Proyecto 2: Microsoft Malware Prediction

1st Isabel Ortiz
Dept. Computer Science
Universidad del Valle de Guatemala
Guatemala, Guatemala
ort18176@uvg.edu.gt

2nd Luis Urbina
Dept. Computer Science
Universidad del Valle de Guatemala
Guatemala, Guatemala
urb18473@uvg.edu.gt

***Abstract*—Malware attacks have increased over the years becoming a serious problem of security and integrity of computer systems, this study aims to develop two Machine Learning models that can predict whether or not a machine is prone to being attacked by malware based on the characteristics of the machine. This was accomplished by using the training and test data sets provided by Microsoft in its competition entitled "Microsoft Malware Prediction". For the development, a sample of 1/256 of the original size of the data sets provided by Microsoft was first taken, then this data set was subjected to a cleaning of columns that generated noise or were irrelevant to the models. Finally, the models based on Decision Trees and Random Forest were built with the resulting test data using the sklearn library for Python. These models yielded an accuracy of 0.529 and 0.554, respectively, an AUC of 0.53 and 0.57, respectively, and cross-testing with k-folds (k=10) yielded similar results for all folds. It was concluded that although both models are fairly accurate in determining which machines are or are not prone to being attacked by malware, Random Forest is the best one for said purpose.**

***Index Terms*—Malware, classification, Random Forest, Decision Trees, ROC, K-folds**

I. Introducción

Los ataques de malware son cualquier tipo de software malicioso diseñado para causar daño a una computadora sin el conocimiento del usuario final [1]. Los atacantes cibernéticos crean, usan y venden malware por muchas razones diferentes, pero se usa con mayor frecuencia para robar información personal, financiera o comercial con fines maliciosos. Los ataques de malware son una de las preocupaciones más grandes relacionadas a la seguridad e integridad de la data de los usuarios de Windows. Esto debido a que una vez que una computadora, ya sea personal o de una organización, ha sido infectada con malware, los atacantes cibernéticos pueden causar daños al usuario u organización de muchas formas.

El incremento de familias o tipos de malware [2] ha dado lugar al desarrollo de múltiples soluciones para computadoras infectadas por malware, ejemplos de estas son: Windows Malicious Software Removal Tool, MalwareBytes y Avast Antivirus. Aunque estas soluciones comerciales pueden ser efectivas, prevenir una infección por malware resulta ser menos costoso que reparar una máquina infectada, independientemente de si la máquina en cuestión es de uso personal u organizacional.

El objetivo principal de este estudio es desarrollar dos modelos que permitan predecir la probabilidad de una máquina con sistema operativo Windows de ser infectada por varias familias de malware, con base en las propiedades de la máquina. Para ello se utilizaron dos datasets que contienen las propiedades de computadoras infectadas y no infectadas según el reporte de amenazas de Windows Defender. El primer dataset (train.csv) contiene información de 8,921,483 dispositivos y el segundo (test.csv) contiene información de 7,853,253 dispositivos. Los modelos a desarrollar fueron Decision Trees y Random forest.

II. Marco Teórico

Para alcanzar el objetivo del proyecto se propusieron dos modelos: Decision Trees y Random Forest. Para cada uno de ellos se calcularon las métricas de evaluación, la gráfica de la curva ROC y La evaluación cruzada con K-folds para k=10, a continuación se presenta una descripción breve de estos conceptos.

A. Decision Trees

Los árboles de decisión son un tipo de algoritmo de Machine Learning supervisado utilizado para la clasificación de datos. En este algoritmo, los datos se dividen continuamente de acuerdo con un parámetro determinado. El árbol puede ser explicado por dos entidades además de la raíz: Los nodos de decisión y las hojas. Los nodos de decisión son los nodos que el algoritmo utiliza para tomar decisiones con base en las características de la data y las hojas son los resultados finales, resultado de las decisiones anteriores [3].

B. Random Forest

Random Forest es un algoritmo robusto de Machine Learning que se puede usar para una variedad de tareas, tales como la regresión y la clasificación. Este modelo se compone de una gran cantidad de pequeños árboles de decisión, llamados estimadores, cada uno de los cuales produce sus propias predicciones. El modelo de bosque aleatorio combina las predicciones de los estimadores para producir una predicción más precisa [5].

Los algoritmos de clasificación mediante árboles de decisión estándar tienen la desventaja de que son propensos a sobreajustarse al conjunto de entrenamiento. El diseño de

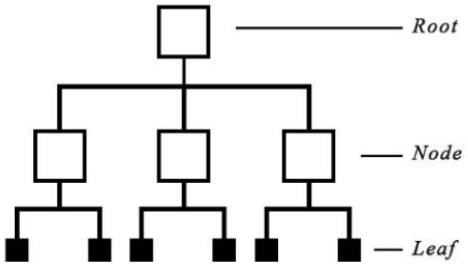

Fig. 1. Composición de un Decision Tree [4]

conjunto del bosque aleatorio permite que el bosque aleatorio compense esto y generalice bien los datos no vistos, incluidos los datos con valores faltantes. Los bosques aleatorios también son buenos para manejar grandes conjuntos de datos con alta dimensionalidad y tipos de características heterogéneas [5]. En la figura 2 se puede apreciar una representación gráfica del funcionamiento del algoritmo Random Forest.

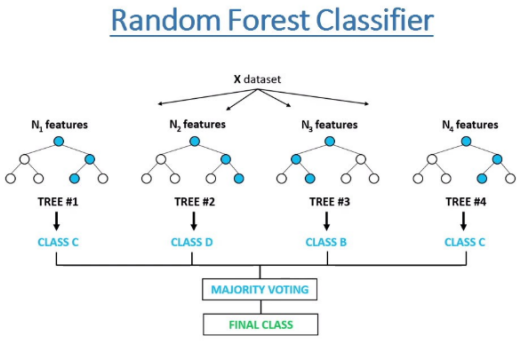

Fig. 2. Funcionamiento del algoritmo Random Forest

C. Accuracy, Precision, Recall y f1

Con el objetivo de identificar qué modelo de los dos seleccionados fue más efectivo, se necesitó valerse de algunas métricas que se puedan utilizar para comparar ambos modelos. Dichas métricas son accuracy, precision y recall.

Accuracy: Es la medida más comúnmente utilizada para determinar qué tan bien funciona un modelo de machine learning, esta métrica indica cuántas veces un modelo acertó en la clasificación de un item específico [6].

Precision: Esta métrica indica qué tan preciso es el modelo al predecir una categoría específica [6].

Recall: Esta métrica indica qué tan bueno es el modelo para detectar una categoría específica [6].

f1: f1 toma en consideración la precision y el recall para indicar qué tan exacto es un modelo al darle más peso a los falsos negativos y falsos positivos mientras que le quita peso a los verdaderos negativos encontrados por el modelo [6].

D. Curva ROC

La curva ROC (Receiver Operator Characteristc, por sus siglas en inglés) es una métrica de evaluación para problemas de clasificación binaria. Esta curva grafica la tasa de verdaderos positivos en contra de la tasa de de falsos positivos en diferentes umbrales de clasificación. El área debajo de la curva (AUC por sus siglas en inglés) es la medida de la capacidad del modelo de distinguir entre clases y es utilizada como un resumen de la curva ROC [7]. La figura 3 muestra una curva ROC con sus componentes.

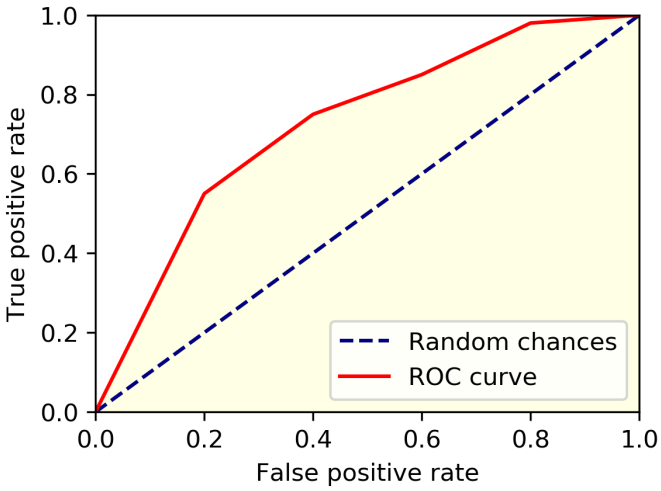

Fig. 3. Curva ROC y sus componentes

E. Validación Cruzada con K-folds

La validación cruzada con K-folds es un método de evaluación de modelos de Machine Learning que permite particionar un dataset en subconjuntos (folds) y ejecutar diferentes simulaciones del modelo con cada uno de estos folds para así permitir a los desarrolladores identificar qué tan bueno es el modelo al ser entrenado y probado con diferentes subconjuntos de datos aleatorios del dataset [8].

III. Metodología

A. Sampling de datos

El primer reto encontrado fue el tamaño de los datasets de entreno y prueba proporcionados por Microsoft. Por lo que se decidió utilizar una muestra pequeña pero significativa de dichos datos. En concreto, se decidió utilizar 1/256 de datos aleatorios del set de entrenamiento (train.csv) que originalmente contenía 8,921,483 datos y 1/256 de datos aleatorios del set de testeo (test.csv) que originalmente contenía 7,853,253 datos.

B. Exploración de los datos

Para la parte de exploración de los datos, el primer paso fue identificar qué columnas del dataset de entreno tenían el mayor porcentaje de datos nulos. Esto indica que dichas columnas tienen demasiado ruido como para ser consideradas

para los modelos.

Posteriormente, se averiguó la cardinalidad de cada columna, es decir, el número de valores únicos de cada columna, se determinó que aquellas columnas con un número alto de valores únicos (mayor a 200) tendrían un impacto negativo en los modelos.

Finalmente, se validó que los registros del dataset estuvieran distribuidos de forma uniforme de acuerdo a la columna HasDetections, que es nuestra variable de interés.

C. Pre-procesamiento

Para el pre-procesamiento, se utilizó la información obtenida en la parte de exploración de datos para limpiar la data y prepararla para los modelos.

Primero, se eliminaron las columnas con un alto porcentaje (mayor a 75) de valores nulos, estas columnas fueron: OrganizationIdentifier, SmartScreen, Census_IsWIMBootEnabled, Census_ThresholdOptIn, Census_InternalBatteryType, Census_IsFlightingInternal, DefaultBrowsersIdentifier, Census_ProcessorClass, y PuaMode.

Luego, se eliminaron las columnas con una alta cardinalidad (mayor a 200 valores únicos) de valores nulos, estas columnas fueron: Unnamed: 0, MachineIdentifier, Census_SystemVolumeTotalCapacity, Census_OEMModelIdentifier, CityIdentifier, Census_FirmwareVersionIdentifier, AvSigVersion, AVProductStatesIdentifier, Census_ProcessorModelIdentifier, Census_InternalBatteryNumberOfCharges, Census_OEMNameIdentifier, Census_PrimaryDiskTotalCapacity, Census_InternalPrimaryDiagonalDisplaySizeInInches, OsBuildLab, GeoNameIdentifier, CountryIdentifier y Census_OSBuildRevision

Posteriormente, por cuestiones de performance, se eliminaron otras columnas del dataset dejando solo las más relevantes, el resultado final fue un dataset con 31 columnas y 69,698 entradas. En la figura 4 se puede apreciar la descripción del dataset final.

Con este dataset limpio, se procedió a identificar las columnas que tuvieran valores NaN y se sustituyeron dichos datos por datos aleatorios de la columna. Las columnas que requirieron que se sustituyeran valores nulos fueron las siguientes: IsProtected, SMode, Firewall, UacLuaenable, Census_ProcessorCoreCount, Census_ProcessorManufacturerIdentifier, Census_PrimaryDiskTypeName, Census_TotalPhysicalRAM, Wdft_IsGamer, y Wdft_IsGamer.

Finalmente, para solucionar el problema de tener muchas columnas con variables categóricas se procedió a volver esas columnas una representación numérica, haciendo uso de la función get_dummies de la librería pandas de Python. Una vez que se obtuvo la representación numérica de las columnas, se eliminó su versión categórica del dataset.

```
RangeIndex: 69698 entries, 0 to 69697
Data columns (total 31 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   EngineVersion                           69698 non-null  category
 1   AppVersion                              69698 non-null  category
 2   HasTpm                                  69698 non-null  int64
 3   LocaleEnglishNameIdentifier             69698 non-null  int64
 4   Platform                                69698 non-null  category
 5   Processor                               69698 non-null  category
 6   OsVer                                   69698 non-null  category
 7   OsBuild                                 69698 non-null  int64
 8   OsSuite                                 69698 non-null  int64
 9   OsPlatformSubRelease                    69698 non-null  category
 10  SkuEdition                              69698 non-null  category
 11  IsProtected                             69421 non-null  float64
 12  SMode                                   65519 non-null  float64
 13  Firewall                                68984 non-null  float64
 14  UacLuaenable                            69608 non-null  float64
 15  Census_MDC2FormFactor                   69698 non-null  category
 16  Census_DeviceFamily                     69698 non-null  category
 17  Census_ProcessorCoreCount               69338 non-null  float64
 18  Census_ProcessorManufacturerIdentifier  69338 non-null  float64
 19  Census_PrimaryDiskTypeName              69598 non-null  category
 20  Census_HasOpticalDiskDrive              69698 non-null  int64
 21  Census_TotalPhysicalRAM                 69057 non-null  float64
 22  Census_OSVersion                        69698 non-null  category
 23  Census_OSArchitecture                   69698 non-null  category
 24  Census_OSBranch                         69698 non-null  category
 25  Census_OSEdition                        69698 non-null  category
 26  Census_OSSkuName                        69698 non-null  category
 27  Census_IsSecureBootEnabled              69698 non-null  int64
 28  Wdft_IsGamer                            67296 non-null  float64
 29  Wdft_RegionIdentifier                   67296 non-null  float64
 30  HasDetections                           69698 non-null  int64
```

Fig. 4. Descripción del dataset luego de la primera parte del pre-procesamiento

D. Modelo 1: Decision Trees

Puesto que la clasificación de los datos, en un nivel simple, resultaba en una decisión binaria (si una máquina sería infectada o no) se escogió el algoritmo de Árboles de decisión como uno de los dos algoritmos para utilizar con el dataset resultante. Para la implementación de este algoritmo primero se dividió el dataset en nuestra columna de la variable de interés (hasDetections) y el resto de las columnas. Luego, se obtuvieron las variables de entreno y prueba (con una distribución de 0.6 y 0.4 de los datos respectivamente). Se construyó y ajustó el modelo haciendo uso de la libreria sklearn y se calcularon las métricas de evaluación: accuracy, precision, recall, roc auc, y f1 score. Finalmente, se calculó la gráfica de la curva ROC y se realizó una evaluación cruzada con K-folds para k=10.

E. Modelo 2: Random Forest

Como se ha mencionado antes, el algoritmo Random Forest resulta ser más robusto que los decision trees para tareas de clasificación de datos en conjuntos con alta dimensionalidad y tipos de características heterogéneas, por lo que este fue el segundo algoritmo escogido para utilizar con el dataset resultante. Para la implementación de este algoritmo primero se dividió el dataset en nuestra columna de la variable de interés (hasDetections) y el resto de las columnas. Luego, se obtuvieron las variables de entreno y prueba (con una

distribución de 0.6 y 0.4 de los datos respectivamente). Se construyó y ajustó el modelo haciendo uso de la libreria sklearn y se calcularon las métricas de evaluación: accuracy, precision, recall, roc auc, y f1 score. Finalmente, se calculó la gráfica de la curva ROC y se realizó una evaluación cruzada con K-folds para k=10.

IV. Resultados

Para el modelo construido con Decision Trees, se obtuvieron las siguientes métricas: 0.529 de accuracy, 0.526 de precision, 0.524 de recall, 0.529 de roc auc, y 0.525 de f1 score. Esto nos indica que el modelo tiene poco más de 50 por ciento de exactitud, es medianamente preciso al predecir una categoría específica (máquinas con riesgo de ser infectadas) así como para detectar una categoría específica (máquinas con o sin riesgo de ser infectadas). Como se puede ver el f1 score es muy similar a la accuracy, por lo que se pueden tomar como iguales. Estos resultados se pueden apreciar en la figura 5.

En la figura 6 se puede apreciar la gráfica de la curva ROC, con un AUC de 0.53, que confirma lo obtenido en las métricas de evaluación, el modelo es medianamente exacto en clasificar de forma correcta los datos.

Finalmente, en la figura 7 se evidencian los resultados de la validación cruzada con 10 folds, la similitud de los resultados indica que el modelo está bien entrenado y que proveerá resultados muy cercanos si se entrena con datos distintos del dataset.

```
La accuracy es de ->  0.5298780487804878
La precision es de ->  0.5268006092696018
El recall es de ->  0.5245558283980933
El roc auc score es de ->  0.5298424004467309
El f1 score es de ->  0.5256758223862773
```

Fig. 5. Métricas de evaluación del modelo construido con Decision Trees

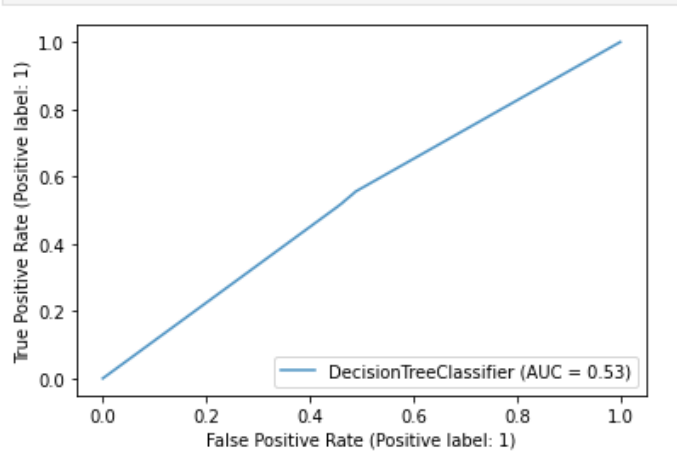

Fig. 6. Curva ROC del modelo construido con Decision Trees

```
[0.52525108 0.52482066 0.53084648 0.53644189 0.52883788 0.5251076
 0.517934   0.52166428 0.51973023 0.53192711]
```

Fig. 7. Resultados de la evaluación cruzada con k = 10 para el modelo construido con Decision Trees

Con el modelo Random Forest, como era de esperarse, se obtuvieron mejores resultados en términos de exactitud, área bajo la curva ROC y la validación cruzada fue estable en todos los folds. Para este modelo, se obtuvieron las siguientes métricas: 0.554 de accuracy, 0.551 de precision, 0.550 de recall, 0.554 de roc auc, y 0.551 de f1 score. Esto nos indica que el modelo tiene poco más de 50 por ciento de exactitud, es medianamente preciso al predecir una categoría específica (máquinas con riesgo de ser infectadas) así como para detectar una categoría específica (máquinas con o sin riesgo de ser infectadas). Como se puede ver el f1 score es muy similar a la accuracy, por lo que se pueden tomar como iguales. Estos resultados se pueden apreciar en la figura 8.

En la figura 9 se puede apreciar la gráfica de la curva ROC, con un AUC de 0.57, que confirma lo obtenido en las métricas de evaluación, el modelo es medianamente exacto en clasificar de forma correcta los datos.

Finalmente, en la figura 10 se evidencian los resultados de la validación cruzada con 10 folds, la similitud de los resultados indica que el modelo está bien entrenado y que proveerá resultados muy cercanos si se entrena con datos distintos del dataset.

```
La accuracy es de ->  0.5548421807747489
La precision es de ->  0.5519287833827893
El recall es de ->  0.5507727863642929
El roc auc score es de ->  0.5548149238932766
El f1 score es de ->  0.5513501789393775
```

Fig. 8. Métricas de evaluación del modelo construido con Random Forest

Para futuros estudios se recomienda utilizar más columnas del dataset original para entrenar los modelos, en caso de que el performance no sea un problema, así como utilizar otro algoritmo de clasificación más robusto como svm, por ejemplo. Además, se recomienda Utilizar un sample más grande del dataset y/o utilizar una proporción más grande para la data de entreno en la creación de los modelos.

V. Conclusiones

- El algoritmo Decison Trees tiene una accuracy de 52.9 por ciento para predecir qué máquinas corren riesgo o no de ser infectadas.
- El algoritmo Random Forest tiene una accuracy de 55.4 por ciento para predecir qué máquinas corren riesgo o no de ser infectadas.

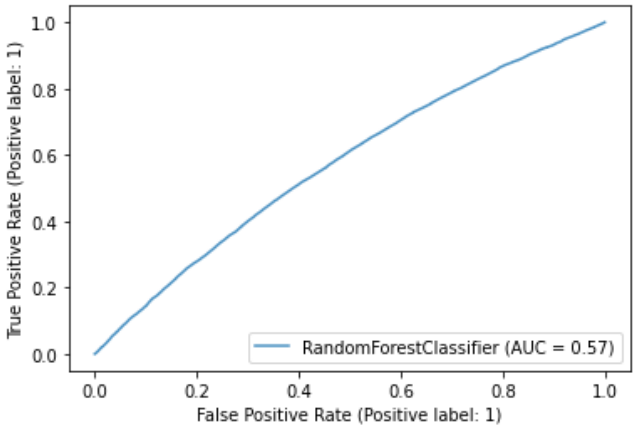

Fig. 9. Curva ROC del modelo construido con Random Forest

```
[0.5492109  0.55466284 0.56183644 0.56628407 0.55494978 0.55423242
 0.55021521 0.56169297 0.55660783 0.56234754]
```

Fig. 10. Resultados de la evaluación cruzada con k = 10 para el modelo construido con Random Forest

- Al ser un algoritmo más robusto, el algoritmo Random Forest resulta ser 2.5 por ciento mejor que el algoritmo Decision Trees para predecir si una máquina corre riesgo o no de ser infectada.

REFERENCES

[1] Rohith, C., y Kaur, G. (2021). "A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus,". 2nd International Conference on Intelligent Engineering and Management (ICIEM). doi: 10.1109/ICIEM51511.2021.9445322.

[2] Dunham, K. (2009). Mobile Malware Attacks and Defense. Estados Unidos. Syngress Publishing, Inc.

[3] Rokach, L., y Maimon, O. (2005). Decision Trees. 10.1007/0-387-25465-X_9.

[4] Alassar, Z. (2020). DECISION TREE AS AN IMAGE CLASSIFICATION TECHNIQUE. Islamic Univesity of Gaza.

[5] Wood, T. (s.f.) What is a Random Forest?. Extraído de https://deepai.org/machine-learning-glossary-and-terms/random-forest

[6] Shung, K. (2018). Accuracy, Precision, Recall or F1?. Extraído de https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

[7] Bradley, A. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. doi:10.1016/s0031-3203(96)00142-2

[8] Berrar, D. (2018). Cross-Validation. Tokyo Institute of Technology. 10.1016/B978-0-12-809633-8.20349-X.