Base de datos de imágenes de productos

Reporte proyecto 2

Jorge Andrés Pérez Barrios[1], Cristopher Jose Rodolfo Barrios Solis[2], Cristopher Sebastian Recinos Ramírez[3]

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

Universidad del Valle de Guatemala, Guatemala

per16362@uvg.edu.gt [1], bar18207@uvg.edu.gt [2], rec16005@uvg.edu.gt [3]

Abstract

—Microsoft Windows is the most widely used operating system in the world, which also means that it is the main focus of the malware industry. Machine learning algorithms can help detect and predict if a machine is likely to be hit with malware. In this document, the results of implementing three ML models in an application that predicts if a machine is vulnerable to malware will be discussed and compared. The models will classify an entry by analyzing some features of a computer and making a decision based on that. The accuracy scores of the three models that were used were, on average, 0.52 for Naive Bayes, 0.52 for Logistic Regression and 0.50 for Decision Tree Classifier model.

Palabras clave—Machine Learning, Naive Bayes, Decision Tree, Logistic Regression, Cybersecurity, threat detection, Microsoft, threat prevention

I. Introducción

Este documento es el informe correspondiente al proyecto de Microsoft Malware Prediction , el cual tiene como objetivo final el análisis de los datos con diferentes tipos de módulos, los cuales fueron tres con el fin de predecir la probabilidad de una maquina con SO windows de ser infectada con por varias familias de malware. Las bases de datos fueron proveídas por una base de datos de Microsoft con la que se entrenó a los modelos.se tiene dos datasets uno de entrenamiento y un dataset de prueba. Los datos de este provienen de computadoras personales, de negocios, etc. El motor de detección y clasificación se realizó con los algoritmos de Logistic Regression, Decision Tree Classifier y Gaussian Naive Bayes, todos 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 de amenazas en base a propiedades de una máquina, de la base de datos proveído por Microsoft, se utilizaron tres algoritmos de aprendizaje de máquina: Decision Tree, Naive Bayes y Logistic Regression.

Estos algoritmos se utilizaron para obtener las propiedades de una máquina al momento de detectar una amenaza y, así, clasificar si una máquina está expuesta a un ataque o es segura.

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.

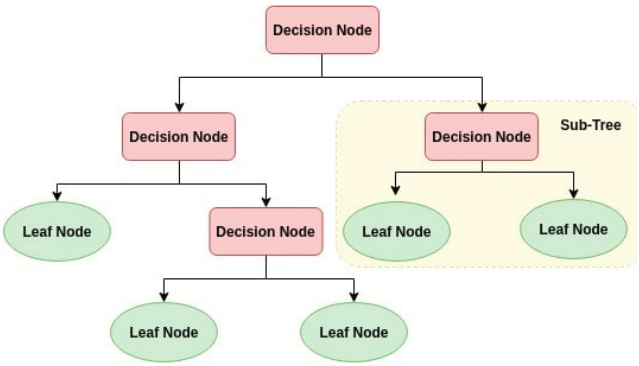

Fig 1: Ejemplo gráfico de un árbol de decisión.

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]

Logistic Regression

La Regresión Logística es un modelo estadístico y una técnica de Machine Learning de aprendizaje automático para clasificación. Es un modelo lineal generalizado, que también funciona para predicciones de probabilidad de que ocurra un

evento. Asimismo, este modelo de clasificación está considerado como una red neuronal de una sola neurona. [7]

La ecuación matemática de este modelo estadístico se puede formular como:

$$y = \sigma(z) = \sigma(WX) = \sigma(\sum(wixi)) = \sigma(\sum(w0x0 + w1x1 + \ldots + wnxn))$$

Ecuación 2: Ecuación general del modelo estadístico de Regresión Logística.

Como se puede observar, esta se compone de dos partes: la primera es la combinación lineal y la segunda es la aplicación de la función logística. Esta función es también llamada sigmoide. Es acotada por 0 (por debajo) y 1 (por arriba) [7]. Esta función está dada por la ecuación matemática:

$$\sigma(z) = 1 + e - z1$$

Ecuación 3: Ecuación general de la función logística.

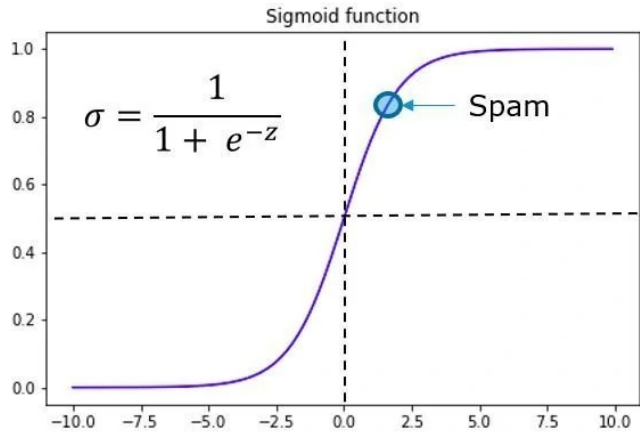

Fig. 2: Gráfico de la función sigmoidal con un ejemplo de detección de spam.

III. METODOLOGÍA

A. Análisis exploratorio de los datos.

Junto con el planteamiento del proyecto, se proporcionaron los conjuntos de datos pertenecientes a Microsoft. Este conjunto de datos cuenta con alrededor de 9 millones de entradas, junto con 83 características, lo que hace que todo el dataset pese aproximadamente 4.1Gb.

Con respecto a las características, se encontraron 26 variables categóricas, 14 variables booleanas y el resto (43 características) variables numéricas. Dentro de las variables categóricas, se encuentran MachineIdentifier que es únicamente un hash identificatorio de la máquina y HasDetections, la cual es la variable independiente. Esta variable, a pesar de ser clasificatoria, es de tipo booleana, por lo que tiene un valor 1 si Windows detecta alguna amenaza o 0 si no.

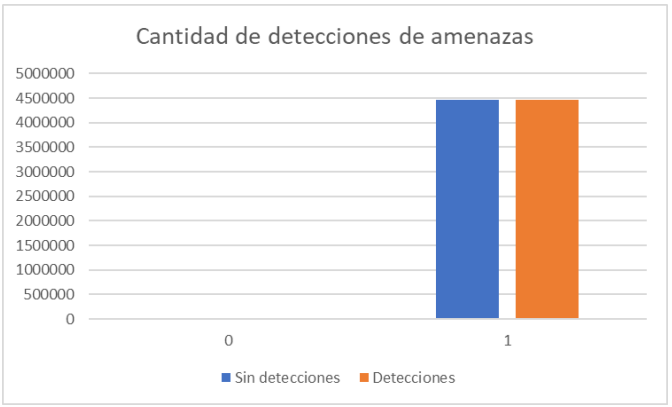

Fig 3: gráfico de barras comparativo entre valores de la variable independiente

	0	1
Cantidad	4462591	4458892
Porcentaje	0.5002	0.4997

Tabla 1: Conteo y porcentaje de valores en la variable independiente.

Como se puede observar en la figura 3 y en la tabla 1, la cantidad de detecciones y no detecciones es casi la misma, por lo que se podría decir que el dataset está bien balanceado en cuanto al label de clasificación.

Por otra parte, se analizó la cantidad de datos faltantes (nan) que estaban presentes en el dataset. Se encontró que 7 columnas presentaban una falta de datos arriba del 60%, siendo la variable PuaMode y Census_ProcessorClass las variables con casi un 100% de datos faltantes. Estas dos variables serán las que se eliminen inmediatamente antes del preprocesamiento.

PuaMode	99.97%
Census_ProcessorClass	99.59%
DefaultBrowsersIdentifier	95.14%
Census_IsFlightingInternal	83.04%
Census_InternalBatteryType	71.05%
Census_ThresholdOptIn	63.52%
Census_IsWIMBootEnabled	63.44%
SmartScreen	35.61%
OrganizationIdentifier	30.84%
SMode	6.03%

Tabla 2: Porcentajes de valores faltantes por columna dentro del dataset.

Como se puede observar en la tabla 2, solo 7 variables son las que tienen una cantidad de datos nulos mayor al 50%. Es posible que todas estas se eliminen. Dos variables son las que tienen una cantidad de datos nulos al 30% y menores al 50% y el resto tiene una cantidad menor al 6%. Para salvar estas variables, ya que muchas de estas son representativas al momento de detectar una amenaza, se sustituyen los valores nulos por 0.

B. Manejo y preprocesamiento de los datos.

Para el preprocesamiento de los datos, se separó esta etapa del proyecto en tres partes: la primera parte consiste en una limpieza de los datos. Primero se corrigieron varias discrepancias en la variable SmartScreen, ya que muchos de los valores apuntaban al mismo concepto, sin embargo estaban escritas de diferente manera. Como por ejemplo, existían valores “Promt” y “Promprt” apuntando al mismo concepto: “Prompt”. Luego se eliminaron las mayúsculas en todas las variables categóricas que lo tuvieran y se convirtieron a minúsculas. Seguidamente, se eliminaron caracteres extraños en la variable SmartScreen y Census_InternalBatteryType. Después se añadieron los tipos de datos categóricas a las respectivas variables, así como la categoría “unknown”.

La segunda parte del preprocesamiento de los datos consistió en la eliminación de las variables innecesarias, las cuales fueron:

- PuaMode
- Census_ProcessorClass
- EngineVersion
- AppVersion
- AvSigVersion
- Census_OSVersion
- OsVer
- OsBuildLab

La mayoría de estas variables fueron eliminadas ya que estas contenían las versiones de software especificadas y tienen varios puntos entre los números, por lo que crearía errores en los modelos. Luego, se convirtieron todos los valores nulos a 0, como anteriormente se mencionó.

La tercera parte del preprocesamiento consiste en asegurarse que los tipos de datos de cada variable fueran los correctos, por lo que se hizo un diccionario y se cargo el dataset con respecto a estos. Luego se eliminaron los identificadores así como MachineIdentifier. Después se seleccionó aproximadamente ⅓ de todo el dataset, ya que este era muy pesado y estaba generando problemas de memoria. En total fueron seleccionados alrededor 4.5 millones de datos al azar. Por último, se crearon variables dummy para las variables categóricas, lo que dio un total de 4.5 millones de entradas, junto con 361 variables. Este dataset llegó a pesar 3.7Gb.

IV. Análisis y Discusión de Resultados

El objetivo principal de este proyecto es predecir la probabilidad de una máquina con Sistema Operativo Windows de ser infectada por varias familias de malware, en base a las propiedades de la máquina. Para resolver este objetivo, se utilizaron tres tipos de algoritmos distintos clasificatorios: Logistic Regression, Decision Tree Classifier y Gaussian 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: 70% para entrenamiento y 30% para pruebas. Asimismo, se aplicó un escalamiento estándar a los datos para un uso del Principal Component Analysis, con 50 componentes principales. Para la compilación de los tres 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. Logistic Regression

Este modelo tuvo la segunda accuracy más alta de los tres modelos, con 51.7%.

330822	356818
306247	380149

Tabla 4: Matriz de confusión del modelo Logistic Regression.

	Precisión	Recall	f1-score	support
Detected	0.52	0.48	0.50	687640
Not Detected	0.52	0.55	0.53	686396
Accuracy			0.52	1374036
macro avg	0.52	0.52	0.52	1374036
weighted avg	0.52	0.52	0.52	1374036

Tabla 5: Métricas obtenidas del modelo Logistic Regression.

Como se puede observar en Tabla 4, este modelo tuvo una cantidad mayor de verdaderos negativos que falsos negativos y una cantidad mayor de falsos positivos que verdaderos positivos.

En cuanto a los positivos, las cantidades son muy parecidas, y esto se ve reflejado en los datos de la Tabla 5, siendo el Recall 0.48 mientras que para negativos fue mayor con 0.55.

B. Decision Tree Classifier

Este modelo tuvo la accuracy más baja de los tres modelos, con 49.8%.

352514	335126
354394	332002

Tabla 6: Matriz de confusión del modelo Decision Tree Classifier.

	Precisión	Recall	f1-score	support
Detected	0.50	0.51	0.51	687640
Not Detected	0.50	0.48	0.49	686396
Accuracy			0.50	1374036
macro avg	0.50	0.50	0.50	1374036

weighted avg	0.50	0.50	0.50	1374036

Tabla 7: Métricas obtenidas del modelo Decision Tree Classifier.

El accuracy score de este modelo es de apenas 49.8% siendo el más bajo de los tres y el único por debajo del 50%. Con esto podemos ver que fue el menos efectivo al realizar esta tarea.

En la Tabla 6 se puede observar que este modelo tuvo una mayor cantidad de verdaderos positivos que falsos positivos, este fue el único modelo de los 3 que obtuvo este resultado.

C. Gaussian Naive Bayes

Este modelo tuvo la accuracy más alta de los tres modelos, con 52.2%.

215652	471988
184650	501746

Tabla 8: Matriz de confusión del modelo Gaussian Naive Bayes.

	Precisión	Recall	f1-score	support
Detected	0.54	0.31	0.40	687640
Not Detected	0.52	0.73	0.60	686396
Accuracy			0.52	1374036
macro avg	0.53	0.52	0.50	1374036
weighted avg	0.53	0.52	0.50	1374036

Tabla 9: Métricas obtenidas del modelo Gaussian Naive Bayes.

Como se puede observar en la Tabla 9 en este caso se tienen buenas métricas, pero tiene mejores en cuanto a máquinas libres de amenazas ya que tiene métricas más altas en esa categoría por lo tanto se puede decir que el mejor modelo es el que tiene un 52.2% de accuracy. Además en la Tabla 8 con la matriz de confusión se ve aceptable.

Así mismo, para replicar este proyecto, se recomienda tener en cuenta la cantidad de datos que se tienen ya que al tener tantos datos puede llegar a complicar la realización del mismo, debido a que dependiendo de la capacidad que tenga la computadora es lo que se podrá realizar, por ejemplo que no pueda mostrarse alguna gráfica que se desee, por lo que hay que tener que utilizar algún otro tipo de análisis.

También se tiene que ver la cantidad de tiempo que se toma en analizar la computadora cada modelo, se requieren computadoras rápidas si se requieren realizar de una manera más ágil en un ambiente formal.

En este caso la cantidad de tiempo que tomó el realizar los análisis fue aceptable, además de que no fue una molestia ya que se vio que podría tardar en realizarlos.

V. Conclusiones

- ❏ Se determinó que el algoritmo con el mejor modelo fue el Naive Bayes, ya que presentó un mejor accuracy teniendo en cuenta los ajustes que se hicieron durante la realización del código.
- ❏ La diferencia de accuracy entre los tres modelos es baja, estando todos cercanos al 50%. por lo tanto se puede determinar de mejor manera cuando se tiene estos tres modelos.
- ❏ El modelo de Decision Tree Classifier fue el menos efectivo para esta tarea. ya que según los datos que se obtuvieron en los otros modelos, se puede ver que tiene ciertas deficiencias en comparación.
- ❏ Se observó que es necesario es necesario tomar en cuenta el uso de tiempo en los análisis, si se quiere obtener de una manera mas rapida.

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/

[7] Martinez Heras, J., (21 de septiembre de 2020). Regresión Logística para Clasificación. Extraido de: https://www.iartificial.net/regresion-logistica-para-clasificacion/#:~:text=La%20regresión%20logística%20es%20una%20técnica%20de%20aprendizaje%20supervisado%20para,más%20usadas%20en%20Machine%20Learning