Instituto Tecnológico y de Estudios Superiores de Monterrey

Análisis de grandes volúmenes de datos

Proyecto | Entrega proyecto final

Fanny Betsabé Fuentes Reyes	A00570705
Luis Miguel Balderas González de Burgos	A01373679
Héctor Eduardo Santillán Padilla	A01633395

06/19/2025

Resumen

Este proyecto consta del análisis del conjunto de datos “Microsoft Security Incident Prediction”, con el fin de desarrollar un modelo que permita anticipar incidentes de seguridad informática dentro de redes corporativas. El dataset contiene variables como el sistema operativo, tipo de dispositivo, ubicación geográfica, uso de VPN, entre otras, que ayudan a identificar patrones relacionados con dispositivos que podrían representar un riesgo. A través de técnicas de Big Data, métricas de calidad de resultados, visualizaciones y algoritmos de aprendizaje automático, obtenemos como resultado un modelo predictivo que permite mejorar la detección temprana de comportamientos extraños, facilitando la toma de decisiones preventivas. Este método no solo fortalece la defensa frente a potenciales amenazas, sino que también proporciona un entendimiento más profundo de los elementos que influyen en la aparición de incidentes, contribuyendo a una creación de estrategia más efectiva en el campo de la ciberseguridad organizacional.

Introducción

En la actualidad, en la que la infraestructura tecnológica dentro de las organizaciones cada vez se encuentran más vulnerables frente a amenazas digitales, la manera de prevenir incidentes de seguridad informática se ha transformado en un elemento estratégico de la ciberseguridad. Los ciberataques no solo provocan daños financieros y operativos, sino que también impactan en la confianza y reputación de las instituciones. Frente a este reto, la implementación de modelos predictivos basados en datos aparece como una respuesta útil para identificar riesgos antes de que se vuelvan una amenaza real.

Este proyecto toma como base el conjunto de datos “Microsoft Security Incident Prediction”, publicado por Microsoft en la plataforma Kaggle. Estas variables se agrupan en vectores de características (features) que alimentan un modelo supervisado cuya variable objetivo es Category_idx, que simboliza el tipo de incidente.

El objetivo de este proyecto es crear un modelo de aprendizaje automático que pueda predecir la clasificación de incidentes de seguridad informática basándose en datos provenientes de dispositivos de la empresa. Para ello, se utilizarán métodos escalables que posibiliten el manejo y análisis de grandes volúmenes de datos. Además de reconocer las combinaciones de variables que ejercen un mayor impacto en la aparición de situaciones de riesgo, con el objetivo de potenciar la habilidad predictiva del modelo y proporcionar datos valiosos para la toma de decisiones en el campo de la ciberseguridad.

Propuesta de Solución

La propuesta consiste en desarrollar un sistema basado en modelos de aprendizaje automático que permitan identificar patrones asociados a la severidad y categoría de incidentes de seguridad informática en redes corporativas. Para ello, se aprovechará el dataset de Microsoft Security Incident Prediction y se aplicarán técnicas de Big Data mediante PySpark, lo que facilita el manejo y procesamiento eficiente de grandes volúmenes de datos.

El objetivo es construir modelos predictivos que ayuden a priorizar incidentes, mejorar la toma de decisiones en los Centros de Operaciones de Seguridad (SOC) y automatizar la respuesta ante amenazas digitales. Para lograrlo, se seguirán etapas rigurosas que incluyen la caracterización de la población de datos, su preparación y división en conjuntos de entrenamiento y prueba, selección de algoritmos, ajuste de hiperparámetros y evaluación basada en métricas adecuadas.

1. Caracterización de la población

La población detectada en el dataset contiene datos sobre dispositivos, características técnicas y categorización de incidentes de seguridad informática. Las variables clave seleccionadas para el análisis incluyen:

- IncidentGrade: nivel de gravedad del incidente,
- EntityType: tipo de entidad involucrada,
- ResourceType: tipo de recurso (por ejemplo, servidores, endpoints),
- OSFamily: familia del sistema operativo,
- Category: clase del incidente, que se utilizó como variable objetivo.

Este conjunto de datos fue limpiado para eliminar registros con valores nulos en las variables clave, garantizando así la calidad del análisis posterior.

2. Recolección de Datos

Considerando que el volumen total de esta base es significativo y el manejo de grandes cantidades de información puede resultar costoso en computación, se sugiere la recolección de una muestra de dimensión limitada, con el objetivo de mantener una eficacia razonable en los periodos de procesamiento y entrenamiento del modelo.

Se implementó un muestreo estratificado por partición. Esto implicó segmentar la base de datos inicial D de acuerdo a las combinaciones de las variables de caracterización para generar particiones uniformes. A partir de estas, se obtuvo un número restringido de instancias por categoría a través de la función sampleBy() de PySpark, escogiendo un 30% para cada estrato de la variable objetivo Category, eligiendo un 30% para cada estrato de la variable objetivo Category.

El resultado final “M”, nos asegura una representación proporcional de las distintas clases de incidentes de seguridad. Esto no solo minimiza el sesgo vinculado al desequilibrio de clases, sino que también disminuye considerablemente los gastos computacionales. Esta táctica se sustentó teóricamente en las sugerencias metodológicas de Kim & Wang (2019) acerca del muestreo en Big Data, además de las directrices prácticas sobre técnicas de muestreo presentadas por Scribbr (s.f.).

3. Construcción de la muestra y estrategia de muestreo

Para disminuir la complejidad computacional sin poner en riesgo la representatividad de la información, se elaboró una muestra equilibrada a partir del dataset que había sido previamente purificado. Esta muestra se produjo a través de la combinación de subconjuntos obtenidos de particiones uniformes de la base de datos, los cuales se determinaron de acuerdo a las variables de caracterización previamente definidas.

El muestreo estratificado implementado en la fase previa mantiene la distribución de clases de la variable objetivo en la muestra final, lo que facilita disponer de una base sólida para el entrenamiento y evaluación de modelos. La muestra obtenida posee una dimensión contenida, apta para ser procesada eficazmente en ambientes distribuidos como PySpark, manteniendo la riqueza estructural de la población inicial.

4. Preparación de conjuntos de entrenamiento y prueba

Con la muestra escogida, se realizó un proceso de preprocesamiento para organizar los datos con el propósito de entrenar modelos de aprendizaje automático a través de PySpark. Primero, se codificaron las variables categóricas IncidentGrade, EntityType, ResourceType y OSFamily utilizando la herramienta StringIndexer, lo que resultó en la creación de nuevas columnas numéricas correspondientes, reconocidas con el sufijo _idx. La variable Category fue transformada en formato numérico con el nombre Category_idx.

Después, las variables codificadas se unieron en un solo vector de características utilizando VectorAssembler, guardando el vector en la columna de características. Este método facilitó la transformación de los datos iniciales en un formato organizado y apropiado para su manejo por los algoritmos de aprendizaje automático existentes en PySpark.

5. Selección de métricas para medir la calidad de los resultados

Para la evaluación de desempeño del modelo se hizo uso del métrica MulticlassClassificationEvaluator con accuracy como métrica principal. Esta métrica señala la cantidad de predicciones correctas en relación al total de casos evaluados. Además, se empleó la matriz de confusión para examinar los errores por clase y valorar el comportamiento del modelo en cada una de las categorías de incidentes.

6. Selección de algoritmos de aprendizaje

Una vez finalizado el preprocesamiento de la información, se llevó a cabo la segmentación del conjunto en dos secciones: un subconjunto para entrenamiento (80%) y otro para prueba (20%). Con esta división se logró el entrenamiento del modelo con una partición representativa de los datos y así mismo mantener una partición para valorar su desempeño evitando sesgos.

El modelo que se implementó fue un clasificador concretamente un modelo de bosque aleatorio (Random Forest), adaptado para tareas de clasificación con diversas categorías. Este modelo trabaja generando diversos árboles a partir de diferentes muestras del conjunto de entrenamiento, combinando sus predicciones para poder conseguir una respuesta final más sólida y exacta.

En el proceso de entrenamiento, el modelo aprendió a reconocer patrones en la información para asociar combinaciones de características con las categorías pertinentes. Después de ser entrenado, se aplicó en el conjunto de prueba para obtener predicciones, las cuales fueron comparadas con las reales. Con esta evaluación, se lograron métricas de evaluación que facilitaron la medición de la exactitud del modelo y su habilidad para generalizar frente a datos nuevos.

7. Técnicas para el ajuste de hiperparámetros

Para mejorar el desempeño del modelo, se realizó un ajuste de hiperparámetros, evaluando diversas combinaciones de valores para los parámetros más significativos. Se evaluaron dos alternativas para la cantidad de árboles (20 y 50) y dos valores para la profundidad máxima de los árboles (5 y 10). Para asegurar una evaluación sólida y prevenir el sobreajuste, se empleó una validación cruzada de cinco segmentos k=5, lo que facilitó el entrenamiento y validación del modelo. Esta metodología simplificó la elección de la configuración más adecuada, optimizando el rendimiento del modelo en cuanto a exactitud y capacidad para generalizar ante datos nuevos, lo cual es de suma importancia en contextos de clasificación multiclase.

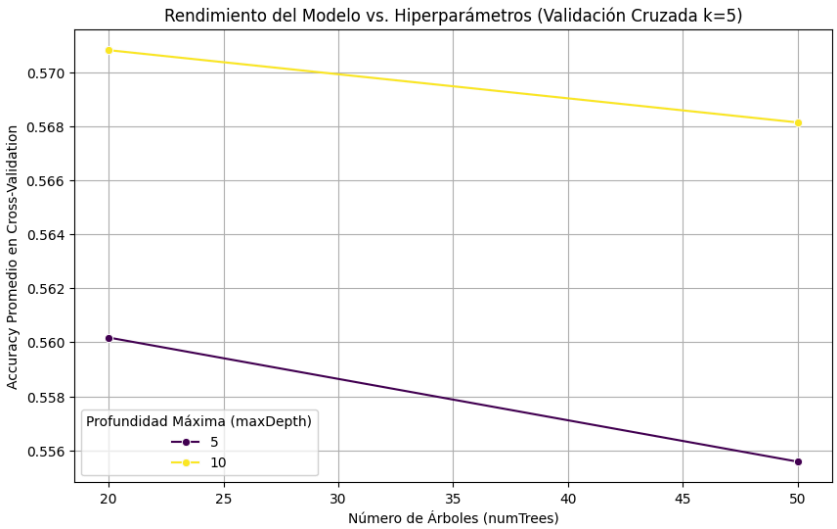
Experimentación

Proceso de ajuste de hiperparámetros

En el procedimiento de modificación de hiperparámetros en el modelo, se analizaron diversas combinaciones del número de árboles y la profundidad máxima de cada árbol a través de una validación cruzada con k = 5. En la imagen se puede ver la representación del desempeño promedio (exactitud) logrado para cada configuración analizada.

Profundidad del árbol: Una profundidad incrementada facilita la captura de relaciones más complejas en los datos, lo que se ve reflejado en un mayor rendimiento. Sin embargo, una profundidad excesiva nos puede dar como resultado un sobreajuste, por lo que este resultado debe ser complementado con otros indicadores tal como lo es la matriz de confusión.

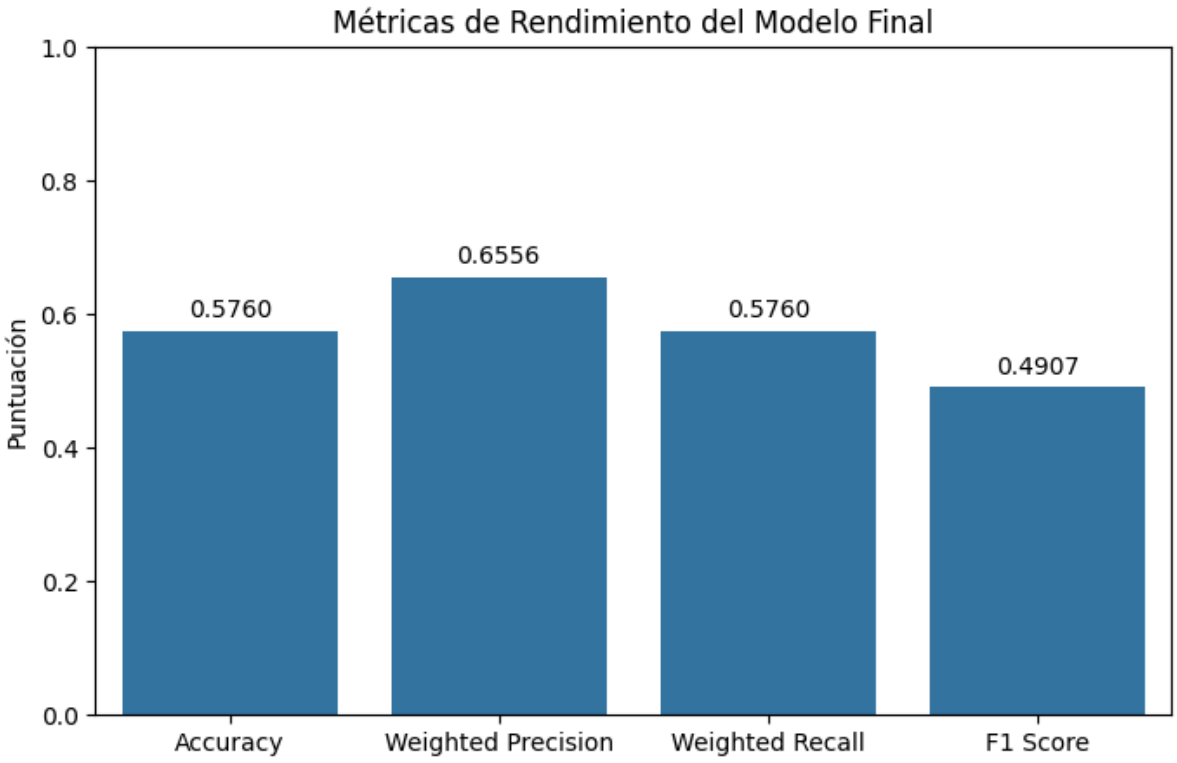
Número de árboles: A pesar de que en teoría un incremento en el número de árboles suele mantener el modelo estable, en este experimento se observó un leve descenso en el rendimiento, lo que podría atribuirse a una varianza más elevada o a un leve sobreajuste en el conjunto de validación.



Evaluación del mejor modelo

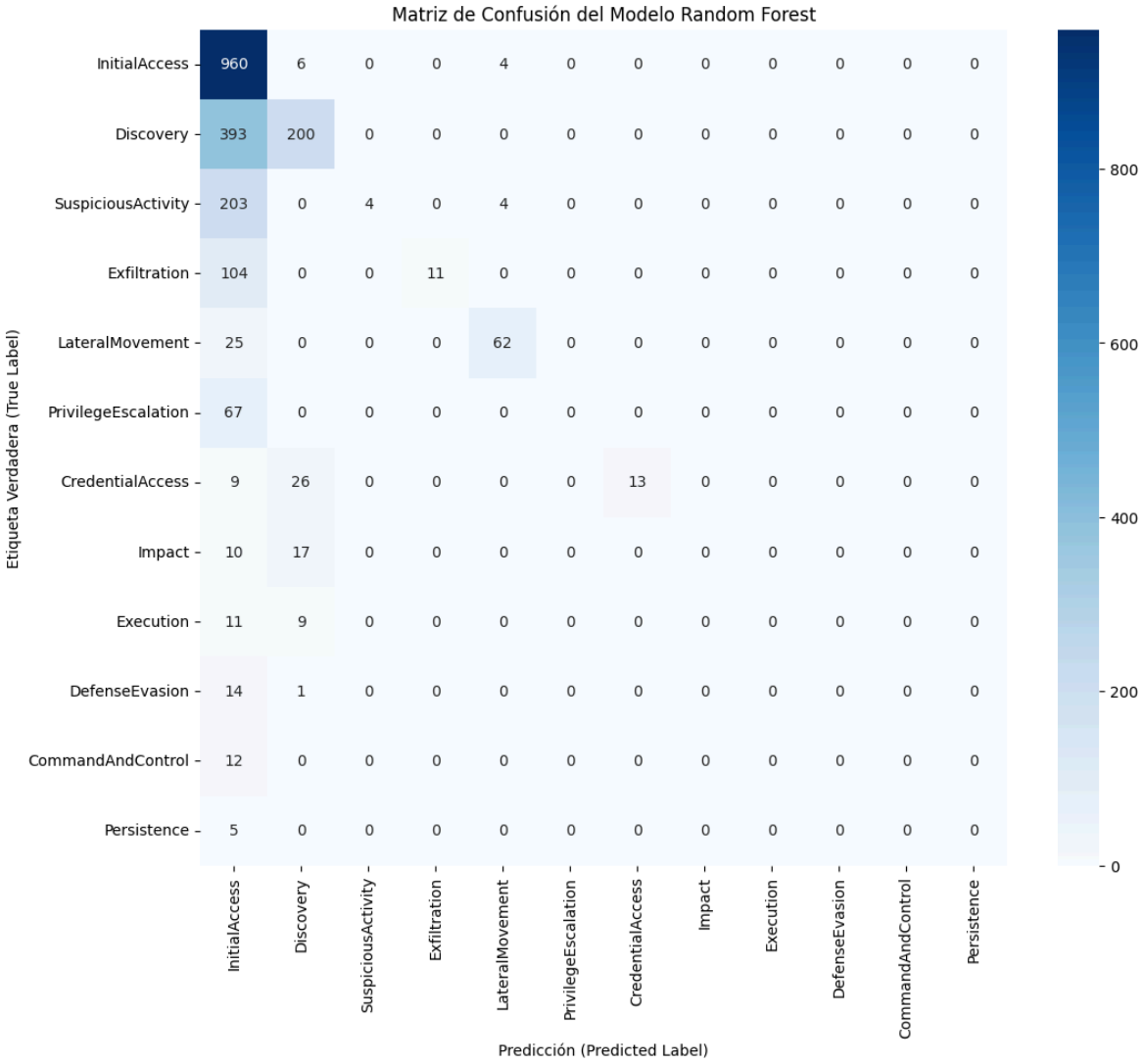
Una vez obtenido la combinación más adecuada de hiperparámetros a través de la validación cruzada, se llevó a cabo el reentrenamiento del modelo final En la imagen se sintetiza su desempeño empleando un grupo de métricas fundamentales: exactitud, precisión ponderada, recall ponderado y puntaje F1.

El modelo alcanzó una accuracy del 57.6%, lo cual nos dice que clasifica correctamente poco más de la mitad de las instancias. Por otro lado tenemos que la precisión ponderada fue de 0.6556, lo cual nos dice que cuando el modelo predice una clase, suele acertar en mayor proporción, especialmente en las clases más frecuentes. También tenemos el recall ponderado el cual coincide con el accuracy con un porcentaje de 0.5760, esto nos dice que el modelo detecta correctamente una proporción similar de instancias verdaderas para cada clase. Y por último la métrica del F1 score, el cual nos indica que aún existe espacio importante de mejora en el equilibrio entre falsos positivos y falsos negativos.



Matriz de Confusión

El modelo Random Forest, según la matriz de confusión, clasifica principalmente las clases InitialAccess (960 aciertos) y Discovery (200 aciertos), aunque esta última a menudo se confunde con InitialAccess (393 errores). Las otras clases muestran un rendimiento deficiente, con escasos aciertos y gran confusión, como la Actividad Suspiciosa, la Exfiltración y el Movimiento Lateral. Algunas clases, como Persistence o CommandAndControl, no se identificaron de manera alguna. Esto muestra un rendimiento desequilibrado entre las clases, probablemente debido al desequilibrio en los datos, lo que impacta la habilidad del modelo para identificar adecuadamente sucesos menos habituales.



Conclusiones

Este proyecto logró demostrar la implementación de métodos de aprendizaje automático supervisado y no supervisado en un escenario de Big Data relacionado con la ciberseguridad, posibilitando el análisis eficaz de grandes cantidades de información y la identificación de patrones valiosos para la categorización de los incidentes.

El método no supervisado consiguió dividir correctamente los datos en cinco grupos claramente establecidos, con un Coeficiente de Silhouette de 0.79, lo que señala una alta calidad en la agrupación. Estos grupos mostraron de manera adecuada la gravedad de los sucesos, incluso en un ambiente de alta dimensión y variabilidad, lo que resalta la importancia de estos modelos para el perfilado inicial de las amenazas.

Por otro lado el modelo supervisado Random Forest se estableció como una alternativa sólida ante grandes cantidades de datos, demostrando un desempeño satisfactorio a pesar del marcado desequilibrio de clases. Con una disposición ideal de 20 árboles y una profundidad máxima de 10, se logró una precisión del 57.6%, con un ponderado de precisión del 65.56%. Sin embargo, la matriz de confusión mostró que el modelo prioriza las clases con mayor representación, como InitialAccess y Discovery.

Como trabajo a futuro mejoraríamos la gestión del balanceo de clases mediante el uso de métodos como SMOTE distribuido o añadiendo una técnica de submuestreo adicional. Además, de buscar implementar más modelos sofisticados como XGBoost distribuido o redes neuronales. En última instancia, utilizamos instrumentos como SHAP o LIME para una mejor interpretación de los resultados y robustecer la toma de decisiones en el ámbito de la ciberseguridad.

Referencias

Lakens Daniël. (2022). Sample size justification. *Collabra : Psychology, 8*(1) doi:https://doi.org/10.1525/collabra.33267

Kim, J. K., & Wang, Z. (2019). Sampling Techniques for Big Data Analysis. *International Statistical Review / Revue Internationale de Statistique*, *87*, S177–S191.

Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B., & Vasilakos, A. V. (2016). Big data: From beginning to future. *International Journal of Information Management*, *36*(6), 1231–1247. https://doi.org/10.1016/j.ijinfomgt.2016.07.009

Vídeo

LINK DEL VIDEO