UNIVERSIDAD NACIONAL DE SAN LUIS FACULTAD DE CIENCIAS FÍSICO, MATEMÁTICAS Y NATURALES
TECNICATURA UNIVERSITARIA EN WEB
SISTEMAS DE CIBERDEFENSA CON IA

Autores: Lucero Pollio Augusto, Montes Pantano Juan Bautista.

Profesor: Agüero Walter.

Mails: -bautistamontesp@gmail.com
-augustoluceropollio@gmail.com

INFORME Y CONTEXTO

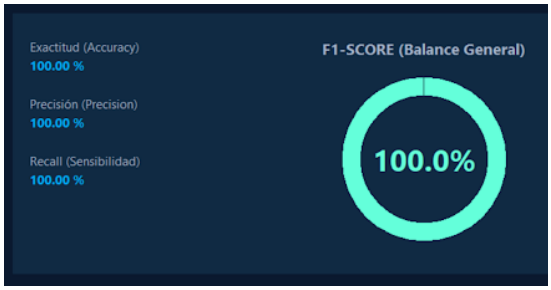
Presentamos en este informe el proceso de creación y los resultados de nuestro **Dashboard de Análisis de Red**, una aplicación de escritorio desarrollada en Tkinter. El objetivo de la herramienta es identificar ataques de denegación de servicio (DoS) utilizando un modelo de `Gradient Boosting`. La clave de nuestro proyecto no fue solo el resultado final, sino el proceso iterativo que seguimos para asegurar que nuestras pruebas fueran verdaderamente representativas de un escenario real.

Nuestro proceso de desarrollo se dividió en tres fases fundamentales:

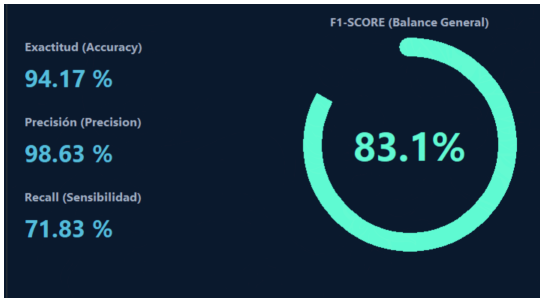
1. Fase Inicial: El Intento con Datos Reales Originalmente, la consigna era utilizar datos sintéticos. Sin embargo, quisimos poner a prueba nuestras habilidades con un desafío mayor: el dataset **KDDTest/KDDTrain**. Pronto nos dimos cuenta de que la complejidad en el preprocesamiento de estos datos era un proyecto en sí mismo y, para mantener el foco, decidimos pivotar.

Error: `icmp value not in ('tcp', 'udp', " 'icmp'")`

2. Fase Intermedia: Datos Sintéticos "Perfectos" Siguiendo la consigna, generamos un dataset "de laboratorio", donde las características del tráfico normal y las de un ataque estaban perfectamente diferenciadas. El modelo arrojó resultados casi perfectos, pero nos generó una duda importante: ¿sería igual de efectivo en un entorno real, donde los ataques no son tan obvios?



3. Fase Final: Datos Sintéticos Realistas y Desafiantes Esta fue la etapa más crucial. Concluimos que para validar realmente nuestro modelo, necesitábamos datos que imitaran las imperfecciones y la complejidad del mundo real. Decidimos someter al modelo a un verdadero *stress test*, forzándolo con un escenario mucho más avanzado que detallaremos a continuación.



DESARROLLO TÉCNICO Y RESULTADOS

Implementación en un Dashboard Interactivo El resultado final se consolidó en una aplicación de escritorio funcional con **Tkinter**. Este dashboard permite visualizar en tiempo real las métricas de rendimiento del modelo, la matriz de confusión y una vista previa del dataset, haciendo el análisis accesible e intuitivo.

Simulación Avanzada de Amenazas

Para asegurar que la evaluación del modelo fuera lo más honesta y rigurosa posible, implementamos dos técnicas avanzadas en la generación de nuestros datos sintéticos:

- **Introducción de "Ataques Sigilosos" (Stealthy Attacks):** El desafío más difícil para un sistema de detección no son los ataques obvios, sino los que intentan pasar desapercibidos. Por ello, generamos intencionalmente una porción significativa de los ataques **(aproximadamente un 30%)** utilizando los mismos parámetros y rangos de valores que el tráfico normal y legítimo. Esto simula amenazas sofisticadas (conocidas como ataques "low-and-slow") que buscan camuflarse en la red, forzando al modelo a aprender a diferenciar patrones extremadamente sutiles.
- **Incremento del "Ruido" Sistémico:** Aumentamos el nivel de "ruido" aleatorio (variabilidad gaussiana) en todo el conjunto de datos. El propósito fue simular un entorno de red menos predecible, con las imperfecciones de medición y las fluctuaciones inherentes a los sistemas reales.

Estas modificaciones constituyeron un **test de estrés deliberado** para el algoritmo. El objetivo no era bajar artificialmente la efectividad, sino obtener una métrica de rendimiento honesta que reflejara la capacidad real del modelo para enfrentarse a amenazas complejas, no solo a anomalías evidentes en un entorno de laboratorio.

GENERACIÓN DE DATOS Y ENTRENAMIENTO El siguiente código representa la implementación final de este generador de datos avanzados, donde los rangos se superponen para crear los ataques sigilosos.

```
Python
# ... (código de generación de datos tn_data y ta_data) ...

# Creación y entrenamiento del modelo con los datos más realistas
modelo = GradientBoostingClassifier(n_estimators=200, random_state=42)
modelo.fit(X_train, y_train)
```

MATRIZ DE CONFUSIÓN: Los resultados, obtenidos contra este dataset altamente desafiante, fueron los siguientes:**Matriz de Confusión Resultante:**

- **Verdaderos Negativos (TN): 2394**
  - Tráfico normal identificado correctamente.
- **Falsos Positivos (FP): 6**
  - Tráfico normal incorrectamente clasificado como ataque (falsas alarmas).
- **Verdaderos Positivos (TP): 431**
  - Ataques reales detectados correctamente.
- **Falsos Negativos (FN): 169**
  - Ataques reales que no fueron detectados.

CONCLUSIÓN

Sometido a un riguroso test de estrés con ataques sigilosos, el modelo alcanzó una efectividad general (F1-Score) del **83.4%**.

Este resultado revela su verdadero perfil: es excelente para evitar falsas alarmas (**99% de Precisión**), pero le cuesta detectar las amenazas más camufladas (**72% de Sensibilidad**).

El verdadero éxito fue haber sometido al modelo a un **test de estrés riguroso**, utilizando datos sintéticos diseñados para imitar las condiciones más desafiantes del mundo real: ruido sistémico y ataques sigilosos. Al demostrar su eficacia en este escenario, hemos probado que su rendimiento es bastante adecuado.

Enlace al video:

Video Antivirus.mp4