UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3094 - Security Data Science
Sección 10
Jorge Andres Yass Coy

Proyecto 2 Predicción de Infección por Malware

Paula Camila González Ortega, 18398
Diana Ximena de León Figueroa, 18607
Maria Inés Vásquez Figueroa, 18250

GUATEMALA, 22 de abril de 2022

Abstract:
Data science has become a basic tool for detection, analysis and management of malware via machine learning models. For this project decision tree, Naive Bayes and KNN models have been chosen based on the telemetry from a computer to detect the probability of being infected by malware. Thereby, first we had to comprehend de data of the telemetry of Windows operating system, understand and explain evaluation metrics for the selection of the best model, all of these to promote research in data science applied to cybersecurity for better management and understanding of the upcoming malware technologies and how to avoid them through preventive methods. The dataset was provided by Microsoft and the data comes from personal and business computers. This dataset was made with the goal of understanding the malware industry and predict with effectiveness which computers are more probable of getting infected. The accuracy results were 0.60, 0.54 and 0.50 for the models Decision Tree, Naives Bayes and KNN respectively, these being low results but Decision Tree being the winner of the all.

Introducción:
Este proyecto consiste en la exploración y preparación de un dataset para calcular la probabilidad de que una computadora sea infectada de un malware en base a su telemetría por medio de modelos de machine learning. Es buena práctica realizar este tipo de análisis dado que la industria de los malware se va reinventando con gran rapidez para lograr burlar la seguridad de las computadoras, y con este tipo de estudios se puede tomar medidas preventivas. La obtención de este dataset fue gracias a Microsoft, quien recopiló esta información de computadoras personales y de negocios para lograr evaluar en base a la telemetría de cada una de ellas. Los modelos utilizados fueron Decision Tree, Naive Bayes y KNN con precisiones de 0.60, 0.54 y 0.50 respectivamente, al final resultados bajos pero siendo Decision Tree el modelo con los mejores resultados.

Marco teórico:

- **Malware:**

Describe cualquier programa o código malicioso que es dañino para los sistemas. El malware intrusivo intenta invadir, dañar o deshabilitar ordenadores, sistemas informáticos, redes, tabletas y dispositivos móviles, asumiendo el control parcial de las operaciones de los dispositivos (Oracle, s. f.).

La intención del malware es sacarle de dinero al usuario de forma ilícita, aunque no puede dañar el hardware de los sistemas o el equipo de red, si se puede robar, cifrar o borrar sus datos, alterar o secuestrar funciones básicas del ordenador y espiar su actividad en el ordenador sin su conocimiento o permiso(Oracle, s. f.) .

Las maneras más comunes en las que el malware obtiene acceso al sistema es mediante el internet y el correo electrónico, es decir básicamente todo el tiempo que está conectado a internet. Las aplicaciones maliciosas pueden ocultarse en aplicaciones aparentemente legítimas, especialmente cuando se descargan a través de sitios web o mensajes y no desde una App Store segura (Oracle, s. f.).

- **Telemetría**

Telemetría es un servicio provisto por Windows, el cual corre en segundo plano y su misión es recopilar información del sistema para mandárselo a Microsoft. Es un tipo de espía que vive pendiente del estado y acciones de la computadora. Esto para identificar cualquier problema de funcionamiento y ayuda a encontrar soluciones a estos problemas antes de que sucedan y encontrar patrones de errores que pueden existir.

- **Windows Defender:**

Es un programa de Windows, como se puede inferir del nombre, que actúa como un antivirus en el dispositivo que busca y soluciona amenazas y evitar que la computadora sea invadida por malware.

- **Firewall:**

Su nombre en español significa cortafuego, y es la parte de un sistema o red que se encarga de bloquear accesos no autorizados por medio de la distinción de los permisos de los recursos en el tráfico de la red para asegurar la integridad de la máquina en cuestión. Estos pueden ser implementados en hardware como software o ambos (Cisco, 2022).

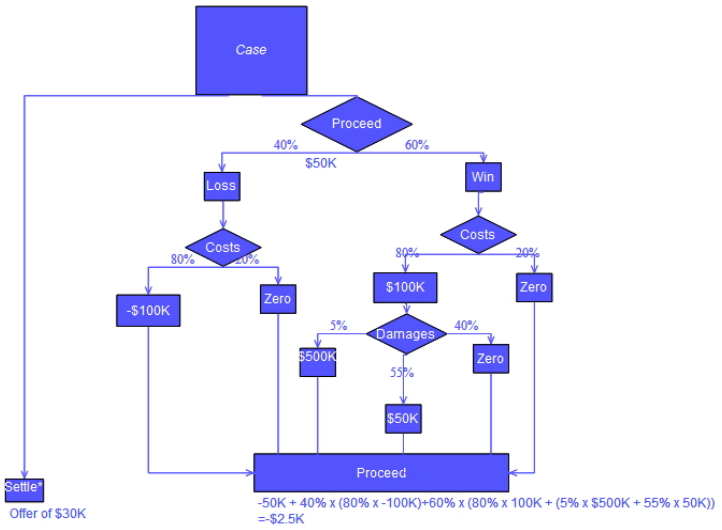
- **Decision Tree:**

En español este modelo se llama “árbol de decisiones” el cual modela las decisiones similar a un árbol, formado por un diagrama de flujo, con sus correspondientes consecuencias, con todos los posibles resultados, sus costos y utilidad. Este modelo funciona por medio de controles y decisiones condicionales. Es utilizado para la investigación de operaciones, análisis de decisiones y aprendizaje automático, esto para identificar estrategias viables entre todos los posibles escenarios (Medvedyev, 2017).

Cada nodo del diagrama de flujo de árbol representa una evaluación y cada rama es el resultado de esta evaluación. Este diagrama funciona como una herramienta de apoyo visual para calcular los valores esperados y alternativas. Consta con 3 tipos de nodos:

- Nodos de decisión: representados por un cuadrado
- Nodos de probabilidad: representados por un círculo
- Nodos finales: representados por un triángulo

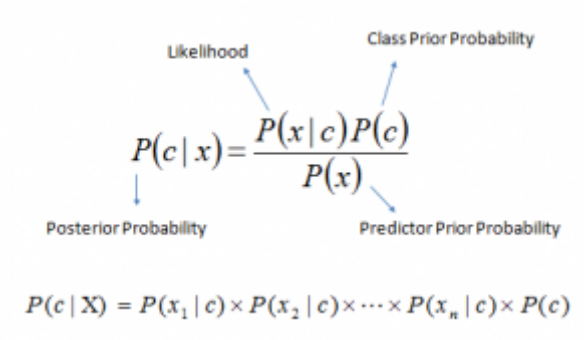
Ejemplo:



- **Naives Bayes:**

Es un modelo de clasificación que se basa en el Teorema de Bayes, el cual se discute a continuación, en base a la suposición de la independencia entre predictores. Es un modelo relativamente sencillo, por lo cual es amigable para principiantes en el mundo de la ciencia de datos, sin embargo, sigue siendo uno de los modelos más sofisticados (Vidha, 2021).

El Teorema de Bayes pretende calcular la probabilidad posterior en base a la probabilidad de ambos eventos y la probabilidad de un evento dado otro, como se puede ver en la siguiente ecuación:



El algoritmo funciona en base a una tabla de frecuencias de la cual se calcula una tabla de probabilidades y por medio de la ecuación superior la calcula la probabilidad posterior para cada evento y la que tenga la probabilidad más alta es el resultado de la predicción (Vidha, 2021).

- **K-Nearest Neighbor (KNN):**

Este modelo es utilizado para aprendizaje automático supervisado, el cual es capaz de clasificar o predecir, esto por medio de la búsqueda de los puntos de datos más similares por cercanía de la fase de entrenamiento y de esto concluyendo las conjeturas de estás clasificaciones. La “k” del nombre hace referencia a la cantidad de puntos vecinos en los n grupos ya establecidos, dado que es un algoritmo supervisado, lo cual lo diferencia de de K-means. Este modelo es basado en instancia, lo cual significa que memoriza las instancias de entrenamiento que son usadas como la base del aprendizaje para la fase de predicción, por lo cual no funciona como la Regresión Logística, donde el algoritmo aprende explícitamente (Garbade, 2018).

Propuesta de metodología:

Para comenzar a familiarizar con el dataset se exploró test.csv y train.csv, ambos con una previa especificación de tipo de variable porque que al cargar los datasets algunas columnas daban conflicto por combinación de variables numéricas y categóricas. Se exploró con una porción de 1,000,000 de registros para cada dataset debido a la limitación de recursos para cargar los datasets completos.

Los datos de test contaban con 82 variables y los de train con 83. La variable “extra” en train.csv consistía en HasDetections, de tipo numérico y en base a la cúal se hicieron varias de las gráficas exploratorias. Con el uso de pandas, numpy y quick-EDA, se exploraron los datos categóricos, numéricos y se elaboró una matriz de correlación para saber la relación

entre variables. Además, para cada set de datos se elaboró un reporte final utilizando pandas profiling.

Cómo fue posible observar desde el análisis exploratorio, algunas de las variables del dataset tenían un alto porcentaje de valores nulos, por lo tanto se eliminaron aquellas columnas que tuvieran un porcentaje mayor al 30% en valores nulos. También se identificaron y eliminaron aquellas columnas que tuvieran alta cardinalidad entre ellas, para evitar que esto pudiera afectar los resultados.

Seguidamente, se aplicó label encoder a las variables categóricas. Para seleccionar las características se utilizó VarianceThreshold para descartar aquellas columnas que tuvieran una varianza muy cercana a cero, es decir eliminar aquellas que tenían el mismo valor en todas las muestras.

Por último se seleccionó la muestra del dataset a utilizar, cabe mencionar que únicamente se consideró el dataset de training dado que era el único que se encontraba etiquetado, dividiendo el dataset en diez segmentos, luego seleccionado el 20% de cada segmento. Una vez obtenido el dataset final, se logró observar un ligero desbalance entre las clases, pero se tomó la decisión de aceptar ese desbalance para tratar de evitar un sobreajuste del modelo. Se dividió el dataset en train, test y validación, se realizó el escalamiento correspondiente.

Luego del proceso detallado con anterioridad se seleccionó “HasDetections” como la variable objetivo para todos los modelos a implementar. Se buscó información sobre modelos ML eficientes e implementables en Python a través de las respectivas librerías. Esto nos permitió elegir los siguientes modelos: Decision Tree, Naive Bayes Gaussiano y K-nearest neighbors. Para todos se siguieron paso similares, entre ellos la sustitución de valores NaN por 0, el fit del clasificador utilizando los datos de entrenamiento, las predicciones con los datos de validación y posteriormente con los de test. Tanto para la data de validación como para la de test, se calcularon las métricas de evaluación, el nivel de precisión y la matriz de confusión.

Finalmente para cada uno de los modelos con el uso de la librería sklearn se realizó la cross-validation utilizando 10 como valor de k. Para la gráfica ROC se implementó una función con base en la información publicada por Medvedyev (2017). Dichas gráficas contaban con dos clases, ya que la variable objetivo podía ser 0 o 1.

Resultados:

Exploración de datos:

A continuación se mostrará y describe los hallazgos más relevantes del dataset
En el dataset, 74 de las columnas contenían missing values, 35 tenía menos del 10% missing values, 2 columnas con missing values entre 10% y 50% y 7 columnas con más del 50% eran missing values. La distribución se puede observar en la siguiente figura:

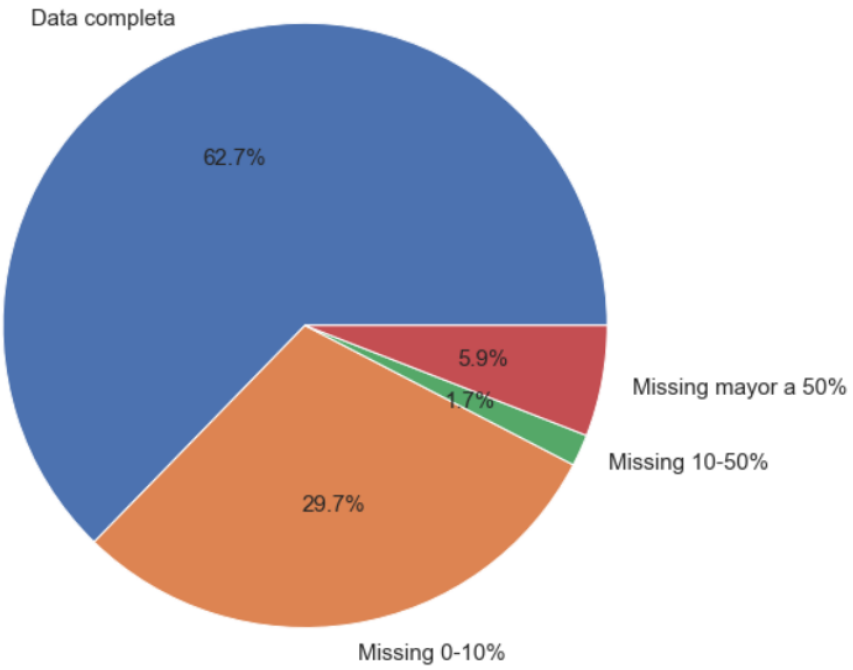


Imagen 1. Exploración de datos faltantes

Se puede observar que a pesar que más del 60% del dataset se encontraba completo, existen muchas inconsistencias y data faltante en el dataset, por lo cual esto puede afectar la efectividad de los modelos.

Igualmente se puede observar los sistemas instalados en los dispositivos y la distribución de estos, como se puede observar a continuación:

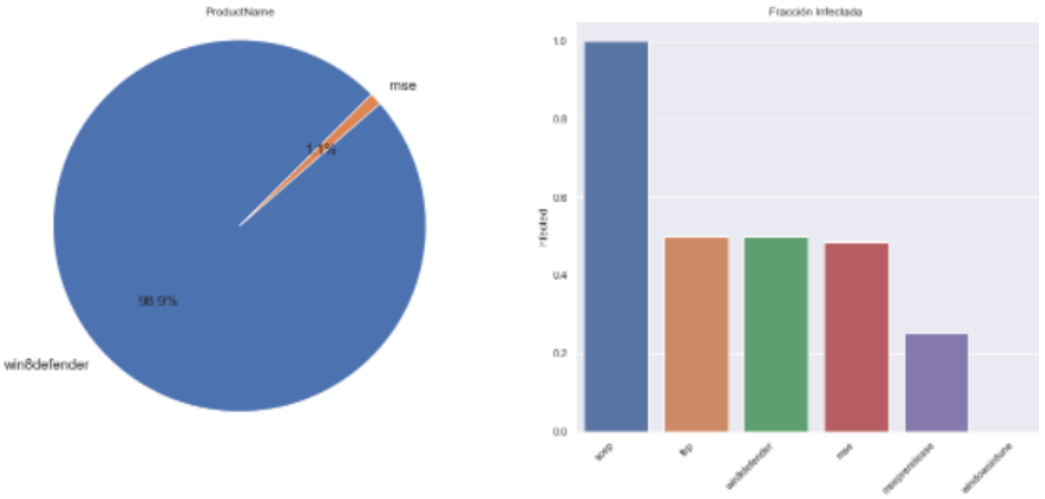


Imagen 2. Antivirus instalados

La mayor parte, el 98.9% de los equipos tienen instalado Windows 8 Defender. Sin embargo, el porcentaje de computadoras infectadas por categoría, fep y win8defenre son similares: alrededor del 50%.

De igual manera se observó los sistemas operativos instalados en las máquinas evaluadas:

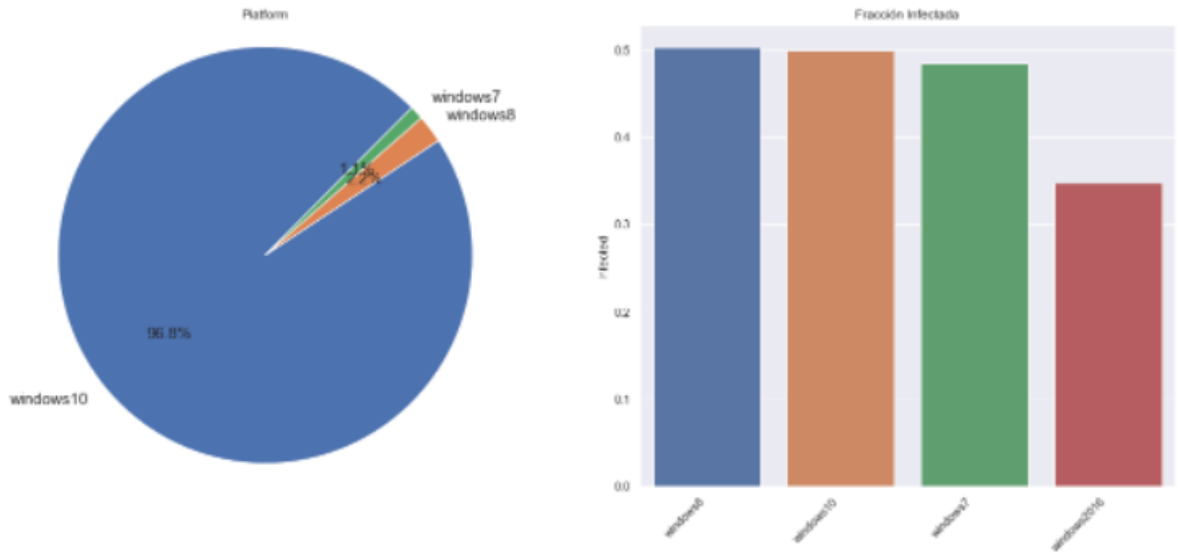
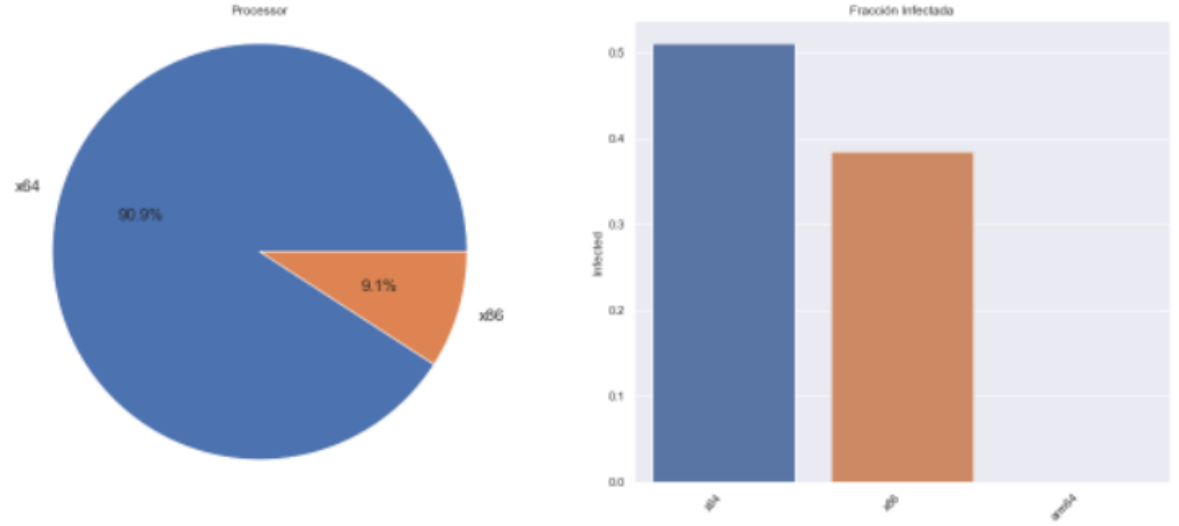


Imagen 2. Sistemas operativos instalados

La mayor parte, el 96.8% de los equipos tienen instalado Windows 10. Sin embargo, el porcentaje de computadoras infectadas por plataforma, windows8 y windows10 son similares: alrededor del 50%.

Las máquinas evaluadas cuentan con diferentes procesadores instalados, y la distribución de estos se puede ver a continuación:



El 90.9% de los equipos cuentan con un procesador x64. Simultáneamente este tipo de procesador, x64, presenta la mayor proporción de casos infectados, aproximadamente el 50%

Luego de realizar el análisis exploratorio, preprocesamiento y limpieza de datos se procedió a implementar los modelos seleccionados, donde son basamos en métricas de evaluación para verificar su precisión, Curva ROC y evaluación cruzada con K - folds con el valor k igual a 10, a continuación se puede observar los resultados para cada modelo:

Decision Tree:

Metricas de evaluacion:

	precision	recall	f1-score	support
0	0.61	0.54	0.57	127534
1	0.59	0.65	0.62	127462
accuracy			0.60	254996
macro avg	0.60	0.60	0.60	254996
weighted avg	0.60	0.60	0.60	254996

Imagen 1. Reporte de clasificación de validation data

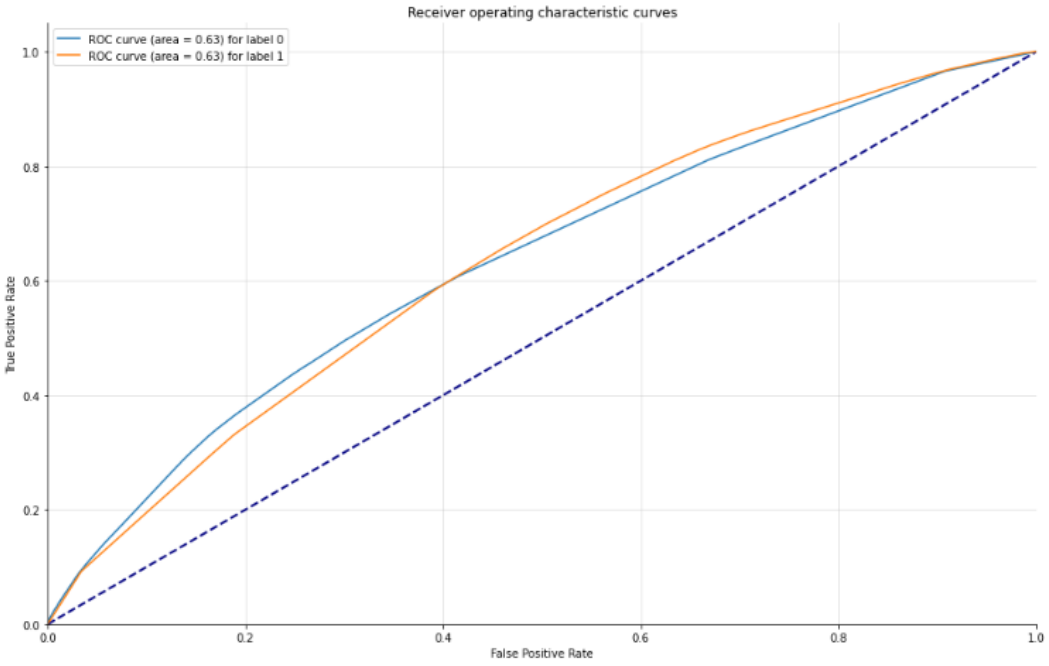


Imagen 2. Curva ROC de Decision Tree para cada valor del target (LABEL)

Naive Bayes:

Metricas de evaluacion:

	precision	recall	f1-score	support
0	0.50	1.00	0.67	127534
1	0.64	0.00	0.00	127462
accuracy			0.50	254996
macro avg	0.57	0.50	0.33	254996
weighted avg	0.57	0.50	0.33	254996

Imagen 3. Reporte de clasificación de validation data

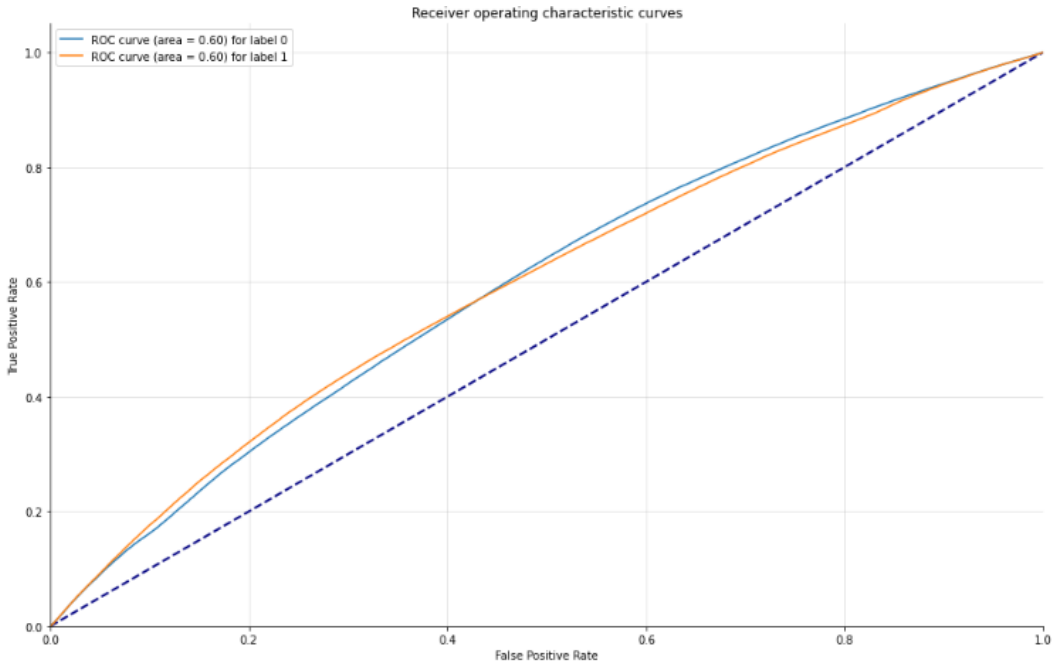


Imagen 4. Curva ROC de Naive Bayes para cada valor del target (LABEL)

KNN:

Metricas de evaluacion:

	precision	recall	f1-score	support
0	0.54	0.53	0.54	259290
1	0.54	0.56	0.55	258427
accuracy			0.54	517717
macro avg	0.54	0.54	0.54	517717
weighted avg	0.54	0.54	0.54	517717

Imagen 5. Reporte de clasificación de validation data

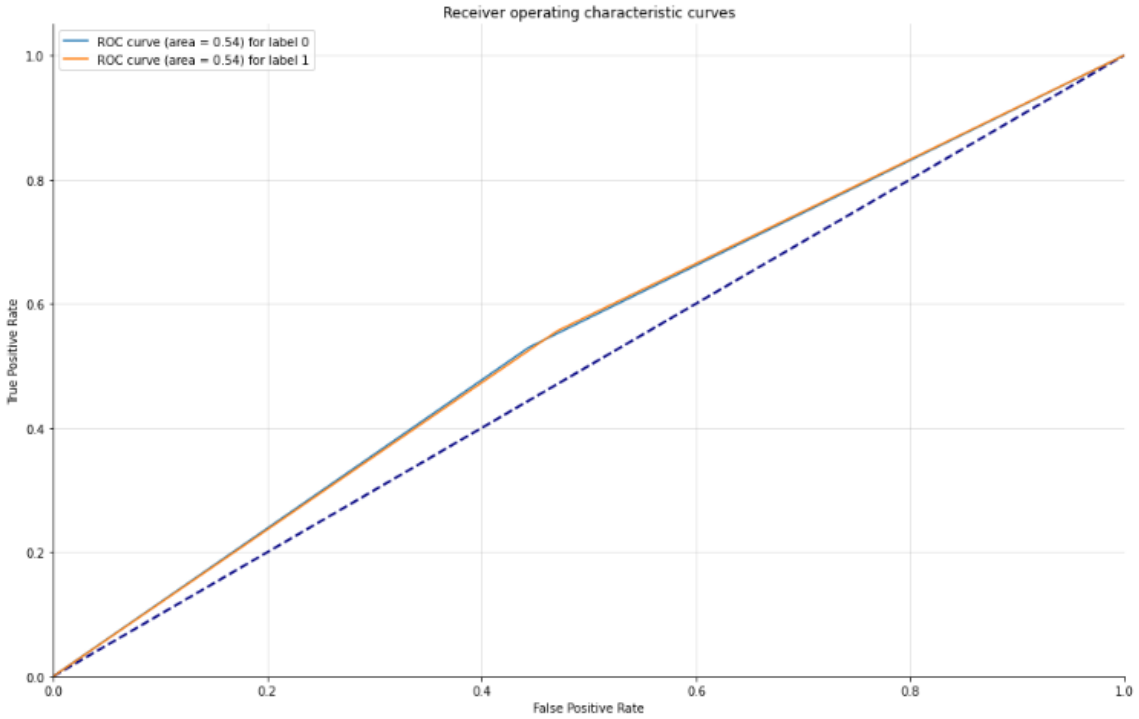


Imagen 6. Curva ROC de KNN para cada valor del target (LABEL)

Como se puede observar en los resultados superiores, el modelo con mejores resultados fue Decision Tree con un accuracy de 0.60, que a pesar de ser bastante bajo, es el más alto de los 3. Este seguido por KNN con un accuracy de 0.54 y por último Naive Bayes con 0.50. Los 3 dieron resultados bastante bajos, pero muy similares entre ellos, por lo cual consideramos que estos resultados se debieron al estado del dataset, probablemente por la cantidad de columnas con datos faltantes que se encuentra en el dataset, como se puede ver en el análisis exploratorio. Las curvas ROC se mantuvieron bastante consistentes, sobre todo para los primeros dos modelos, y ambos labels (0 y 1) dieron el mismo resultado para todos los modelos, pero dado que no se mantuvieron más cercanos a la esquina superior izquierda no fueron los resultados esperados.

En cuanto a la facilidad de diseñar los modelos, de los 3 el modelo KNN fue el que más tiempo consumió en entrenarse, tomando incluso más de un día, por lo cual a pesar que fue el segundo más alto en precisión, no lo recomendaríamos por el tiempo que ocupa.

Conclusiones:

- El modelo con la mejor precisión fue Decision Tree, con un valor de 0.60, que a pesar de ser bajo fue el que más se adecuo a nuestro dataset.
- El modelo KNN no lo recomendamos para ser replicable para este dataset dado que tardó más de 24 horas en entrenarse, siendo muy poco ágil y por los resultados que brindó no vale la pena invertirlo.
- La gran cantidad de columnas con datos faltantes pudo haber sido la causante de los resultados tan bajos obtenidos en los modelos, probablemente por la cantidad de inconsistencias que esto conlleva.

Referencias bibliográficas:

- Cisco. (2022, 1 abril). *What Is a Firewall?* Recuperado 19 de abril de 2022, de https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html
- Medvedyev, O. (2017, 26 julio). *ROC for multiclass classification*. Stack Overflow. https://stackoverflow.com/questions/45332410/roc-for-multiclass-classification
- GeeksforGeeks. (2021, 22 junio). *Decision Tree*. Recuperado 21 de abril de 2022, de https://www.geeksforgeeks.org/decision-tree/
- Vidha, S. (2021, 26 agosto). *Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples*. Analytics Vidhya. Recuperado 21 de abril de 2022, de https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
- Garbade, M. J. (2018, 13 septiembre). *Understanding K-means Clustering in Machine Learning*. Medium. Recuperado 22 de abril de 2022, de https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
- Oracle. (s. f.). ¿Qué es el malware? Definición de malware. Oracle España. Recuperado 10 de abril de 2022, de https://www.oracle.com/es/database/security/que-es-el-malware.html
- Universidad de Sevilla, & Valle, A. (2018, marzo). *Curvas ROC (Receiver-Operating-Characteristic) y sus aplicaciones.* Universidad de Sevilla. https://idus.us.es/bitstream/handle/11441/63201/Valle%20Benavides%20Ana%20Roc%C3%ADo%20del%20TFG.pdf?sequence=1&isAllowed=y
- N. (2020, 15 julio). *Algoritmo k-Nearest Neighbor*. Aprende Machine Learning. Recuperado 22 de abril de 2022, de https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/

Link a repositorio: https://github.com/dianaxime/SDS-Proyecto2