INTELIGENCIA ARTIFICIAL PARA LAS CIENCIAS E INGENIERÍAS
PROYECTO ENTREGA FINAL

ANDROID DETECTION MALWARE

INTEGRANTES

JUAN PABLO MANOTAS CAÑAS, CC: 1020487220, BIOINGENIERIA
CARLOS DANIEL QUIROS CARBAJAL, CC:1214743514, INGENIERIA DE SISTEMAS
JHAROL SEBASTIAN AGUDELO RAMOS, CC: 1214740141, INGENIERIA MECANICA

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN - ANTIOQUIA

2023

INTRODUCCIÓN

Descripción del problema

El problema predictivo que se busca resolver es la detección de malware en dispositivos Android a través de técnicas de aprendizaje automático enfocado en detectar y clasificar aplicaciones maliciosas para dispositivos Android. Una de las formas precisas de identificar la sospecha de aplicaciones es monitorear la red en la que está conectado el dispositivo Android. El aprendizaje automático es un subconjunto de la inteligencia artificial que se centra en el desarrollo de programas informáticos que pueden acceder a los datos y utilizarlos para aprender por sí mismos. Esta tecnología se puede utilizar para construir modelos que evalúan los datos entrantes para hacer predicciones y detectar anomalías. Esto se puede aplicar a la detección de aplicaciones maliciosas de Android mediante la creación de un modelo que busque patrones asociados con el comportamiento del malware. El modelo podría usar una variedad de funciones, como permisos solicitados, llamadas API realizadas, actividad de red, etc. Una vez entrenado, el modelo puede implementarse para clasificar cualquier aplicación nueva como maliciosa o no maliciosa.

Enlace

El conjunto de datos utilizado sería el proporcionado en el enlace de Kaggle: https://www.kaggle.com/datasets/subhajournal/android-malware-detection

Este dataset contiene características extraídas de 5560 aplicaciones de Android, donde se han etiquetado como malware o no malware.

Métricas de desempeño

• Métricas de machine learning: precisión (accuracy), sensibilidad (recall), especificidad (specificity) y F1-Score.
• Métricas de negocio: tiempo de respuesta, tasa de falsos positivos, tasa de falsos negativos y costo asociado a la implementación de la solución.

Criterio de desempeño

El desempeño deseable en producción sería lograr una alta precisión (accuracy) en la clasificación de aplicaciones de Android como malware o no malware, al mismo tiempo que se logra un bajo costo asociado a la implementación de la

solución y una tasa de falsos positivos y falsos negativos aceptable para el negocio. En este sentido, un valor de precisión por encima del 95% y una tasa de falsos positivos y falsos negativos inferior al 5% podrían ser indicadores de un buen desempeño en producción.

Métricas de negocio

Las métricas empresariales son de vital importancia para prever la bancarrota, un estado que afecta tanto al negocio en sí como a la economía global. La capacidad de anticipar la bancarrota resulta especialmente crucial en actividades relacionadas con instituciones financieras. El propósito de esta labor consiste en realizar pronósticos sobre la posible bancarrota de una empresa o negocio. Mediante un análisis eficaz y eficiente de estas predicciones, se logra optimizar las decisiones de préstamo adoptadas por la organización.

EXPLORACIÓN DESCRIPTIVA DEL DATASET

Luego de la selección del dataset, se creó un nuevo notebook que tiene como fin generar un nuevo dataset que cumple con los requerimientos del proyecto(ver notebook 01 – exploración de datos y preprocesado.ipynb).

Para esto se importó el dataset original (355630 datos × 86 columnas) y se realizó una selección de 10000 datos aleatorios con el propósito de hacerlo más manejable con el recurso de procesamiento disponible generando así un nuevo dataset (10000 datos × 84 columnas) que posteriormente se subió a github.

Luego de esto, como el nuevo dataset no tenía valores nulos se llevó a cabo una eliminación de 500 filas de tres columnas con datos poco relevantes para nuestro fin. Por lo cual, se cumple el requisito de al menos tener un 5% de datos faltantes en al menos 3 columnas (5% en 4 columnas). Una vez realizado, las siguientes columnas fueron afectadas:

```
na_values = df_modified.isna().sum()
na_values[na_values != 0]
```

```
Fwd PSH Flags     500
 Bwd PSH Flags    500
 Fwd URG Flags    500
 Bwd URG Flags    500
```

Una vez cumplido este requerimiento, se procedió a analizar las variables categóricas, al observar que no contaba con la cantidad de variables categóricas necesarias para cumplir la exigencia de tener al menos el 10% de las columnas categóricas en el proyecto, fue necesario realizar un procedimiento para convertir variables numéricas a categóricas.

Para cumplir con la cantidad de variables categoricas se seleccionaron 5 columnas con datos fáciles de categorizar (pocas variables únicas). Se definió la categorización de cada una como se muestra a continuación:

Down/Up Ratio, PSH Flag count, ACK Flag Count, RST Flag Count, SYN Flag Count.

```
def categorizar(valor):
    if valor <= 3:
        return 'low'
    elif valor >= 5:
        return 'high'
    else:
        return 'medium'
```

```
def categorizar_psh(valor):
    if valor == 0.0:
        return 'send'
    else:
        return 'receive'
```

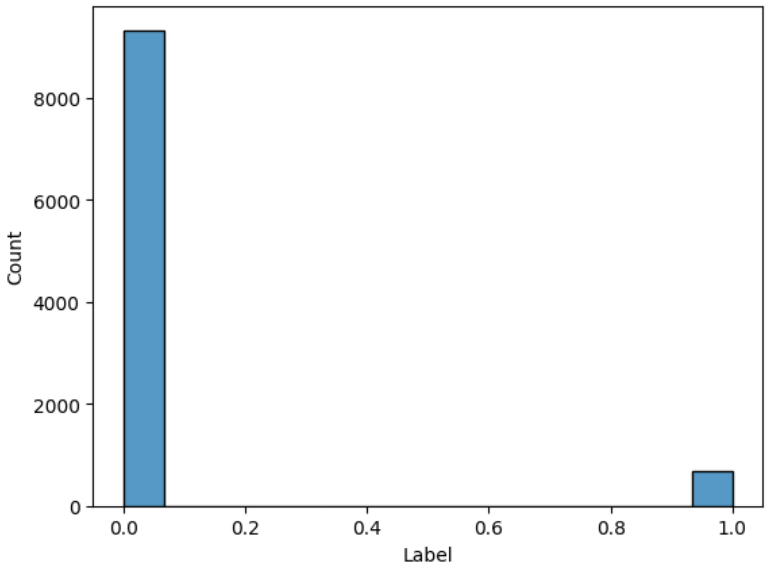
```
def categorizar_act(valor):
    if valor == 0.0:
        return 'not require'
    else:
        return 'require'
```

```
def categorizar_rst(valor):
    if valor == 0.0:
        return 'reset'
    else:
        return 'no reset'
```

```
def categorizar_syn(valor):
    if valor == 0.0:
        return 'no conection
    else:
        return 'conection'
```

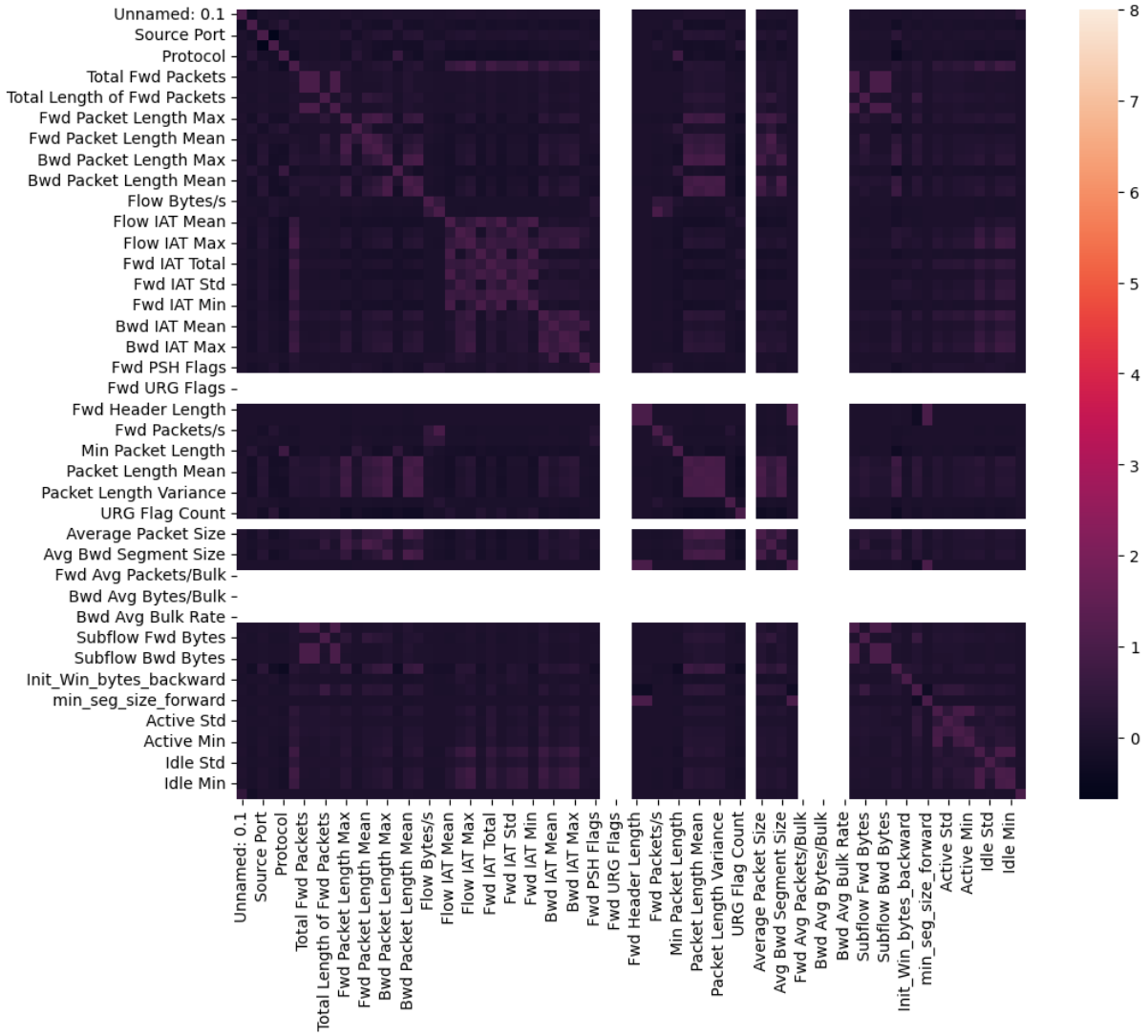
La variable objetivo es ‘ Label’ y se decidió que todo lo relacionado con valores android (Android_Adware, Android_Scareware, Android_SMS_Malware) se nombraría con cero y los relacionados con benigno (Benign) se nombrarían con uno.

Luego se grafica variable de salida, se encuentra que el valor 0 es el más presente en el dataset.



Tras haber realizado la selección de datos y habiendo cumplido con todos los requerimientos del proyecto, se encontró lo siguiente:

- Tamaño del dataset: El tamaño del dataset es (1000, 84).
- Valores nulos: Las columnas Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, tienen 500 valores nulos.
- Variables categóricas: Las columnas Flow ID, Source IP, Destination IP, Down/Up Ratio, PSH Flag count, ACK Flag Count, RST Flag Count, SYN Flag Count, son categóricas.
- Tipos de datos: Los tipos de datos que están presentes son float64 en su mayoría, algunos int64 y otros object en las variables que se convirtieron a categóricas.
- Inspección de datos numéricos: Se calcula el promedio, la desviación estándar, valor mínimo, máximo y percentiles de las columnas numéricas además se graficó la matriz de correlaciones.



ITERACIONES DE DESARROLLO

Para la realización de este proyecto se siguieron las siguientes iteraciones implementando los diferentes modelos que se encuentran en el notebook del proyecto las cuales se van a describir acontinuacion.

RF (Random Forest) Para este modelo, se realizaron corridas con árboles [5,10,20,50,100, 150] y variables para la selección del mejor umbral [5,10,15,20,25] a través de combinaciones entre sí.

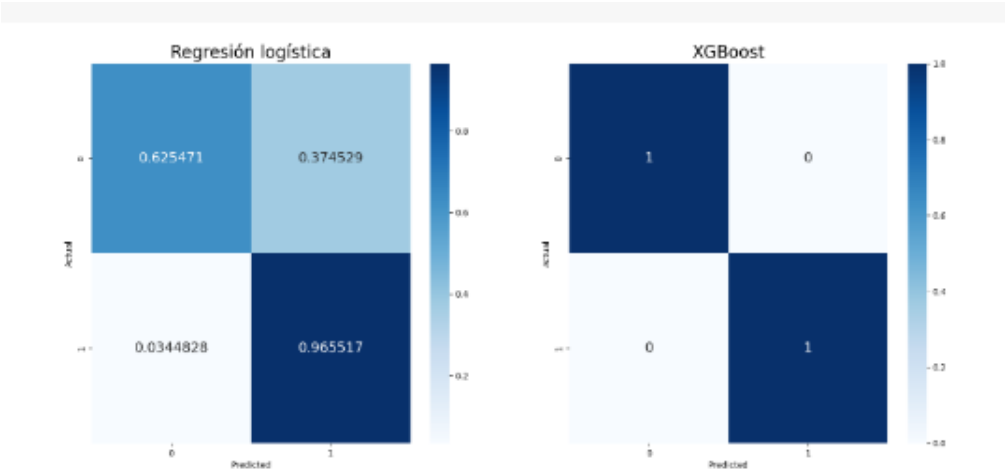
Gradient Boosting Con la función GradientBoostingClassifier de sklearn se varía el n_estimators entre los valores [20,50,100,200,300], por ejemplo, empleando un valor de n_estimators en 300

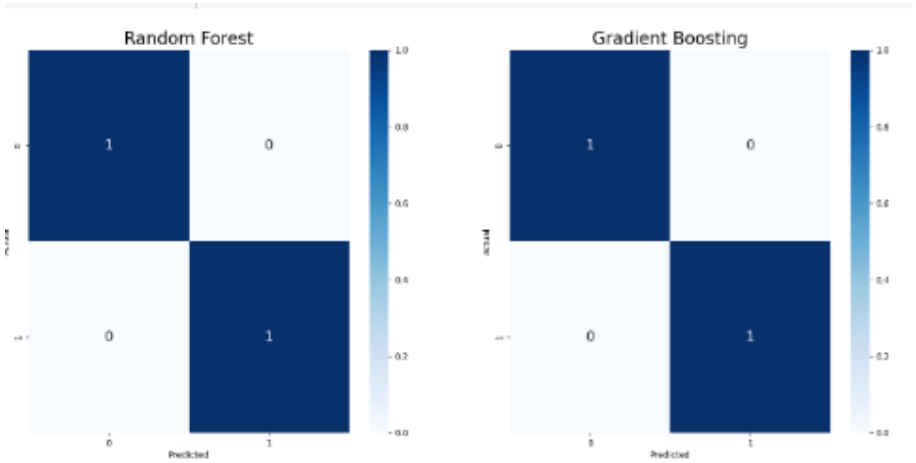
SVM (Support Vector Machine) Para este caso se realizó un experimento con kernel ‘rbf

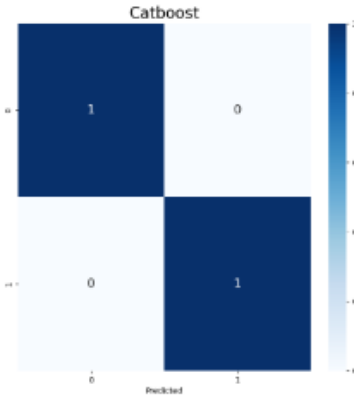
Redes neuronales: se tiene una red neuronal con la siguiente configuración:
- Una capa densa de 128 neuronas y activación ReLu
- Una capa de Dropout de 0.3
- Una capa densa de 64 neuronas y activación ReLu
- Una capa densa de 32 neuronas y activación ReLu
- Una capa densa de 16 neuronas y activación ReLu
- Una capa de Dropout de 0.3
- Una capa densa de salida con 1 neurona y activación sigmoide.

Para esta configuración se utilizó el optimizador Nadam con un learning rate de 0.001, se utilizó como función de pérdida Categorical Cross Entropy. El entrenamiento se realizó con un batch size de 256 durante 30 épocas y un split de validación del 20%.

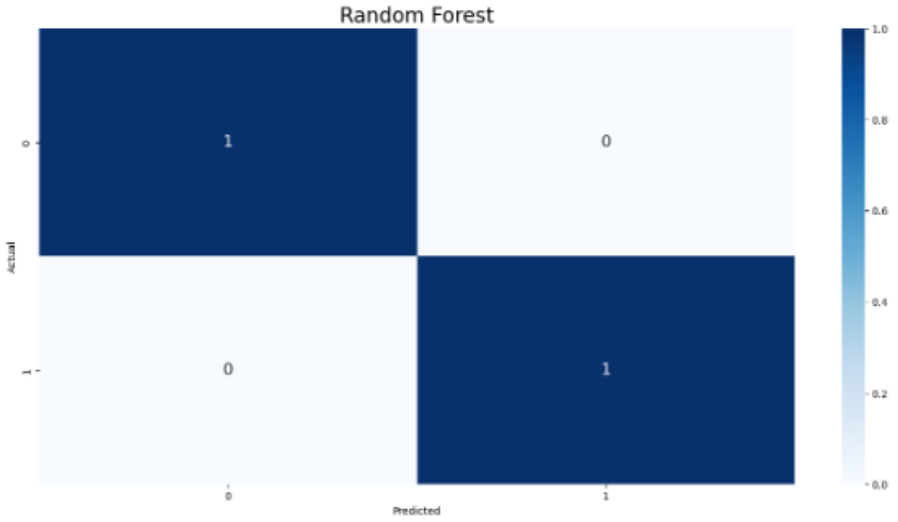
Acontinuacion se presentan las matrices de confusión creadas para los 5 modelos (regresión logística, XGBoost, Random Forest y Gradient Boosting)

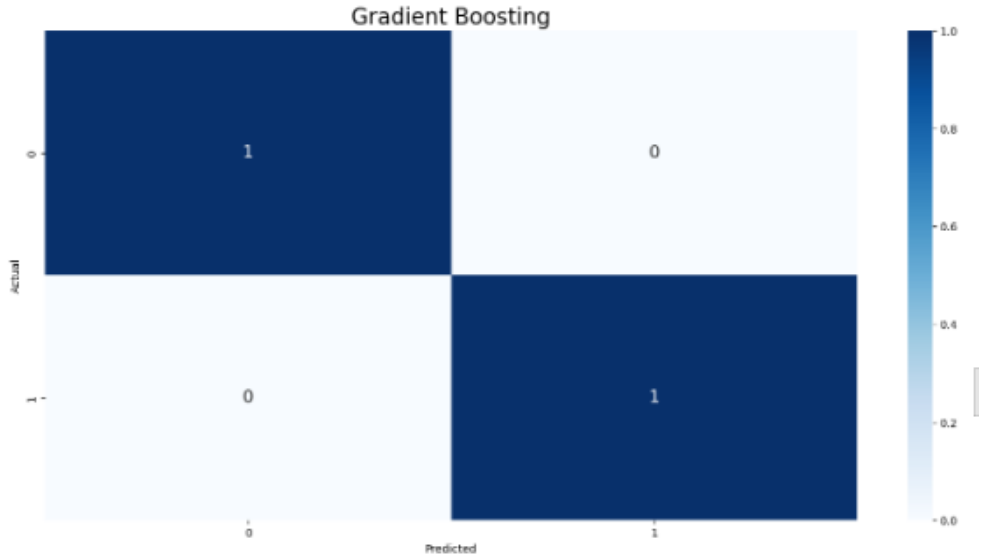






En la siguiente grafica presentamos el mejor modelo entregado considerando todos los modelos anteriores con respecto a los datos iniciales





RETOS Y CONSIDERACIONES DE DESPLIEGUE

Los retos que se tenían al principio del proyecto en cuanto al rendimiento del modelo
eran:

- Valor de precisión por encima del 95%
- Tasa de falsos positivos y falsos negativos inferior al 5%

La tasa de acierto se observa mayor al 95% debido a que los modelos están clasificando de forma desbalanceada siendo la clase mayoritaria la que nos indica que se tiene un rendimiento mucho mayor al 85%, por lo tanto no es una métrica

que tenga mucha influencia a la hora de desplegar el modelo. Los valores de sensibilidad de todos los modelos son inferiores al 80% por lo tanto no se cumple con el requerimiento asociado a esta métrica que nos indica la fracción de los verdaderos positivos clasificados. Los valores de especificidad de todos los modelos son inferiores al 90% por lo tanto no se cumple con el requerimiento asociado a esta métrica que nos indica la fracción de los verdaderos negativos. Se concluye que no se pueden desplegar los modelos debido a que no se cumplieron los requerimientos mínimos para que se tenga un buen rendimiento a nivel de producción.

CONCLUSIONES

Al realizar el proyecto podemos llegar a las siguientes conclusiones

La mejor manera de obtener un buen modelo de machine learning es aplicando diferentes algoritmos predictivos y combinación de hiperparámetros, en este caso de clasificación binaria sobre el mismo conjunto de datos, debido a que los algoritmos más comunes como regresión logística no siempre ofrecen el mejor resultado. Análogamente es recomendable evaluar el desempeño de cada modelo con diversas métricas de validación y no tomar decisiones solo con la exactitud (accuracy), sino también analizar otras métricas como el AUC, curva ROC, precisión y matriz de confusión.

Regresión Logística (con SMOTE): El modelo de Regresión Logística aplicado junto con la técnica SMOTE mostró una precisión del 85.19%. Si bien esto indica una capacidad moderada para clasificar correctamente las clases de malware y aplicaciones legítimas, es posible que se requieran técnicas y modelos más avanzados para mejorar la precisión y la capacidad de detección de malware.

XGBoost y Random Forest: Tanto el modelo XGBoost como el modelo Random Forest lograron una precisión perfecta del 100% en todas las métricas de evaluación para la detección de malware en Android. Esto indica un rendimiento sobresaliente y una alta capacidad de clasificación. Estos modelos podrían ser opciones muy efectivas para la detección de malware en este contexto.

En resumen, los modelos de XGBoost y Random Forest demostraron ser altamente efectivos en la detección de malware en aplicaciones de Android, logrando una precisión del 100% en todas las métricas de evaluación. Estos modelos podrían ser opciones recomendadas para la detección de malware en proyectos relacionados con Android. Sin embargo, es importante tener en cuenta que la detección de malware es un desafío en constante evolución, por lo que es

fundamental mantenerse actualizado con las últimas técnicas y enfoques para garantizar la eficacia continua en la detección de nuevas amenazas