Informe Módulo 6

Machine Learnig

Fecha: 13 de Abril de 2024

Autor: **Azael Ramírez Pérez**

Mail: **keepcoder_test@gmail.com (ficticio)**

Empresa: **KeepCoder.inc (ficticio)**

Contenido

Problemática.....3
Solución actual del problema.....4
Propuesta de solución al problema.....5
KPIs – Indicadores de negocio.....6
Validación de la propuesta.....7
Experimentación.....9

Problemática

Es verdad que durante los últimos años se han incrementado de manera exponencial los ataques informáticos causando graves daños monetarios a organizaciones en todo el mundo.

Actualmente diversas compañías dedican muchos recursos al desarrollo de antivirus; sin embargo la complejidad, la velocidad de propagación y la capacidad polimórfica que posee actualmente el malware moderno representan enormes desafíos.

Motivados por encontrar nuevas alternativas, la comunidad de científicos de datos ha descubierto que la utilización de técnicas de Machine Learning y Deep Learning para la detección y clasificación de malware puede ofrecer una opción más que competitiva.

El presente trabajo propone utilizar el algoritmo KNN (K-Nearest Neighbors) de aprendizaje automático supervisado utilizado para tareas de clasificación y regresión, en este caso se utilizara para clasificar distintas muestras de malware.

Solución actual del problema

La mayoría de empresas dedicadas al desarrollo de anti-virus, por mencionar algunas Kaspersky, Norton, AVG hacen su mejor esfuerzo para hacer frente y detectar Malware, sin embargo de manera tradicional este tipo de software utilizan el método **signature-based** para la detección.

La **signature** es una secuencia corta de bytes que sirve para identificar Malwares conocidos, sin embargo este método no es fiable pues no pude identificar ataques zero-day, o malwares polimórficos que utilizan técnicas de ofuscación, dado que no cuentan con registros de los mismos.

Propuesta de solución al problema

Se pretende usar el algoritmo KNN (K-Nearest Neighbors) pues puede analizar las características de un archivo (como tamaño, llamadas a la API, secciones de código) y compararlo con archivos conocidos como malware en el espacio de características. En el caso de que los k vecinos más cercanos del archivo nuevo pertenecen principalmente a la clase de malware, se puede clasificar como potencialmente malicioso.

KPIs – Indicadores de negocio

El seguimiento de KPIs específicos para KNN en la detección de malware es crucial para evaluar su eficacia y optimizar su rendimiento. La selección de los KPIs adecuados, el establecimiento de objetivos realistas y la medición regular del rendimiento son esenciales para tomar decisiones informadas sobre la implementación y el uso del algoritmo.

Se listan los KPIs específicos para KNN:

- **Precisión:** Porcentaje de archivos correctamente clasificados como malware o benignos.
- **Precisión por clase:** Precisión individual para la detección de malware y archivos benignos.
- **Tasa de falsos positivos:** Porcentaje de archivos benignos erróneamente clasificados como malware.
- **Tasa de falsos negativos:** Porcentaje de archivos maliciosos erróneamente clasificados como benignos.
- **Tiempo de detección:** Tiempo promedio que tarda el algoritmo en clasificar un nuevo archivo.
- **Memoria utilizada:** Cantidad de memoria que utiliza el algoritmo durante la clasificación.
- **Sensibilidad a la elección de K:** Medir cómo la precisión varía con diferentes valores de K.
- **Capacidad de detección de malware nuevo:** Evaluar la eficacia del algoritmo para detectar malware que no se ha visto antes.

Validación de la propuesta

Para la validación del algoritmo KNN se usara principalmente la técnica de validación cruzada la cual es utilizada principalmente en el aprendizaje automatico (machine learning) para evaluar el rendimiento de un modelo predictivo.

La idea principal detrás de la validación cruzada es dividir el conjunto de datos en subconjuntos más pequeños, llamados pliegues (folds en inglés), y luego utilizar estos pliegues de manera rotativa para entrenar y evaluar el modelo.

El proceso típico de validación cruzada implica los siguientes pasos:

- **División de datos:** El conjunto de datos se divide en k subconjuntos aproximadamente iguales, llamados pliegues. Un valor común para k es 5 o 10, aunque esto puede variar según el tamaño del conjunto de datos y las preferencias del usuario.
- **Iteración:** El modelo se entrena k veces, utilizando k-1 pliegues para el entrenamiento y el pliegue restante para la evaluación en cada iteración.
- **Evaluación del rendimiento:** Después de cada iteración, se evalúa el rendimiento del modelo utilizando el pliegue de prueba (el pliegue que no se utilizó para el entrenamiento en esa iteración). Se registran las métricas de rendimiento, como precisión, sensibilidad, especificidad, etc.
- **Promedio de resultados:** Finalmente, se promedian los resultados de rendimiento obtenidos en las k iteraciones para obtener una estimación general del rendimiento del modelo.

La validación cruzada ayuda a obtener una estimación más precisa del rendimiento del modelo, ya que utiliza todos los datos disponibles tanto para el entrenamiento como para la evaluación. Esto ayuda a reducir la variabilidad en la estimación del rendimiento que podría surgir de una sola división de datos en entrenamiento y prueba. Además, la validación cruzada proporciona una forma de evaluar cómo se generaliza el modelo a diferentes subconjuntos de datos, lo que ayuda a detectar problemas de sobreajuste o subajuste.

Experimentación

El algoritmo K-Nearest Neighbors es un algoritmo de clasificación de datos que clasifica un objeto desconocido asignándole una clase que represente a la mayoría de sus vecinos más cercanos. Sean los objetos el conjunto de grafos D, conjunto de clases C y un conjunto de entrenamiento T rS = (Gj , cj)M j=1 donde Gj ∈ D es un grafo y cj ∈ C es la clase del grafo.

El clasificador KNN induce a la T rS a una función de mapeo f : G → C que asigna una clase a un gráfico desconocido del conjunto de prueba TeS. El problema de K-Nearest Neighbors se puede definir con la siguiente expresión:

$$(G*, c*) = \arg_{(Gj ,cj)\in T\ rS} \min d(Gi, Gj)\forall Gi \in T\ eS$$

Donde d(Gi, Gj) es la métrica utilizada para calcular una disimilitud entre Gi y Gj . En definición, el número de llamadas a la función de disimilitud es igual a M , donde M es el tamaño del conjunto de entrenamiento. Para extender la expresión mencionada anteriormente de K-Nearest Neighbors, se introduce K en el conjunto de KNN dentro de la consulta del gráfico Gi ∈ T eS. Sea K = (G1, c1), ..., (Gj , cj), ..., (Gk, ck) un conjunto de gráficos junto con sus etiquetas de clase (Gj , cj) ∈ T rS la expresión anterior quedaría :

$$K = \arg_{(Gj ,cj)\in T\ rS} \text{sort } d(Gi, Gj)\forall Gi \in T\ eS$$

Donde sort es una función que realiza una ordenación ascendente de valores d(Gi, Gj), k es el número de valores retenidos para elegir el número de vecinos más cercanos de Gi. Para utilizar la expresión formulada anteriormente en un contexto de clasificación se debe definir un operador de votación.

El operador de votación máxima es una función $\tau : K \rightarrow C$ definida por:

$$\mathbf{c*j = argcj \in C \max mcj\ (K)}$$

Donde mcj es una función que cuenta el número de observaciones que caen en cada clase cj . El clasificador KNN ha sido muy utilizado. El uso de este clasificador es muy sencillo ya que es una técnica no paramétrica y por tanto no necesita conocimientos sobre la distribución

de clases. Además, cuando se define la métrica, el clasificador KNN puede proporcionar una explicación del resultado de la clasificación y, por lo tanto, el clasificador KNN tiene una ventaja sobre los otros clasificadores que se consideran modelos de caja negra.

Se ha demostrado que cuando hay suficientes patrones de entrenamiento, el error de clasificación es menor que el error de Bayes.