Predicción de infección por malware

Sara Zavala, Amado García[1]

[1]*Facultad de Ingeniería,*
Ingeniería en Ciencias de la Computación y TI, Universidad del Valle de Guatemala,Ciudad de Guatemala, Guatemala.

Nowadays, the internet and technology have been adopted by the masses. And with it, the danger of malicious attacks by cybercriminals have increased. These attacks are done via Malware and have resulted in billions of dollars of financial damage. This makes the prevention of malicious attacks an essential part of the battle against cybercrime. In this paper, we are applying machine learning algorithms to predict the malware infection rates of computers based on its features.

I. INTRODUCCIÓN

Malware es un término genérico utilizado para describir una variedad de software hostil o intrusivo: virus informáticos, gusanos, caballos de Troya, software de rescate, spyware, adware, software de miedo, etc. Puede tomar la forma de código ejecutable, scripts, contenido activo y otro software. El malware hostil, intrusivo e intencionadamente desagradable intenta invadir, dañar o deshabilitar ordenadores, sistemas informáticos, redes, tabletas y dispositivos móviles, a menudo asumiendo el control parcial de las operaciones de un dispositivo. Sea cual sea su tipo, todo malware sigue el mismo patrón básico: El usuario descarga o instala involuntariamente el malware, que infecta el dispositivo.
Aunque se han logrado avances significativos en el campo de la predicción y prevención de malware y la ciberseguridad en general, parece que los ciberdelincuentes siempre van un paso por delante de la curva. El volumen y la intensidad de los ataques cibernéticos solo han aumentado con una mayor dependencia de Internet por parte de individuos y organizaciones por igual.

A continuación, se describe el uso de dos modelos de Machine Learning, Support vector machine (SVM) y Random Forest.

II. MARCO TEÓRICO

A. Random Forest

Random forest es un algoritmo de aprendizaje automático supervisado que se usa ampliamente en problemas de clasificación y regresión. Construye árboles de decisión en diferentes muestras y toma su voto mayoritario para la clasificación y el promedio en caso de regresión.

Una de las características más importantes del Random Forest es que puede manejar el conjunto de datos que contiene variables continuas como en el caso de la regresión y variables categóricas como en el caso de la clasificación y este ofrece mejores resultados para problemas de clasificación.

El "forest" que construye es un conjunto de árboles de decisión, generalmente entrenados con el método de "embolsado". La idea general del método de embolsado es que una combinación de modelos de aprendizaje aumenta el resultado general.

En pocas palabras: el random forest crea múltiples árboles de decisión y los fusiona para obtener una predicción más precisa y estable. Una gran ventaja del random forest es que puede usarse tanto para problemas de clasificación como de regresión, que forman la mayoría de los sistemas de aprendizaje automático actuales.

Si no se sabe cómo funciona un árbol de decisión o qué es una hoja o un nodo, una buena descripción sería: “En un árbol de decisión, cada nodo interno representa una 'prueba' en un atributo (por ejemplo, si una moneda si sale cara o cruz), cada rama representa el resultado de la prueba y cada nodo hoja representa una etiqueta de clase (decisión tomada después de calcular todos los atributos). Un nodo que no tiene hijos es una hoja”.

Al observar la importancia de la característica, puede decidir qué características descartar posiblemente porque no contribuyen lo suficiente (o, a veces, nada en absoluto) al proceso de predicción. Esto es importante porque una regla general en el aprendizaje automático es que cuantas más funciones tenga, más probable es que su modelo sufra sobreajuste y viceversa.

Otra gran cualidad del algoritmo de random forest es que es muy fácil medir la importancia relativa de cada característica en la predicción. Sklearn proporciona una gran herramienta para esto que mide la importancia de una característica al observar cuánto los nodos de árbol que usan esa característica reducen la impureza en todos los árboles del bosque. Calcula esta puntuación automáticamente para cada función después del entrenamiento y escala los resultados para que la suma de todas las importancias sea igual a uno.

B. Máquina de Soporte Vectorial

SVM (máquina de vectores de soporte) es una técnica de clasificación y regresión que aprovecha al máximo la precisión de las predicciones de un modelo sin ajustar excesivamente los datos de entrenamiento. SVM es ideal para analizar datos con un gran número de campos de predictores (por ejemplo, miles).

SVM tiene aplicaciones en multitud de disciplinas, incluyendo la gestión de relaciones con los clientes (CRM), el reconocimiento facial y de otras imágenes, bioinformática, extracción de conceptos de minería de texto, detección de intrusiones, predicción de estructura de proteínas y reconocimiento de la voz.

El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. “De la mejor forma posible” implica el hiperplano con el margen más amplio entre las dos clases, representado por los signos más y menos en la siguiente figura. El margen se define como la anchura máxima de la región paralela al hiperplano que no tiene puntos de datos interiores. El algoritmo solo puede encontrar este hiperplano en problemas que permiten separación lineal; en la mayoría de los problemas prácticos, el algoritmo maximiza el margen flexible permitiendo un pequeño número de clasificaciones erróneas.

SVM funciona correlacionando datos a un espacio de características de grandes dimensiones de forma que los puntos de datos se puedan categorizar, incluso si los datos no se puedan separar linealmente de otro modo. Se detecta un separador entre las categorías y los datos se transforman de forma que el separador se puede extraer como un hiperplano. Tras ello, las características de los nuevos datos se pueden utilizar para predecir el grupo al que pertenece el nuevo registro.

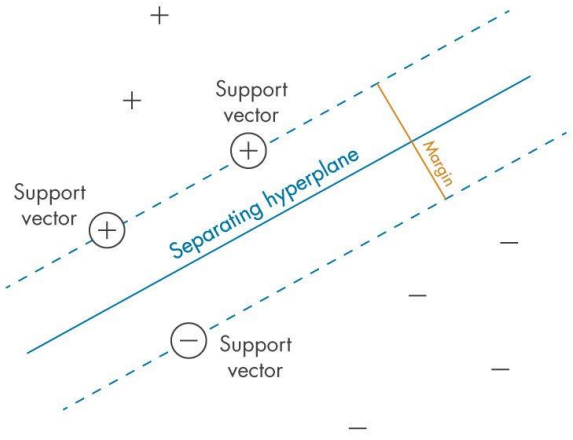

Figura 1: Gráfica de SVM

III. METODOLOGÍA

A. Análisis Exploratorio

Se cargaron los dos datasets llamados ***train.csv*** y ***test.csv*** y se realizó una breve visualización de las 83 columnas que contenía cada uno. Estos archivos contenían el mismo tipo de data.

B. Preprocesamiento

Estos datasets se trabajaron por separado en un inicio. Se contaban con alrededor de 8921483 datos en cada uno de los sets. Para trabajar de manera más fácil y ágil, se creó un data set con menor cantidad de datos y se guardó con el nombre ***new_train.csv.*** Se hizo lo mismo con el data set de test.

C. Selección de Características

El data set con el que se trabajó constaba de 52 columnas en las cuales se albergan alrededor de tres mil millones de datos. Cada una de las columnas tenían características distintivas de las variables que el data set manejaba, sin embargo, debido a la magnitud de los datos y las necesidades que se tenía, se tuvo que realizar una depuración de las mismas para poder quedarnos con menos cantidad de datos, pero siempre sin perder las partes fundamentales del dataset, o al menos perder el menor porcentaje posible.

Las columnas que se consideraron poco relevantes para este análisis fueron las siguientes:

- 'MachineIdentifier', 'DefaultBrowsersIdentifier', 'ProductName','IsBeta','RtpStateBitfield',
- 'IsSxsPassiveMode','AVProductsEnabled','HasTpm','Platform','OrganizationIdentifier','OsVer', 'IsProtected',
- 'AutoSampleOptIn','SMode', 'Firewall','PuaMode','UacLuaenable','SmartScreen', 'Census_ProcessorClass','Census_InternalBatteryType',
- 'Census_IsFlightingInternal','Census_ThresholdOptIn','Census_IsWIMBootEnabled','Census_DeviceFamily', 'Census_HasOpticalDiskDrive',
- 'Census_IsFlightsDisabled','Census_IsVirtualDevice','Census_IsPenCapable','Census_IsAlwaysOnAlwaysConnectedCapable',
- 'Census_FlightRing', 'Census_IsPortableOperatingSystem

```
# Vamos a seguir con el sample TRAIN
train_sample = train_sample.drop(columns=['Unnamed: 0', 'MachineIdentifier', 'DefaultBrowsersIdentifier', 'ProductName','IsBeta','RtpStateBitfield',
                                          'IsSxsPassiveMode','AVProductsEnabled','HasTpm','Platform','OrganizationIdentifier','OsVer', 'IsProtected',
                                          'AutoSampleOptIn','SMode', 'Firewall','PuaMode','UacLuaenable','SmartScreen', 'Census_ProcessorClass','Census_InternalBat
                                          'Census_IsFlightingInternal','Census_ThresholdOptIn','Census_IsWIMBootEnabled','Census_DeviceFamily', 'Census_HasOpticalDisk
                                          'Census_IsFlightsDisabled','Census_IsVirtualDevice','Census_IsPenCapable','Census_IsAlwaysOnAlwaysConnectedCapable',
                                          'Census_FlightRing', 'Census_IsPortableOperatingSystem'],axis=1)
```

Figura 2: Dropeo de columnas no útiles

Ya que se depuraron las columnas, luego fue necesario realizar un cambio en el tipo de variables que se tenían. Las variables cualitativas fueron cambiadas de manera que estas pudieran ser cuantificadas y ser reconocidas de esta manera. Esto se realizó con el fin de poder comparar datos de manera más sencilla. Estos cambios se pueden notar en las columnas de las etiquetas y de los tipos de protocolos existentes.

De igual manera se establecieron agrupaciones de datos de las variables para mejorar el analisis y realizar mejores predicciones.

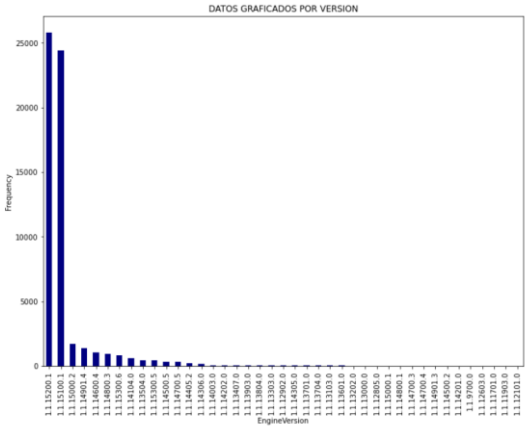

Figura 3: Gráfica de dataset

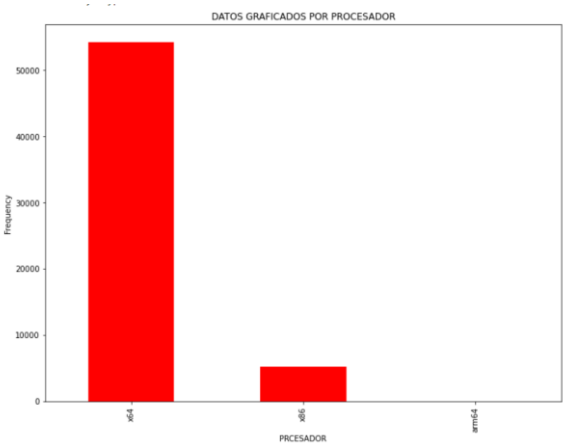

Figura 4: Gráfica de dataset

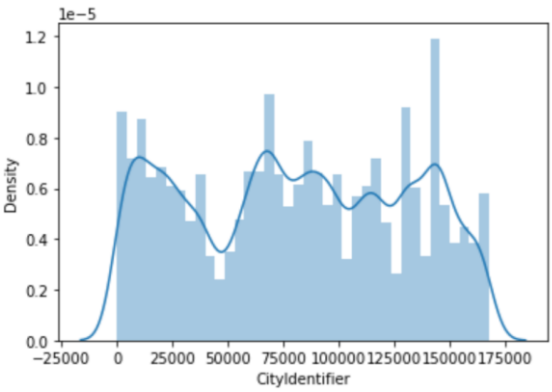

Figura 5: Gráfica de dataset

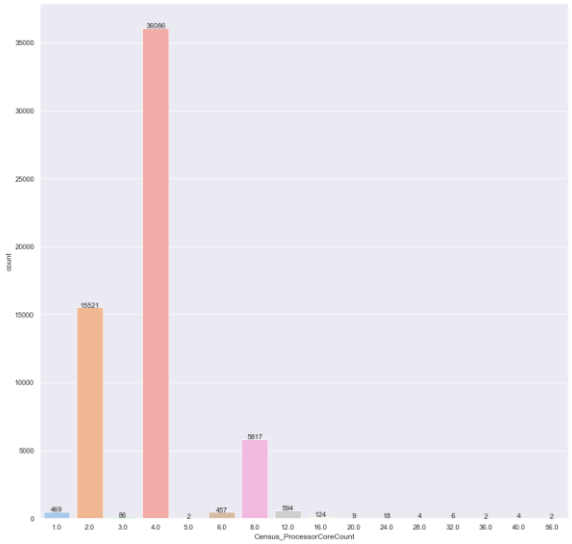

Figura 6: Gráfica de dataset

D. Implementación de Modelos

Como se estipuló al principio, se pondrán en comparación los resultados de dos modelos distintos, el SVM y Random Forest.

Comenzamos con Random forest, en este caso trabajamos con 51715 filas × 32 columnas. Se declararon todas las variables de testeo que se utilizarían y luego se creo una matriz de confusión junto a un heatmap, estos nos dan ciertos parámetros de como se encuentran categorizados los datos que se utilizan. Por ultimo se obtuvieron los valores de las predicciones y la precisión de estos

Y en segunda instancia tenemos el modelo SVM, en el cual se utilizaron de igual manera las variables de testeo para realizar estas pruebas. Se empezó con crear un *svclassifier*, y este fue de tipo linear para comenzar. Luego obtuvimos la respectiva matriz de confusión. Cabe destacar que en este modelo se utilizó una parte bastante pequeña del dataset, ya que al manejar una gran cantidad de datos tardaba demasiado tiempo.

Luego realizamos la predicción de acorde a los valores obtenidos y conseguimos los *k-valores.* Luego se repitió el mismo procedimiento, solo que esta vez se cambió a un *poly* kernel.

IV. RESULTADOS

Random Forest

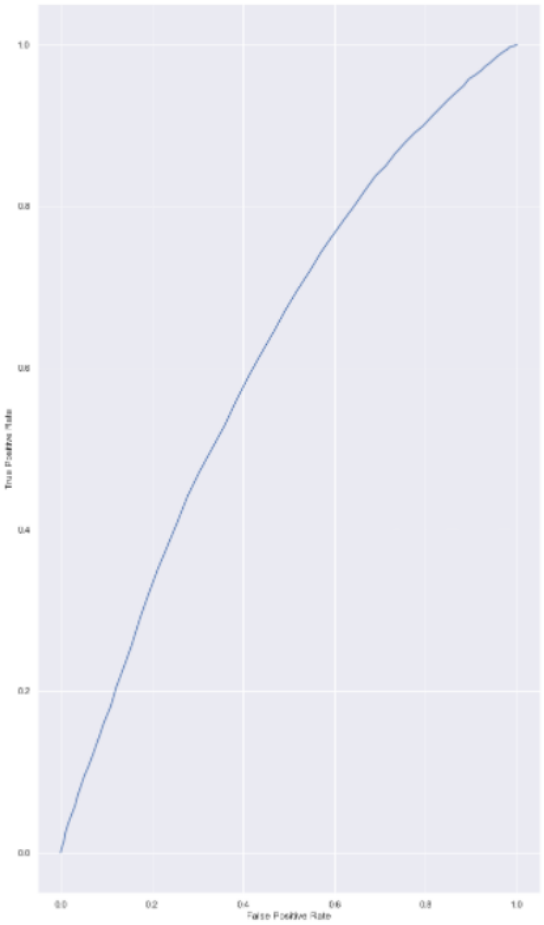

Figura 6: Gráfica ROC RandomF

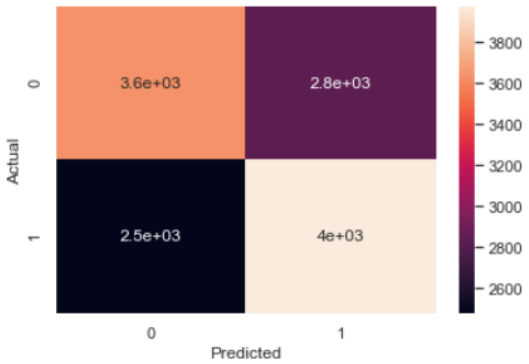

Figura 7: Matriz de Confusión RF

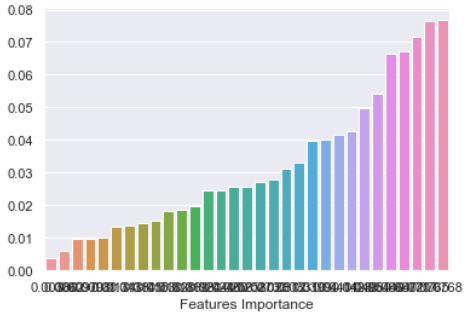

Figura 8: FeatureImportances RF

SVM

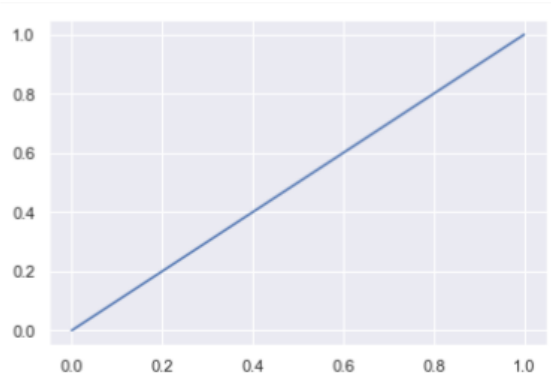

Figura 9: Gráfica ROC

```
k_val
```

```
array([0.5, 0.5, 0.5, 0. ])
```

```
[[1 0]
 [1 0]]
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2
```

Figura 10: Matriz de Confusión RF

Al analizar los resultados de los dos algoritmos implementados, se pudo observar el desempeño de estos. El algoritmo de random forest nos devolvió una certeza del 90 %, sin embargo, esto no es del todo confiable. Esto se pudo deber a que el modelo calló en sobreajuste. Por lo que no es recomendable utilizar este modelo para la predicción.

Seguido de esto, se trabajó con la implementación de SVM. Derivado del tamaño de datset, trabajar con este algoritmo resultó ser complicado. Por lo tanto, se redujo más el dataset, esto causó que la muestra no fuera representativa. Los resultados no fueron certeros, se puede apreciar en la matriz de confusión que se obtuvo una precisión del 50 %.

Con esto podemos decir que las mejores predicciones fueron realizadas por el modelo de Random forest, independientemente del sobreajuste al que se pudo haber llegado, la eficacia, rapidez y predicción fueron mayores como se puede ver en las gráficas y matrices correspondientes.

V. CONCLUSIONES

- Como parte del preprocesamiento, fue necesario descartar todas las columnas que tuvieran datos fuera de los parámetros establecidos.
- El modelo de SVM es el menos efectivo debido aque por la cantidad de datos este fue elque más tiempo tardó para realizar procesos.
- Las matrices de confusión indican que, el modelo que tuvo la mayor exactitud para predecir lo solicitado fue el del árbol de decisión. Se puede observar que tiene un porcentaje bastante alto en comparacióna los otros dos.

V. DISCUSIÓN

Al analizar los resultados de los tres algoritmos implementados, se pudo observar el desempeños de estos. El algoritmo de Arboles de Decisión nos devolvió una certeza del 100 %, sin embargo, esto no es del todo confiable. Esto se pudo deber a que el modelo calló en sobreajuste. Por lo que

VI. BIBLIOGRAFIA

- Palo Alto Network. (2020). Malware. https://www.paloaltonetworks.com/cyberpedia/what-is- malware
- How to balance a dataset in Python. (2021). Retrieved 26 February 2022, from https://towardsdatascience.com/how-to- balance-a-dataset-in-python-36dff9d12704
- scikit-learn.(2017-2020) Support Vector Machine. https://scikit-learn.org/stable/modules/svm
- A. bin Asad, R. Mansur, S. Zawad, N. Evan and M. I. Hossain, "Analysis of
- Malware Prediction Based on Infection Rate Using Machine Learning Techniques," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 706-709, doi: 10.1109/TENSYMP50017.2020.9230624