ANÁLISIS DE INTELIGENCIA ARTIFICIAL EN LA SEGURIDAD DE CONEXIONES DE INFORMACIÓN

ARTIFICIAL INTELLIGENCE ANALYSIS IN INFORMATION CONNECTION SECURITY

Carlos Roberto Martínez Martínez [1]

RESUMEN

Este estudio evalúa la aplicación de la inteligencia artificial (IA) y el procesamiento de lenguaje natural (PLN) en la ciberseguridad, centrando su análisis en la detección de amenazas digitales para reforzar la seguridad de la información, como medio para el desarrollo sostenible organizacional. Aborda el pseudo código de los algoritmos, examinando su eficacia a través del análisis de registros de tráfico de servidores. Los resultados demuestran que modelos como DistilBert y GPT-2 sobresalen en precisión predictiva, a pesar de requerir tiempos de entrenamiento prolongados, en contraste con otros modelos como SVM, XGBoost e Isolation Forest, que, si bien son rápidos, muestran menor precisión. Este estudio sugiere un potencial significativo de estas tecnologías para anticipar y gestionar anomalías de seguridad informática, recomendando el desarrollo de prototipos de sistemas de IA para una evaluación más extensa y la implementación de sistemas de alerta temprana efectivos.

PALABRAS CLAVE: ciberseguridad, aprendizaje automático, procesamiento de lenguaje natural, algoritmos informáticos.

ABSTRACT

This study evaluates the application of artificial intelligence (AI) and natural language processing (NLP) in cybersecurity, focusing its analysis on the detection of digital threats to reinforce information security, as a means for organizational sustainable development. It addresses the pseudo code of the algorithms, examining their efficacy through the analysis of server traffic logs. The results demonstrate that models such as DistilBert and GPT-2 excel in predictive accuracy, despite requiring extended training times, in

[1] Ingeniero en Sistemas Informáticos.
Maestro en Ciencias
Maestro en Seguridad Informática.
Facultad de Ingeniería y Arquitectura, Universidad Católica de El Salvador

contrast with other models like SVM, XGBoost, and Isolation Forest, which, while fast, show less precision. This study suggests significant potential for these technologies to anticipate and manage cybersecurity anomalies, recommending the development of AI system prototypes for a more extensive evaluation and the implementation of effective early warning systems.

KEYWORDS: cybersecurity, machine learning, natural language processing, computer algorithms.

I. INTRODUCCIÓN

El presente estudio se enfoca en analizar el impacto de la inteligencia artificial (IA) y el procesamiento de lenguaje natural (PLN) en reforzar la seguridad de la información, un pilar esencial para el desarrollo sostenible. En un mundo cada vez más interconectado, la proliferación digital facilita el acceso al conocimiento y estimula el crecimiento económico, pero también expone a riesgos cibernéticos que comprometen la seguridad de datos críticos. Frente a este escenario, la investigación evalúa la eficacia de las técnicas IA y de PLN en la detección de amenazas digitales, demostrando cómo estructurar dichas tecnologías avanzadas que clave para avanzar hacia un entorno digital seguro y sostenible.

Mediante el análisis de registros de tráfico de servidores, se examinan diversas aplicaciones de IA y PLN en ciberseguridad, para la identificación de intentos de ataques, con el fin de evaluar su factibilidad para la protección de infraestructuras digitales, destacando potencial de estas tecnologías emergentes para enfrentar desafíos actuales en seguridad de la información, en consonancia con los Objetivos de Desarrollo Sostenible (ODS) (Pérez, 2020), particularmente los relacionados con la industria, innovación e infraestructura (ODS #9), y la promoción de sociedades pacíficas e inclusivas (ODS #16). Al investigar cómo la IA y el PLN pueden fortalecer las defensas cibernéticas, este estudio busca aportar a la construcción de sociedades en las que la información de las personas pueda estar asegurada, promoviendo así el desarrollo sostenible a través de la innovación tecnológica y la resiliencia de la infraestructura.

II. MARCO TEÓRICO

La seguridad de la información se ha convertido en una preocupación primordial en la era digital, impulsando la necesidad de métodos avanzados de detección y prevención de amenazas. La IA, específicamente a través del aprendizaje automático, proporciona mecanismos para clasificar y analizar comportamientos dentro de los sistemas de información, empleando algoritmos como la regresión logística, y clasificadores basados en árboles. La regresión logística, un método estadístico, se utiliza para predecir la

probabilidad de un evento categorizando los datos en grupos, como acciones maliciosas o benignas, basándose en una función que asigna pesos a diferentes características de los datos que se usan para entrenar los modelos. Este proceso permite identificar patrones que indican comportamientos potencialmente dañinos. Por otro lado, los clasificadores de árboles, operan construyendo múltiples nodos de decisión para aislar casos anómalos dentro de los datos. Este método es especialmente eficaz para detectar nuevas amenazas, ya que se centra en la rareza de las observaciones en lugar de en la conformidad con un patrón de ataque conocido.

El PLN, complementario a la IA, se dedica al análisis del lenguaje humano, permitiendo la interpretación automática de textos para identificar patrones de respuesta a interrogantes. Mediante modelos como GPT, el PLN puede generar y comprender texto con un alto grado de coherencia y relevancia contextual, lo cual puede ser aprovechado para para analizar comunicaciones cibernéticas y detectar tácticas de ingeniería social como el *phishing*, donde el contenido textual es manipulado para engañar al receptor.

La integración de IA y PLN en la ciberseguridad transforma la capacidad de los sistemas para procesar y analizar grandes volúmenes de datos (Sarker *et al.*, 2021; Georgescu, 2020), identificando amenazas con una precisión y velocidad superiores a las de los métodos tradicionales. Estas tecnologías permiten una clasificación eficiente de las actividades y eventos, facilitando la detección temprana de acciones sospechosas y la implementación de medidas preventivas o correctivas. Además, la capacidad de aprendizaje y adaptación de estos sistemas asegura una mejora continua en la identificación de amenazas, ajustándose a las evoluciones en las tácticas de los ciberdelincuentes. Este enfoque tecnológico no solo optimiza la detección y respuesta ante las amenazas, sino que también alivia la carga operativa sobre los equipos humanos a cargo de la ciberseguridad, permitiéndoles concentrar sus esfuerzos en la mitigación de riesgos más complejos. Así, la aplicación de IA y PLN se erige como un componente esencial en la estrategia de seguridad de la información.

En el ámbito del análisis de datos y la inteligencia artificial, el lenguaje de programación Python se ha consolidado como la plataforma predominante, distinguiéndose por su sintaxis intuitiva y legible, además de que cuenta con un robusto sistema de librerías especializadas, lo cual a su vez facilita el manejo y el procesamiento de grandes volúmenes de datos. Entre estos recursos, pueden destacarse los siguientes:

Scikit-learn

Conocida también como Sklearn, es un pilar en el aprendizaje automático dentro de Python, destacándose por su rica colección de algoritmos tanto de aprendizaje supervisado como no supervisado. Estos algoritmos abarcan desde modelos lineales hasta árboles de decisión, pasando por métodos de ensamble como Random Forests y Gradient Boosting. Random Forest es un algoritmo de aprendizaje supervisado que utiliza el método de ensamble para la clasificación y la regresión (Shah, 2020). Se basa en la construcción de múltiples árboles de decisión durante el entrenamiento y produce la salida por votación mayoritaria para la clasificación y promediando el resultado para la regresión de los datos de entrada. Cada árbol se entrena con una muestra aleatoria de dichos datos, lo que hace que el modelo sea robusto ante el sobreajuste. Gradient Boosting por su parte, funciona construyendo secuencialmente modelos y corrigiendo errores de los modelos anteriores en cada paso iterativo mediante el uso del gradiente del error. Esta técnica optimiza una función de pérdida agregando árboles que corrigen los residuos del ensamble construido hasta el momento, lo que resulta en una mejora progresiva del rendimiento del modelo.

Numpy

Esta librería proporciona soporte para arreglos y matrices multidimensionales de gran tamaño, junto con una extensa colección de funciones matemáticas de alto nivel para el procesamiento y análisis de datos. Esto facilita operaciones avanzadas de álgebra lineal, transformadas de Fourier, y generación de números aleatorios, muy usados para el manejo de datos numéricos y el análisis estadístico en proyectos de ciencia de datos e IA.

PyTorch

Emergió como una solución destacada para el aprendizaje profundo, ofreciendo un enfoque basado en tensores con aceleración GPU y un modelo de construcción de redes neuronales altamente interactivo y dinámico. La capacidad de PyTorch para permitir modificaciones en tiempo real del grafo computacional lo hace excepcionalmente adecuado para la investigación y el desarrollo experimental en IA (Du *et. al.*, 2023), posibilitando una iteración rápida y una experimentación flexible con arquitecturas de modelos complejos.

Transformers

Desarrollada por Hugging Face, revoluciona el procesamiento de lenguaje natural al proporcionar acceso a modelos preentrenados como BERT, GPT-2, y T5, que han establecido nuevos estándares en tareas de PLN gracias a su capacidad para comprender y generar lenguaje humano con un nivel de precisión sin precedentes (Rothman, 2021). Esta librería permite a los desarrolladores implementar soluciones de PLN avanzadas sin la necesidad de entrenar modelos desde cero, acelerando significativamente el desarrollo de

aplicaciones que requieren comprensión del lenguaje natural. Adicionalmente, la integración con la librería de OpenAI ofrece acceso directo a modelos de inteligencia artificial avanzados como GPT-3, abriendo un abanico de posibilidades para la generación de texto y la automatización de tareas basadas en lenguaje, desde chatbots hasta sistemas complejos de respuesta automática.

III. DESARROLLO

Dado que el presente estudio se trata de enfocar las mencionadas tecnologías de IA para el análisis de seguridad en de transmisión de información a través de conexiones cibernéticas, para realizar un estudio de aplicación y comparación entre algoritmos, se utilizaron datos generados por un Sistema de Detección de Intrusos o en Inglés, IDS (Intrusion Detection System). Este servicio genera logs que contienen información detallada sobre las actividades de red en un sistema específico (Kim *et al.* 2020). Se capturó la dirección IP desde la cual se originaron actividades de conexión hacia el sistema evaluado, la acción que llevó a cabo dicha dirección IP, el puerto al que intentó acceder, el protocolo o servicio asociado a ese puerto, un sello temporal exacto de cuándo tuvo lugar la actividad y, en algunos casos, detalles adicionales como la cantidad de datos transferidos o la rapidez con que se realizaron ciertas acciones. A continuación, se ofrece un extracto de dicho log para fines demostrativos:

1: 203.0.113.40 connected to port 139 using NetBIOS at 2023-09-18 12:55:10.10
2: 192.168.1.10 accessed port 22 using SSH at 2023-09-18 12:10:05.
3: 10.0.0.5 accessed port 3389 using RDP at 2023-09-18 12:12:20.
4: 203.0.113.26 had 15 login attempts on port 22 using SSH within 2 minutes at 2023-09-18 12:15:10.
5: 198.51.100.17 was blocked trying port 445 using SMB at 2023-09-18 12:20:45.
6: 192.0.2.146 accessed port 80 using HTTP and downloaded 500MB data at 2023-09-18 12:30:30.
7: 203.0.113.29 tried accessing non-standard port 7878 at 2023-09-18 12:35:40.
8: 203.0.113.30 made 50 requests to port 80 within 1 minute at 2023-09-18 12:45:00.
9: 192.168.1.20 accessed port 21 using FTP at 2023-09-18 12:47:30.
10: 10.0.0.8 made an unsuccessful attempt to access port 3306 (MySQL) at 2023-09-18 12:50:00.

Los administradores y profesionales de seguridad utilizan esos tipos de datos para detectar y responder a actividades sospechosas o maliciosas en tiempo real o realizar análisis de informática forense después de los hechos. Además, en situaciones donde se requiere una investigación detallada, como después de un incidente de seguridad, este tipo registros son indispensables para entender cómo y cuándo ocurrió un evento particular. Este log de ejemplo se destaca por incluir sellos temporales precisos y ofrecer detalles más granulares sobre cada actividad cibernética, como intentos de inicio de sesión específicos, cantidades exactas de datos transferidos y detalles sobre la rapidez de ciertas acciones. Mediante este banco de

información, el presente estudio evaluó los siguientes algoritmos de IA para mejorar la alerta temprana, utilizando modelos de ML y NLP:

Bosques de aislamiento (Isolation Forest)

Es un algoritmo eficaz para la detección de anomalías en colecciones extensas de datos, basado en la premisa de que las anomalías son menos frecuentes y distintas de los valores producidos por acontecimientos normales. Opera aislando observaciones de forma aleatoria (Carletti *et al.*, 2023; Xu *et al.*, 2023): en cada iteración, selecciona una característica al azar y luego divide los datos con un valor promedio comprendido entre los valores máximos y mínimos de esa característica. Este proceso se repite recursivamente, generando una estructura de árbol para cada muestra. Las anomalías, al ser diferentes, tienden a aislarse más rápidamente, lo que significa que los caminos hacia estas observaciones en el árbol serán más cortos. El algoritmo construye muchos de estos árboles para formar un "bosque" y utiliza la longitud promedio de los caminos en estos árboles para determinar la normalidad o anormalidad de las observaciones. Un camino más corto sugiere una mayor probabilidad de ser una anomalía. Este enfoque permite a Isolation Forest manejar eficazmente conjuntos de datos grandes y de alta dimensión, destacando por su rapidez y eficiencia en la identificación de puntos atípicos.

Los pasos de pseudo-código para implementar este algoritmo en el contexto propuesto de análisis de ciberseguridad, son:

```
INICIO
    # importar librerías necesarias
    IMPORTAR pandas, sklearn
    # leer datos de servidor en archivo CSV
    log_data ⇐ CARGAR 'web_server_log.csv'
    # Filtrar datos con base en criterios de amenaza
    filtered_data ⇐ SELECCIONAR log_data
    # Extraer características relevantes del modelo
    X ⇐ EXTRACCIÓN DE ['request_size', 'response_time']
    # Creación y entrenamiento del modelo
    clf ⇐ IsolationForest(contaminación=0.05)   clf.ENTRENAR(X)
    # Predecir anomalías en los datos con base en el modelo
    anomalies ⇐ clf.PREDECIR(X)
    # Reportar al usuario las anomalías encontradas
    PARA CADA fila EN anomalies HACER
```

IMPRIMIR(fila)

FIN

La salida proporcionada por la operación del script anterior, con base a la detección de anomalías en el archivo de log del IDS, es como se muestra en la figura 1.

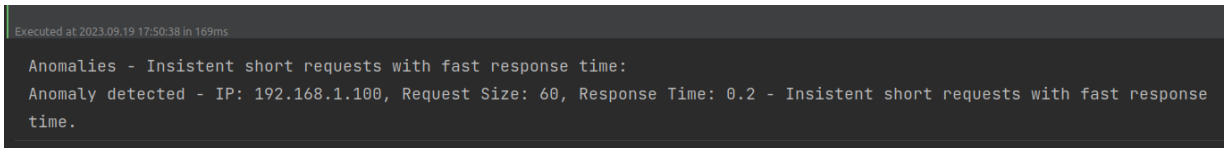

Figura 1. Salida de consola de identificador por clasificador de árbol

XGBoost

Su nombre significa eXtreme Gradient Boosting. Es un algoritmo de aprendizaje automático que sigue una estrategia de conjunto, mejorando la precisión de las predicciones a través de la adición secuencial de árboles de decisión (Raghunath *et al.*, 2022; Yang *et al.*, 2022). Inicia con una muestra simple de datos y, paso a paso, añade nuevos árboles que se enfocan en corregir los errores cometidos por los cálculos resultantes en árboles anteriores. En cada iteración, evalúa qué tan bien está realizando las predicciones mediante una función objetivo, que no solo mide la exactitud sino también incluye un componente para controlar la complejidad del modelo, ayudando a prevenir el sobreajuste. Para cada nuevo árbol que se añade, XGBoost busca el que mejor mejore el rendimiento del conjunto hasta el momento, teniendo en cuenta tanto la corrección de errores previos como la simplicidad del modelo. Este proceso se repite hasta alcanzar un número predeterminado de árboles o hasta que la mejora en la precisión se estabilice. Además, XGBoost maneja eficientemente los datos faltantes y permite el uso de regularización para hacer el modelo más generalizable. La salida proporcionada por la operación del script anterior, con base a la detección de anomalías en el archivo de log, es como se muestra en la Fig. 2. Los pasos de pseudo código para implementar este algoritmo en el contexto propuesto de análisis de ciberseguridad, son:

INICIO

 # importar librerías necesarias

IMPORTAR pandas, sklearn, xgboost

 # leer datos de servidor en archivo CSV

data ⇐ CARGAR 'web_server_log.csv'

 # categorizar datos de características y etiquetas

X ⇐ data['request_size', 'response_time']

```
y ⇐ data['anomaly']
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test ⇐
train_test_split(X, y, test_size=0.2, random_state=42)
# Entrenar el modelo con los datos
model ⇐ XGBClassifier.ENTRENAR(X_train, y_train)
# Predecir anomalías en los datos con base en el modelo
anomalies ⇐ model.PREDECIR(X)
# Reportar al usuario las anomalías encontradas
PARA CADA fila EN anomalies HACER
        IMPRIMIR(fila)
FIN
```

```
Dato de Prueba 1: [  1   1   1   1 200]
Es un ataque (Real): No
Es un ataque (Predicción): No
--------------------------------------------------
Dato de Prueba 2: [  1   2   1   2 200]
Es un ataque (Real): No
Es un ataque (Predicción): No
--------------------------------------------------
Dato de Prueba 3: [  4   1   1   8 404]
Es un ataque (Real): Sí
Es un ataque (Predicción): Sí
--------------------------------------------------
Dato de Prueba 4: [  3   1   1   6 200]
Es un ataque (Real): No
Es un ataque (Predicción): No
--------------------------------------------------
Accuracy: 1.0
```

Figura 2. Salida de las predicciones de XGBoost

Regresión con Máquinas de Vectores de Soporte

Conocida en Inglés como Support Vector Regression (SVM), son un tipo de algoritmo de aprendizaje supervisado utilizado para clasificación y regresión. En su forma más simple, para problemas de clasificación, SVM busca encontrar el hiperplano que mejor separa las clases de datos en el espacio de características. Este hiperplano es elegido de manera que maximiza el margen entre los puntos de las diferentes clases, es decir, la distancia entre el hiperplano y los puntos más cercanos de cada clase, conocidos como vectores de soporte (Mohan *et al.*, 2020; Sun *et al.*, 2020). SVM se destaca por su capacidad para manejar espacios de alta dimensión y situaciones donde el número de dimensiones supera al número

de muestras. Para casos en los que los datos no son linealmente separables, SVM utiliza una técnica llamada el truco del kernel para transformar el espacio de características a uno donde sea posible encontrar una separación lineal, permitiendo así la clasificación de datos complejos. Este enfoque hace de SVM una herramienta poderosa y versátil para tareas de clasificación y regresión en una amplia gama de aplicaciones. Los pasos de pseudo código para implementar este algoritmo en el contexto propuesto de análisis de ciberseguridad, son:

```
INICIO
        # importar librerías necesarias
        IMPORTAR pandas, sklearn
        # leer datos de servidor en archivo CSV
        data ⇐ CARGAR 'web_server_log.csv'
        # Extraer características relevantes
        X ⇐ data[['request_size', 'response_time']]
        y ⇐ data['anomaly']
        # normalizar las características de los datos
        scaler ⇐ StandardScaler()
        X ⇐ scaler.FIT_TRANSFORM(X)
        # Dividir los datos en conjuntos de entrenamiento y prueba
        X_train, X_test, y_train, y_test ⇐
          train_test_split(X, y, test_size=0.2, random_state=42)
        # Inicializar el modelo y entrenarlo con los datos
        model ⇐ SVC()
        model.ENTRENAR(X_train, y_train)
        # Predecir anomalías en los datos con base en el modelo
        anomalias ⇐ model.PREDECIR(X_test)
   # Reportar al usuario las anomalías encontradas
        PARA CADA fila EN anomalies HACER
                IMPRIMIR(fila)
FIN
```

La salida proporcionada por la operación del script anterior, con base a la detección de anomalías en el archivo de log, es como se muestra en la figura 3. Nótese que en el arreglo de respuestas, el valor real

coincide con el de la predicción, sean estos ceros o unos. Para el caso de estudio, la correspondencia entre ambas variables fue del 100%.

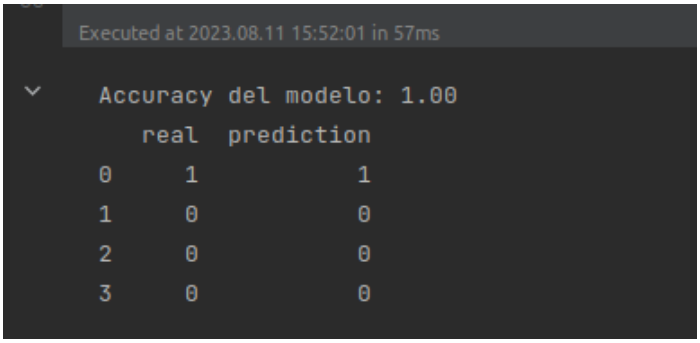

Figura 3. Salida de las predicciones de SVM

DistilBERT

Es una herramienta de inteligencia artificial que simplifica el complejo modelo BERT (Bidirectional Encoder Representations from Transformers) que fue desarrollado inicialmente por Google, manteniendo su capacidad para entender el lenguaje humano (Silva & Akabane, 2022). Funciona analizando textos completos para capturar el significado de cada palabra en su contexto específico, gracias a la arquitectura de transformadores de palabras y significados. Este modelo destila el conocimiento de BERT, conservando su precisión en una estructura más pequeña y menos demandante computacionalmente. Emplea mecanismos de atención para evaluar la relevancia de las palabras, lo que permite al modelo concentrarse en la información crucial para entender el mensaje. DistilBERT es eficaz en tareas como comprensión de textos y análisis de sentimientos, ofreciendo resultados rápidos y precisos con menor requerimiento de recursos, lo que facilita su implementación en diversos sistemas de procesamiento de lenguaje natural aún en contextos de tecnicismos de ciberseguridad (Bokolo et al., 2023). Los pasos de pseudo código para implementar este algoritmo en el contexto propuesto de análisis de ciberseguridad, son:

INICIO

 # importar librerías necesarias

IMPORTAR pandas, torch, transformers

 # leer datos de servidor en archivo CSV

data ⇐ CARGAR 'web_server_log.csv'

 # Preparar un modelo preentrenado con base en los datos

tokenizador ⇐ DistilBertTokenizer.Preentrenar()

codificaciones ⇐ tokenizador.TOKENIZAR(data)

 # Convertir las etiquetas de amenazas en valores numéricos

```
        etiquetas ⇐ [INT(amenaza) PARA 'threat' EN data]
        # Entrenar el modelo con base en las etiquetas categorizadas
        modelo ⇐ DistilBertForSequenceClassification.Preentrenar(
    'distilbert-base-uncased', num_etiquetas=COUNT(etiquetas))
        entrenador ⇐ Trainer(modelo)
        entrenador.ENTRENAR()
        # Predecir anomalías en los datos con base en el modelo
        anomalias ⇐ entrenador.predict(dataset)
        # Reportar al usuario las anomalías encontradas
        PARA CADA fila EN anomalies HACER
                IMPRIMIR("Anomalía detectada: "+fila)
FIN
```

La salida proporcionada por la operación del script anterior, con base a la detección de anomalías en el archivo de log, es como se muestra en la figura 4.

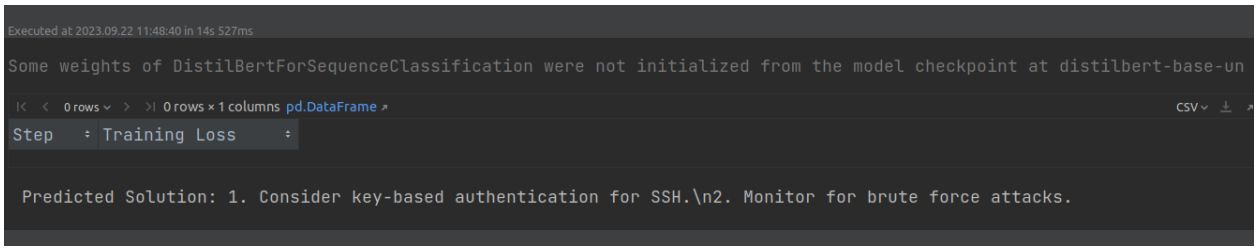

Figura 4. Salida de consola del pronosticador de soluciones con Distilbert local

Transformador Preentrenado Generativo

GPT (Generative Pre-trained Transformer) es un modelo de lenguaje que utiliza la arquitectura Transformer para procesar datos de texto (Beltrán & Mojica, 2020). El proceso comienza con el entrenamiento previo en un vasto conjunto de datos de texto, donde el modelo aprende a predecir la siguiente palabra en una secuencia dada, absorbiendo así un amplio conocimiento del lenguaje y de varios temas. Esta fase de entrenamiento previo permite que GPT adquiera una comprensión contextual de las palabras y frases, aprendiendo patrones de lenguaje, gramática, y hechos del mundo real. Una vez completado el entrenamiento previo, GPT puede ser afinado (fine-tuned) en tareas específicas de procesamiento de lenguaje natural (Kublik & Saboo, 2022; Kurniadi *et al.*, 2023), como la generación de texto, traducción, resumen, y más. En este paso, se ajusta el modelo preentrenado con un conjunto de datos más pequeño y específico para la tarea en cuestión, lo que le permite especializarse en esa tarea

manteniendo el conocimiento general aprendido durante el entrenamiento previo. GPT procesa el texto de entrada analizando las relaciones y dependencias entre las palabras, utilizando su capacidad para generar respuestas contextualmente relevantes basadas en el aprendizaje previo y la información específica de la tarea. Los pasos de pseudo código para implementar este algoritmo en el contexto propuesto de análisis de ciberseguridad, son:

```
INICIO
    # importar librerías necesarias
    IMPORTAR pandas, torch, transformers
    # leer datos de servidor en archivo CSV
    data ⇐ CARGAR 'web_server_log.csv'
    # pre entrenar el tokenizador del modelo GPT
    tokenizador ⇐ GPT2Tokenizer.Preentrenar('gpt2')
    modelo ⇐ GPT2LMHeadModel.Preentrenar('gpt2')
    # Convertir los datos en un formato adecuado para el modelo
    encodings ⇐ tokenizador.TOKENIZAR(data)
    # Convertir las etiquetas de 'amenaza' en
    # los registros de datos a valores enteros
    etiquetas ⇐ [INT(amenaza) PARA 'threat' EN data]
    # crear un modelo de datos encodificados
    dataset ⇐ CREAR_CONJUNTO_DATOS(encodings, etiquetas)
    # preparar y entrenar el modelo GPT
    entrenador ⇐ Trainer(modelo, dataset)
    entrenador.ENTRENAR()
    # cargar los datos que se van a evaluar con el modelo
    verificar ⇐ CARGAR 'casos_server_log.csv'
    # Preparar los nuevos datos para introducirlos al modelo
    inputs = tokenizador(verificar)
    # Predecir anomalías en los datos con base en el modelo
    anomalias = modelo(inputs)
    # Reportar al usuario las anomalías encontradas
    PARA CADA fila EN anomalies HACER
        IMPRIMIR("Anomalía detectada: "+fila)
FIN
```

La salida proporcionada por la operación del script anterior, con base a la detección de anomalías en el archivo de log, es como se muestra en la figura 4.

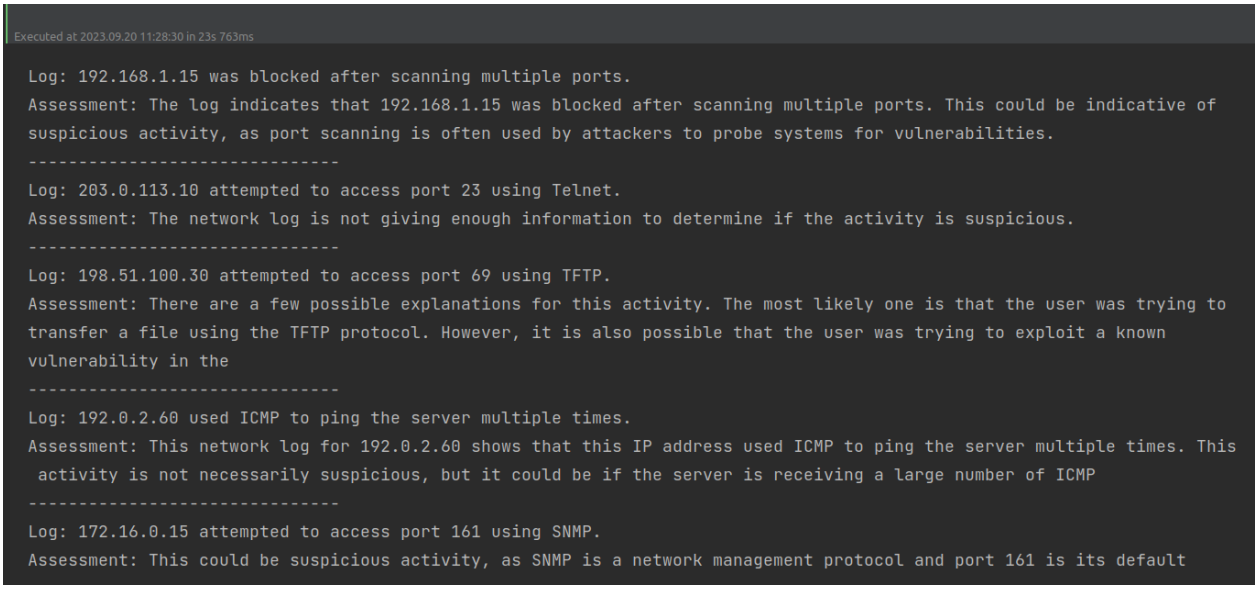

Figura 5. Salida del analizador GPT aplicado a ciber-seguridad

Luego de compilados los modelos, estos fueron sometidos al procesamiento de los datos de prueba y se calcularon las métricas de confiabilidad de cada uno de ellos. La evaluación reflejó una variabilidad en la eficacia de los modelos de aprendizaje automático para la tarea predictiva. El modelo Isolation Forest evidenció una capacidad predictiva medianamente aceptable, con un coeficiente de determinación de 0.789, indicando un buen ajuste en los datos pronosticados. El modelo XGBoost demostró una eficiencia comparable a Isolation Forest (Sahin, 2020), con un coeficiente de 0.788. En contraste, el modelo Support Vector Machine (SVM) presentó una capacidad predictiva reducida, con el coeficiente más bajo entre los modelos estudiados, situándose en 0.623. Sin embargo, los modelos basados en transformadores, DistilBert y GPT-2, sobresalieron significativamente sobre los demás, con errores cuadráticos medios de 0.240 y 0.005, respectivamente, y coeficientes de determinación que excedieron el 0.9, lo cual indica una alta precisión y confiabilidad predictiva. Estos resultados ponen de manifiesto la superioridad de las arquitecturas avanzadas de procesamiento de lenguaje natural en capturar la complejidad y la varianza de los datos de prueba (Adel *et al.*, 2022), concretamente en tareas donde la precisión es una prioridad.

La fase de entrenamiento y prueba reveló diferencias notables en términos de eficiencia temporal ante el procesamiento y pronóstico de los datos de prueba. Los modelos tradicionales, como Isolation Forest y XGBoosting, demostraron tiempos de entrenamiento y prueba mínimos, menores a 1 segundo. SVM, aunque con un tiempo de prueba mayor a 6 segundos, refleja la viabilidad de este método para un sistema de

consulta rápida en escenarios de uso práctico (Abedi, 2022). Por otro lado, DistilBert y GPT-2, incurrieron en tiempos significativamente mayores de entrenamiento, más de 10 minutos para el primero y más de 100 para el segundo. Sin embargo, los tiempos de prueba de datos para detección de anomalías fueron razonablemente bajos, con 15.3 segundos para DistilBert y 2 minutos y 0.13 segundos para GPT-2, el cual demostró ser el más ventajoso (Frohling, 2021). A pesar de la mayor inversión de tiempo en la etapa de entrenamiento, la fase de prueba refleja la factibilidad de su aplicación en un sistema distribuido para predecir eventos anómalos de seguridad informática. Considerando que la fase de prueba simula el despliegue real de estos modelos, el uso intensivo de datos de logs de servidores podría alimentar eficientemente los modelos, permitiendo obtener resultados confiables en un marco temporal adecuado para la toma de decisiones preventivas.

V. CONCLUSIONES

La investigación ha demostrado que los modelos predictivos de aprendizaje automático tienen el potencial de anticipar y administrar anomalías de seguridad informática. El rendimiento de las pruebas confirma la eficacia de estos sistemas para alertas tempranas. Los modelos como DistilBert y GPT-2, debidamente configurados para ejecutar regresión de datos de conexiones, destacaron por su precisión en la predicción de valores de detección de posibles amenazass, a pesar de su respectivo tiempo prolongado de entrenamiento. En contraste, a pesar de su rapidez, los modelos como XGBoosting e Isolation Forest ofrecieron menor precisión, lo que podría comprometer las predicciones en situaciones críticas.

Como recomendación para futuros trabajos, podría desarrollarse un prototipo de sistema que emplee estos modelos de IA para evaluar series temporales extensas, abarcando los datos concernientes a la totalidad de eventos de tráfico de una red durante un lapso prolongado de tiempo. Este prototipo permitiría una evaluación más amplia de la efectividad de las tecnologías en diferentes condiciones y facilitaría la implementación de sistemas de alerta que proporcionen pronósticos confiables y oportunos, que son esenciales para la prevención de desastres en la seguridad de datos y la protección de la información personal de los usuarios.

VI. REFERENCIAS

Abedi, R., Costache, R., Shafizadeh-Moghadam, H., & Pham, Q. B. (2022). Flash-flood susceptibility mapping based on XGBoost, random forest and boosted regression trees. Geocarto International, 37(19), 5479-5496. https://doi.org/10.1080/10106049.2021.1920636

Adel, H., Dahou, A., Mabrouk, A., Abd Elaziz, M., Kayed, M., El-Henawy, I. M., ... & Amin Ali, A. (2022). Improving crisis events detection using distilbert with hunger games search algorithm. Mathematics, 10(3), 447. https://doi.org/10.3390/math10030447

Beltrán, N. C. B., & Mojica, E. C. R. (2020). Procesamiento del lenguaje natural (PLN)-GPT-3: Aplicación en la Ingeniería de Software. Tecnología Investigación y Academia, 8(1), 18-37.

Bokolo, B. G., Chen, L., & Liu, Q. (2023). Detection of Web-Attack using DistilBERT, RNN, and LSTM. In 2023 11th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-6). IEEE.

Carletti, M., Terzi, M., & Susto, G. A. (2023). Interpretable anomaly detection with diffi: Depth-based feature importance of isolation forest. Engineering Applications of Artificial Intelligence, 119, 105730.

Du, T., Kanodia, A., & Athey, S. (2023). Torch-Choice: A PyTorch Package for Large-Scale Choice Modelling with Python. arXiv preprint arXiv:2304.01906.

Frohling, L., & Zubiaga, A. (2021). Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover. PeerJ Computer Science, 7, e443. https://doi.org/10.7717/peerj-cs.443

Georgescu, T. M. (2020). Natural language processing model for automatic analysis of cybersecurity-related documents. Symmetry, 12(3), 354.

Kim, A., Park, M., & Lee, D. H. (2020). AI-IDS: Application of deep learning to real-time Web intrusion detection. IEEE Access, 8, 70245-70261.

Kublik, S., & Saboo, S. (2022). GPT-3. O"Reilly Media, Incorporated.

Kurniadi, D., Septiana, Y., & Sutedi, A. (2023). Alternative Text Pre-Processing using Chat GPT Open AI. Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI, 12(1).

Mohan, L., Pant, J., Suyal, P., & Kumar, A. (2020). Support vector machine accuracy improvement with classification. In 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 477-481). IEEE. https://doi.org/10.1109/CICN49253.2020.9242572

Pérez Martell, R. (2020). La tecnología y los objetivos de desarrollo sostenible. La tecnología y los objetivos de desarrollo sostenible, 1-297.

Raghunath, K. K., Kumar, V. V., Venkatesan, M., Singh, K. K., Mahesh, T. R., & Singh, A. (2022). XGBoost regression classifier (XRC) model for cyber attack detection and classification using inception v4. Journal of Web Engineering, 21(4), 1295-1322.

Rothman, D. (2021). Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more. Packt Publishing Ltd.

Sahin, E. K. (2020). Assessing the predictive capability of ensemble tree methods for landslide susceptibility mapping using XGBoost, gradient boosting machine, and random forest. SN Applied Sciences, 2(7), 1308. https://doi.org/10.1007/s42452-020-3060-1

Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. SN Computer Science, 2, 1-18.

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and KNN models for the text classification. Augmented Human Research, 5, 1-16.

Silva Barbon, R., & Akabane, A. T. (2022). Towards transfer learning techniques—BERT, DistilBERT, BERTimbau, and DistilBERTimbau for automatic text classification from different languages: a case study. Sensors, 22(21), 8184.

Sun, C. C., Cardenas, D. J. S., Hahn, A., & Liu, C. C. (2020). Intrusion detection for cybersecurity of smart meters. IEEE Transactions on Smart Grid, 12(1), 612-622.

Xu, H., Pang, G., Wang, Y., & Wang, Y. (2023). Deep isolation forest for anomaly detection. IEEE Transactions on Knowledge and Data Engineering.

Yang, C. T., Chan, Y. W., Liu, J. C., Kristiani, E., & Lai, C. H. (2022). Cyberattacks detection and analysis in a network log system using XGBoost with ELK stack. Soft Computing, 26(11), 5143-5157.