Ejercicio: Clasificación de Malware usando el dataset personalizado

Integrantes:

- Ivonne Campoverde Pilco
- Jimmy Tarira Pérez

Objetivos:

- Construir un clasificador de malware utilizando un dataset personalizado creado específicamente para este propósito.
- Aplicar técnicas de reducción de dimensionalidad.
- Aplicar modelos de aprendizaje.
- Analizar resultados obtenidos de la aplicación de los modelos de aprendizaje.

Tareas:

1. **Preparación del dataset:**
   - Explora el dataset y familiarízate con las características.
   - Procesar el dataset para manejar valores nulos o inconsistentes.
   - Aplica técnicas de reducción de dimensionalidad si es necesario.
   - Separa el dataset en conjuntos de entrenamiento, validación y prueba.
2. **Selección de técnicas de aprendizaje automático:**
   - Investiga diversas técnicas de aprendizaje automático para la clasificación de malware.

     - **Análisis de Firmas:**
       Implica la comparación de patrones conocidos de malware (firmas) con los archivos en busca de coincidencias.
       Puede utilizarse técnicas estadísticas que proporciones precisión en la detección de firmas y adaptarse a nuevas variantes de malware.
     - **Análisis Heurístico:**
       Análisis del comportamiento sospechoso de un programa para determinar si es malicioso o no.
       Se utilizan algoritmos de aprendizaje automático que identifican patrones y comportamientos maliciosos basándose en conjuntos de datos de comportamiento de malware y benigno.
     - **Análisis de Secuencias Temporales:**
       Observación del comportamiento de ejecución de un programa durante un período de tiempo y detecta actividades maliciosas basadas en patrones temporales.
       Se aplican modelos como redes neuronales recurrentes (RNN) y modelos LSTM, para capturar patrones temporales.
     - **Clasificación Basada en Características Estáticas:**
       Analiza características estáticas de los archivos, como el código binario, para clasificar el malware.
       Se utilizan algoritmos de aprendizaje automático, como máquinas de soporte vectorial (SVM) y árboles de decisión, pueden ser entrenados con características estáticas para la clasificación.
     - **Análisis de Redes:**

Examina las comunicaciones de red y patrones de tráfico para identificar comportamientos maliciosos.
Los clasificadores basados en árboles o redes neuronales, pueden procesar datos de red y detectar patrones maliciosos.

- **Aprendizaje Profundo (Deep Learning):**
  Utiliza arquitecturas de aprendizaje profundo, como redes neuronales profundas, para aprender representaciones complejas y características para la clasificación de malware.
- **Ensembles de Modelos:**
  Combina múltiples modelos de aprendizaje automático para mejorar la robustez y la precisión de la clasificación.
- Se aplican técnicas como Bosques Aleatorios (Random Forest) y Gradient Boosting son utilizadas para crear ensembles que pueden adaptarse a una variedad de escenarios de malware.

- Selecciona una o más técnicas adecuadas (modelos de ML) para este problema, considerando el desbalance de clases.
- Justifica la elección de las técnicas seleccionadas.

3. **Entrenamiento del modelo:**
   - Implementa las técnicas de aprendizaje automático elegidas.
   - Entrena el modelo utilizando el conjunto de entrenamiento.
   - Optimiza los hiperparámetros del modelo para obtener la mejor precisión posible, considerando el desbalance de clases.
4. **Evaluación del modelo:**
   - Evalúa el rendimiento del modelo en los conjuntos de validación y prueba.
   - Calcula métricas de evaluación apropiadas para el problema, como precisión, sensibilidad, especificidad y AUC-ROC.
   - Analiza los resultados de la evaluación e identifica posibles mejoras.
5. **Interpretación del modelo:**
   - Analiza los resultados del modelo para identificar las características más importantes para la clasificación de malware.
   - Visualiza los resultados del modelo para comprender mejor su funcionamiento.

Informe:

1. **Link a repositorio público de GitHub con la siguiente información en el readme:**

- https://github.com/BonecitaCP/clasificaci-nMalware-/tree/main
- Archivo ejercicio-malware: aplica los modelos de aprendizaje
  - Random Forest
  - Gradient Boosting
  - Regresión lineal
  - Árbol de decisión
- Archivo entregable2_Malware_PCA: aplica los modelos de aprendizaje
  - Random Forest
  - Gradient Boosting
- Archivo entregable2_Malware_TSNE: aplica los modelos de aprendizaje
  - Random Forest
  - Gradient Boosting

2. Descripción del dataset personalizado.

2.1. Características generales

- Tamaño: 373 muestras (301 maliciosas, 72 benignas)
- Desbalance: Prevalencia de muestras maliciosas
- Características: 531 características representadas como F_1 a F_531
- Etiqueta: Columna indicando si el archivo es malicioso o no (true para malware)
- Nota: Las características binarias se representan numéricamente por simplicidad (F_1, F_2, etc.)

2.2. Cambios

- Etiqueta: Cambio a True o False para aplicar técnicas de reducción de dimensionalidad
- Agregar columna **Int** para validar los datos de la etiqueta en 1 (malicioso) o 0 (No malicioso)

2.3. Proceso

1. Se mantuvo el número de registros del dataset general, en los dos ejercicios se agregaron nuevas columnas para diferenciar los datos correspondientes a las etiquetas, en común se agrego una columna de tipo **int** que permite visualizar la etiqueta no maliciosos como 0 y maliciosos como 1, esto permitió aplicar las técnicas de reducción de dimensionalidad.
2. Se validaron los datos, el tamaño tanto del dataset principal, como de los conjuntos de prueba y entrenamiento, se validaron los valores nulos y se rellenaron, aunque realmente en la validación inicial no se encontraron datos faltantes, pero para aplicar las técnicas de reducción de dimensionalidad era necesario generar estas validaciones, así como, la estandarización de la escala de los datos.
3. Para obtener un conjunto de datos que permita la aplicación de los modelos de datos, en las dos técnicas de reducción se configuro el parámetro n_components=2 y con ello se aplico la validación de la varianza entre X y Y.

3. Técnicas de aprendizaje automático utilizadas y justificación de su elección.

- Random Forest
- Gradient Boosting
- Regresión lineal
- Árbol de decisión

4. Resultados del entrenamiento y la evaluación del modelo.

- **Regresión lineal**
    - Se obtuvo una precisión del 64% y como era de suponerse la precisión positiva del 100% debido a que mayormente la presencia en la muestra es positiva, en la matriz de confusión se puede apreciar que los falsos positivos son de 27 a comparación de los aciertos que son de 31 se nota que no obtenemos las predicciones adecuadas.
- **Árbol de decisión**
    - Se obtuvo una precisión del 94% dejando el número de árbol infinito, lo cual mejora notablemente la predicción. Quizás una cuestión en este caso es que, si se da un falso positivo a una aplicación como antivirus, este bloqueará el archivo y si es una DLL importante en Windows afectará el desarrollo del sistema operativo, por esta razón buscamos otro modelo que mejore la predicción.
- **Random Forest**
    - Se obtuvo una precisión del 97.3%, es una mejora que aparentemente es poca a comparación del árbol de decisión, pero en producción tiene mucha relevancia ya que afectará menos al desarrollo del sistema operativo o dónde se lo aplique.

- **Gradient Boosting**
    - Se obtuvo una precisión de 97.3% y un balance con respecto a los aciertos en positivo y negativo siendo casi iguales lo que es óptimo; estos resultados son casi iguales a los de RandomForest, pero por la técnica del balance de las muestras concluyo que el modelo de GradientBoosting es el que predice de mejor manera en este caso en particular

5. Análisis crítico de los resultados del modelo.

- La complejidad del conjunto de árboles dificulta la capacidad de explicar cómo se obtuvieron los resultados, ya que su inclinación en la curva del balance óptimo está inclinada hacia el lado derecho o sea hacia la complejidad eso en producción podría dar un sesgo hacia los positivos malware y no se podría obtener los mismos resultados buenos obtenidos en entrenamiento y test.
- Debido a que, en el conjunto de datos, el número de registros no maliciosos es considerablemente menor con respecto a los registros que denotan malware, la posibilidad de que los modelos o técnicas de aprendizaje reporten falsos positivos es mayor, por lo cual se debe tomar en cuenta métodos que permitan diferenciar de manera más optima los datos tanto de entrenamiento como de test.
- Los métodos de Random Forest y Gradient Boosting, fueron ejecutados por dos usuarios en diferentes máquinas, sin embargo, al definir los parámetros de división de los datos de entrenamiento y test bajo los mismos parámetros, los resultados coincidieron en cuento a predicción, exactitud, precisión, sensibilidad, especificidad, y la curva discriminativa.
- Por otra parte, se pudo observar que en cuanto a exactitud si se puede notar diferencia en los resultados obtenidos al aplicar diferentes técnicas de reducción de dimensionalidad, en base a los resultados obtenidos, el valor de exactitud obtenido con la técnica TSNE--Distributed Stochastic Neighbor Embedding (t-SNE), fue mas exacta que con la aplicación de PCA -Análisis de Componentes Principales.

6. Interpretación del modelo y características más relevantes.

- La aplicación de técnicas de reducción de dimensionalidad elimino de manera significativa el tiempo de procesamiento y diferenciar tendencias y patrones de los registros.
- Al aplicar los métodos de aprendizaje se pudo comparar los resultados de la regresión lineal, con los resultados de la aplicación de varias cantidades de árboles, sin embargo, los resultados no reflejaron mayor diferencia.
- Con el método RandomForest se obtuvo resultados más precisos a comparación de la regresión lineal, sin embargo, no se considera como el modelo óptimo de aprendizaje.
- Finalmente debido al desbalance del dataset se seleccionó el GradientBoosting debido a que con este modelo se obtienen muestras sintéticas y con eso se balancean los datos, y se obtuvo el mejor resultado
- Las configuraciones y juegos de parámetros aplicados en cada modelo permitieron diferenciar y concluir que el método de GradientBoosting presento un nivel de precisión más óptimos con respecto a los otros métodos