Athena (Android Malware Detection)

Mateo Aguirre Duque

Departamento de ingeniería de sistemas

Universidad de Antioquia

Inteligencia Artificial para las Ciencias e Ingenierías

Raul Ramos P.

Mayo 28 de 2023

Contenido

Introducción

La inteligencia artificial (IA) ha revolucionado múltiples campos al ofrecer un potencial sin precedentes para analizar grandes volúmenes de datos. En particular, el aprendizaje automático dentro del ámbito de la detección de malware en dispositivos Android se presenta como una poderosa herramienta. La constante evolución de las amenazas y la sofisticación de los ataques plantean desafíos significativos. Sin embargo, gracias a la IA y sus técnicas de aprendizaje automático, es posible desarrollar modelos precisos capaces de identificar y clasificar aplicaciones maliciosas.

Este trabajo se enfocará en explorar las aplicaciones prácticas de la inteligencia artificial en la detección de malware en dispositivos Android, abordando el proceso de construcción de modelos de aprendizaje automático. Se analizarán características relevantes, como las actividades en la red, para identificar comportamientos sospechosos y proteger a los usuarios de posibles amenazas. Además, se examinarán los beneficios de implementar estos modelos en la clasificación de nuevas aplicaciones como maliciosas o no maliciosas.

Planteamiento del problema

En la actualidad, nuestros dispositivos móviles almacenan una gran cantidad de información personal, bancaria y profesional, y también son utilizados para acceder a redes sociales, descargar archivos y aplicaciones. Sin embargo, esta conveniencia viene acompañada de un riesgo significativo: la posible presencia de archivos y aplicaciones con una procedencia dudosa, lo que compromete la integridad y seguridad de nuestros dispositivos.

El objetivo de este proyecto es abordar este problema mediante la creación de modelos predictivos capaces de detectar patrones en las actividades de red de nuestros dispositivos. Estos modelos se enfocarán en identificar la presencia de aplicaciones potencialmente maliciosas o que contengan malware. Para lograrlo, se analizarán las actividades de red, como las comunicaciones salientes y entrantes, las solicitudes de datos y las conexiones establecidas por las aplicaciones en uso.

El desarrollo de estos modelos predictivos permitirá a los usuarios de dispositivos móviles contar con una herramienta eficaz para identificar y prevenir la instalación de aplicaciones maliciosas en sus dispositivos. Esto contribuirá a salvaguardar la seguridad y privacidad de la información personal, bancaria y profesional, brindando una capa adicional de protección ante las crecientes amenazas cibernéticas.

Dataset

El dataset utilizado en este proyecto, denominado Android_Malware.csv, está disponible en la plataforma Kaggle. Contiene información sobre diferentes comportamientos en la red de dispositivos Android. Cada entrada en el dataset representa un registro de comportamiento,

con un total de 355,630 registros y 85 columnas de datos. La última columna, llamada "Label", indica el tipo de malware que potencialmente afecta al dispositivo. Sin embargo, el objetivo del proyecto no es clasificar el tipo de malware, sino indicar si el dispositivo está infectado o no. Por lo tanto, se realizará una modificación en la última columna para reflejar únicamente si el dispositivo está infectado o no.

Características

El dataset contiene información sobre diversas características extraídas de flujos de red en dispositivos. Estas características incluyen la duración de los flujos, el número de paquetes en dirección hacia adelante y hacia atrás, el tamaño total de los paquetes, la longitud de los paquetes, la velocidad de flujo de bytes y paquetes, el tiempo entre paquetes, los indicadores de banderas (como FIN, SYN, RST, etc.), la relación de descarga y carga, y varias estadísticas de tiempo de actividad e inactividad de los flujos. Estas características proporcionan información relevante para detectar y analizar posibles malwares en dispositivos.

Métrica de desempeño

Para evaluar el desempeño de los modelos de este proyecto se usaron las métricas recall, precision y f1 score que son métricas comúnmente utilizadas para evaluar el rendimiento de un modelo de clasificación. A continuación, se explica cómo funcionan y se detallan las ecuaciones correspondientes a cada métrica:

Recall (Recuperación)

El recall mide la capacidad de un modelo para identificar correctamente todos los casos positivos. Es decir, cuántos de los casos positivos reales fueron correctamente detectados por el modelo. Se calcula dividiendo el número de verdaderos positivos (TP) entre la suma de los verdaderos positivos y los falsos negativos (FN).

recall = TP / (TP + FN)

Precision (Precisión)

La precisión evalúa la exactitud de las predicciones positivas realizadas por el modelo. Mide cuántos de los casos identificados como positivos por el modelo son realmente positivos. Se calcula dividiendo el número de verdaderos positivos (TP) entre la suma de los verdaderos positivos y los falsos positivos (FP).

precision = TP / (TP + FP)

F1 score

La puntuación F1 es una medida que combina tanto el recall como la precisión en un solo valor. Es útil cuando se busca un equilibrio entre estas dos métricas. Se calcula como la media armónica entre el recall y la precisión.

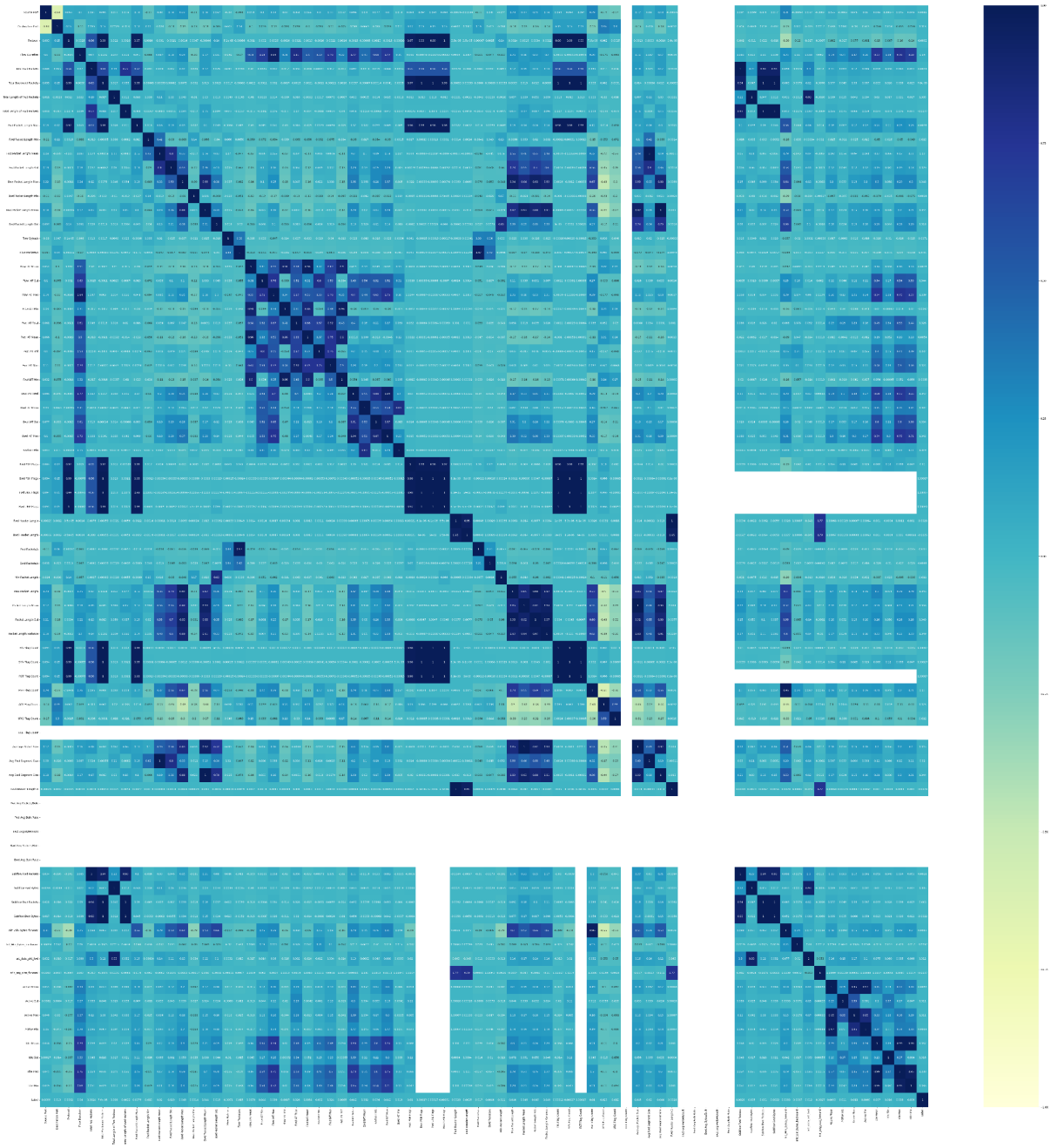
f1_score = 2 * (recall * precision) / (recall + precision)

Exploración de datos

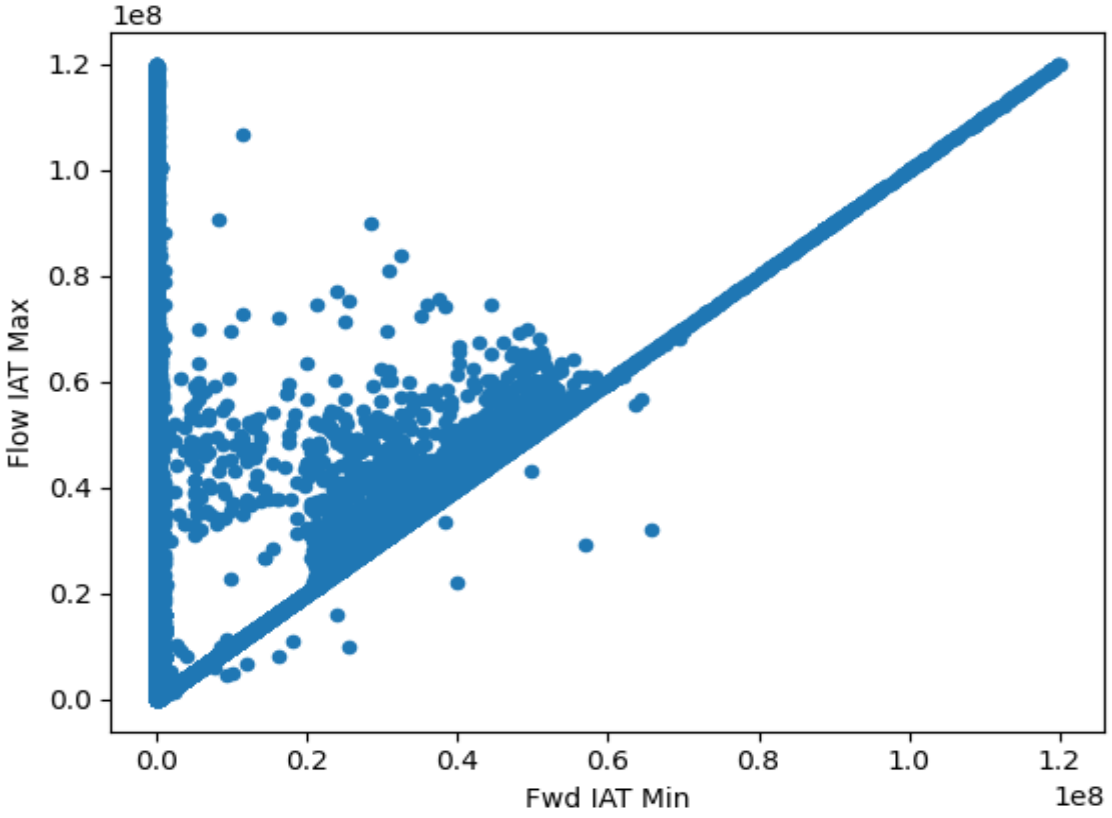
Durante la exploración de datos se revisó las correlaciones lineales entre la variable objetivo y todas las otras variables, en este análisis se descubrió que ninguna de las variables muestra

una correlación significativa con la variable objetivo, pero sí poseen correlaciones fuertes entre ellas, algunas de las correlaciones más fuertes son:

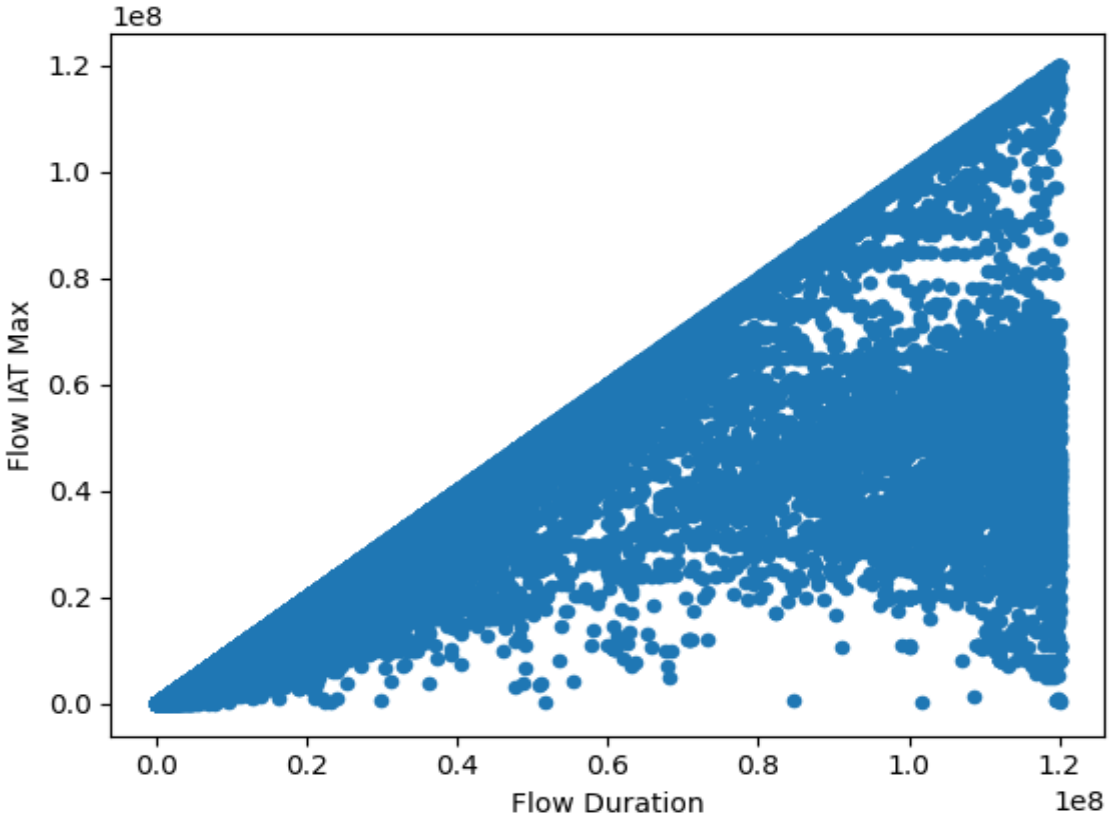
- Fwd IAT Min con Flow IAT Max
- Flow Duration con Flow IAT Max
- Bwd Packet Length Max con Max Packet Length



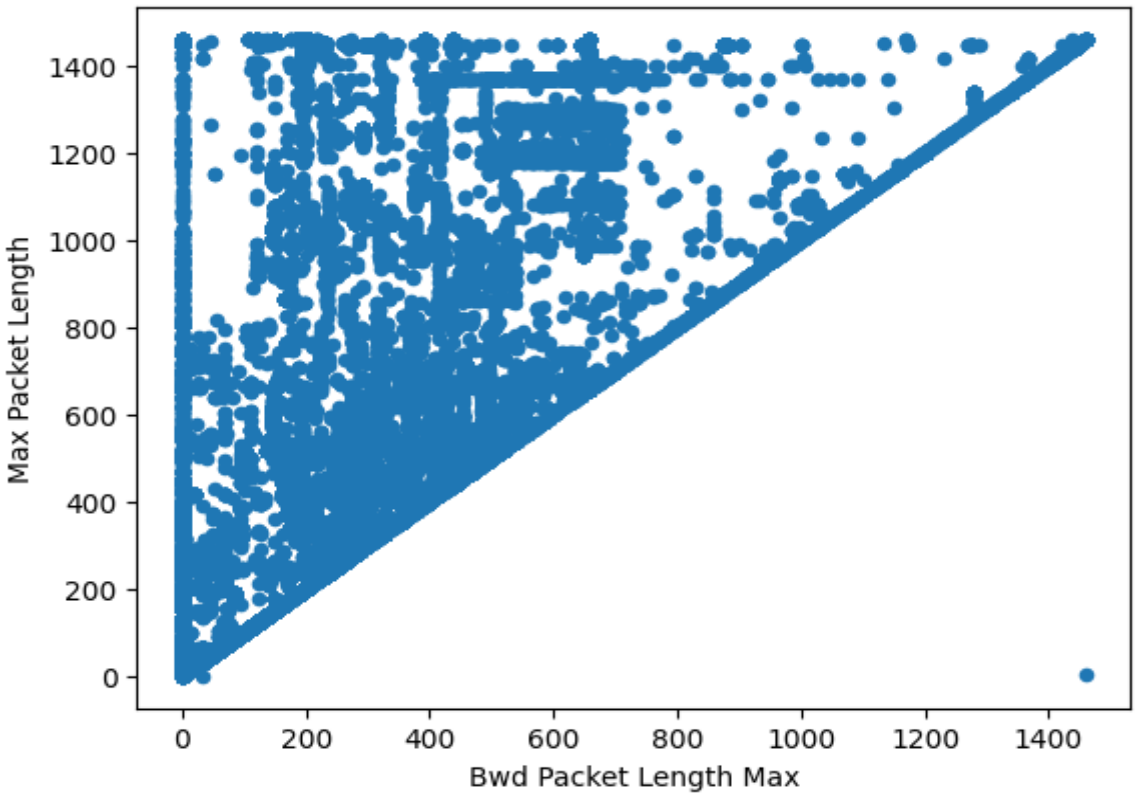
Mapa de calor



Fwd IAT Min vs Flow IAT Max



Flow Duration vs Flow IAT Max



Bwd Packet Length Max vs Max Packet Length

Posteriormente, se examinó la cantidad de valores faltantes en el dataset de Kaggle, este dataset está bastante completo a nivel de datos faltantes, ya que solo 5 registros presenta este problema y no corresponde ni al 1% de los datos. Dentro del análisis de los valores faltantes se logró percibir que el problema que si tiene el dataset son los valores atípicos.

Como para el proyecto es necesario que el dataset posea como mínimo un 5% de valores faltantes, se creó una función que coge 13 columnas del dataset y reemplaza sus valores de forma aleatoria hasta alcanzar ese 5% de valores faltantes.

Iteración I

Preprocesado

El llenado de los datos se realizó con 2 métodos diferentes, uno basado en los cuartiles y la distribución normal, y otro basado únicamente en la distribución normal. Los cuantiles se usan con el fin de quitar posibles valores outliers y mejorar el llenado con valores aleatorios a partir de la distribución normal. Después del llenado de los datos faltantes se realizó una reducción del problema con el fin de remover las 4 categorías de la variable objetivo y establecer una escala binaria que indica si el registro posee malware o no lo posee.

Para finalizar con el preprocesado del dataset se le realizó una partición de 70% de valores para entrenar, 20% de valores para test y 10% para visualizar el rendimiento del modelo y saber que tan bueno es en comparación a otros modelos.

Modelos

Random Forest Classifier

El modelo Random Forest Classifier es un algoritmo de aprendizaje automático que se basa en la combinación de múltiples árboles de decisión. Cada árbol de decisión se entrena con diferentes subconjuntos de datos y características aleatorias, y luego se combina para obtener predicciones más precisas y estables. En esta iteración se entrenó el modelo con los parámetros “n_estimators” igual a 10 y “random_state” igual a 0, con esta configuración se logró el siguiente desempeño.

precision = 0.9358192442981959
recall = 0.9943484944389185
f1 = 0.9641964664104401

Con estos resultados da la impresión de que el modelo está sobre ajustado ya que el dataset aún tiene ciertos problemas con valores atípicos y con sesgos.

Iteración II

Preprocesado

En esta iteración, antes de realizar la partición de los datos se implementó un modelo LOF que es una técnica utilizada en inteligencia artificial para detectar valores atípicos en conjuntos de datos, con este modelo se eliminaron todos los datos atípico y pasamos de tener 355.626 registros a tener 316.187. Al eliminar valores atípicos, se redujo el impacto de puntos extremos que pueden distorsionar los resultados y afectar negativamente el rendimiento del modelo. Con esto se mejoró la precisión y confiabilidad de las predicciones al proporcionar una representación más precisa de los datos subyacentes.

Además de eliminar estos valores atípicos en esta iteración se implementó una metodología para balancear el dataset, ya que en la iteración anterior se detectó que los datos estaban muy sesgados y esto puede ocasionar que los modelos queden sobre ajustados. La metodología de balanceo que se usó fue que por aca 2 datos con etiqueta de malware hay 1 dato con etiqueta benigno, esto redujo el dataset a tan solo 63.477 registros.

Después de estas dos operaciones de preprocesado de datos se realizó la partición de datos de la misma forma que en la iteración anterior.

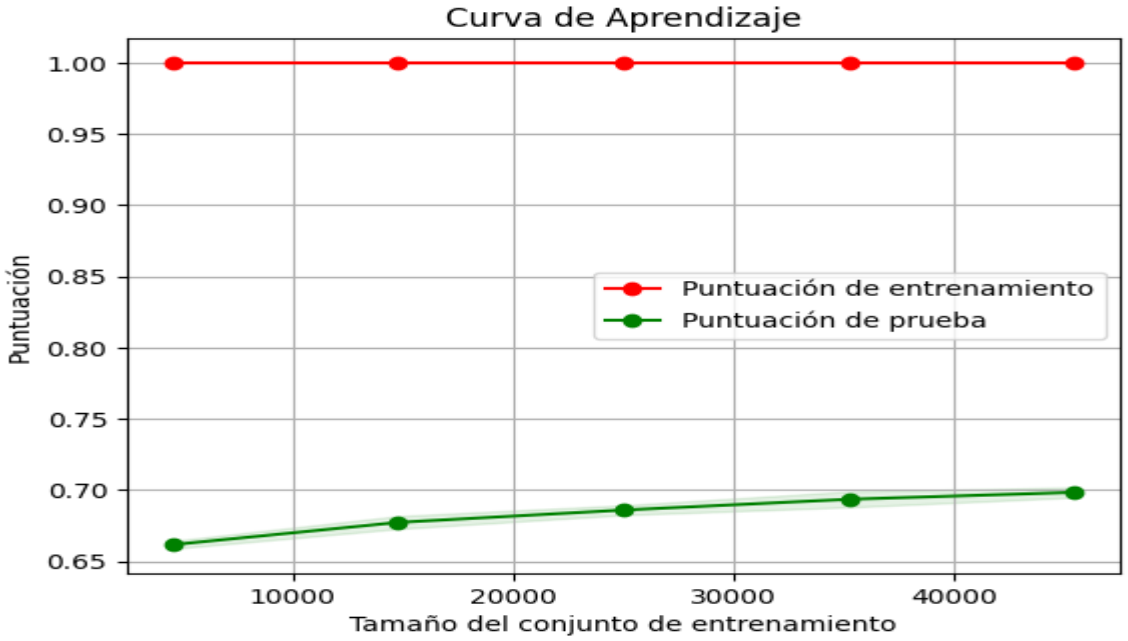
Modelos

Random Forest Classifier

El modelo Random Forest Classifier es un algoritmo de aprendizaje automático que se basa en la combinación de múltiples árboles de decisión. Cada árbol de decisión se entrena con diferentes subconjuntos de datos y características aleatorias, y luego se combina para obtener predicciones más precisas y estables. En esta iteración se entrenó el modelo con los parámetros “n_estimators” igual a 10 y “random_state” igual a 0, con esta configuración se logró el siguiente desempeño y la siguiente curva de aprendizaje.

precision = 0.7242277992277992
recall = 0.8894025604551921
f1 = 0.7983613534794637

Curva de aprendizaje



Curva de aprendizaje Random Forest Classifier

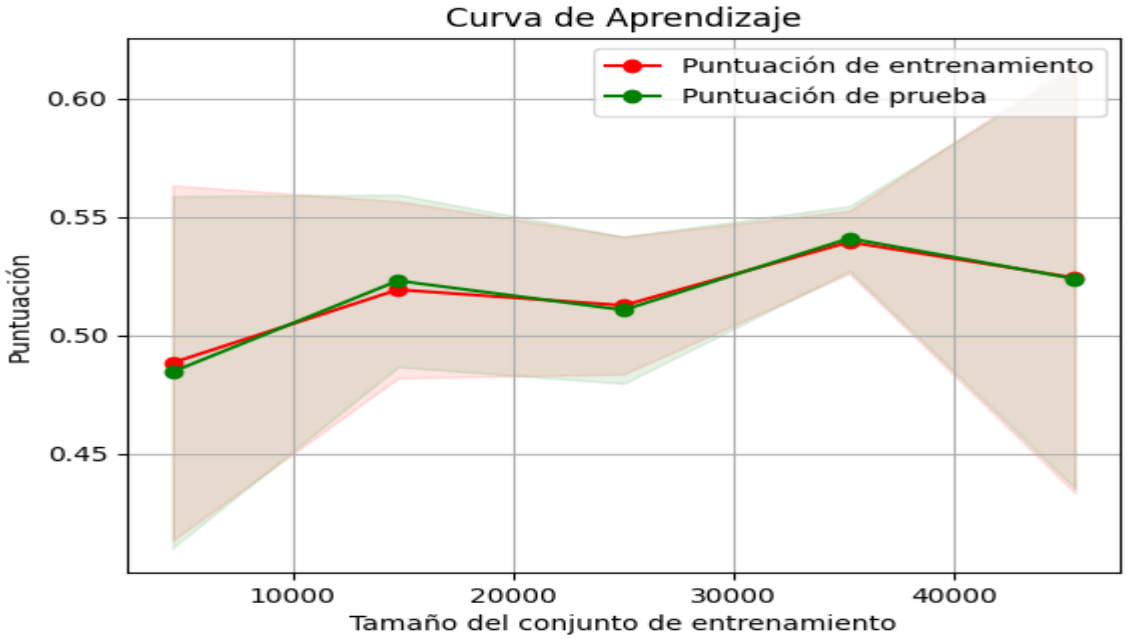
En esta curva se nota que el modelo está muy sobre ajustado a los datos de entrenamiento y que los datos de test no son capaz de pasar del 0.70 lo cual es una métrica algo baja.

Linear SVC

El modelo LinearSVC (Support Vector Classifier) es un algoritmo de aprendizaje automático utilizado para la clasificación de datos. Se basa en el concepto de máquinas de vectores de soporte y se utiliza para separar y clasificar datos en diferentes categorías. A diferencia de otros clasificadores lineales, LinearSVC utiliza una formulación de pérdida hinge que maximiza el margen entre las muestras de diferentes clases. Esto significa que busca la mejor separación lineal entre las clases, tratando de encontrar un hiperplano óptimo que divida los datos. En esta iteración se entrenó el modelo con el parámetro "max_iter" igual a 10000, con esta configuración se logró el siguiente desempeño y la siguiente curva de aprendizaje.

precision = 0.6687229437229437
recall = 0.7324561403508771
f1 = 0.6991400769404843

Curva de aprendizaje



Curva de aprendizaje Linear SVC

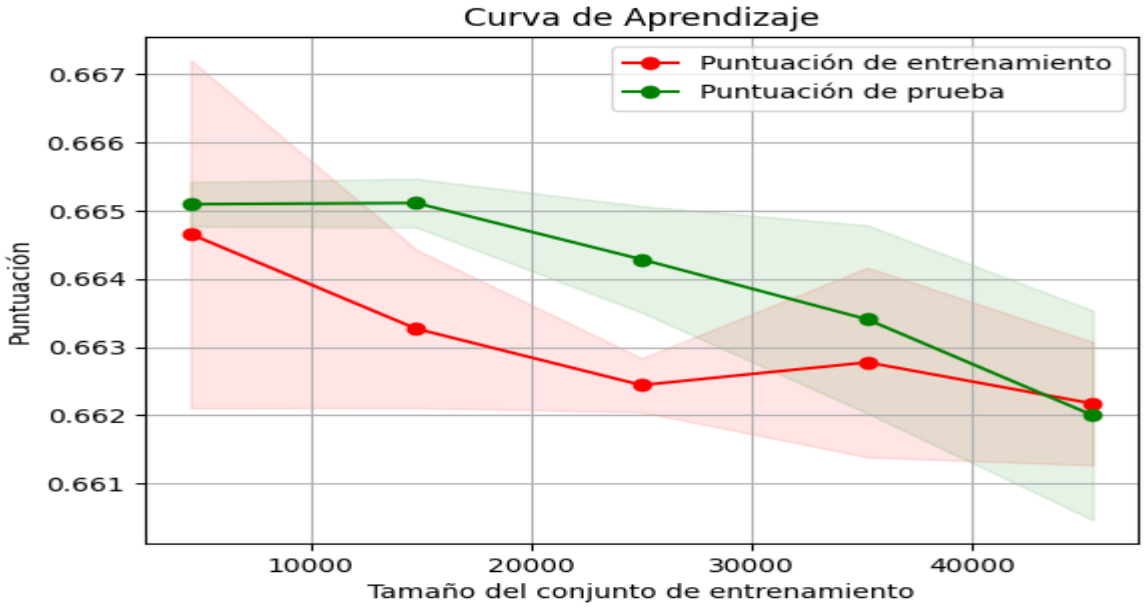
En esta curva se nota que el modelo tiene un comportamiento muy similar para todos los datos de entrenamiento y todos los datos de prueba, pero no no es capaz de pasar de 0.55 lo cual es una métrica muy baja.

SVC con kernel sigmoid

El modelo SVC con kernel sigmoid es un algoritmo de aprendizaje automático utilizado para la clasificación de datos. A diferencia del kernel lineal o el kernel radial utilizado en otros modelos SVC, el kernel sigmoid utiliza una función de activación sigmoidal para transformar los datos y encontrar una frontera de decisión no lineal. La función de activación sigmoidal se asemeja a una curva en forma de "S" y se utiliza para mapear los valores de entrada a un rango entre 0 y 1. En esta iteración se entrenó el modelo con el parámetro “gamma” igual a 1, con esta configuración se logró el siguiente desempeño y la siguiente curva de aprendizaje.

precision = 0.6675218471899302
recall = 0.9869606448553817
f1 = 0.7964034626237505

Curva de aprendizaje



Curva de aprendizaje Linear SVC

En esta curva se nota que el modelo tiene un comportamiento muy particular ya que entre más datos le pasamos al modelo poco a poco va bajando su rendimiento, lo que alcanza a bajar puede ser considerado despreciable pero es interesante ver ese comportamiento.

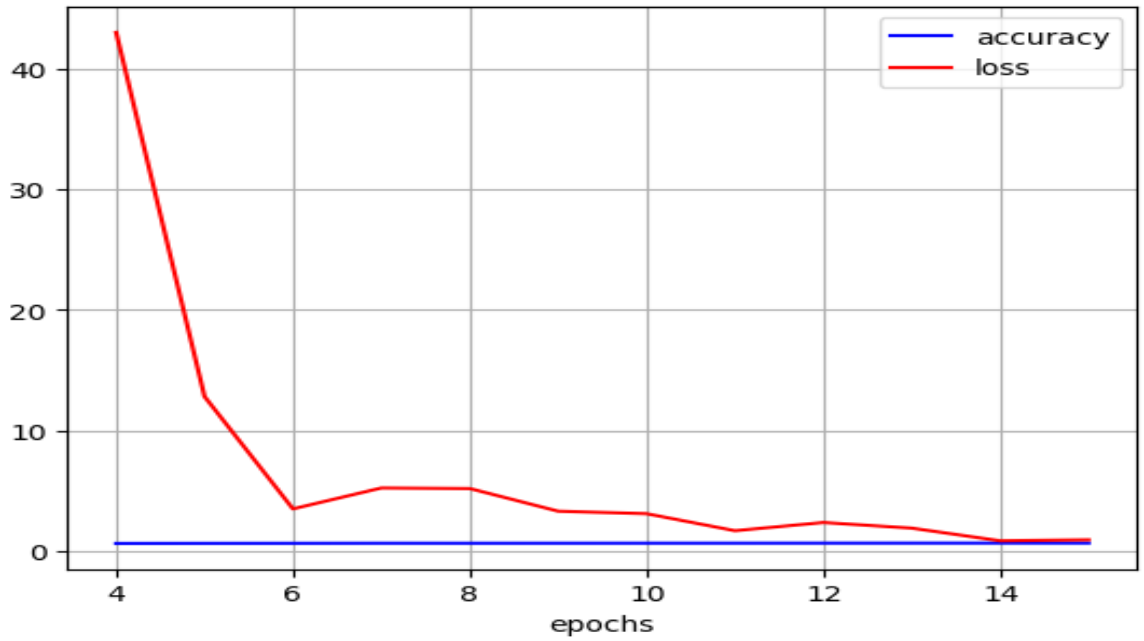
Red neuronal Sequential de Keras

Las redes neuronales Sequential de Keras son un tipo de modelo utilizado en aprendizaje profundo para resolver problemas de clasificación, regresión y otras tareas de aprendizaje automático. Estas redes se componen de una secuencia lineal de capas de neuronas interconectadas. En esta iteración se entrenó el modelo con una capa de entrada de tipo dense con funcion de activacion relu y las unidades en 512, en las capas ocultas se usaron dos capas tipo dense con funcion de activacion relu y las unidades en 256, y por último se usó una capa de salida de tipo dense con función de activación sigmoid y las unidades en 1.

Como esta red neuronal retorna es una probabilidad y no una clasificación directa se establece un threshold de 0.65 para establecer si posee o no malware. En el entrenamiento se usaron 15 épocas, un “batch_size” igual a 126 y el optimizador adam, con esta configuración se logró el siguiente desempeño y la siguiente curva de aprendizaje.

precision = 0.6706202917134156
recall = 0.9822749148961145
f1 = 0.7970662475591751

Curva de aprendizaje



Curva de aprendizaje Red Neuronal

KMeans

Los modelos KMeans son una técnica popular en el campo del aprendizaje automático no supervisado utilizada para agrupar datos en clústeres. El algoritmo KMeans busca agrupar los datos en k grupos, donde k es un número predeterminado de clústeres definido por el usuario. En esta iteración se entrenó el modelo con los parámetros “init” igual a random, “n_clusters” igual a 2, “n_init” igual a 10, “max_iter” igual a 500 y “random_state” igual a 0, con esta configuración se logró el siguiente desempeño.

precision = 0.6713744075829384
recall = 0.8314356145087451
f1 = 0.7428811159473492

Iteración III

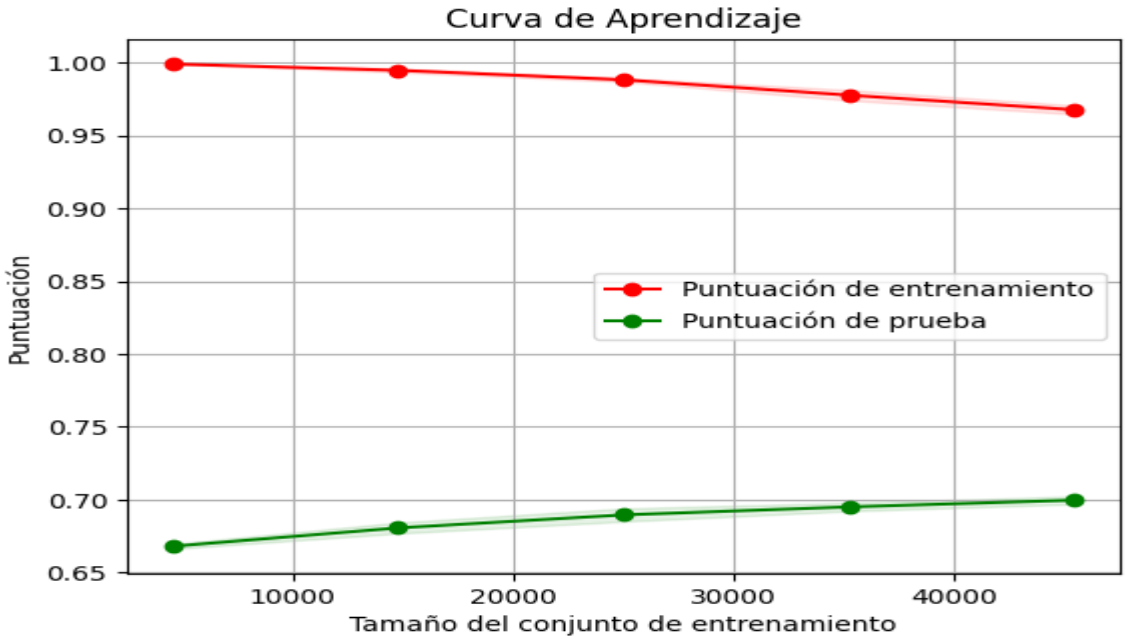
Modelos

Random Forest Classifier

En esta iteración a este modelo se le trató de quitar el sobre ajuste estableciendo los siguientes valores de parámetros: “n_estimators” igual a 200, “max_depth” igual a 22 y “random_state” igual a 0, con esta nueva configuración se logró el siguiente desempeño y la siguiente curva de aprendizaje.

precision = 0.7101789911408425
recall = 0.9312470365101944
f1 = 0.8058262385885732

Curva de aprendizaje



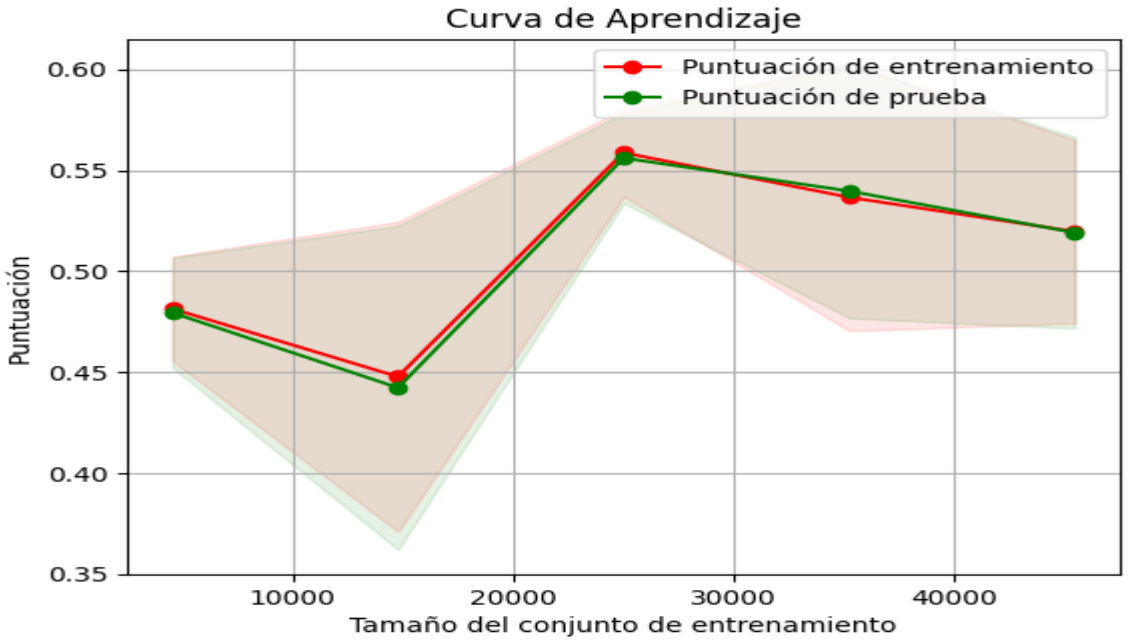
Curva de aprendizaje Random Forest Classifier

Linear SVC

En esta iteración a este modelo se le trató de mejorar el desempeño estableciendo un nuevo parámetro que es una semilla de inicio a la hora de entrenar, el parámetro es "random_state" al cual se le asigna un valor de 50, con esta nueva configuración se logró el siguiente desempeño y la siguiente curva de aprendizaje.

precision = 0.6749376963918085
recall = 0.7383831199620673
f1 = 0.7052363430512313

Curva de aprendizaje



Curva de aprendizaje Linear SVC

Red neuronal Sequential de Keras

En esta iteración a este modelo no se realizó ningún cambio en sus parámetros y en la estructura de esta.

KMeans

En esta iteración a este modelo no se realizó ningún cambio en sus parámetros ni en la forma de usarlo.

Retos y condiciones de despliegue del modelo

Para que alguno de estos modelos sea apto para ser desplegado se debe garantizar que su rendimiento sea óptimo ya que no es útil una herramienta que bloquee o notifique sobre posible malwares cuando realmente no existe ninguna amenaza, lo único que provocaría un comportamiento de este estilo es hacer que las personas confíen cada vez menos en la marca Android ya que todo el tiempo se la pasaría reportando posibles malwares que no existen.

Otro reto grande para que un modelo de estos sea desplegado es garantizar los permisos que estos requieren para acceder a toda la información de la red del dispositivo, ya que en estos tiempos cada marca monta sus propias capaz de modificación del dispositivo y cada aplicación puede administrar ciertos recursos de red a su conveniencia lo cual puede ser una limitante para recopilar en tiempo real toda la información necesaria de la red para realizar las predicciones.

Conclusiones

- Es muy importante estar validando que los modelos no presenten bias u overfitting, ya que muchas veces presentan buenos resultados en el entrenamiento o en las pruebas por fuera de muestra, pero al ir a producción o probando con los datos de demostración ya no les va tan bien.

- El proceso de preprocesado es muy importante para poder obtener buenos modelos ya que si los datos tienen muchos outliers, muchos huecos o simplemente estan malos, lo más posible es que los modelos queden muy mal entrenados y a la hora de ponerlos en un entorno más real no den los resultados esperados.
- Es fundamental evaluar y comparar el desempeño de los diferentes modelos, considerando métricas como la precisión, el recall y la puntuación F1, para seleccionar el modelo más adecuado y confiable para la detección de malware en dispositivos Android.

Bibliografía

- Android Malware Detection, kaggle 2023
  https://www.kaggle.com/datasets/subhajournal/android-malware-detection
- Evaluating the Performance of Machine Learning Models, Towards Data Science
  Abril 18 de 2020
  https://towardsdatascience.com/classifying-model-outcomes-true-false-positives-negatives-177c1e702810