Base de datos de imágenes de productos

Reporte proyecto final

Jorge Andrés Pérez Barrios

Departamento de Ing. Ciencias de la Computación y TI, Facultad de Ingeniería

Universidad del Valle de Guatemala, Guatemala

per16362@uvg.edu.gt

Abstract

—Machine Learning models such as Decision Tree and Naive Bayes can be useful in detecting potential threats in a network. In this document, the results of implementing both of these ML models in an application that detects potential cyber attacks like Distributed Denial of Service (R-U-Dead-Yet and Slowloris) and SYN Scan will be discussed and compared. The models will classify an entry by analyzing some features of said network entry into four categories: SYN Scan, R-U-Dead-Yet, Slowloris or normal flow. The accuracy scores of both models were, on average, 0.796 for Naive Bayes and 0.999 for Decision Tree Classifier model. This last one was overfitted.

Palabras clave—Machine Learning, Deep Learning, Naive Bayes, Decision Tree, SIMARGL, Cybersecurity, threat detection

I. Introducción

Este documento es el informe correspondiente al proyecto de SIMARGL, el cual tiene como objetivo final realizar un motor de detección y clasificación de amenazas de red, clasificándolos en dos tipos de ataques DDoS, SYN Scan o flujo normal. Las bases de datos fueron proveídas por una base de datos de Kaggle con la que se entrenó a los modelos. Esta base de datos es llamada SIMARGL, la cual contiene datos actualizados de ataques de red modernos. El motor de detección y clasificación se realizó con los algoritmos de Decision Tree Classifier y Multinomal Naive Bayes, ambos de la librería ScikitLearn. Estos modelos se discutirán y compararán para decidir cuál es el mejor en este tipo de problemas.

Antecedentes

II. Marco Teórico

A. Para poder crear un motor de detección y clasificación de amenazas de red de la base de datos proveído por Kaggle, se utilizaron dos algoritmos de aprendizaje de máquina: Decision Tree y Naive Bayes.

Estos algoritmos se utilizaron para obtener el grupo al que pertenece la amenaza del dataset y así poder obtener su clasificación.

Decision Tree

El modelo de árboles de decisión son algoritmos estadísticos de clasificación de machine learning que predicen o clasifican observaciones, basándose en reglas de decisión.[1] Está estructurado en el diagrama de árbol de decisión, que es similar a un diagrama de flujo, donde una característica es representada por un nodo. Así mismo, cada rama representa una regla de decisión y cada resultado es representado por una hoja. [2]

Cada clasificación comienza desde el nodo raíz, pasa por las ramas hasta llegar a una hoja. Cada nodo es

un caso de prueba para cierta característica hasta descender a una de las posibles respuestas al caso de prueba. Este proceso es recursivo hasta agotar los atributos o instancias. [2] Ya que es un modelo clasificatorio, clasifica la información en función de otras variables, según ciertas propiedades o en la relación entre diferentes variables para predecir el valor de otra. [3] Estos modelos se utilizan en Big Data para predecir la probabilidad de un resultado, en base a ciertas condiciones.

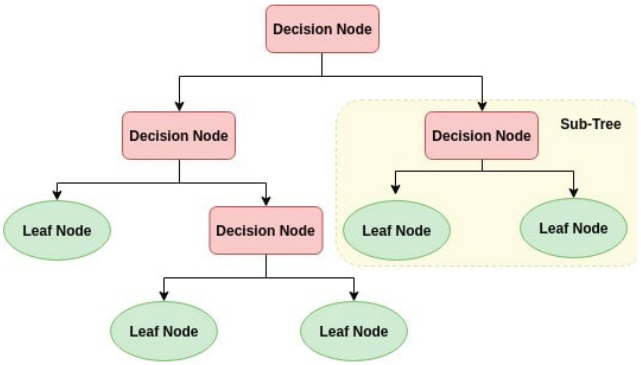

Fig 1: Ejemplo gráfico de un árbol de decisión.

Ventajas:

- No requieren un pre-procesamiento complejo.
- Puede utilizarse para variables tanto cualitativas como cuantitativas.
- Son fáciles de utilizar junto con otros modelos de decisiones.
  [4]

Desventajas:

- Son inestables; pueden llegar a un resultado diferente con un pequeño cambio en la entrada.
- No se garantiza que sean el modelo más óptimo.
- Requieren de mucha práctica para evitar el sesgo.
  [4]

Naive Bayes

Es uno de los modelos de probabilidad más simples y mayormente utilizados en la clasificación de un resultado. Se basa en las reglas de Bayes, las cuales predicen la probabilidad condicional que un atributo pertenezca a una clase $P(c_i|d_j)$ a partir de la probabilidad de los atributos de cada clase seleccionada $P(d_j|c_i)$ y la probabilidad de la clase en el set de entrenamiento $P(c_i)$. [5]

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)}$$

Ecuación 1: ecuación de la probabilidad de un atributo en el modelo de Naive Bayes.

El modelo Multinomial Naive Bayes permite el análisis multivariado considerando la frecuencia de cada uno de los términos en los atributos en lugar de tener una ocurrencia binaria. [5]

Ventajas:

- Puede integrar múltiples variables para la clasificación de los datos
- Es fácil de integrar con características conocidas.
- Fácil de manejar y entender y no requiere de mucha práctica.
  [6]

Desventajas:

- Los predictores son independientes entre sí, lo que pueda que no se ajuste correctamente a los datos
- Requiere de gran esfuerzo en procesamiento computacional, a medida que incrementa las características.
- Requiere que los datos se distribuyan normalmente.
  [6]

III. METODOLOGÍA

A. Análisis exploratorio de los datos.

Junto con el planteamiento del proyecto, se proporcionaron los conjuntos de datos pertenecientes a SIMARGL. Estos conjuntos de datos estaban separados en cuatro partes. Sin embargo, para fines prácticos del proyecto, se trabajó únicamente con los primeros dos set de datos. Estos sets de datos contenían alrededor de 3.5 y 8.6 millones de entradas, respectivamente. El primer dataset contiene dos labels y el segundo contiene tres, por lo que el conteo de clasificaciones es el siguiente:

- Normal flow (dataset 1) - 1073852 entradas
- SYN Scan - aggressive - 2496814 entradas
- Normal flow (dataset 2) - 5496206 entradas
- Denial of Service R-U-Dead-Yet - 2276947 entradas
- Denial of Service Slowloris - 864054 entradas.

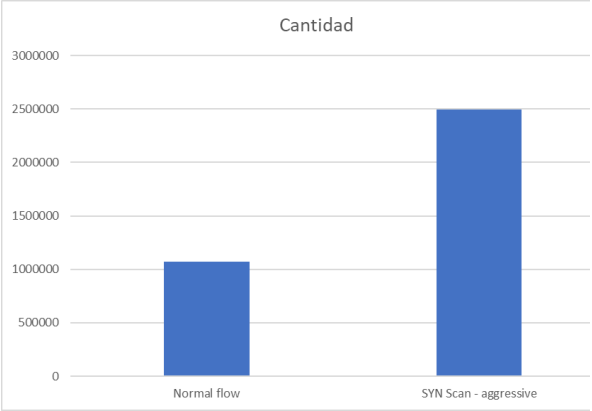

Fig 2: Gráfico de barras de la cantidad de labels en dataset 1.

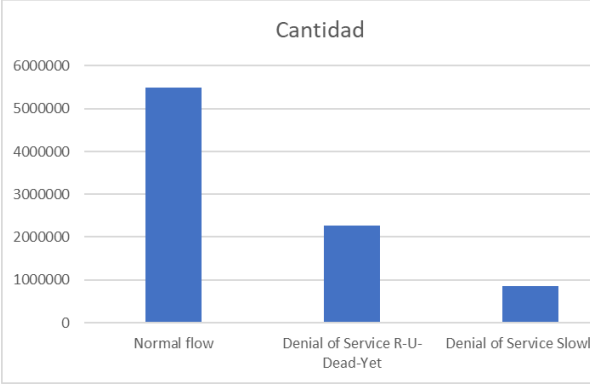

Fig 3: Gráfico de barras de la cantidad de labels en dataset 2.

Label	Cantidad	Frecuencia
Normal flow	1073852	30.07%
SYN Scan - aggressive	2496814	69.93%
Total	3570666	100%

Tabla 1: Tabla de frecuencias de labels del dataset 1

Label	*Cantidad*	*Frecuencia*
Normal flow	*5496206*	*63.63%*
Denial of Service R-U-Dead-Yet	*2276947*	*26.36%*
Denial of Service Slowloris	*864054*	*10.00%*
Total	*8637207*	*100*

Tabla 2: Tabla de frecuencias de labels del dataset 2

Como se puede observar en las tablas 1 y 2, los datos no están del todo balanceados. Especialmente en la tabla 2, se puede observar que las entradas clasificadas como Slowloris ocupan solo el 10% de los datos, a diferencia de las entradas clasificadas como Normal flow. Para resolver esta diferencia tan grande entre los datos, se tomaron muestras representativas de cada uno de los labels clasificatorios a un tamaño similar al de Slowloris: 800,000 entradas por clasificación (400,000 entradas clasificadas como Normal flow del dataset 1 y 400,000 del dataset 2), para manejar un total de 3.2 millones de datos, los cuales se juntaron en un solo dataset.

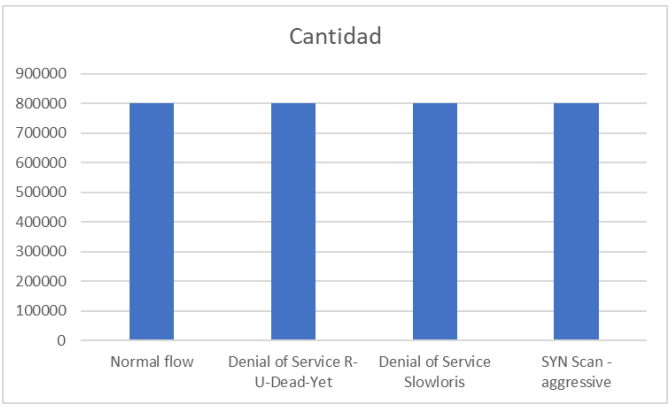

Fig 4: Gráfico de barras de cantidad de labels en dataset.

Label	*Cantidad*	*Frecuencia*
Normal flow	*800000*	*25*
Denial of Service R-U-Dead-Yet	*800000*	*25*
Denial of Service Slowloris	*800000*	*25*
SYN Scan - aggressive	*800000*	*25*
Total	*3200000*	*100*

Tabla 3: Tabla de frecuencias de dataset.

Luego de balancear los datos, se hizo un análisis exploratorio más profundo sobre las características. Sin embargo, como el dataset cuenta con 50 características, se decidió revisar una por una para descartar características con los mismos valores para cada clasificación o irrelevantes para su análisis, antes de graficar una matriz de correlaciones. Se eliminaron las siguientes características:

- biflow_direction
- direction
- firewall_event
- flow_active_timeout
- flow_end_milliseconds
- flow_end_sec
- flow_id
- flow_inactive_timeout
- flow_start_milliseconds
- flow_start_sec
- frame_length
- max_ip_pkt_len
- min_ip_pkt_len
- ooorder_in_pkts
- ooorder_out_pkts
- retransmitted_in_bytes
- retransmitted_in_pkts
- retransmitted_out_bytes
- retransmitted_out_pkts
- src_tos
- dst_tos
- sampling_interval

Estas características se eliminaron ya que no aportan valor a los modelos, e incluso, pueden llevar overfitting y confundir a los modelos. Las características restantes son, en su mayoría, numéricas junto con unas cuantas clasificatorias. Luego, se eliminaron las características de direcciones IP, tal como IPV4_DST_ADDR e IPV4_SRC_ADDR, ya que la dirección IP fuente y destino confunden los modelos. Así mismo, se eliminó la variable L7_PROTO_NAME, ya que cuenta con demasiados protocolos como para enumerarlos.

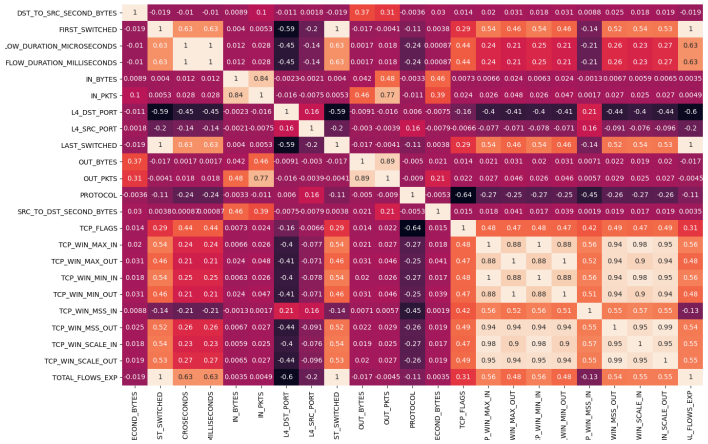

Fig 5: Matriz de correlaciones.

Como se muestra en la figura 5, las 25 características que quedaron muestran una correlación relativamente alta entre sí, a excepción de la variable de puerto destino y protocolo, sin embargo estas dos muestran una correlación fuerte alguna otra variable. Las características que mayor correlación tuvieron fueron las que muestran el flujo de ventanas TCP, al igual que la variable de flags TCP. Con estas características ya se puede empezar a preprocesar y arreglar los datos.

B. Manejo y preprocesamiento de los datos.

Para el preprocesamiento de los datos, se aplicaron cambios en tres variables: PROTOCOL_MAP, DST_TO_SRC_SECOND_BYTES y SRC_TO_DST_SECOND_BYTES. Para la variable protocol_map, se extrajeron y eliminaron las tuplas que contienen los valores “ipv6-icmp”, ya que se decidió que el protocolo IPv6 es muy nuevo y no tiene el suficiente peso dentro del dataset como para ser relevante. Luego de eso, se hizo una ramificación binaria para el resto de valores: 0 para los protocolos tcp y 1 para los protocolos udp, icmp y gre. Esto fue así ya que el protocolo TCP es el que

mayormente está presente en el dataset, por lo que hacer una clasificación numérica no binaria llevaría confusión a los modelos.

Por otra parte, para las otras dos variables de flujo de bytes por segundo, se tuvo que aplicar un preprocesamiento de eliminación de comas (,) ya que los valores de esta variable eran de tipo string, no numéricos. De igual manera, se eliminaron los valores decimales en estas variables para que al final estas características fueran de tipo entero. Luego se extrajo un reporte sobre las características finales.

IV. Análisis y Discusión de Resultados

El objetivo principal de este proyecto es realizar un motor de detección y clasificación de amenazas de red, clasificándolos en dos tipos de ataques DDoS, SYN Scan o flujo normal. Para resolver este objetivo, se utilizaron dos tipos de algoritmos distintos clasificatorios: Decision Tree Classifier y Multinominal Naive Bayes, anteriormente mencionados. Se trabajó cada uno individualmente, con el mismo set de datos y la misma data de train y test, para su comparación. Dicho dataset se le aplicó una separación de la siguiente manera: 55% entrenamiento, 15% validación y 30% pruebas. Para la compilación de ambos modelos, se trabajó dentro de la misma computadora. Las características de hardware de esta son las siguientes:

- Procesador Intel i7 11700K, 5.00 GHz, 8 núcleos con 16 subprocesos.
- Sistema de enfriamiento líquido Corsair H100i para el procesador.
- 32 GB de memoria RAM marca Kingston HyperX Fury, 2666MHz.
- 1.7 GB de almacenamiento SSD marca Kingston.
- Motherboard Asus TUF Gaming Z590-Plus WiFi.

A. Decision Tree Classifier

Este algoritmo fue el que menos se tardó tanto en la compilación del modelo como en la predicción de los datos de verificación, con tiempos promedio de 5.633s y 0.086s, respectivamente. Sin embargo, a pesar de que los tiempos de compilación y verificación fueron excelentes, las métricas del modelo dictan el caso contrario, a pesar que en la tabla 4, en la matriz de confusión, se obtuvieron buenos resultados con un solo falso positivo R-U-Dead-Yet clasificado como Slowloris.

239938	0	0	0
0	240220	0	0
0	0	240219	1
0	0	0	239621

Tabla 4: Matriz de confusión del modelo Decision Tree Classifier.

	Precisión	Recall	f1-score	support
Normal flow	1.0	1.0	1.0	239938
SYN Scan	1.0	1.0	1.0	240220
R-U-Dead-Yet	1.0	1.0	1.0	240220
Slowloris	1.0	1.0	1.0	239621
Accuracy			1.0	959999
macro avg	1.0	1.0	1.0	959999
weighted avg	1.0	1.0	1.0	959999

Tabla 5: Métricas obtenidas del modelo Decision Tree Classifier.

Como se puede observar en las tabla 5, tanto las métricas de precisión, recall y f1-score, se obtuvo un valor de 1.0, lo que significa que hubo un overfitting al momento de compilar el modelo. Esto se puede respaldar ya que el

accuracy score de este modelo dio como resultado 0.9999, lo que significa que, efectivamente, hubo algún overfitting dentro de las características.

B. Multinomial Naive Bayes

Este algoritmo fue el más lento de los dos, con tiempos promedio de 6.1274s en la compilación y 0.1374s en la verificación. A pesar de que se tardó casi el doble de tiempo que el modelo Decision Tree Classifier, puede decirse que fue un tiempo de compilación y predicción óptimos para este problema. Sin embargo, lo que diferencia el modelo anterior con este son las métricas. Como se muestra en la tabla 6, la matriz de confusión, a pesar de que hubieron más falsos positivos y negativos, los verdaderos positivos predominaron con un 79.63%, lo que significa que no hubo algún tipo de overfitting en el modelo.

168964	18507	52467	0
0	230693	9527	0
6100	3348	125319	105453
0	0	115	239506

Tabla 6: Matriz de confusión del modelo Multinomial Naive Bayes.

	Precisión	Recall	f1-score	support
Normal flow	0.97	0.7	0.81	239938
SYN Scan	0.91	0.96	0.94	240220
R-U-Dead-Yet	0.67	0.52	0.59	240220
Slowloris	0.69	1.0	0.82	239621
Accuracy			0.80	959999
macro avg	0.81	0.80	0.79	959999
weighted avg	0.81	0.80	0.79	959999

Tabla 7: Métricas obtenidas del modelo Decision Tree Classifier.

El accuracy score de este modelo fue de 0.7963, lo que respalda que fue un excelente modelo, ya que este valor es aceptado. De igual manera, analizando las métricas, se puede decir que la mejor métrica fue la de presión, con valores de 0.97 y 0.91 en la clasificación de flujo normal y SYN Scan, respectivamente. Y, a pesar de tener 0.67 y 0.69 para R-U-Dead-Yet y Slowloris, respectivamente, esto es aceptado ya que estos dos ataques son de tipo DDoS, lo que significa que son muy parecido en sus características, pero puede diferenciarlos con cierta precisión.

Así mismo, para replicar este proyecto, se recomienda utilizar un dispositivo con más capacidad de memoria RAM y, por ende, un tamaño de datos más grande que solo 800,000 entradas por clasificación y, así mismo, más clasificaciones. Por otra parte, se recomienda implementar otro modelo de clasificación en lugar del Decision Tree Classifier, ya que este tiene el problema de overfitting debido a sus problemas de optimización anteriormente mencionadas.

V. Conclusiones

- Los algoritmos de clasificación Naive Bayes superan a los algoritmos Decision Tree en datos con pocas clasificaciones.
- El modelo Decision Tree Classifier cuenta con un problema de optimización y sesgo para problemas de esta índole.
- Los algoritmos Naive bayes aprenden de manera más lenta pero mas optima que los algoritmos Decision Tree Classifier
- Ambos algoritmos consumen muy pocos recursos de hardware tanto en su entrenamiento como en la verificación.

Referencias

[1] IBM (2021). Modelos de árboles de decisión. Extraído de: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=trees-decision-tree-models

[2] SitioBigData. (14 de diciembre de 2019). Árbol de decisión en Machine Learning (Parte 1). Extraído de: https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/

[3] Universidad Internacional de La Rioja (2022). Árboles de decisión: en qué consisten y aplicación en Big Data. Extraido de: https://www.unir.net/ingenieria/revista/arboles-de-decision/

[4] artyco. (2021). Qué es un árbol de decisión y su importancia en el Data Driven. Extraido de: https://artyco.com/que-es-un-arbol-de-decision-y-su-importancia-en-el-data-driven/

[5] Anguiano-Hernandez, E., Consejo Nacional de Ciencia y Tecnología CONACYT, (29 de abril de 2009). Extraído de: https://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/MGP_RepProy_Abr_29.pdf

[6] Numérica, (20 de octubre de 2020). Científico de Datos: Ventajas y Desventajas de Naive Bayes en Machine Learning. Extraído de: https://us.numerica.mx/articulos-educativos-juegos-de-entretenimiento/ventajas-y-desventajas-de-naive-bayes-en-machine-learning/