Group Assignment - Data Driven Security

Group Assignment base repository for the Data Driven Security subject of the CyberSecurity Management Msc.

Creación de modelo estadístico para la detección de malware en APKs.

Requerimientos

Para este proyecto, necesitaremos un conjunto de APKs clasificadas según sean malware y benignas. En nuestro caso, usaremos un conjunto publicado por el Canadian Institute for Cybersecurity, el cual realizó en 2017 también un estudio para intentar determinar si una app es malware o no en base al tráfico de red que generaba. El estudio recoge los datos de 10.854 apps, de las cuales 4.354 fueron detectadas como malware y 6500 como benignas. El origen de las apps proviene de diferentes fuentes entre las cuales 6.000 tienen el origen en la play store Oficial. A partir del análisis de las app's se clasificaron en 4 categorías: Adware, Rasonmware, Scareware, SMS Malware.

Objetivos

Nuestro objetivo principal es intentar crear un modelo que nos permita determinar si una aplicación es malware o no usando únicamente los datos que se pueden extraer utilizando la herramienta `apkanalyzer` del SDK oficial de Android. Para ello, testearemos dos modelos de clasificación supervisada: KNN y SVM.

Adquisición de datos

Como hemos dicho, hemos utilizado las APKs a partir del proyecto publicado por el Canadian Institute for Cybersecurity. Para extraer la información correspondiente, realizamos un script en python, el cual generaba una tabla en CSV para que pudiesemos tratar los datos extraidos a partir de la herramienta `apkanalyzer`. Dicha herramienta nos permitió crear un conjunto de datos de cada apk con las siguientes columnas:

- `name`: El nombre de la aplicación
- `type`: El tipo de la aplicación, es decir, si es benigna o algún tipo de malware
- `apk_size`: El tamaño del archivo apk
- `n_files`: El número de archivos contenidos en el apk
- `n_permissions`: El número de permisos

Además de estos datos básicos, mediante el `apkanalyzer` también podemos obtener un árbol con las referencias del código de la aplicación, lo que nos permitió obtener columnas con el número de paquetes, clases, métodos y campos definidos y referenciados, así como el tamaño de éstos.

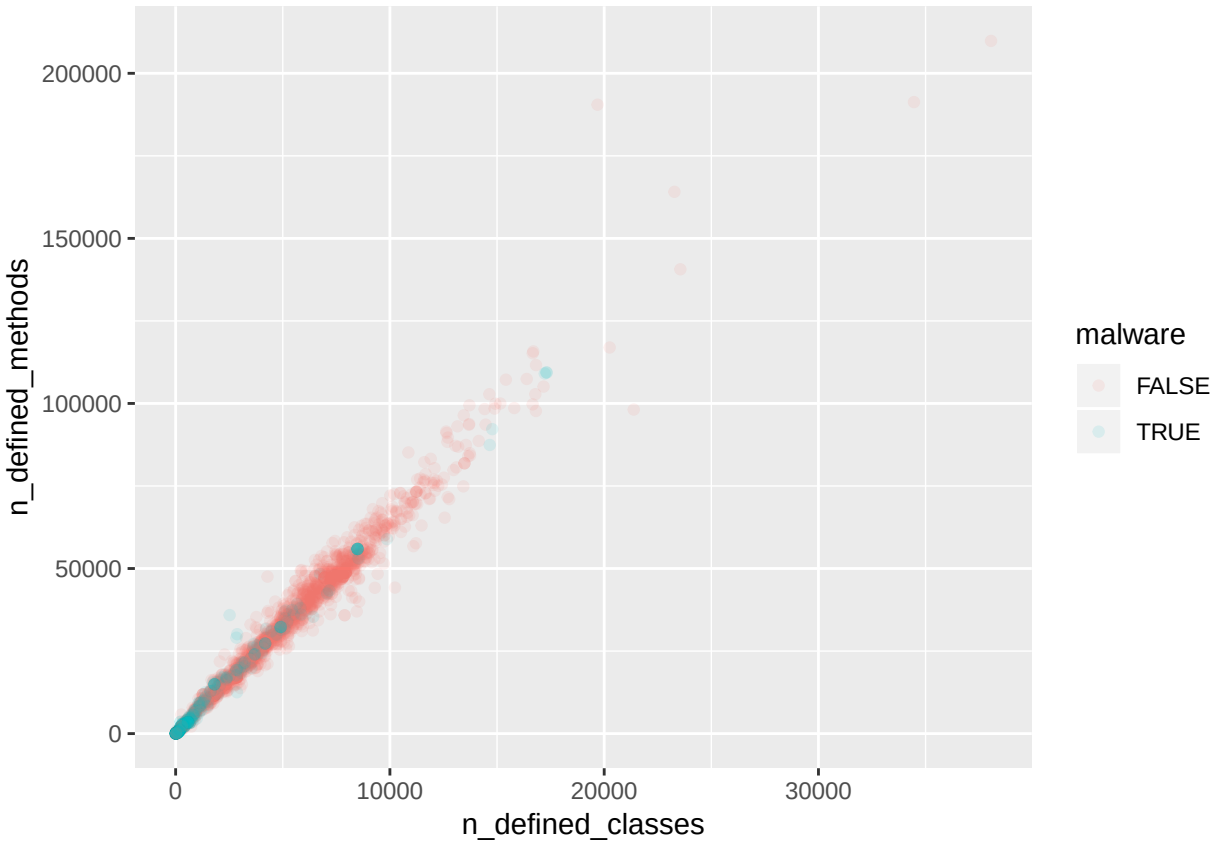
Las columnas correspondientes son: `n_defined_packages`, `n_referenced_packages`, `n_defined_classes`, `n_referenced_classes`, `n_defined_methods`, `n_referenced_methods`, `n_defined_fields`, `n_referenced_fields`.

Finalmente, tambien tenemos el campo `max_depth` que indica la profundidad máxima de un objeto en el árbol de código, lo que nos puede indicar en cierta manera la complejidad de la aplicación.

Limpieza y transformaciones

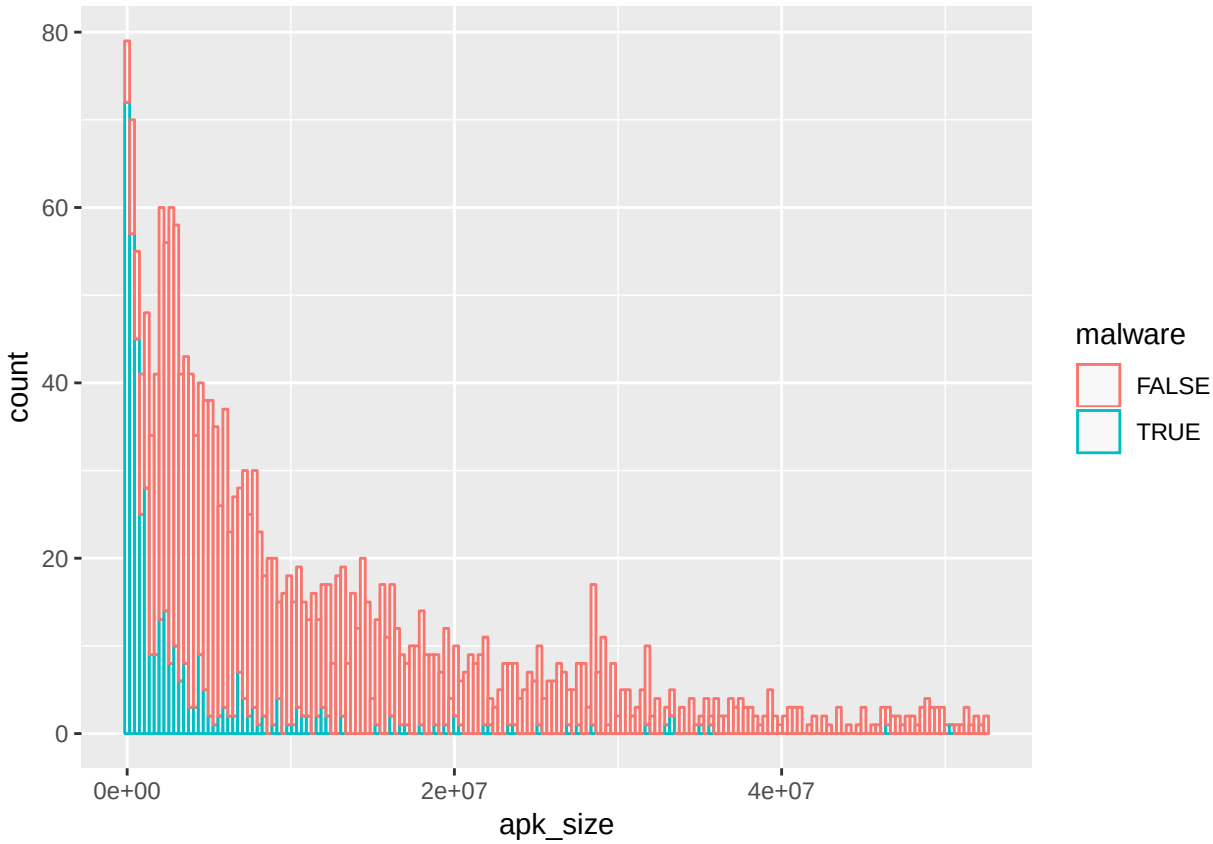
Primeramente, decidimos clasificar cada aplicación en función de si es o no malware, en vez de clasificarlas específicamente por el tipo de malware que eran para intentar mejorar los resultados y simplificar el análisis.

Primeramente, eliminamos buena parte de las columnas relacionadas con el código, puesto que todas ellas mantienen entre sí una relación lineal bastante marcada. Por ejemplo, si representamos el número de métodos definidos `n_defined_methods` en función del numero de clases definidas `n_defined_classes`, obtenemos el siguiente gráfico:

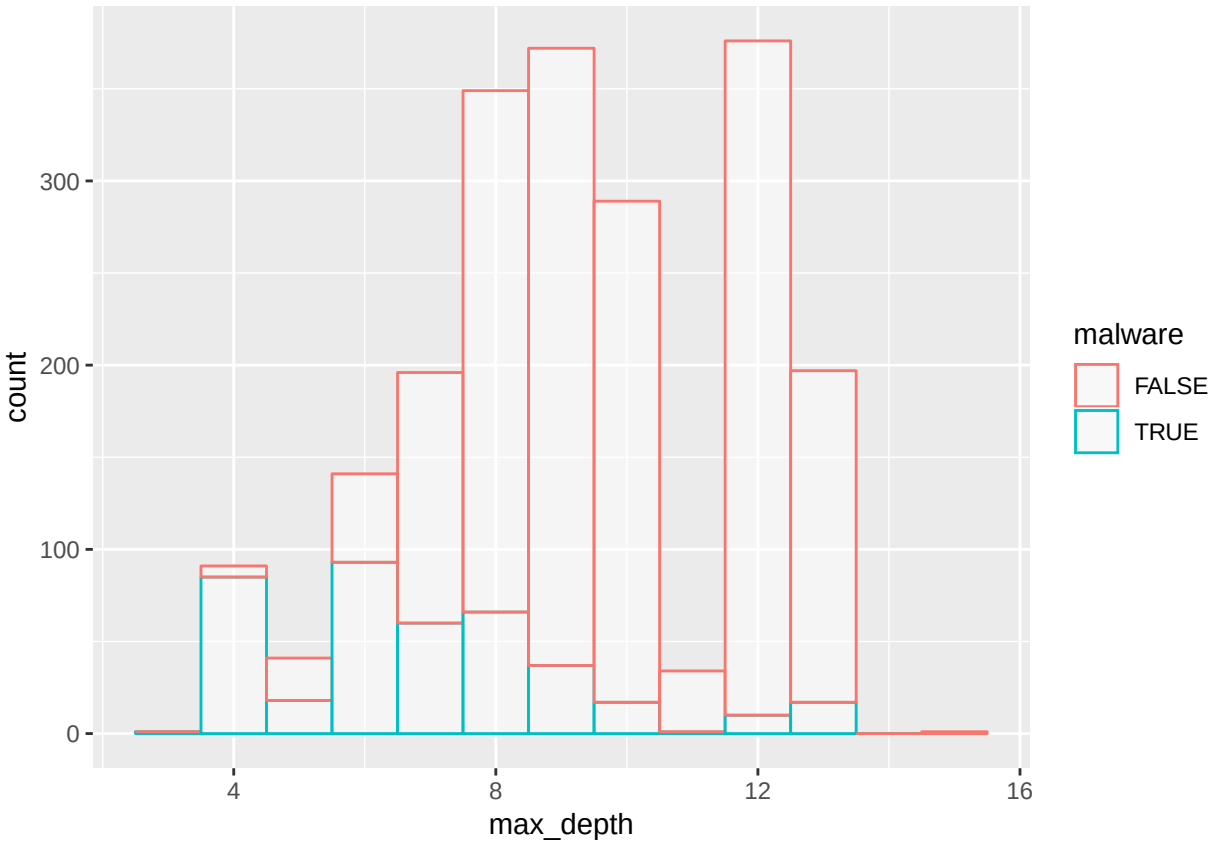


Por lo tanto, decidimos descartar todas estas columnas salvo la de `n_defined_methods` a la hora de realizar el test de nuestros modelos.

Si observamos un histograma del tamáño de las aplicaciones, podemos observar claramente como la mayoría de las que son malware son principalmente aplicaciones de poco tamaño, de manera que esta también puede ser una buena variable a tener en cuenta a la hora de implementar los modelos:



Finalmente, observando el histograma de la columna `max_depth` (complejidad),



decidimos crear una nueva variable para nuestro modelo en función de si esta columna era menor o igual o mayor a 4, ya que vimos en nuestras pruebas que este valor era el que mejor resultado nos daba en los tests.

Data analysis

Results / Conclusions.