modelo.

```
1. # Crear un clasificador Naive Bayes
2. nb_classifier = GaussianNB()
3. nb_classifier.fit(X_train, y_train)
4. y_pred_nb = nb_classifier.predict(X_test)
```

Segmento de código 15. Estructuración del código para el algoritmo de Naive Bayes

El segmento de código 15 se basa en :
Preparar el clasificador: Se establece el tipo de clasificador que se utilizará, en este caso, es una versión de Naive Bayes llamada "GaussianNB".

Entrenar el clasificador: Se alimenta al clasificador con ejemplos de entrenamiento para que aprenda cómo funcionan las características en relación con las etiquetas. Esto es como enseñarle al clasificador a reconocer patrones.

Hacer predicciones: Una vez entrenado, el clasificador se utiliza para adivinar las etiquetas de un conjunto de datos que no ha visto antes como el conjunto de prueba. Utiliza las características de estos datos para hacer predicciones sobre qué etiqueta sería la más apropiada. Guardar las predicciones: Las etiquetas adivinadas se guardan en una variable para su posterior uso o análisis.

2.3 Análisis comparativo entre los modelos de Machine Learning

En el entrenamiento y análisis de los 4 algoritmos de Machine Learning se desarrolló con 10 componentes principales que representan alrededor del 73% de la varianza total y también sin aplicar análisis de componentes principales es decir con el total 47 etiquetas.

Estas pruebas de aplicar PCA y sin aplicar PCA se lo realiza con la finalidad de analizar si es factible aplicar componentes principales a entornos IoT donde se tenga una buena precisión, exactitud y un menor consumo de recursos.

En cada algoritmo de Machine Learning se utilizó el mismo conjunto de datos y para evaluar se aplicó la técnica de validación cruzada donde con 5 interacciones se divide al conjunto de datos en 5 partes. En cada interacción se realiza la obtención de métricas de accuracy, precisión y consumo de recursos, para finalmente obtener un valor promedio de las 5 interacciones.

Para el análisis de tiempo, uso de CPU, memoria RAM, se lo realizo con el siguiente Segmento de código 16, la funcionalidad de este código es tomar medidas o información antes de ejecutar el algoritmo y después de haberlo ejecutado con aquello se logra una diferencia de valores lo que permite tener un resultado aproximado del consumo de recursos.

```
1. start_time = time.time() #valor inicial del tiempo
2. # Uso anterior del CPU
3. cpu_before_training = psutil.cpu_percent(interval=None)
4. memory_before1 = psutil.virtual_memory().used # Uso anterior RAM
5. #----------------------------------
6. #Codigo de ejecución del algoritmo de Machine Learning
7. #----------------------------------
8. memoryPrediccionRL = psutil.virtual_memory().used - memory_before1 # Obtener variación de memoria RAM.
9. cpuPrediccionRL = psutil.cpu_percent(interval=None) - cpu_before_training # Obtener variación del uso del CPU
10. timePrediccionRL = time.time() - start_time #comparacion de tiempos.
```

Segmento de código 16. Código para extracción de consumo de recursos.

Análisis comparativo con y sin PCA en el algoritmo Regresión Logística.

Tabla 11. Tabla con métricas de evaluación y consumo de recursos en RL. Fuente: El Autor.

	Exactitud	Precisión	Tiempo (s)	RAM (bytes)	CPU (%)
10 PCA	0.9751	0.9640	0.6390	7955251	14.58
Sin PCA	0.9905	0.9841	10.9037	30794547	31.57

Los resultados de la Tabla 11, demuestran que, para este modelo de regresión logística, la adición de Análisis de Componentes Principales afecta las métricas de rendimiento del clasificador. En ambas ocasiones, la precisión es notablemente alta, pero cuando se

utilizó PCA, cayó un 2%. El modelo PCA todavía tiene un alto nivel de precisión con un valor del 96,40% a pesar de esta reducción. Dadas las ventajas adicionales que puede ofrecer la reducción del tamaño de la PCA en términos de efectividad y conocimiento de los datos, es fundamental pensar detenidamente si una reducción de precisión de este tipo es aceptable.

Tanto el mismo escenario se repite para la exactitud, con la diferencia del 2% ya que al aplicar PCA se tiene el 97.5% que es un valor excelente de clasificación y un valor perfecto del 99% al no aplicar PCA.

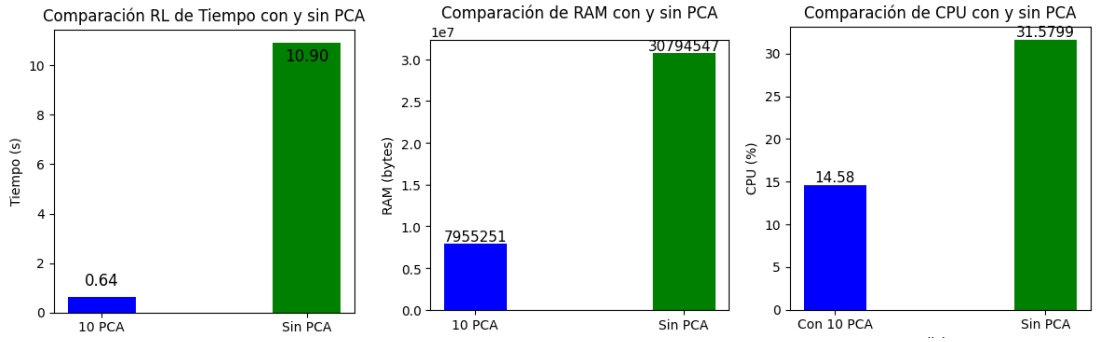

Figura 13. Comparaciones de consumo de recursos RL. Fuente: El Autor.

Desde el punto de vista de la eficiencia computacional tenemos la figura 13, se visualiza una ventaja considerable al utilizar PCA que sin PCA, donde se requiere más tiempo y recursos. Esto es una indicación que al reducir la dimensión del conjunto de datos se está ayudando a acelerar el proceso de clasificación, es importante considerar si el ahorro de tiempo y recursos se traduce en una disminución de la precisión.

Se observa que la sección que no se aplica PCA utiliza aproximadamente el doble en potencia en CPU y 4 veces más en RAM. Esto se toma en cuenta para sistemas con limitación de recursos , donde la eficiencia es crítica.

Análisis comparativo con y sin PCA en el algoritmo Redes Neuronales Convolucionales

Como se puede observar en la Tabla 12, la precisión del modelo equipado con PCA es ligeramente mayor 99,14 % que la del modelo sin PCA, que alcanza el 98,76 por ciento. Este aumento en la precisión podría sugerir que el PCA tiene un efecto beneficioso sobre la capacidad de clasificación del modelo.

El análisis de la precisión revela un patrón similar. Al lograr una precisión del 98,44 por ciento, el modelo equipado con PCA superó al modelo sin PCA en un 98,07 por ciento.

Parece que la adición de PCA mejora la precisión de la clasificación del modelo, aunque la diferencia es insignificante.

Tabla 12. Tabla con métricas de evaluación y consumo de recursos en CNN. Fuente: El Autor.

	Exactitud	Precisión	Tiempo (s)	RAM (bytes)	CPU (%)
10 PCA	0.9914	0.9844	77.9458	32084787	23.3
Sin PCA	0.9876	0.9807	194.8033	181986918	31.40

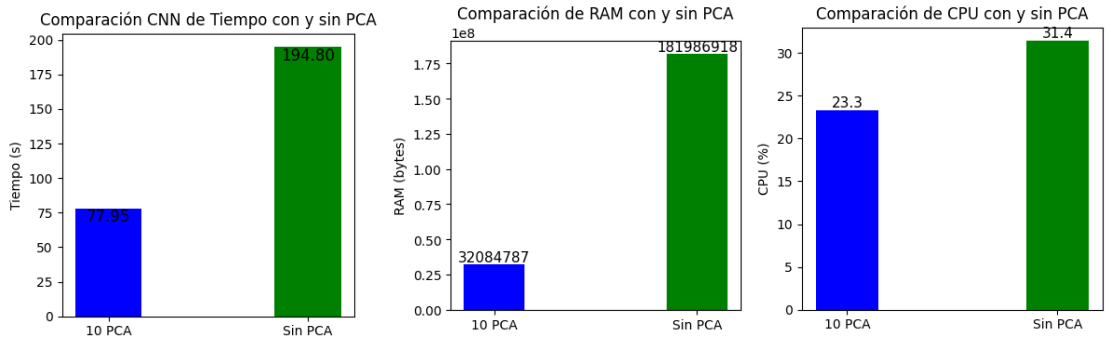

Figura 14. Comparaciones de consumo de recursos en CNN. Fuente: El Autor.

En la Figura 14, se reconoce que la solicitud de PCA genera costos en términos de uso de recursos y tiempo de procesamiento. Se necesitan 77,94 segundos para el modelo PCA, frente a 194,80 segundos para el modelo sin PCA, esto nos demuestra que con menos cantidad de registros de entrada puedo obtener mejores resultados si el análisis PCA es correctamente implementado. Además, se observa que el modelo sin PCA utiliza 181.986.918 bytes de RAM y el 31,4 por ciento de la potencia de la CPU, en comparación con los aproximadamente 32.084.787 bytes de RAM y el 23,3 por ciento de potencia de la CPU del modelo PCA.

Análisis comparativo con y sin PCA en el algoritmo KNN.

Tabla 13. Tabla con métricas de evaluación y consumo de recursos en KNN. Fuente: El Autor.

	Exactitud	Precisión	Tiempo (s)	RAM (bytes)	CPU (%)
10 PCA	0.9894	0.9885	8.67	12674662	14.88
Sin PCA	0.9897	0.9891	478.0688	146492620	36.8

Los resultados de la Tabla 13, ofrecen una perspectiva para contrastar la efectividad de KNN frente al uso de componentes principales. Ambos métodos muestran un equilibrio alto en términos de precisión. La precisión del método con PCA es del 98,8%, mientras que la precisión del método sin PCA es del 98,9%. Es justo decir que ambas opciones

exhiben un rendimiento comparable por no decir igual ya que mientras el método sin PCA tiene una exactitud del 98,9%,y el método con PCA también tiene 98.9%.

Esta pequeña variación sugiere que la optimización PCA tendría poco impacto en la capacidad del modelo KNN sino se analizaría con el consumo de los recursos de la Figura 15. Se tiene en comparación con el tiempo de procesamiento del método sin PCA de 478,0688 segundos, el método con PCA muestra un tiempo de procesamiento de sólo 5,7217 segundos. Al utilizar 12.674.662 bytes de RAM y utilizar el 18,74 por ciento de la CPU en lugar de 146.492.620 bytes de RAM y el 36,8 por ciento de la CPU en modo PCA sin PCA, el método PCA reduce significativamente el consumo de recursos.

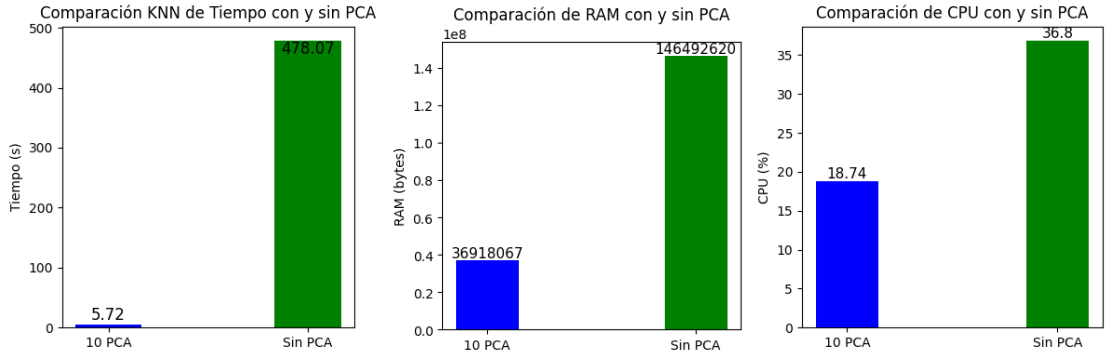

Figura 15. Comparaciones de consumo de recursos en KNN. Fuente: El Autor.

Análisis comparativo con y sin PCA en el algoritmo Naive Bayes.

Tabla 14. Tabla con métricas de evaluación y consumo de recursos en NB. Fuente: El Autor.

	Exactitud	Precisión	Tiempo (s)	RAM (bytes)	CPU (%)
10 PCA	0.8923	0.8236	0.1080	14297497	9.72
Sin PCA	0.9313	0.8917	0.4909	100645273	13.18

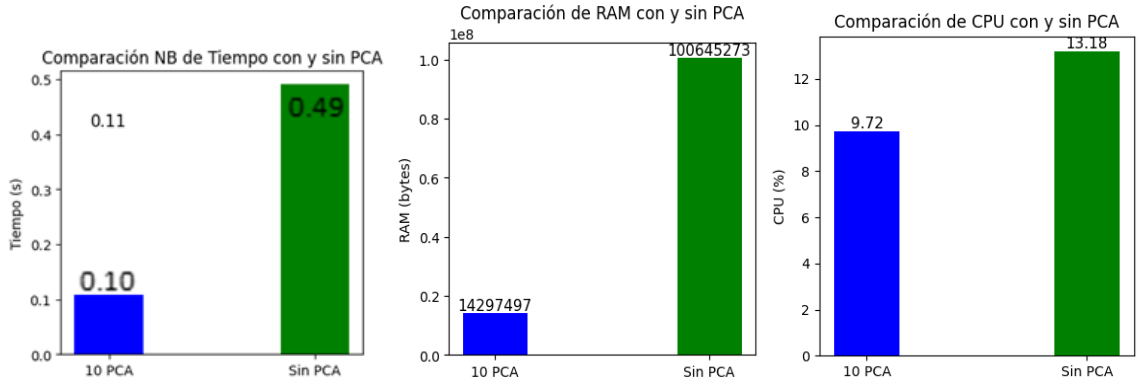

Figura 16. Comparaciones de consumo de recursos en NB. Fuente: El Autor.

Cuando se compara los valores de exactitud al utilizar la técnica de componentes principales y no utilizarla, se puede ver en la Tabla 14, que el método sin PCA arroja una exactitud de 93.1% mientras que el método con PCA arroja una exactitud del 89.2 % . Esta diferencia de grupo alrededor del 4% podría ser una señal de que la técnica de reducción de dimensión del PCA interfiere con la capacidad del modelo para clasificar datos con tanta precisión como el modelo sin PCA. En comparación con el método con PCA, el método sin PCA muestra una precisión de 89.1%, que es mayor. Esta variación puede demostrar cómo la reducción de dimensión producida por PCA puede afectar la capacidad del modelo Naive Bayes.

En términos de recursos expresados en la Figura 16, el modelo con PCA requiere 14,297,497 bytes de RAM y un 9.72% de la CPU, en comparación con los 100,645,273 bytes de RAM y el 13.18% de la CPU utilizados por el modelo sin PCA.

Para esta sección de los 4 tipos de ataques más frecuentes que se encontró en el conjunto de datos original , se realizó el análisis por cada uno de estos tipos de tramas o ataques maliciosos con su entrenamiento y evaluación de los 4 algoritmos de Machine Learning con la finalidad de saber cuál es más adecuado para detectar ese tipo de ataque.

Evaluación en ataque del tipo Generic

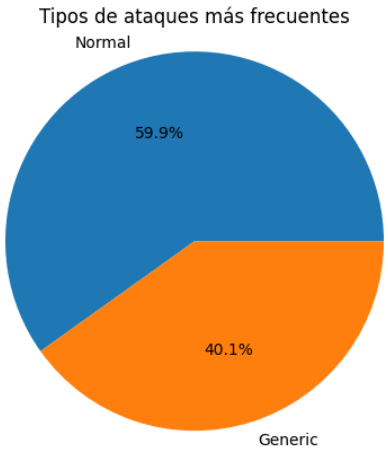

Figura 17. Porcentaje de tramas normales y tramas del tipo Generic (maliciosas). Fuente: El Autor.

El tipo de ataque Generic también conocido como Scan Generic, es un tipo de ataque de escaneo para enumerar hosts, servicios que se ejecutan en una red [23]. En la Figura 17 se tiene su porcentaje de distribución, siendo el de mayor presencia en el conjunto de

datos, para este caso se puede observar que no existe un desbalance en clases de tramas normales o maliciosas.

Tabla 15. Métricas de evaluación en clasificación de tramas normales y Generic. Fuente: El Autor.

	Exactitud	**Precisión**	**Tiempo (s)**	**RAM (bytes)**	**CPU (%)**
RL	0.9830	0.9726	2.7448	12514918	17.86
CNN	0.9947	0.9916	47.8991	22401024	35.98
KNN	0.9973	0.9968	7.3531	156928409	28.12
NB	0.9103	0.8268	0.1329	45875	15.12

Como se observa en la Tabla 15, Naive Bayes destaca en rendimiento ya que sus valores en tiempo, RAM y CPU son los más bajos comparados al resto solamente afectado por 82.7% de precisión y un valor de 91% de exactitud .Se puede observar como KNN y CNN tienen valores casi perfectos en exactitud y precisión, pero con un consumo más elevado de sus recursos.

Regresión logística realiza una combinación de exactitud, precisiones favorables con 98.3% y de 97.3% respectivamente y consumo de recursos como se observa en la Tabla 15, un tiempo de 2.7s mucho menor que CNN y KNN además de valores RAM y CPU menores.

Tabla 16. Matrices de confusión al clasificar tramas normales y Generic. Fuente: El Autor.

Matriz de confusión RL		Real		**Matriz de confusión CNN**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	63969	1697	Predicción	**P**	64567	426
	N	932	94432		**N**	334	95703
Matriz de confusión KNN		Real		**Matriz de confusión NB**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	64669	229	Predicción	**P**	63750	13153
	N	232	95900		**N**	1151	82976

Tanto La Matriz de confusión de RL y NB presenta cierto porcentaje de errores considerables entre los falsos positivos y falsos negativos además muestra tendencia en mayor clasificación para verdaderos negativos como se observa en la Tabla 16. La menor cantidad de falsos negativos y falsos positivos está en KNN y CNN teniendo así 461 errores en KNN y 760 en CNN.

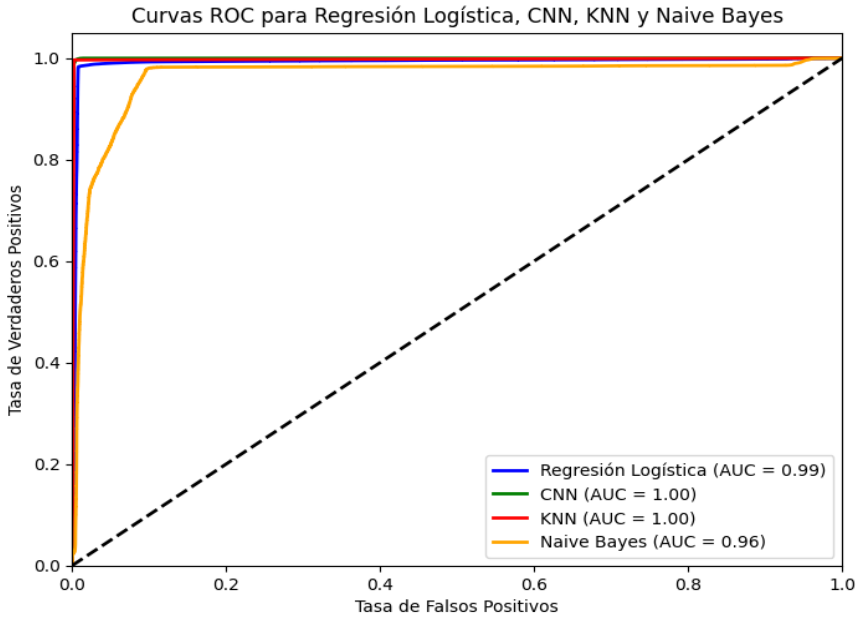

Figura 18. Comparación de curvas ROC en clasificación de tramas normales y Generic. Fuente: El Autor.

En la Figura 18, se verifican mismas tendencias de las gráficas de Regresión Logística, CNN Y KNN mientras que una pequeña desviación de Naive Bayes. La curva ROC con un valor AUC de 0,96 demuestra las prometedoras características de rendimiento del modelo. La curva ROC muestra que el modelo tiene una gran capacidad para distinguir entre grupos positivos y negativos porque el área bajo la curva (AUC) es cercana a 1. Estos hallazgos demuestran que el modelo puede predecir resultados con altos niveles de precisión.

Evaluación en ataque del tipo Exploits

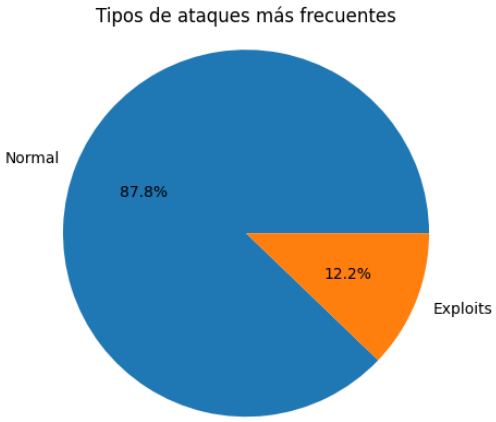

Figura 19. Porcentaje de tramas normales y tramas del tipo Exploits (maliciosas). Fuente: El Autor.

Los tipos de ataques Exploits, son códigos que se consideran maliciosos que se aprovechan de fallos en los sistemas informáticos [24]. En la Figura 19, se tiene su porcentaje de distribución, siendo el 2do con más presencia en el conjunto de datos, para este caso se puede decir que la mayoría de las tramas son normales y solo el 12.2% las maliciosas.

Tabla 17. Tabla con métricas de evaluación y consumo de recursos en clasificación de Exploits. Fuente: El Autor.

	Exactitud	**Precisión**	**Tiempo (s)**	**RAM (bytes)**	**CPU (%)**
RL	0.9754	0.8938	1.4041	5976064	11.54
CNN	0.9892	0.9440	43.7596	25573785	23.92
KNN	0.9926	0.9700	8.4387	9019399	21.50
NB	0.9419	0.7481	0.1086	50790	18.92

El modelo "CNN" muestra una alta exactitud de 0,9892 y una precisión de 0,9440. Estos datos son observados en la Tabla 17. Sin embargo, en algunas circunstancias en las que el rendimiento es crucial, el tiempo de procesamiento de 43,75 segundos y los recursos utilizables 25.573.785 bytes de RAM y 23,2 % de CPU pueden verse como limitaciones.

El modelo "KNN", por otro lado, presenta una alta exactitud de 0,9926 y una sorprendente precisión de 0,9700. Para aquellos que quieran equilibrar precisión y eficiencia, este modelo es una buena opción debido a su tiempo de procesamiento de 8,44 segundos y uso de recursos 9.019.399 bytes de RAM y procesador de 21,5 %.

Tabla 18. Matrices de confusión al clasificar tramas normales y Exploits. Fuente: El Autor.

Matriz de confusión RL		Real		**Matriz de confusión CNN**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	12116	1388	Predicción	**P**	13084	775
	N	1269	94970		**N**	301	95583
Matriz de confusión KNN		Real		**Matriz de confusión NB**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	12929	432	Predicción	**P**	10543	3228
	N	456	95926		**N**	2842	93130

La matriz de confusión de Naive Bayes visualizada en la Tabla 18, tiene más alto valor de cantidad de errores en comparación con el resto de los modelos. CNN y KNN poseen la más baja cantidad de errores siendo así 888 registros mal clasificados con el modelo KNN y 1075 registros mal clasificados de CNN.

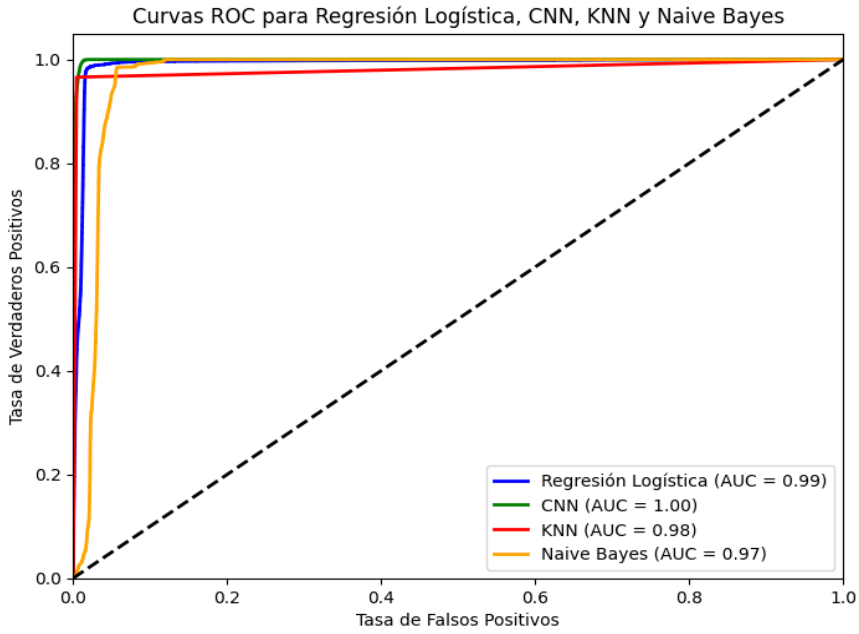

Figura 20. Comparación de curvas ROC en clasificación de tramas normales y Exploits. Fuente: El Autor.

Las curvas ROC presentadas en la Figura 20, están con excelentes tendencias de clasificación para detectar verdaderos positivos presentándoles como mejores opciones Regresión Logística y CNN y con ciertas desviaciones de estas tendencias a KNN y Naive Bayes.

Evaluación en ataque del tipo Fuzzers

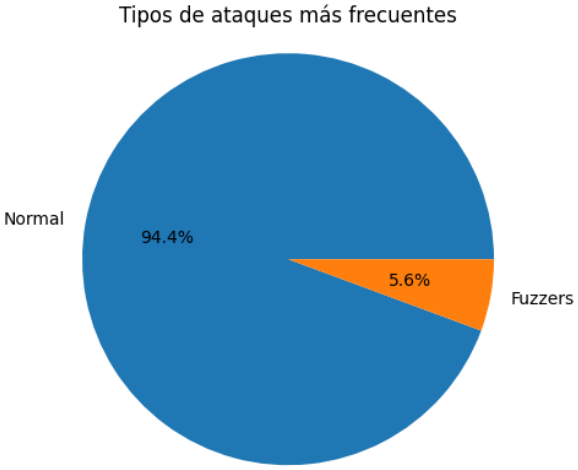

Figura 21. Porcentaje de tramas normales y tramas del tipo Fuzzers (maliciosas). Fuente: El Autor.

Los ataques de tipo Fuzzers son pruebas automatizadas que envían o introducen a las entradas, datos inválidos aleatorios con la finalidad de producir algún error [25]. En la Figura 21, se tiene a este pequeño segmento de tramas maliciosas representadas solo con el 5.6% del total del conjunto de datos.

Tabla 19. Tabla con métricas de evaluación y consumo de recursos. Fuente: El Autor.

	Exactitud	**Precisión**	**Tiempo (s)**	**RAM (bytes)**	**CPU (%)**
RL	0.9839	0.7976	0.8030	1708851	15.32
CNN	0.9862	0.8144	40.0283	33155481	22.74
KNN	0.9840	0.8552	5.4378	6614220	21.79
NB	0.9715	0.6735	0.0932	3355443	15.36

Los valores de la Tabla 19 nos indica que el modelo "KNN" tiene una precisión de 0,8552 y una exactitud de 0,9840. En comparación con otros modelos, este modelo tiene un fuerte equilibrio entre precisión y eficiencia con un tiempo de procesamiento de 5,44 segundos y un consumo de recursos de 6.614.220 bytes de RAM y 21,79% en CPU.

En la muestra "NB" se muestran una exactitud de 0,9715 y una precisión de 0,6735. Aunque su velocidad no es tan rápida como la de otros modelos, su tiempo de respuesta extremadamente rápido 0,0932 segundos y su escaso uso de recursos 3,355,443 bytes de RAM y 15,36 % de CPU aún pueden ser útiles en escenarios donde el rendimiento es crucial.

Tabla 20. Matrices de confusión resultantes de la clasificación de tramas normales y Fuzzers. Fuente: El Autor.

Matriz de confusión RL		Real		**Matriz de confusión CNN**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	5458	1363	Predicción	**P**	5638	1295
	N	234	95089		**N**	54	95157
Matriz de confusión KNN		Real		**Matriz de confusión NB**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	4866	836	Predicción	**P**	5465	2645
	N	826	95616		**N**	227	93807

En la Tabla 20, se tiene las 4 matrices de confusión destacando a la que posee menos cantidad de registros mal clasificados. Esta matriz CNN posee tan solo 1349 errores de clasificación y de forma contraria a la matriz de confusión NB que posee 2872 registros mal clasificados de un total del conjunto de datos con 102144 registros.

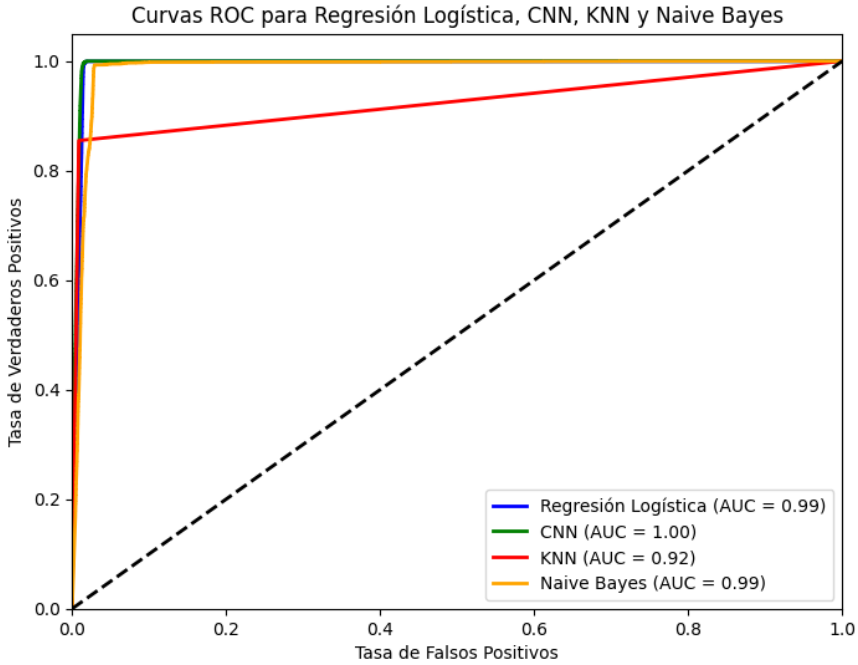

Figura 22. Comparación de curvas ROC en clasificación de tramas normales y Fuzzers. Fuente: El Autor.

Las curvas ROC presentadas en la Figura 22, están con excelentes tendencias de clasificación para detectar verdaderos positivos presentándoles como mejores opciones Regresión Logística y CNN y con ciertas desviaciones de estas tendencias a KNN y Naive Bayes. De forma general todos los modelos tienen buena oportunidad de clasificación de tramas buenas y maliciosas.

Evaluación en ataque del tipo DoS

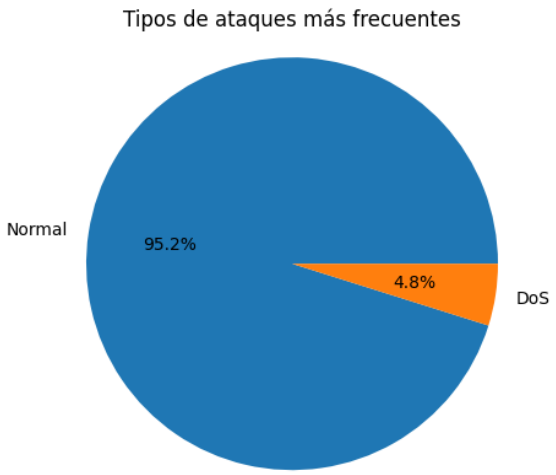

Figura 23. Porcentaje de tramas normales y tramas del tipo DoS (maliciosas). Fuente: El Autor.

Los ataques de denegación de servicio (DoS) buscan bloquear o prohibir el acceso a los recursos de una red o equipos, saturan los recursos con múltiples peticiones [23]. Este último análisis que se presenta en la Figura 23. Tan solo se tiene el 4.8% de tramas maliciosas del total del conjunto de datos.

Tabla 21. Tabla con métricas de evaluación y consumo de recursos. Fuente: El Autor.

	Exactitud	**Precisión**	**Tiempo (s)**	**RAM (bytes)**	**CPU (%)**
RL	0.9861	0.8858	1.0488	158105	17.04
CNN	0.9919	0.9176	43.2967	17949491	26.9199
KNN	0.9960	0.9606	5.6645	131891	28.98
NB	0.9808	0.7427	0.0867	154828	13.16

El modelo "RL" tiene una precisión de 0,8858 y una exactitud de 0,9861. Logra un equilibrio entre precisión y eficiencia que puede ser apropiado para una variedad de aplicaciones con un tiempo de ejecución de 1,04 segundos, utilizando 158.105 bytes de RAM y 17,04 por ciento de CPU.

El modelo CNN muestra una sólida exactitud de 0,9919 y una sorprendente precisión de 0,9176. Sin embargo, en circunstancias en las que el rendimiento es crucial, puede ser importante tener en cuenta el tiempo de procesamiento de 43.3 segundos y el uso de recursos 17.949.491 bytes de RAM y 26.92% CPU.

El modelo KNN muestra una exactitud de 0,9960 y una precisión de 0,9606. Su capacidad para realizar tareas extremadamente precisas queda demostrada por su tiempo de procesamiento de 5,6645 segundos y un uso de recursos de 131.891 bytes de RAM y 28,98 de CPU.

Tabla 22. Matrices de confusión resultantes de la clasificación de tramas normales y DoS. Fuente: El Autor.

Matriz de confusión RL		Real		**Matriz de confusión CNN**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	3971	503	Predicción	**P**	4592	386
	N	913	95904		**N**	292	96021
Matriz de confusión KNN		Real		**Matriz de confusión NB**		Real	
		P	**N**			**P**	**N**
Predicción	**P**	4679	230	Predicción	**P**	4494	1600
	N	205	96177		**N**	390	94807

Las matrices de confusión mostradas en la Tabla 22, poseen cantidades de registros para unas más favorables que para otras, así tenemos a la matriz de confusión con 100856 aciertos de clasificación y corresponde al modelo KNN consecuentemente de destacar en poseer la menor cantidad de falsos positivos y negativos. Y la matriz con más errores de clasificación es la del modelo Naive Bayes con 1990 tramas mal clasificadas.

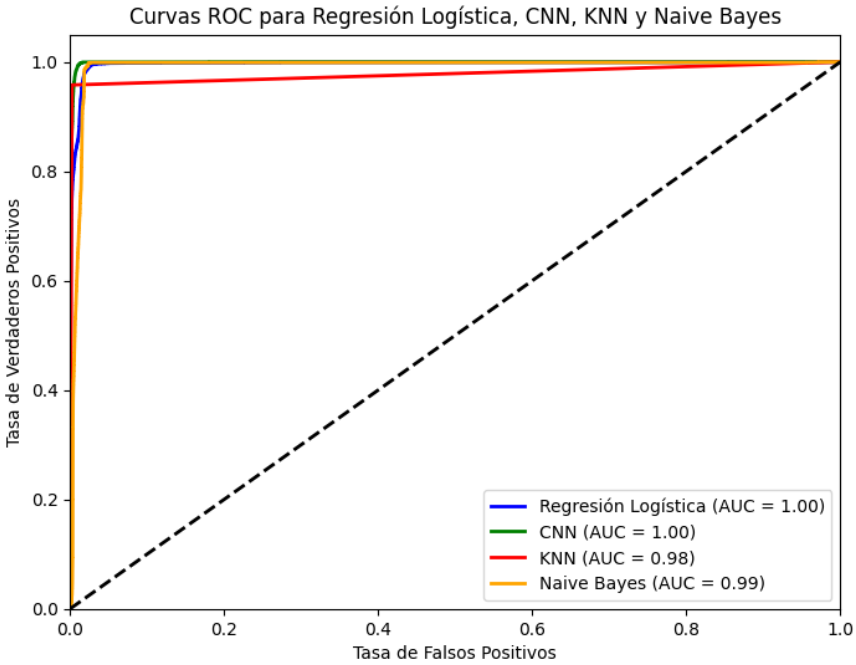

Figura 24. Comparación de curvas ROC en clasificación de tramas normales y DoS. Fuente: El Autor.

En el análisis de la Figura 24, las curvas ROC con AUC mayor a 0.98 muestran resultados prometedores. Muestran que los modelos tienen una gran capacidad para distinguir entre grupos positivos y negativos, con una alta probabilidad de clasificar correctamente los eventos a pesar de que los 3 últimos tipos de ataques no se encontrarían balanceados ya que los porcentajes de tramas maliciosas son pequeños. Para este caso todas las gráficas están en la misma tendencia con la curvatura casi a 90 º.

La estructuración completa del código de los algoritmos, preprocesamiento y graficas se encuentra en los archivos almacenados en la nube de Drive que está en el siguiente enlace compartido al público: ANEXO 1

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Comparación de métricas de los modelos de Machine Learning

En el contexto IoT ya antes mencionado, se decía que el poder computacional de dispositivos IoT es limitado por aquella razón en la evaluación del capítulo anterior se obtuvo el resultado que aplicando análisis de componentes principales se obtiene menos consumo de recursos, además los tiempos de respuesta mejoran y con respecto a los valores de precisión y exactitud de cada modelo estarán a la par del análisis de la correcta elección de cuantos componentes representan la mayor cantidad de características del conjunto de datos inicial.

En conclusión, el modelo de Machine Learning más favorecido por uso de PCA en el conjunto de datos es KNN ya que los tiempos de entrenamiento y predicción se reduce drásticamente en comparación de utilizar con las 47 etiquetas de entrada del conjunto de datos original comparando con el resto de los algoritmos que ocuparon mucho menos tiempo al evaluar sin aplicar PCA.

En conclusión, del análisis sobre los 4 tipos de ataques más frecuentes, permitió tener una idea más clara sobre cuál de los 4 algoritmos de Machine Learning sobresale en términos de precisión, exactitud y consumo de recursos moderado para cada tipo de ataque. Siendo así por el ataque de tipo Generic aquí sobresale el algoritmo de Regresión Logística, en el ataque de tipo Exploits sobresale el algoritmo de KNN, en el ataque de tipo Fuzzers sobresale el algoritmo KNN y finalmente en el ataque de tipo DoS sobresale el algoritmo Regresión Logística.

Como conclusión del algoritmo más sencillo, rápido y de bajo consumo es Naive Bayes, pero su desventaja radica en que sacrifica sus valores de precisión y exactitud.

A partir del análisis del capítulo 2 se evaluó cada una de las métricas de precisión, exactitud , tiempos de ejecución, uso de recursos. Se obtuvieron resultados por el análisis con 10 PCA esto permitió reunir valores que se los puede visualizar en la nueva Tabla 23, con la evaluación de las mismas métricas, pero ya adjuntándose a cada uno de los modelos de machine Learning evaluados.

Tabla 23. Comparación de métricas de cada modelo evaluado utilizando 10 PCA.

Métricas	**RL**	**CNN**	**KNN**	**NB**
Exactitud	0.9751	0.9914	0.9894	0.8923

Precisión	0.9640	0.9844	0.9885	0.8236
Tiempo entrenamiento (s)	0.6390	77.9458	8.67	0.1080
Uso de RAM (bytes)	7955251	32084787	12674662	14297497
Uso de CPU %	14.58	23.30	14.88	9.72

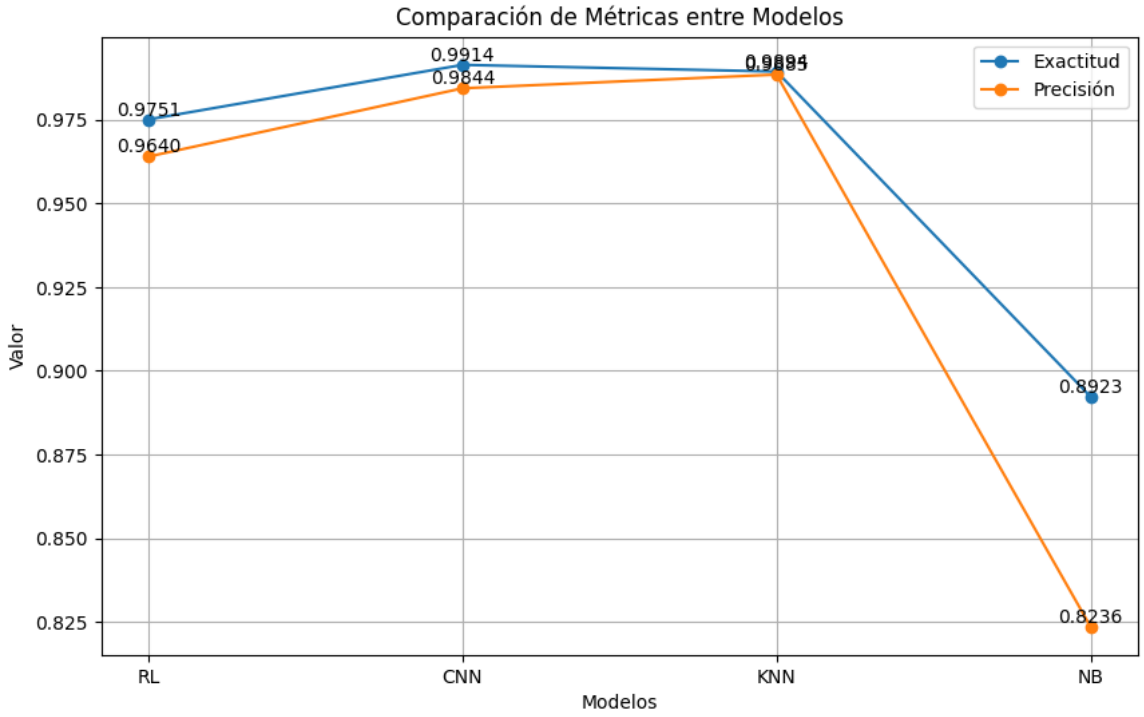

Figura 25. Comparación de métricas de exactitud y precisión entre los modelos de Machine Learning. Fuente: El Autor.

En la Figura 21, se puede observar los valores de exactitud y la precisión tomándose en cuenta juntos, los modelos CNN y KNN tienen los valores más altos para ambas métricas. Esto sugiere que la precisión de la identificación de clases y la reducción de falsos positivos están bien equilibradas en estos dos modelos.

En esta comparación, el modelo KNN podría considerarse superior a CNN debido a su ligera ventaja en ambas métricas. Sin embargo, hay factores adicionales a tener en cuenta, incluido el uso de recursos, los tiempos de entrenamiento y predicción y el tiempo, que pueden ser cruciales en los entornos de IoT. Por lo tanto, se deben seguir evaluando el resto de las métricas.

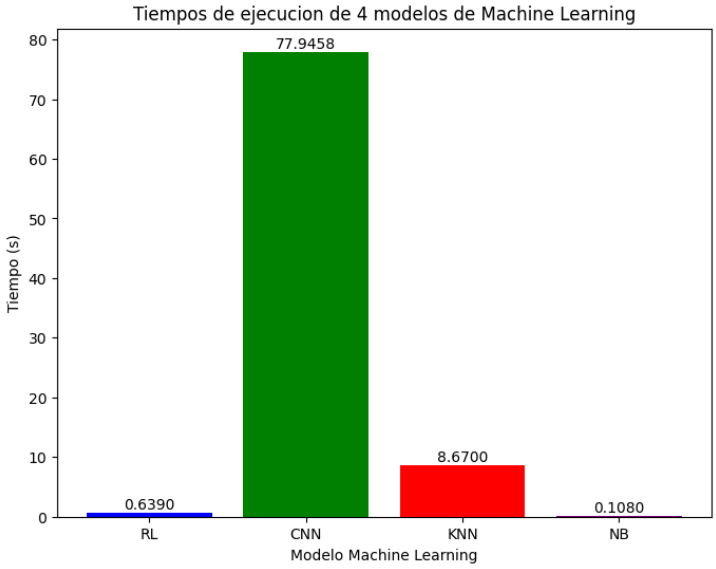

Figura 26. Comparación de tiempos de ejecución. Fuente: El Autor.

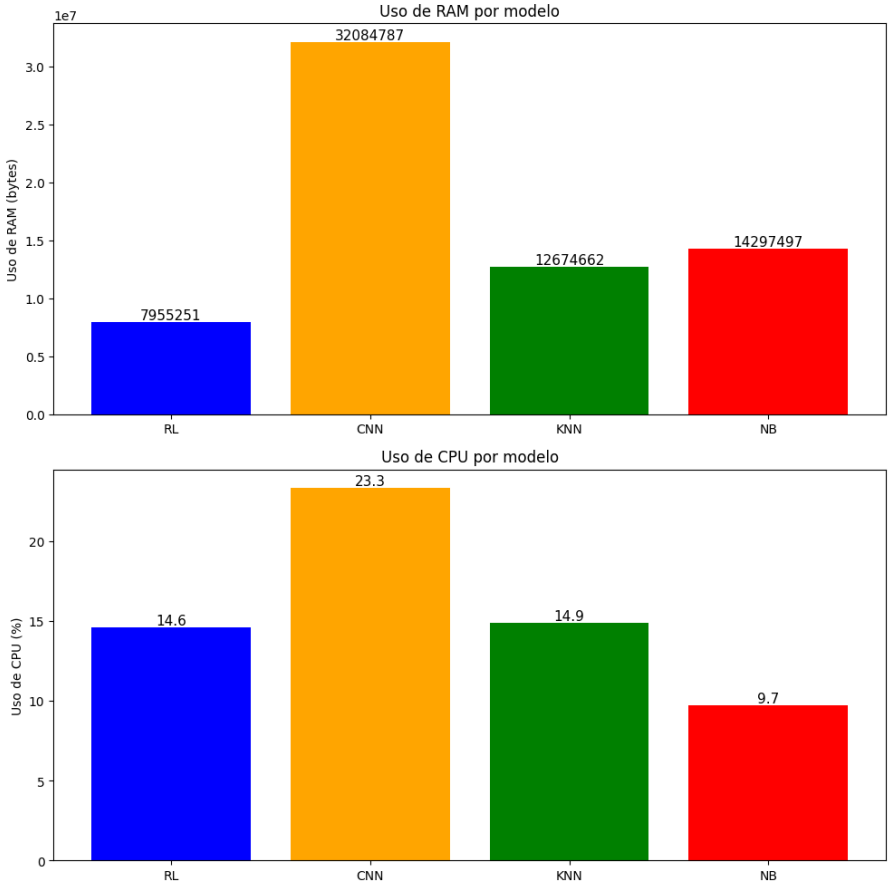

Figura 27. Comparaciones en el uso de RAM y el uso de CPU

Al observar las métricas de la Figura 26, se identifican diferencias en los comportamientos de tiempo. El modelo Naive Bayes sobresale por su tiempo de ejecución excepcionalmente bajo, lo que sugiere una rápida convergencia durante el proceso de aprendizaje y prueba. En cambio, el que requiere más tiempo en entrenamiento es el modelo Redes Neuronales Convolucionales, en comparación con el modelo de Regresión Logística y KNN, lo que podría indicar requerir un mayor poder computacional en las operaciones de entrenamiento y pruebas.

En la Figura 26, el modelo RL muestra tiempos de ejecución notablemente más rápidos que KNN, lo que lo convierte en una opción favorable en aplicaciones que requieren respuestas rápidas.

Al examinar las métricas de la Figura 27, se obtiene una visión de sus demandas en términos de eficiencia computacional. El modelo NB se destaca por su uso eficiente de CPU, lo que sugiere una gestión de bajo consumo en comparación con los demás modelos. En términos de carga de la CPU Y RAM, CNN muestran un uso elevado en comparación con el modelo RL y KNN, esto implica que la CNN requieren una mayor potencia de procesamiento para llevar a cabo sus operaciones de clasificación y predicción. Además, los modelos RL y KNN muestra una carga de CPU razonable y casi similares.

Para el cumplimiento del objetivo general planteado en el proyecto se empezará a descartar por eliminación los modelos de aprendizaje automático que no cumplan con los requisitos para la ejecución de entornos IoT donde es primordial que sea razonable el consumo de recursos y clasificación de tramas buenas y maliciosas.

Desde el primer análisis tenemos que eliminar CNN porque es el que más recursos consume y tiene razón por que es un algoritmo de cierto grado de complejidad para el tratamiento de imágenes.

Segundo análisis consta de que tan preciso o exacto es el modelo para clasificar tramas buenas o malas y el modelo NB podría tener dificultades en clasificar ciertas instancias por lo que eliminaríamos como opción de modelo para análisis de tráfico en redes IoT.

Tercer análisis comparando el uso de recursos entre KNN y RL y además de la precisión y exactitud de estos modelos, se tiene que RL y KNN utiliza una cantidad razonable de recursos, pero KNN nos provee de una mejor precisión y exactitud al momento de clasificar lo que nos asegura una buena fiabilidad del modelo al identificar tramas buenas

y maliciosas. Por estas razones eliminamos a RL como primera opción de clasificación de tramas en tráfico en redes IoT.

Tomando nota como observación general de estos dos últimos modelos es importante destacar que RL es más rápido entrenándose y prediciendo con respecto a KNN. Por lo que seleccionaría como una segunda opción a RL en caso de requerir velocidad de entrenamiento, predicción y consumo de recursos razonable.

3.2 Conclusiones

Se obtuvo con éxito un conjunto de datos integro y ordenado. Esto se logró abordando datos atípicos, transformación de datos cualitativos, faltantes, resultando en la coherencia de los datos. Para consecuentemente utilizar en la evaluación de los modelos de machine Learning y esto se demostró obteniendo valores de precisión y exactitud superiores a 0.87.

Se evito sesgos gracias a la reorganización de los datos y aplicación de técnica de balanceo y normalización asegurando así que con este conjunto de datos los modelos de machine Learning se entrenen y evalúen de forma equilibrada.

En la evaluación de los modelos de machine Learning donde se varia su cantidad de variables independientes se logró identificar que al aplicar análisis de componentes principales se obtiene un significativo ahorro de recursos de procesamiento y almacenamiento temporal.

Con el entrenamiento y pruebas se logró discernir cuales modelos tienen mejores valores de precisión y exactitud , valores en tiempos de ejecución, valores en el uso de recursos para sistemas IoT.

Para los sistemas IoT que poseen bajo poder computacional y se mantenga el objetivo de identificar tramas buenas y maliciosa con alta precisión y exactitud se determinó a través de un análisis riguroso que el modelo *k-Nearest Neighbors* cumple con estos requerimientos.

Con el análisis y estudio de cómo fue creado el conjunto de datos, se identificó los pasos claves para poder recrear un futuro análisis en nuestros propios entornos simulados o reales.

3.3 Recomendaciones

Cuando se esté en la parte de normalización del conjunto de datos se debe separar de la salida ya que si la salida es binaria y se normaliza resultara con errores los modelos en momento de entrenarlos.

La utilización de los datos debe hacerse en porcentajes que sean manejables para las características de la PC y del IDE utilizado.

En la medición de uso de recursos se lo debe aplicar cuando el modelo se lo esté evaluando con validación cruzada de otra forma las mediciones serán demasiado cortas y no tendrá sentidos sus valores.

Es importante tener en cuenta que el balanceo de datos debe realizarse cuidadosamente para evitar la introducción de sesgos en el conjunto de datos. Además, no siempre es necesario equilibrar los datos, ya que algunos modelos de Machine Learning pueden manejar datos desequilibrados de manera efectiva.

La búsqueda de información no debe limitarse solo por medios electrónicos, en las bibliotecas puede existir información que nos ayude de forma precisa.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] A. Polania, "Evaluación de modelos de Machine", Tesis, Ingeniería de Sistemas y Computación, UA , Bogota, 2021.

[2] A Mohammed, " Doubling the Number of Connected Devices in Narrow-band Internet of Things while Maintaining System Performance", *arXivLabs*, vol. 1, pp. 1, Agosto 2022.

[3] N. Moustafa, (2021,Junio 2). The UNSW-NB15 Dataset (1era edicion), [En línea]. Available: https://research.unsw.edu.au/projects/unsw-nb15-dataset.

[4] N. Moustafa, " Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic ", Ph.D. dissertation , Tecnologias de la informacion, UNSW, Australia, Bedegal, 2017.

[5] A. Azevedo,"KDD, semma and CRISP-DM: A parallel overview", IADIS European Conference on Data Mining 2008, Amsterdam, pp 1-5, Julio 2008.

[6] J. Roman, " Industria 4.0: la transformación digital de la industria", Conferencia de directores y Decanos de Ingeniería Informática, Madrid, Valencia, pp 1-10, 2016.

[7] A. L. Muhammad, " Security Analysis of Network Anomalies Mitigation Schemes in IoT Networks", *IEEE Xplore*, vol. 8, pp. 2 - 8, Febrero 2020.

[8] G. Salvador, "Big Data: Preprocesamiento y calidad de datos", Soft Computing and Intelligent Information Systems, Granada, 2016.

[9] F. Herrera, (2004,Mayo 6), Preprocesamiento de Datos (1era edicion), [En línea]. Available: http://www.lsi.us.es/redmidas/IIreunion/trans/prepro.pdf.

[10] M. Pérez, (2017, Junio 13), La Gestión de Datos de Investigación (1era edicion), [En línea]. Available:https://repositorio.uam.es/bitstream/handle/10486/678601/gestion_perez_us_ 2017_2.pdf?sequence=2.

[11] C. Russo, "Tratamiento Masivo de Datos Utilizando Técnicas de Machine Learning", Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Buenos Aires, 2016.

[12] M. A. Hall, "Practical feature subset selection for machine learning", presentado en Computing and Mathematical Sciences Papers, Berlin, 1998.

[13] J. Zamora, "Comparativa y análisis de algoritmos de aprendizaje automático para la predicción del tipo predominante de cubierta arbórea", Trabajo Fin Master, Ingenieria Informatica, UCM,Madrid,2018.

[14] S. Indolia, A. Kumar y P. Asopa, " Conceptual Understanding of Convolutional Neural Network-A Deep Learning Approach", *ELSEVIER*, vol. 132, pp. 1-10, 2018.

[15] E. Garcia, "Introducción a las redes neuronales de convolución. Aplicación a la visión por ordenador", Trabajo fin de grado, Matematicas, U. Z, Zaragoza, 2019.

[16] S. Tahsien, H. Karimipour y P. Spachos, "Machine learning based solutions for security of Internet of Things (IoT): A survey", *ELSEVIER*, vol. 161, pp. 1-22, 2020.

[17] G. I Webb, M. University (2017), " Naïve Bayes ", [Online], Available: https://www.researchgate.net/profile/Geoffrey-Webb/publication/306313918_Naive_Bayes/links/5cab15724585157bd32a75b6/Naive-Bayes.pdf

[18] C. Acosta y D Sebastian, "Detección de amenazas en una red IoT a usando técnicas de Machine Learning", Tesis, Ingeniería de Sistemas y Computación, UA , Bogota, 2021.

[19] M. Ono A. Maćkiewicz y W. Ratajczak, “Principal components analysis (PCA)”, Computers & Geosciences, vol. 19, núm. 3, pp. 303–342, 1993.

[20] K. Rajalingham, (2020),linuxhint ,[Online], Available: https://linuxhint.com/install-zeek-bro/

[21] R. Valdovinos, “Tecnicas de submuestreo,toma decisiones y analisis de diversidad en aprendizaje supervisado con sistemas multiples de clasificacion”, Tesis Doctoral,Dto. Lenjuages y sistemas informaticos, UJI, Castello de la Plana, 2006.

[22] K. Pizarro y O. Martinez, “Análisis factorial exploratorio mediante el uso de las medidas deadecuación muestral kmo y esfericidad de bartlett para determinar factores principales”, *JSR*, Vol. 5, pp 7-10, Octubre 2020.

[23] Kaspersky, (2022), kaspersky THREATS, [Online], Available: https://threats.kaspersky.com/mx/threat/Scan.Generic.TCP/

[24] J. Albors, (22/12/2022), welivesecurity by eset, [Online], Available: https://www.welivesecurity.com/la-es/2022/12/22/exploits-que-son-como-funcionan/

[25] KeepCoding, (19/05/2023), TechSchool, [Online], Available: https://acortar.link/VIqoDq

5 ANEXOS

Anexo 1:

Archivos, códigos utilizados y generados

https://drive.google.com/drive/folders/1xCLIcKyOiR-3Qk10v1wj2XQSgS-wK-qk?usp=sharing