Universidad de La Sabana

Facultad de Ingeniería

Proyecto de Machine Learning

Alejandro Gil Gallego (274361), Juan David Sánchez González (272906)

alejandrogiga@unisabana.edu.co, juansangon@unisabana.edu.co

Maestría en Análitica Aplicada

Profesor: Hugo Franco, Ph.D.

Chía, Colombia
Octubre 2025

Resumen

Este proyecto busca identificar de manera correcta la detección de sitios web de phishing usando técnicas de aprendizaje supervisado como los modelos y además implementando flujos de trabajo con Prefect. Para este trabajo se hizo uso de un dataset relacionado con la detección de sitios web de phishing. Se entrenaron y compararon diferentes algoritmos de clasificación, evaluando tanto sus versiones básicas como usando normalización con StandarScaler. Por medio de las diferentes métricas de desempeño se observó y seleccionó el modelo con el mejor rendimiento para la deteccion de sitios web de phishing. El proyecto elaborado demuestra la aplicación práctica del workflow de ciencia de datos basado en aprendizaje supervisado en un problema real de ciberseguridad.

Abstract

This project seeks to correctly identify phishing website detection using supervised learning techniques such as models and also implementing workflows with Prefect. For this work, a dataset related to phishing website detection was used. Different classification algorithms were trained and compared, evaluating both their basic versions and using normalization with StandardScaler. Through different performance metrics, the model with the best performance for phishing website detection was observed and selected. The developed project demonstrates the practical application of the data science workflow based on supervised learning in a real cybersecurity problem.

Index Terms

Phishing, Clasificación, Gradient Boosting, Random Forest, kNN, SVM.

I. Introducción

El curso de Machine Learning I parte de la maestría en Análitica Aplicada dictado por Hugo Franco, Ph.D. en la Universidad de la Sabana tiene como parte de sus objetivos el entendimiento y puesta en práctica de los aprendizajes adquiridos durante las 5 sesiones de clase. Como proyecto de finalización del curso se realizó un modelo de aprendizaje de máquina supervisado de clasificación. Este tiene como objetivo distinguir si un sitio web es posiblemente un sitio web de phishing o si no lo es. Este documento narra todo lo hecho por el grupo de trabajo para llegar al resultado final entregado. Entre los modelos utilizados se encuentra Gradient Boosting, Random Forest, kNN, y SVM.

I-A. Marco conceptual

El presente estudio realizado se fundamenta en los siguientes conceptos claves del mundo de la ciberseguridad informática relacionados al phishing.

I-A1. Phishing: El phishing es un tipo de ciberataque que busca aprovecharse de personas o de entidades. Este tipo de ataques se centra en hacer ingeniería social; no busca explotar el sistema de seguridad de un sistema como otros tipos de ciberataques. Por el contrario, lo que busca es inducir al error humano, un error que les permita crear una brecha de seguridad donde estas personas puedan acceder a datos sensibles personales o empresariales [2].

I-A2. Avances de los ciberatacantes: El phishing ha existido desde que la World Wide Web fue lanzada al público, y al igual que esta ha evolucionado con los años para adaptarse a las nuevas tecnologías y ser cada vez más difícil de detectar por los usuarios menos experimentados, los que comúnmente se convierten en las victimas de estos ciberataques, de esta forma técnicas de phishing se han adecuado con el paso de los años como lo es la ofuscación de URLs, que dificultar y reconocer el destino de la dirección URL, manipulando uno o varios componentes de una URL, utilizando simbolos poco comunes [3].

I-B. Antecedentes

La detección de sitios web de phishing mediante técnicas de aprendizaje automático se ha investigado en varios aspectos a lo largo de los años, con diversos estudios proponiendo y evaluando diferentes enfoques metodológicos. A continuación, se presentan los trabajos más relevantes relacionados:

I-B1. Optimización de redes neuronales mediante inteligencia de enjambre: En este estudio propusieron el ajuste automático de hiperparámetros para redes neuronales profundas utilizando algoritmos de inteligencia de enjambre aplicados a la clasificación de sitios web de phishing. Los resultados demostraron mejoras significativas, alcanzando incrementos de 2.5-3.8 % en accuracy y hasta 24 % en F1-score comparado con configuraciones tradicionales en cuatro datasets diferentes de phishing [5].

I-B2. Comparación de random forest, SVM y redes neuronales: En este estudio, hicieron una comparación entre Random Forest, Support Vector Machines y redes neuronales con backpropagation para la detección de phishing. Los tres algoritmos alcanzaron desempeños superiores al 97 %, demostrando la efectividad de múltiples modelos supervisados para este problema de clasificación binaria [6].

I-B3. Deep Learning con optimizador Adam para detección de phishing: Aquí, aplicaron técnicas de Deep Learning con el optimizador Adam para clasificar sitios web de phishing, donde lograron mejor precisión superando métodos como SVM y Adaboost. Lo que demuestró que las redes neuronales pueden identificar patrones complejos en las URLs mejor que los algoritmos convencionales [7].

I-B4. Deep learning semi-supervisado para detección: Se exploraron técnicas de deep learning semi-supervisado combinando features, de contenido HTML y servicios externos para detección de phishing. Los modelos propuestos demostraron mejoras cuando se aprovecha información no etiquetada adicional, permitiendo generalización superior en escenarios con datos limitados de entrenamiento [8].

Estos estudios demuestran la efectividad de algoritmos de aprendizaje supervisado. Sin embargo, son pocas las investigaciones basadas en la comparación de múltiples modelos o algoritmos con y sin preprocesamiento, así como en el análisis de complejidad computacional.

II. Datos empleados

Para el desarrollo de este proyecto se empleó el dataset público "Datasets for Phishing Websites Detection", el cual proporcionó información importante sobre características extraídas de URLs para la detección de sitios web de phishing. Esta base de datos fue tomada debido a que está respaldada por informes técnicos y validados, lo que asegura que los datos estén organizados de forma correcta y sean confiables para usarlos en modelos de aprendizaje supervisado para este proyecto.

II-A. Base de datos de Phishing Websites

Fuente: G. Vrbančič, I. Jr. Fister, V. Podgorelec. Datasets for Phishing Websites Detection. Data in Brief, Vol. 33, 2020 [1]
Indicador: Phishing (1 si es, 0 si no es)
Versión: 1 (2020)
Licencia: C.C. by 4.0

Esta base de datos constituye el conjunto completo de variables tanto independientes como dependiente del estudio, representando un problema de clasificación binaria para identificar si un sitio web es legítimo o fraudulento (phishing) [2].

Características técnicas - small variant:

- **Total de instancias:** 58,645 registros
- **Distribución de clases:**
  - **Sitios web legítimos (etiquetados como 0):** 27,998
  - **Sitios web de phishing (etiquetados como 1):** 30,647
- **Número total de características:** 111 variables
- **Variable objetivo:** Variable binaria 'phishing' que indica si el sitio web es legítimo o fraudulento.
- **Tipo de problema:** Clasificación binaria supervisada

Características técnicas - full variant:

- **Total de instancias:** 88,647 registros
- **Distribución de clases:**
  - **Sitios web legítimos (etiquetados como 0):** 58,000
  - **Sitios web de phishing (etiquetados como 1):** 30,647
- **Número total de características:** 111 variables
- **Variable objetivo:** Variable binaria 'phishing' que indica si el sitio web es legítimo o fraudulento.
- **Tipo de problema:** Clasificación binaria supervisada

II-B. Interés

La seguridad informática no solo es importante para los ingenieros encargados de la construcción del software y de la infraestructura, es algo que ha vuelto vital y parte de la cultura del internet, si quieres evitar problemas has de saber cuidarte, por esta razón ha sucedido históricamente que la existencia de contenido malicioso ha sido perseguido, y este se halla tenido que adaptar, generando y ciclo de mejora tanto del atacante como del defensor, en los últimos años el público general no ha llegado a estar a la altura de este nivel de cambio como lo menciona Netskope ya que en el año 2024 aumento en un 190 % comparado con 2023, pasando de 3 usuarios de 1000 que entraron a una página de phishing a más de 8 usuarios de 1000 [4]. Por esto es importante entender como estas páginas pueden ser detectadas sin necesidad de entrar en ellas. Por ende este proyecto tiene como objetivo la creación de un modelo de aprendizaje de maquina que clasifica entre si una página es phishing o no. En un futuro este modelo puede ser utilizado y adaptado a navegadores web para evitar que los usuarios entren a estos sitios web.

II-C. Complejidad

La complejidad del dataset manejado se distribuye de la siguiente forma:

II-C1. Small Dataset::

- **Dimensionalidad:** 58,645 instancias x 111 variables
- **Imbalance de clases:**
  - **Sitios Web legítimos:** 47.74 %
  - **Sitios Web de phishing:** 52.26 %

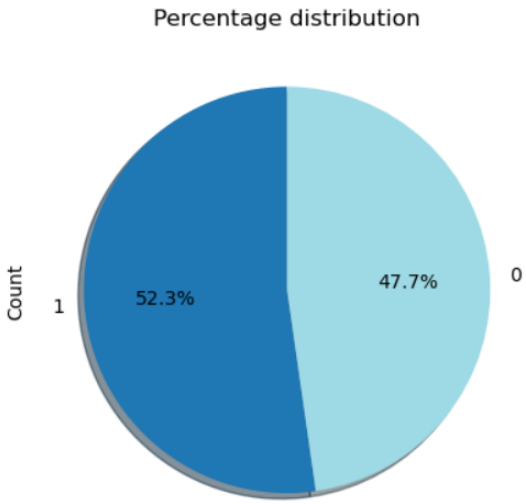

Figura 1. Distribución porcentual de clases para el dataset small

- **Correlación de características:** Las correlaciones están mejor representadas gráficamente en los resultados.

II-C2. Full Dataset::

- **Dimensionalidad:** 88,647 instancias x 111 variables
- **Imbalance de clases:**
  - **Sitios Web legítimos:** 65.43 %
  - **Sitios Web de phishing:** 34.57 %

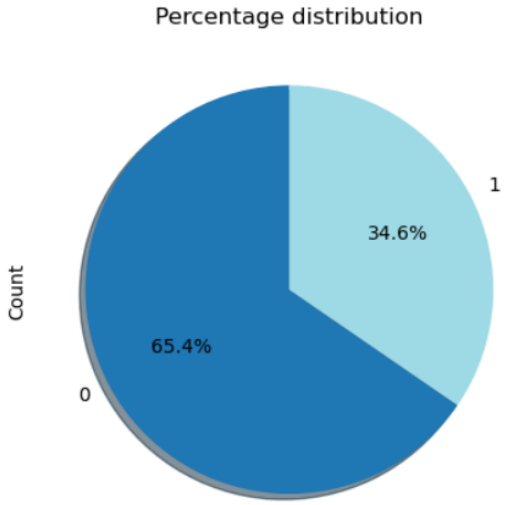

Figura 2. Distribución porcentual de clases para el dataset full

- **Correlación de características:** Las correlaciones están mejor representadas gráficamente en los resultados.

II-D. Utilidad práctica

Como se mencionó en el interés del proyecto el modelo desarrollado tiene como objetivo disminuir la cantidad de ataques de phishing, clasificar los dominios web y de esta forma proteger a los usuarios de la World Wide Web. Se considera que el código puede ser adaptado para alguna de las siguientes ideas:

- Servicio PaaS de predicción del riesgo de una página web a tiempo real, alto coste computacional.
- Creación de listas de investigación de páginas web, dependiendo de una segunda instancia el dominio se mantiene o se desecha.
- Creación de un sistema de recomendación al usuario sobre el riesgo de una página web.

III. METODOLOGÍA

La metodología empleada en este proyecto siguió un enfoque basado en el diseñó e implementación basado en aprendizaje supervisado que implementó todos los procesos de forma orquestada mediante Prefect para garantizar un buen funcionamiento de todos los procesos. El proceso de implementación se llevo a cabo para las dos variantes del dataset, en primer lugar para el dataset small y luego el mismo flujo de trabajo se implementó para el dataset full pero con algunos cambios. Para abordar el proyecto, se implementó una metodología estructurada en cuatro etapas principales:

- **Fase 1 - Análisis exploratorio de los datos:** Se realizó la revisión de cómo están distribuidas las clases, búsqueda de datos faltantes o repetidos, y análisis estadístico descriptivo.
- **Fase 2 - Preprocesamiento:** En esta fase se ejecutó la división estratificada 80-20 en conjuntos de entrenamiento y prueba.
- **Fase 3 - Entrenamiento de Modelos:** Aquí se hizo la implementación de los cinco algoritmos supervisados, adicionalmente se hizo el escalado de características con StandarScaler para cada modelo.
- **Fase 4 - Análisis y comparación:** Análisis de las diferentes métricas de desempeño para comparar los diferentes modelos.

Esta metodología permitió que el análisis pudiera llevarse a cabo de la mejor manera. Estos aspectos fueron clave para asegurar que los resultados obtenidos fueran confiables de acuerdo al contexto del proyecto.

III-A. Fase 1 - Análisis exploratorio de los datos:

En primer lugar, se inició con la carga del conjunto de datos, que como ya se mencionó anteriormente el dataset incluia 111 características extraidas de URLs y una variable objetivo binaria que indica si el sitio web es legítimo o phishing.

Se realizó un análisis de la distribución de clases para identificar posibles desbalances en los datos. Para el caso del dataset small (Figura 3), los resultados mostraron que este se encontraba moderadamente balanceado, con 27.998 instancias de sitios legítimos que equivalen al 47.7 % y 30.647 instancias de sitios de phishing que equivale al 52.3 %. Esta distribución permitio trabajar sin aplicar SMOTE como técnica de balanceo.

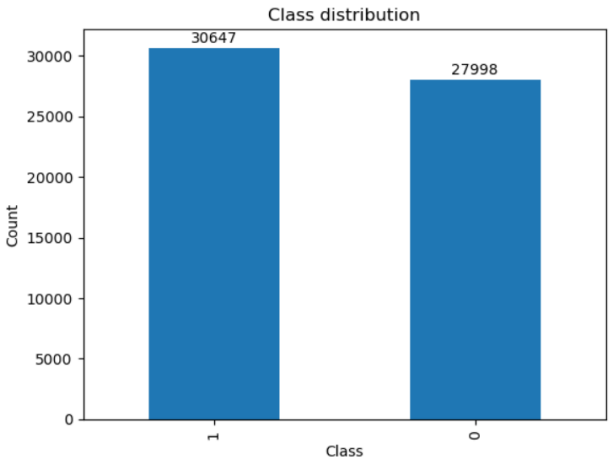

Figura 3. Análisis distribución de clases para el dataset small

Por otro lado, para la variante Full del dataset (Figura 4) fue necesario aplicar SMOTE (Figura 5), ya que presentaba un desequilibrio con 58,000 instancias de sitios legítimos que representan el 65.4 % y 30,647 instancias de sitios de phishing que representan el 34.6 %.

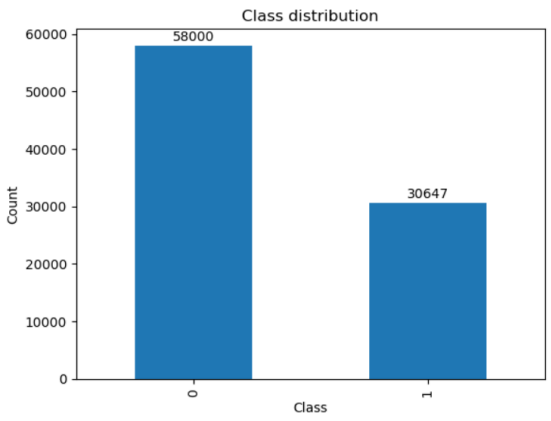

Figura 4. Análisis distribución de clases para el dataset full

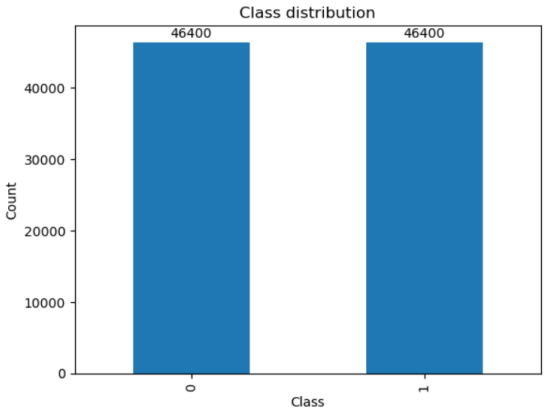

Figura 5. Análisis distribución de clases para el dataset full tras usar SMOTE

Posteriormente, se verificó la calidad de los datos mediante la búsqueda de valores faltantes y otras características que pudieran requerir limpieza. Sin embargo, el dataset no presentó valores nulos ni inconsistencias relevantes, por lo que no fue necesario realizar un proceso de limpieza de datos. Adicionalmente, se generó un análisis descriptivo de todas las características, calculando medidas como la media, desviación estandar, valores mínimos y máximos y otros datos que permitieron comprender la distribución de cada variable.

Finalmente, se realizó un análisis de correlación entre las características y la variable objetivo en el cual se identificaron las 15 características con mayor correlación con la variable phishing. Este análisis ayudó a entender cuáles son los elementos más importantes para identificar sitios web de phishing.

III-B. Fase 2 - Preprocesamiento:

En esta fase se prepararon los datos para entrenar los modelos. Primero se dividió el dataset en dos partes, la matriz de características que incluye las 111 variables independientes y la variable objetivo que contiene la variable binaria que indica si un sitio es phishing o no.

Se aplicó una división estratificada de los datos con una proporción 80-20, en donde el 80 % de las instancias para entrenamiento y el 20 % restante para prueba. Esto aseguró que ambos conjuntos de datos conservaran la misma proporción entre sitios legítimos y sitios de phishing, manteniendo un equilibrio en la distribución de las clases. El siguiente algoritmo muestra como se ejecutó este preprocesamiento:

```
Algoritmo 1. Preprocesamiento de datos

Inicio
  Función Preprocesamiento(datos):
    // Separar características y variable objetivo
    Extraer todas las columnas excepto phishing como características
    Extraer columna phishing como variable objetivo

    // Dividir datos en entrenamiento y prueba
    Aplicar división estratificada 80-20
    Mantener proporción de clases en ambos conjuntos
    Establecer semilla aleatoria en 42

    Retornar conjuntos de entrenamiento y prueba
  Fin Función
Fin
```

La implementación detallada del algoritmo 1 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

III-C. Fase 3 - Entrenamiento de Modelos:

Como se mencionó al iniciar el informe, se implementaron cuatro modelos de clasificación supervisada, los cuales fueron el Gradient Boosting, Random Forest, Knn, SVM. A continuación, se detalla cada uno de ellos.

III-C1. Gradient Boosting: Para este modelo se utilizó el clasificador GradientBoostingClassifier, el cual construye múltiples árboles de decisión de forma secuencial, donde cada nuevo árbol corrige los errores del anterior. Se configuró con 100 estimadores, una tasa de aprendizaje de 0.1 y una profundidad máxima de 3 niveles. El entrenamiento se realizó mediante el método fit() y las predicciones con predict(). Se aplicó validación cruzada con 5 particiones para evaluar la estabilidad del modelo durante el entrenamiento.

```
Algoritmo 2. Clasificación con Gradient Boosting

Función gradient_boosting(X_train, X_test, y_train, y_test): modelo
   Inicializar clasificador Gradient Boosting con parámetros definidos
   Entrenar modelo con X_train y y_train
   Realizar predicciones sobre X_test
   Evaluar desempeño con validación cruzada y exactitud
   retornar modelo
```

La implementación detallada del algoritmo 2 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

III-C2. Random Forest: El modelo de Random Forest se realizó usando el clasificador RandomForestClassifier, configurado en este caso para construir 100 árboles de decisión. Cada árbol se entrena con una parte aleatoria de los datos y con un grupo distinto de características, lo que ayuda a evitar el sobreajuste y a que el modelo genere mejores resultados en nuevos datos [9]. El modelo también se entrenó con la función fit() y su rendimiento se evaluó con una validación cruzada de 5 partes, además también se midió su precisión con el conjunto de prueba.

```
Algoritmo 2. Clasificación con Random Forest

Función random_forest(X_train, X_test, y_train, y_test): modelo

   Inicializar clasificador Random Forest
   Entrenar modelo con X_train y y_train
   Realizar predicciones sobre x_test
   Evaluar desempeño con métrica de exactitud

   retornar modelo
```

La implementación detallada del algoritmo 2 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

III-C3. K-Nearest Neighbors: Para este modelo se utilizó el clasificador KNeighborsClassifier con k igual a 5 vecinos más cercanos. lo que significa que clasifica cada punto según las cinco instancias más cercanas en el conjunto de entrenamiento. Se realizaron pruebas con diferentes valores de k para identificar la configuración óptima.

```
Algoritmo 3. Clasificación con Knn

Función knn(X_train, X_test, y_train, y_test): modelo

   Inicializar clasificador KNN con k=5
   Entrenar modelo con X_train y y_train
   Realizar predicciones sobre X_test
   Comparar diferentes valores de k (1, 3, 5, 7)
   Evaluar desempeño con validación cruzada y exactitud
   retornar modelo
```

La implementación detallada del algoritmo 3 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

Adicionalmente, se implementó una alternativa con escalado de características mediante StandardScaler. La validación cruzada de 5 particiones permitió evaluar su desempeño durante el entrenamiento.

```
Algoritmo 4. Clasificación con Knn escalado

Función knn(X_train, X_test, y_train, y_test): modelo, X_test_escalado

   Inicializar StandarScaler
```

```
   Escalar datos de entrenamiento y prueba
   Inicializar clasificador KNN con k=5
   Entrenar modelo con datos escalados
   Realizar predicciones sobre X_test_escalado
   Comparar diferentes valores de k (1, 3, 5, 7)
   Evaluar desempeño con validación cruzada y exactitud
   retornar modelo, X_test_escalado
```

La implementación detallada del algoritmo 4 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

III-C4. Support Vector Machine: Se implementaron cuatro distintas configuraciones del clasificador SVM. La primera utilizó SVC con kernel RBF, que permite capturar relaciones no lineales entre las diferentes variables. La segunda configuración que se aplicó fue el escalado de características con StandardScaler antes del entrenamiento que permitió mejorar el rendimiento del modelo.

```
Algoritmo 5. Clasificación básica de SVM

Función svm(X_train, X_test, y_train, y_test): modelo

   Inicializar clasificador SVM con kernel RBF
   Entrenar modelo con X_train y y_train
   Realizar predicciones sobre X_test
   Evaluar desempeño con validación cruzada y exactitud
   retornar modelo
```

La implementación detallada del algoritmo 5 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

Por otro lado, la tercera utilizó LinearSVC una versión más rápida de SVM que es muy útil en datasets grandes y se pensó usar para poderla comparar debido al uso del dataset full que tenía muchos más registros. Y la última fue también con los datos escalados para LinearSVC.

```
Algoritmo 6. Clasificación con SVM LinearSVC

Función svm_linearsvc(X_train, X_test, y_train, y_test): modelo

   Inicializar clasificador LinearSVC con parámetros definidos
   Entrenar modelo con X_train y y_train
   Realizar predicciones sobre X_test
   Evaluar desempeño con validación cruzada y exactitud
   retornar modelo
```

La implementación detallada del algoritmo 6 se encuentra en el archivo adjunto Comparison.ipynb, disponible en el repositorio de GitHub del proyecto [10].

Todas las versiones anteriores mencionadas se entrenaron con fit() y se evaluaron mediante validación cruzada de 3 particiones ya que este algoritmo requiere más tiempo y recursos para entrenarse.

III-D. Fase 4 - Análisis y comparación:

En esta fase se evaluó el desempeño de todos los modelos entrenados mediante múltiples métricas de clasificación. Para cada modelo se calculó el accuracy en el conjunto de prueba y se utilizó validación cruzada para obtener una mejor estimación del desempeño a través de múltiples divisiones de los datos.

También se implementaron matrices de confusión para observar el desempeño de cada modelo, mostrando así cuantas predicciones fueron correctas y cuantas fueron erróneas. Además se usaron reportes de clasificación que permitieron analizar más métricas detalladas por clase, como precision, recall y f1-score. Adicionalmente, se calcularon las curvas ROC y el área bajo la curva AUC, para medir la capacidad del modelo para diferenciar entre las clases positiva y negativa.

Finalmente, se crearon curvas de aprendizaje para algunos modelos, mostrando cómo cambia el rendimiento al aumentar la cantidad de datos de entrenamiento y ayudando a detectar si el modelo presenta sobreajuste o subajuste.

IV. Resultados

IV-A. Análitica descriptiva

Como parte del entendimiento de los resultados fue de vital importancia el entendimiento del dataset, o en el caso del proyecto de los dos dataset, el full y el small. El primer proceso realizado fue la generación de una estadística descriptiva de la repartición de las clases de los dataset, como se puede llegar a ver en las figuras 1 y 2 para la distribución porcentual de los datasets, y el análisis de distribución en las figuras 3 y 4. Debido a que el dataset full se encontraba imbalanceado se realizó el proceso de SMOTE dando como resultado una distribución como la vista en la figura 5. Cabe aclarar que en los resultados siguientes que se refiere al dataset full, se hace referencia al dataset full con SMOTE.

Posterior a esto se buscaron correlaciones entre las variables de los datasets. Los resultados son los de las figuras 6 y 7.

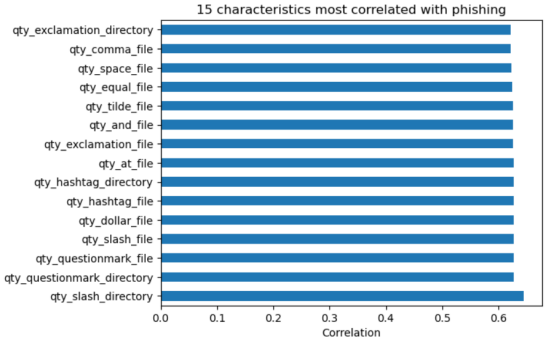

Figura 6. Top 15 variables más correlacionadas del dataset small.

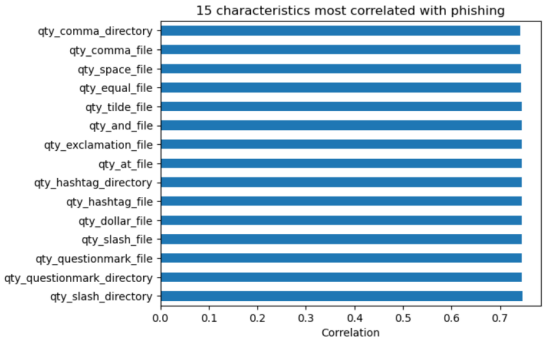

Figura 7. Top 15 variables más correlacionadas del dataset full

A partir de estas figuras 6 y 7 se puede evidenciar que para el dataset small (Figura 6) la variable con mayor correlación es qty_slash_directory con un 0.64. Esta variable hace referencia a la cantidad de / que tiene el directorio de la página web. Por parte del dataset full (Figura 7) entre las 15 variables con mayor correlación todas se encuentran en un 0.74, teniendo mejor correlación las variables del dataset full que las del dataset small.

IV-B. Comparación de modelos

Como parte de la aplicación de lo aprendido durante el curso se utilizaron varios modelos, comparando sus resultados y su forma de trabajar con el dataset y la calidad de sus resultados.

IV-B1. Gradient Boosting: El modelo arrojó una precisión del 0.93 para el dataset small y del 0.95 para el dataset full. Las figuras 8 y 9 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

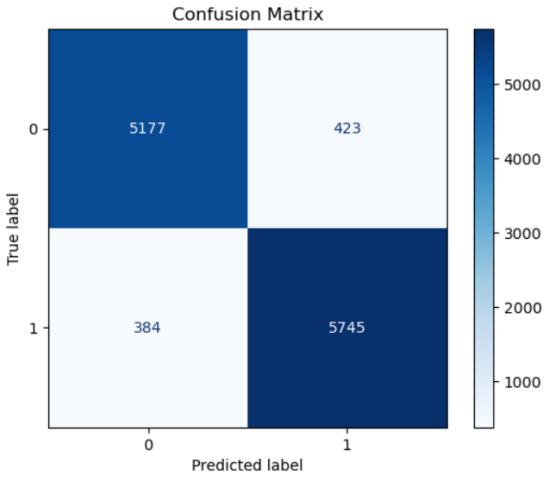

Figura 8. Matriz de confusión de Gradient Boosting del dataset small.

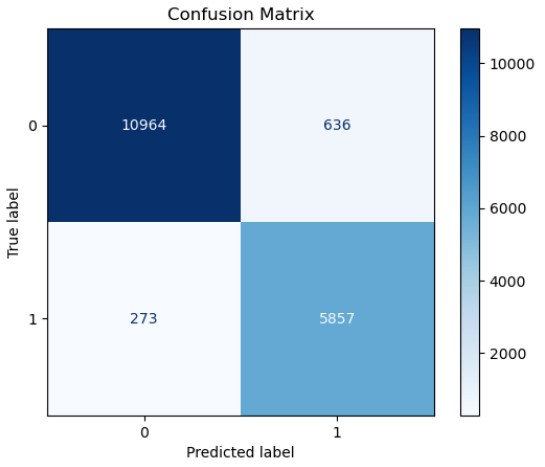

Figura 9. Matriz de confusión de Gradient Boosting del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.98 y para el dataset full fue de 0.99. Como lo muestran las figuras 10 y 11 respectivamente.

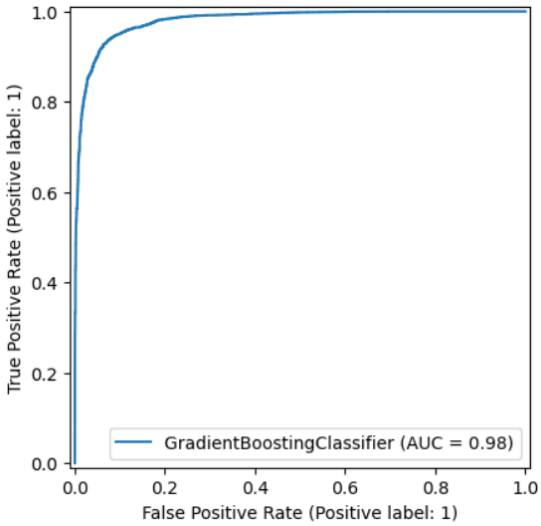

Figura 10. AUC-ROC de Gradient Boosting del dataset small.

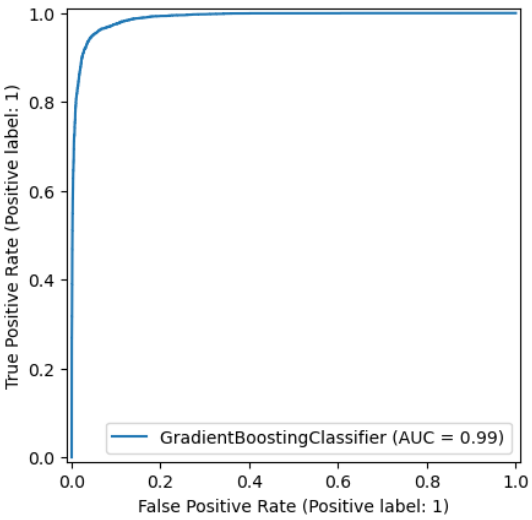

Figura 11. AUC-ROC de Gradient Boosting del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 12 y 13 respectivamente.

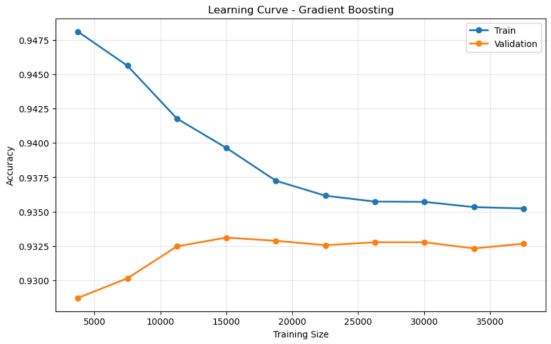

Figura 12. Linea de aprendizaje de Gradient Boosting del dataset small.

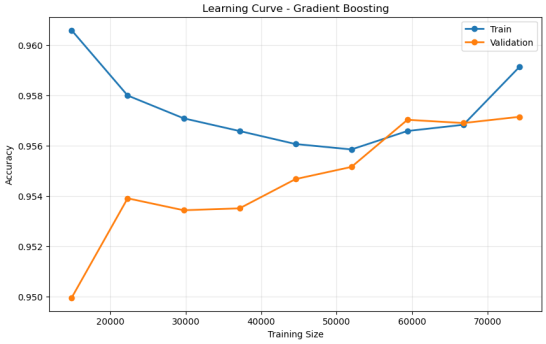

Figura 13. Linea de aprendizaje de Gradient Boosting del dataset full

En el caso del dataset small a partir de la Figura 12 se puede identificar que el modelo sufre de underfitting ya que las dos curvas estan cerca pero las dos bajas. Por parte del dataset full el comportamiento es bueno como lo muestra la figura 13.

IV-B2. Random Forest: El modelo arrojó una precisión del 0.93 para el dataset small y del 0.96 para el dataset full. Las figuras 14 y 15 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

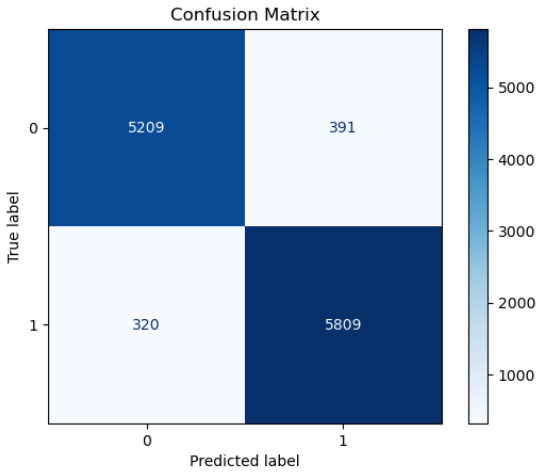

Figura 14. Matriz de confusión de Random Forest del dataset small.

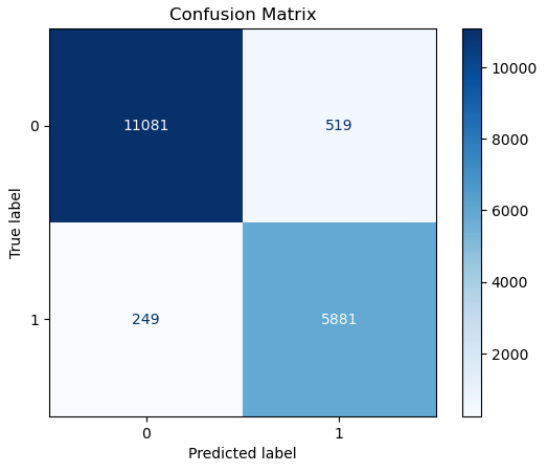

Figura 15. Matriz de confusión de Random Forest del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.99 y para el dataset full fue de 0.99. Como lo muestran las figuras 16 y 17 respectivamente.

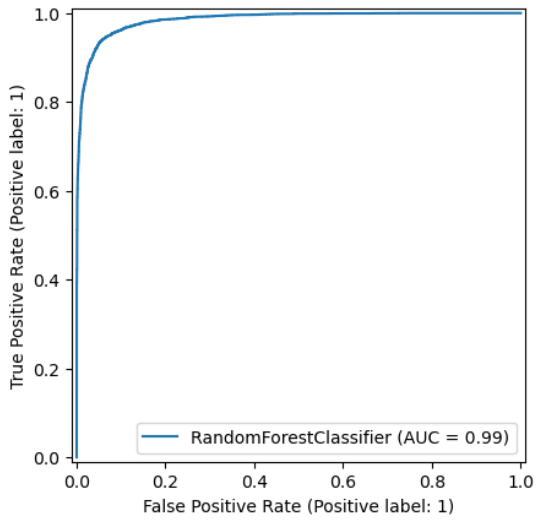

Figura 16. AUC-ROC de Random Forest del dataset small.

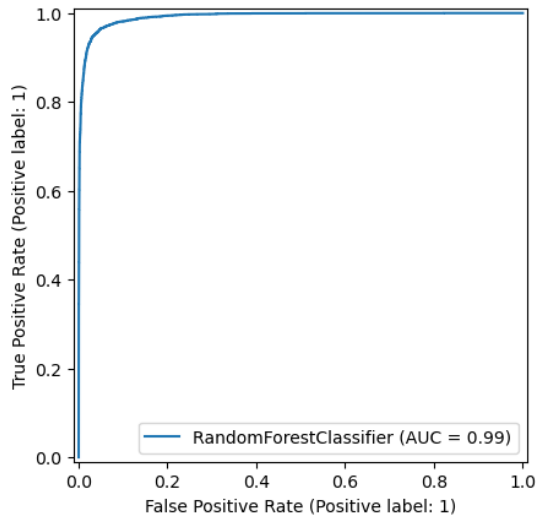

Figura 17. AUC-ROC de Random Forest del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 18 y 19 respectivamente.

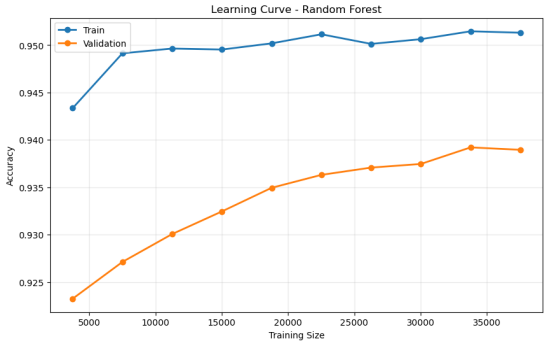

Figura 18. Linea de aprendizaje de Random Forest del dataset small.

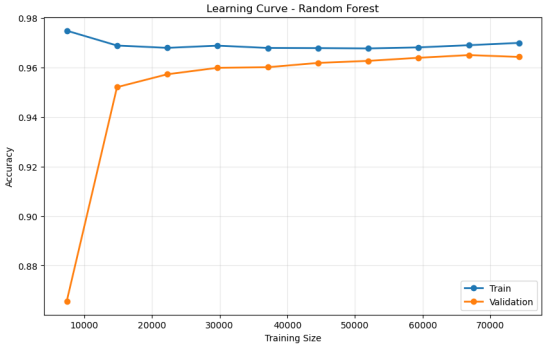

Figura 19. Linea de aprendizaje de Random Forest del dataset full

En el caso del dataset small a partir de la Figura 18 se puede identificar que el modelo sufre de overfitting ya que las dos curvas estan muy separadas, la validación no se acerca al test. Por parte del dataset full el comportamiento es ideal como lo muestra la figura 19.

IV-B3. K-Nearest Neighbors: El modelo arrojó una precisión del 0.85 para el dataset small y del 0.87 para el dataset full. Las figuras 20 y 21 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

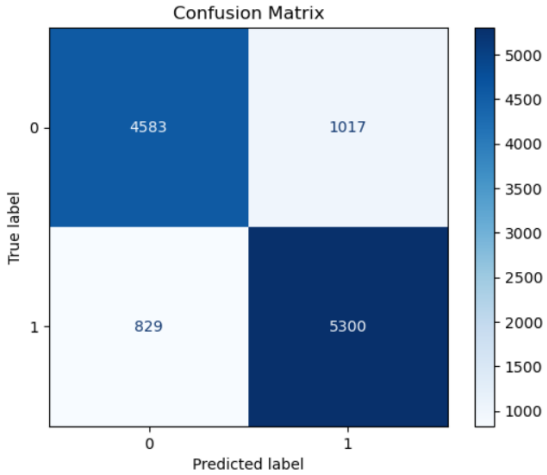

Figura 20. Matriz de confusión de K-Nearest Neighbors del dataset small.

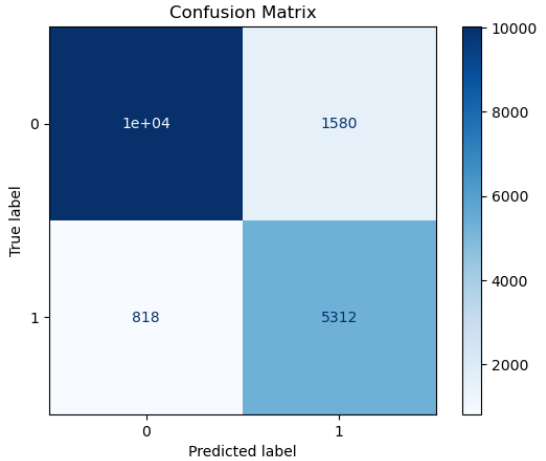

Figura 21. Matriz de confusión de K-Nearest Neighbors del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.91 y para el dataset full fue de 0.93. Como lo muestran las figuras 22 y 23 respectivamente.

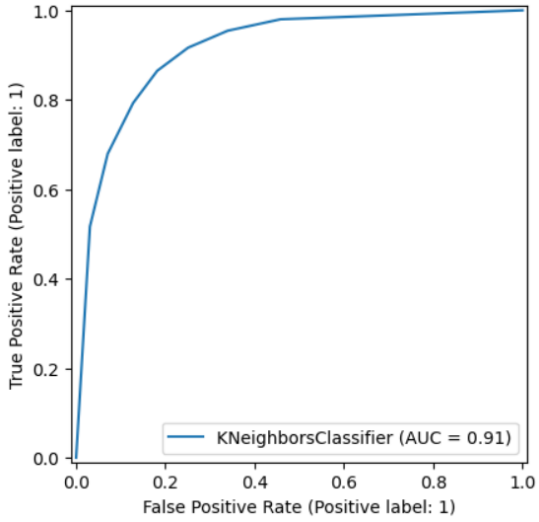

Figura 22. AUC-ROC de K-Nearest Neighbors del dataset small.

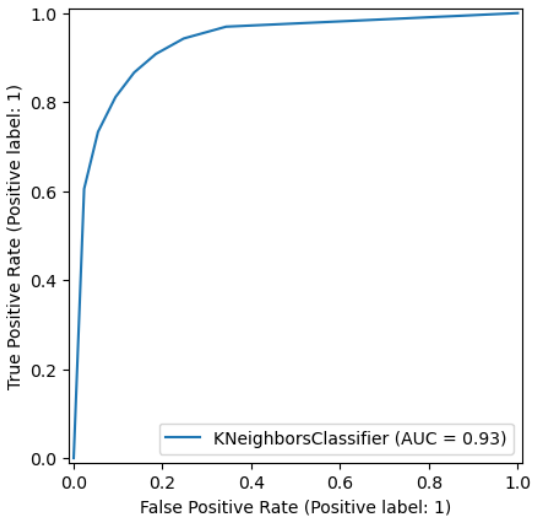

Figura 23. AUC-ROC de K-Nearest Neighbors del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 24 y 25 respectivamente.

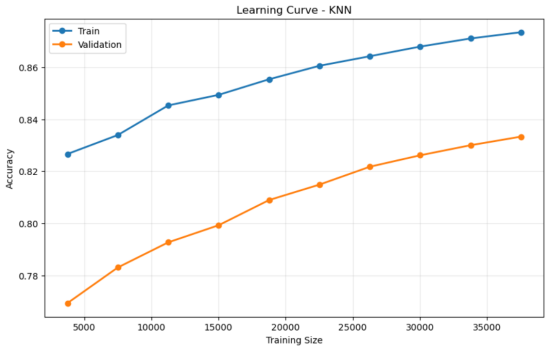

Figura 24. Linea de aprendizaje de K-Nearest Neighbors del dataset small.

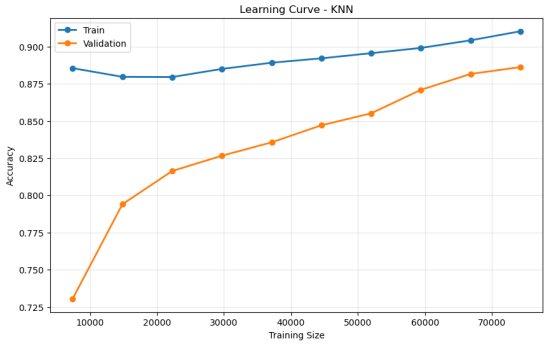

Figura 25. Linea de aprendizaje de K-Nearest Neighbors del dataset full

En el caso del dataset small a partir de la Figura 24 se puede identificar que el modelo sufre de overfitting ya que las dos curvas estas a pesar de incrementar, la validación no se acerca al test, por lo cual no converge. Por parte del dataset full el comportamiento es ideal como lo muestra la figura 25.

IV-B4. K-Nearest Neighbors Scaled: El modelo arrojó una precisión del 0.93 para el dataset small y del 0.95 para el dataset full. Las figuras 26 y 27 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

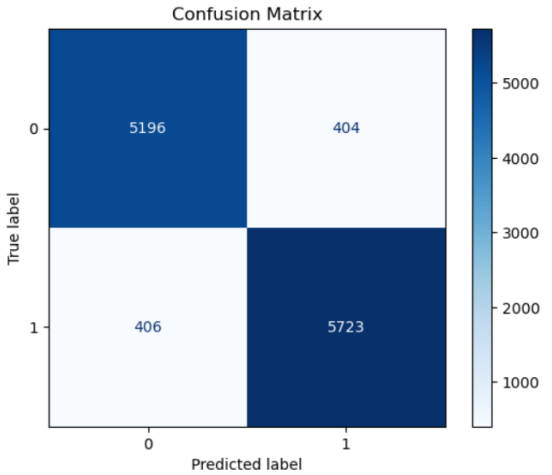

Figura 26. Matriz de confusión de K-Nearest Neighbors Scaled del dataset small.

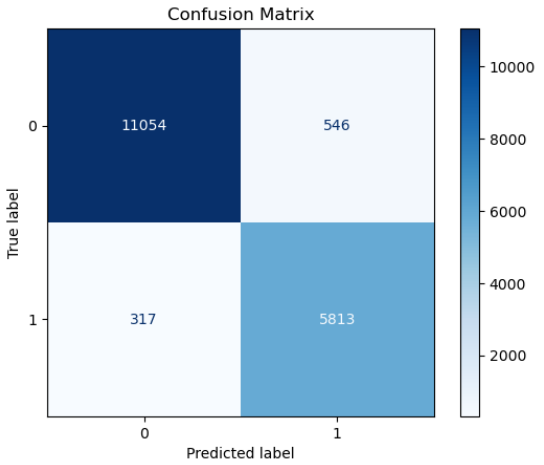

Figura 27. Matriz de confusión de K-Nearest Neighbors Scaled del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.97 y para el dataset full fue de 0.98. Como lo muestran las figuras 28 y 29 respectivamente.

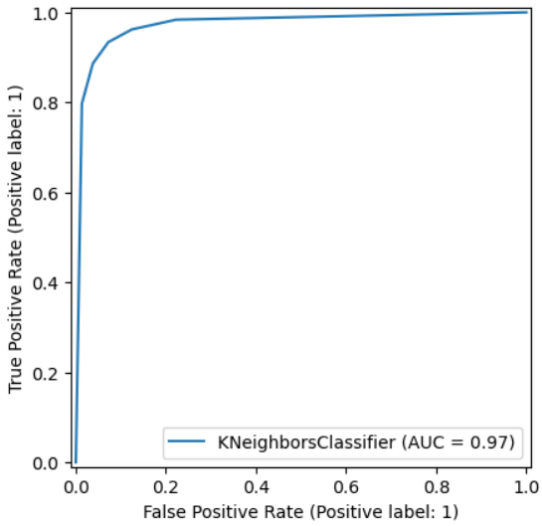

Figura 28. AUC-ROC de K-Nearest Neighbors Scaled del dataset small.

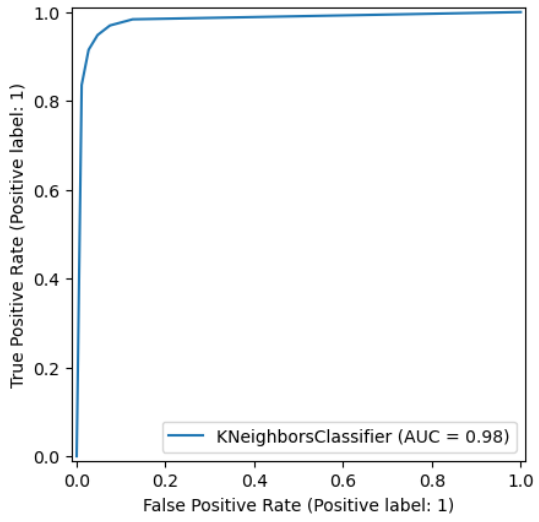

Figura 29. AUC-ROC de K-Nearest Neighbors Scaled del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 30 y 31 respectivamente.

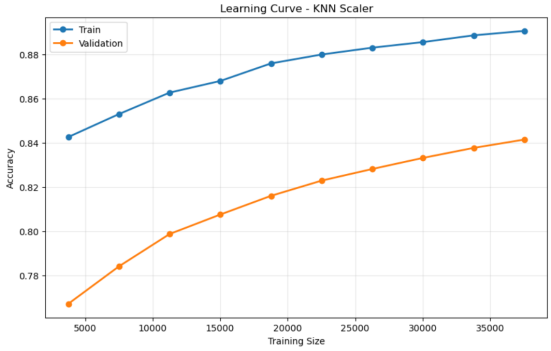

Figura 30. Linea de aprendizaje de K-Nearest Neighbors Scaled del dataset small.

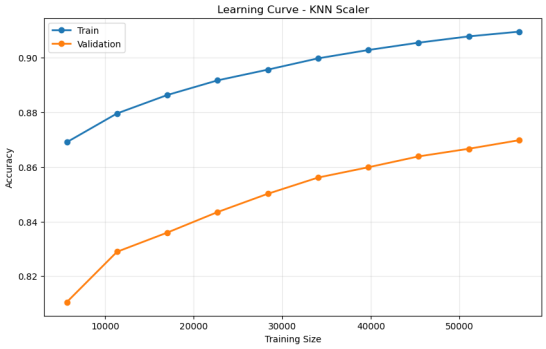

Figura 31. Linea de aprendizaje de K-Nearest Neighbors Scaled del dataset full

En el caso del dataset small a partir de la Figura 30 se puede identificar que el modelo sufre de overfitting ya que las dos curvas estas a pesar de incrementar, la validación no se acerca al test, por lo cual no converge. Este mismo comportamiento le sucede al dataset full como lo muestra la figura 31.

IV-B5. SVM: El modelo arrojó una precisión del 0.72 para el dataset small y del 0.74 para el dataset full. Las figuras 32 y 33 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

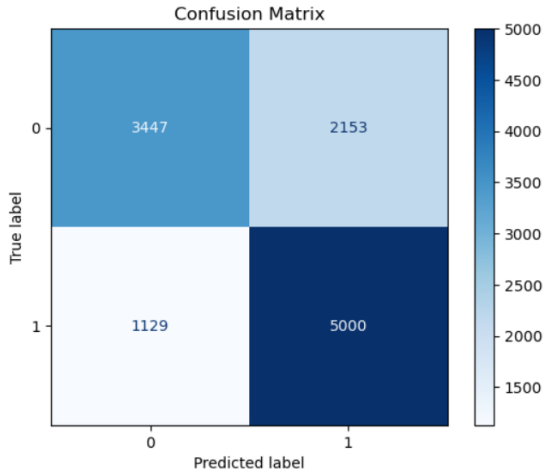

Figura 32. Matriz de confusión de SVM del dataset small.

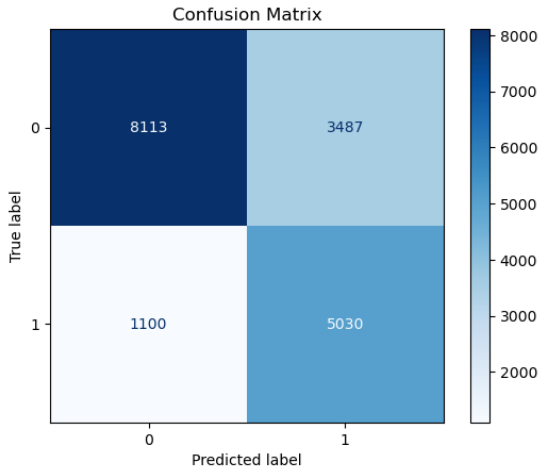

Figura 33. Matriz de confusión de SVM del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.77 y para el dataset full fue de 0.81. Como lo muestran las figuras 34 y 35 respectivamente.

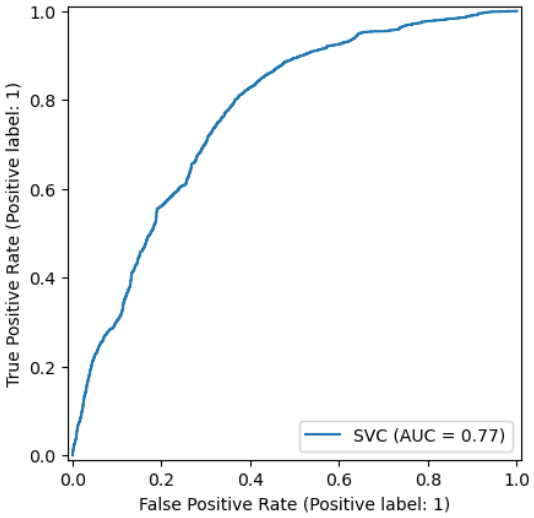

Figura 34. AUC-ROC de SVM del dataset small.

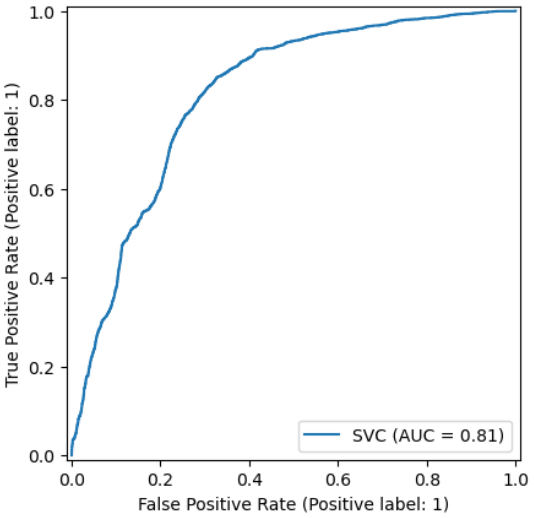

Figura 35. AUC-ROC de SVM del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 36 y 37 respectivamente.

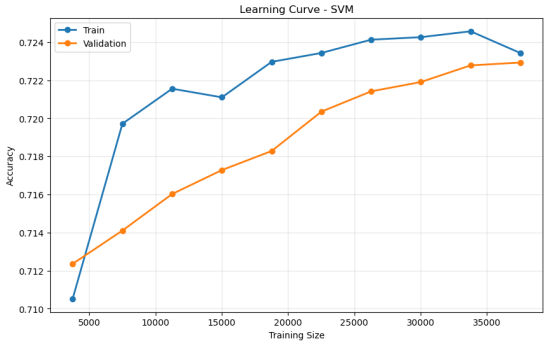

Figura 36. Linea de aprendizaje de SVM del dataset small.

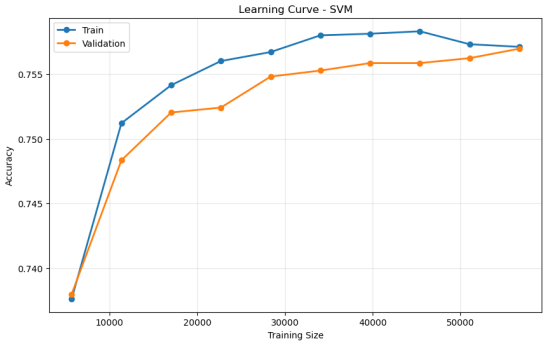

Figura 37. Linea de aprendizaje de SVM del dataset full

En el caso del dataset small a partir de la Figura 36 se puede identificar que el modelo converge y teniendo un buen resultado. Por parte del dataset full el comportamiento es ideal de la misma forma que con el dataset small como lo muestra la figura 37.

IV-B6. SVM Scaled: El modelo arrojó una precisión del 0.92 para el dataset small y del 0.94 para el dataset full. Las figuras 38 y 39 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

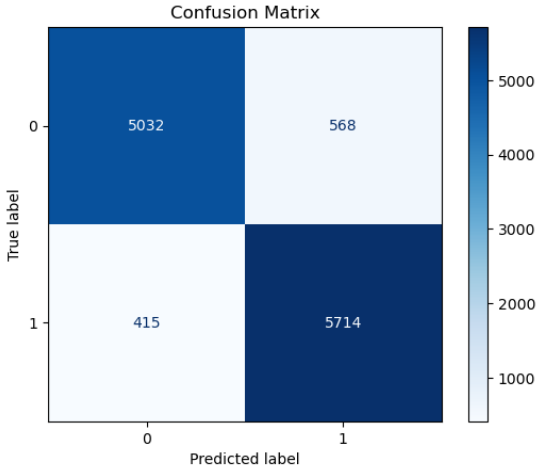

Figura 38. Matriz de confusión de SVM Scaled del dataset small.

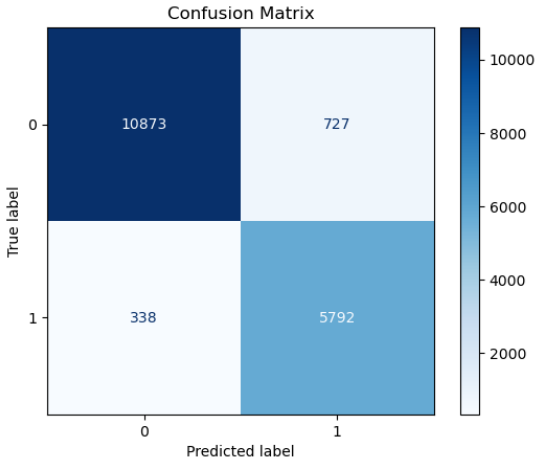

Figura 39. Matriz de confusión de SVM Scaled del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.97 y para el dataset full fue de 0.98. Como lo muestran las figuras 40 y 41 respectivamente.

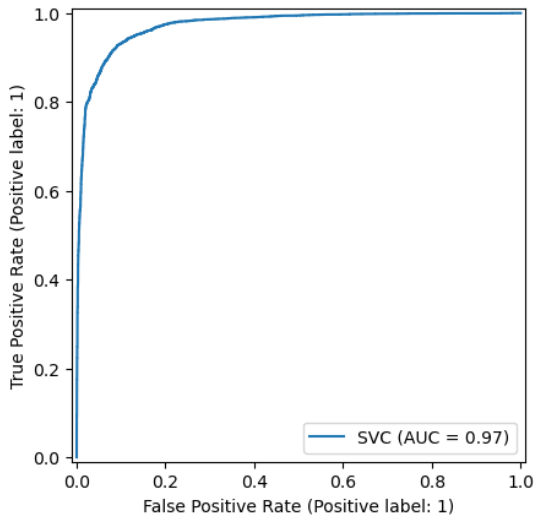

Figura 40. AUC-ROC de SVM Scaled del dataset small.

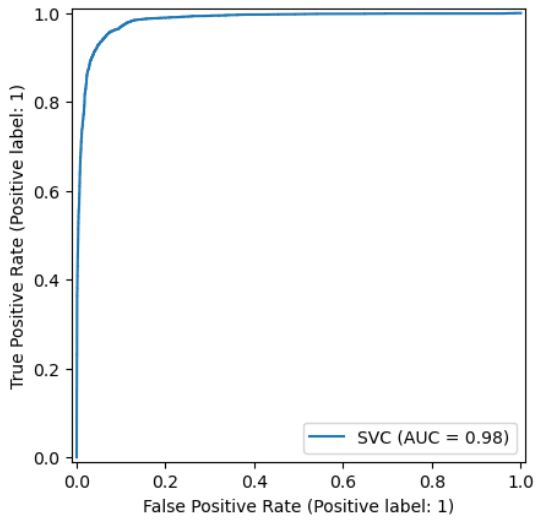

Figura 41. AUC-ROC de SVM Scaled del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 42 y 43 respectivamente.

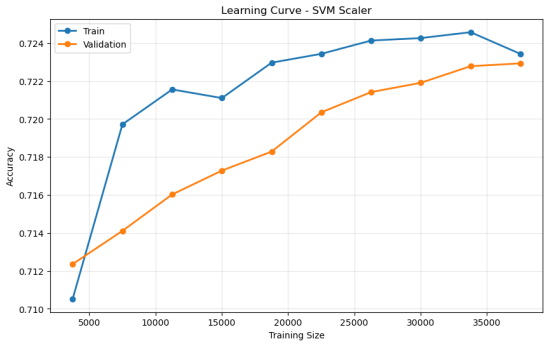

Figura 42. Linea de aprendizaje de SVM Scaled del dataset small.

images/SVMscalCurveFull.png

Figura 43. Linea de aprendizaje de SVM Scaled del dataset full

En el caso del dataset small a partir de la Figura 42 se puede identificar que el modelo sufre de overfitting ya que las dos curvas estas a pesar de incrementar, la validación no se acerca al test, por lo cual no converge. Este mismo comportamiento le sucede al dataset full como lo muestra la figura 43.

IV-B7. Linear SVC: El modelo arrojó una precisión del 0.88 para el dataset small y del 0.91 para el dataset full. Las figuras 44 y 45 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

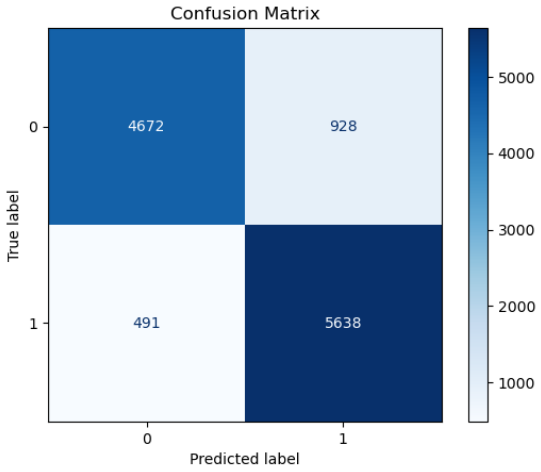

Figura 44. Matriz de confusión de SVM Linear SVC del dataset small.

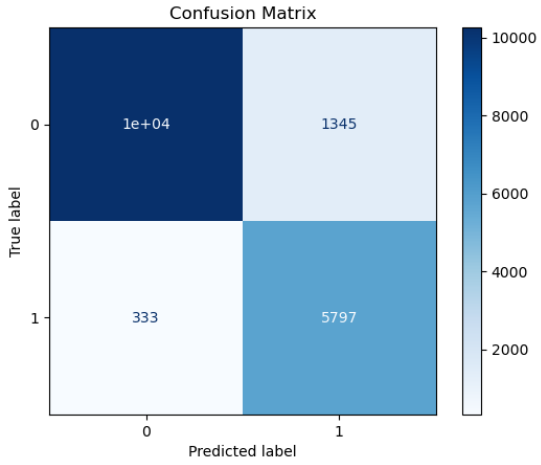

Figura 45. Matriz de confusión de SVM Linear SVC del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.95 y para el dataset full fue de 0.97. Como lo muestran las figuras 46 y 47 respectivamente.

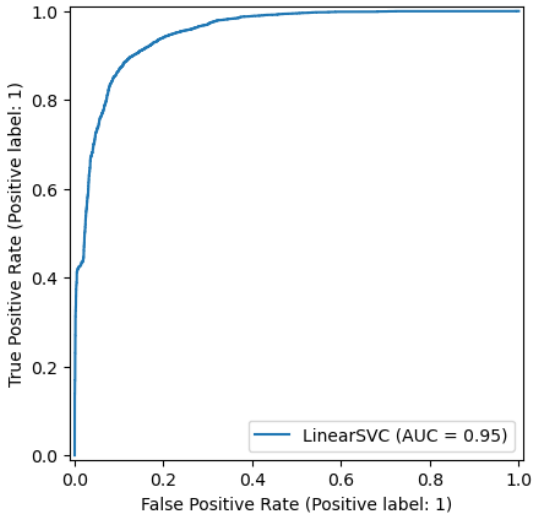

Figura 46. AUC-ROC de SVM Linear SVC del dataset small.

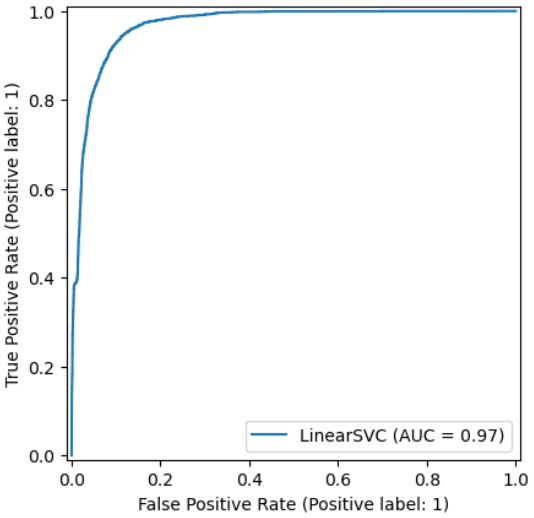

Figura 47. AUC-ROC de SVM Linear SVC del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 48 y 49 respectivamente.

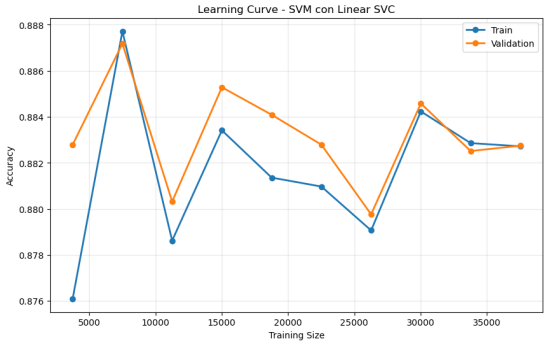

Figura 48. Linea de aprendizaje de SVM Linear SVC del dataset small.

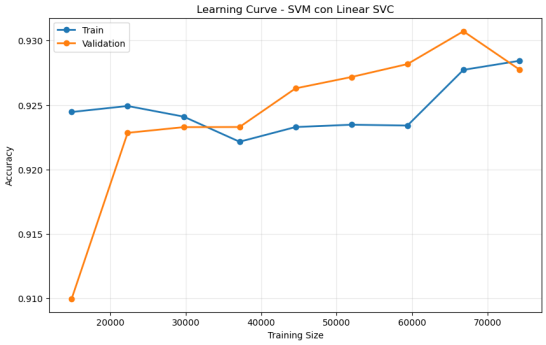

Figura 49. Linea de aprendizaje de SVM Linear SVC del dataset full

En el caso del dataset small a partir de la Figura 48 se puede identificar que el modelo no sufre de overfitting pero se puede evidenciar que el SVM llego a su capacidad por la aparición de ruido que genera las oscilaciones aleatorias, a pesar de esto esta bien. Por otro lado la figura 49 muestra el caso del dataset full con un buen comportamiento.

IV-B8. Linear SVC Scaler: El modelo arrojó una precisión del 0.90 para el dataset small y del 0.92 para el dataset full. Las figuras 50 y 51 muestran las matrices de confusión del dataset small y del dataset full respectivamente.

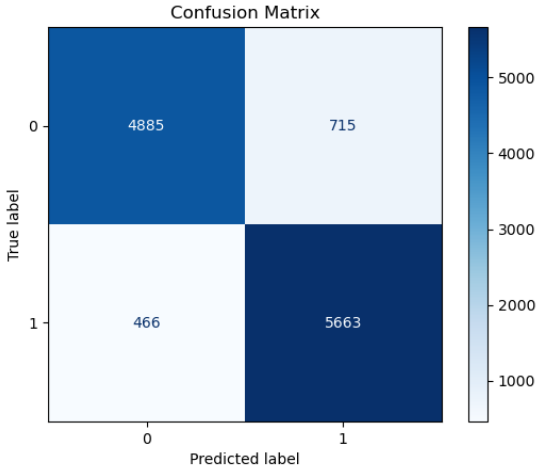

Figura 50. Matriz de confusión de SVM Linear SVC con Scaled del dataset small.

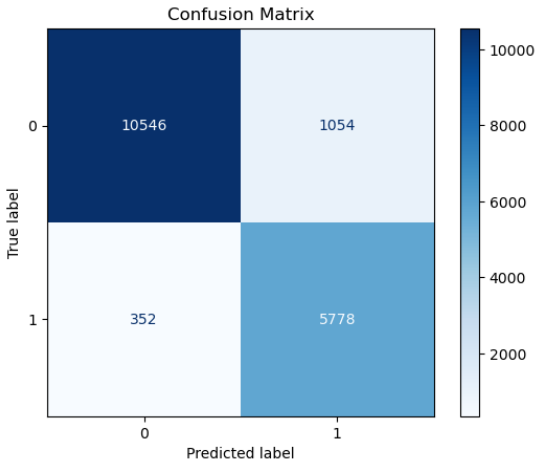

Figura 51. Matriz de confusión de SVM con Scaled del dataset full

Posteriormente se evaluó el resultado del modelo según AUC-ROC, para el dataset small el resultado fue de 0.96 y para el dataset full fue de 0.98. Como lo muestran las figuras 52 y 53 respectivamente.

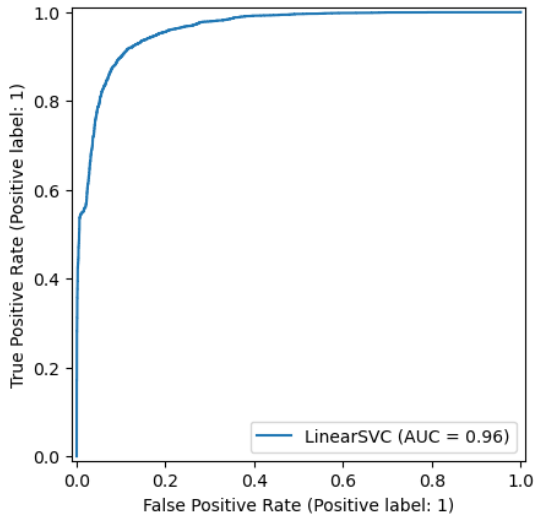

Figura 52. AUC-ROC de SVM Linear SVC con Scaled del dataset small.

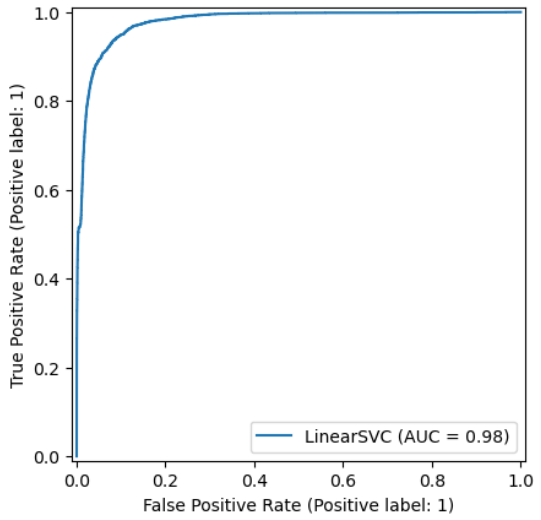

Figura 53. AUC-ROC de SVM Linear SVC con Scaled del dataset full

Por último se graficó la curva de aprendizaje del modelo para el dataset small y el full, como lo muestra la figura 54 y 55 respectivamente.

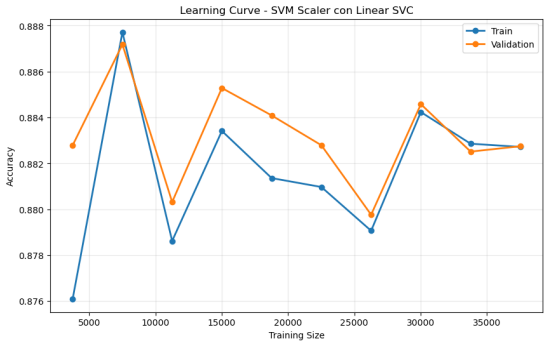

Figura 54. Linea de aprendizaje de SVM Linear SVC con Scaled del dataset small.

images/SVMlinearscalCurveFull.png

Figura 55. Linea de aprendizaje de SVM Linear SVC con Scaled del dataset full

En el caso del dataset small a partir de la Figura 54 se puede identificar que el modelo no sufre de overfitting pero se puede evidenciar que el SVM llego a su capacidad por la aparición de ruido que genera las oscilaciones aleatorias, a pesar de esto esta bien. Por otro lado la figura 55 muestra el caso del dataset full con un buen comportamiento.

V. Discusión y conclusiones

Tomando en cuenta los resultados obtenidos se logró evidenciar que en todos los modelos el mejor resultado siempre fue el dado dataset full comparado con el small, esto se debe a que el dataset full representa la totalidad de los datos y el utilizar SMOTE nos permitió mantener de una mejor manera la naturaleza de la información comparado con el dataset small. En todos los casos el rendimiento fue mejor, así como el AUC-ROC.

Realizando una comparación entre los distintos AUC-ROC los mejores modelos fueron Gradient Boost, Random Forest, Linear SVC y kNN como se puede evidenciar en la figura 56 para el dataset small y figura 57 para el dataset full.

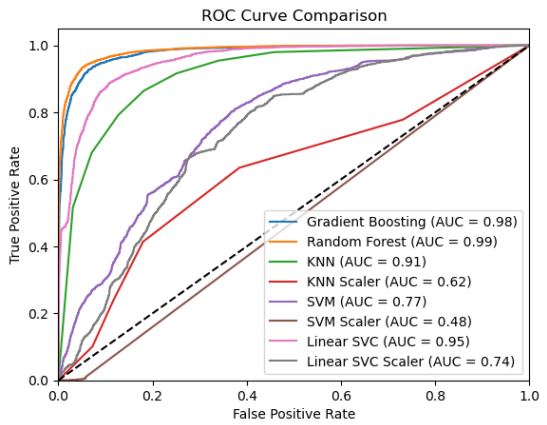

Figura 56. Comparación de la curva ROC de todos los modelos del dataset small.

images/rocCurveComparisonFull.png

Figura 57. Comparación de la curva ROC de todos los modelos del dataset full.

Del mismo modo se realizó una comparación entre la precisión de cada modelo donde los mejores fueron Gradient Boosting, Random Forest, kNN y Linear SVC. Como lo muestran las figuras 58 y 59.

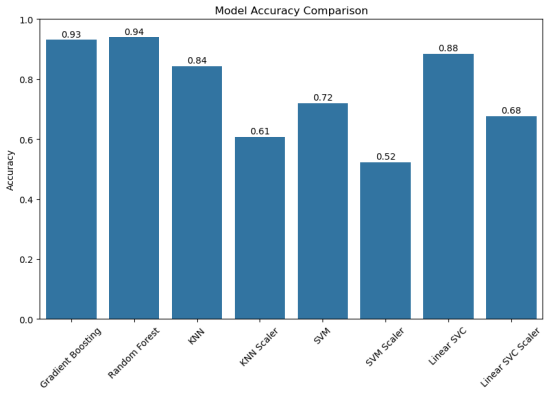

Figura 58. Comparación del accuracy de todos los modelos del dataset small.

images/ModelAccuracyComparisonFull.png

Figura 59. Comparación del accuracy de todos los modelos del dataset full.

En conclusión mejor modelo utilizado para el dataset fue el modelo de Random Forest con el dataset full, presentando el mejor desempeño y con la mejor linea de aprendizaje. Este modelo es el ideal para manejar este tipo de datos y lograr generar las mejores predicciones sobre el peligro de un sitio web según las carácteristicas tratadas por el dataset.

REFERENCIAS

[1] G. Vrbančič, I. Jr. Fister, V. Podgorelec. Datasets for Phishing Websites Detection. Data in Brief, Vol. 33, 2020, DOI: 10.1016/j.dib.2020.106438. Available: https://github.com/GregaVrbancic/Phishing-Dataset?tab=readme-ov-file . [Accessed: 5-Sep-2025]

[2] M. Kosinski, “What is Phishing?”, IBM Think, [Online]. Available: https://www.ibm.com/think/topics/phishing . [Accessed: 20 Oct. 2025].

[3] Skula, Ivan & kvet, michal. (2024). URL and Domain Obfuscation Techniques - Prevalence and Trends Observed on Phishing Data. 10.1109/SAMI60510.2024.10432841.

[4] Marcus Law, “Netskope Data Shows Phishing Success Rate Tripled in 2024,” Cyber Magazine, Jan. 07, 2025. [Online]. Available: https://cybermagazine.com/articles/netskope-data-shows-phishing-success-rate-tripled-in-2024 . [Accessed: Oct. 20, 2025].

[5] G. Vrbančič, I. Jr. Fister, y V. Podgorelec, ”Parameter setting for deep neural networks using swarm intelligence on phishing websites classification,Ïnternational Journal on Semantic Web and Information Systems, vol. 15, no. 4, pp. 167-192, 2019, doi: 10.1142/S021821301960008X.

[6] S. Sindhu, S. P. Patil, A. Sreevalsan, y F. Rahman, ”Phishing detection using random forest, SVM and neural network with backpropagation,ïn 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 2020, pp. 391-394, doi: 10.1109/ICSTCEE49637.2020.9277256.

[7] L. Lakshmi, M. P. Reddy, C. Santhaiah, y U. J. Reddy, ”Smart phishing detection in web pages using supervised deep learning classification and optimization technique ADAM,”Wireless Personal Communications, vol. 118, pp. 3549-3564, 2021, doi: 10.1007/s11277-021-08165-y.

[8] Y. Zhang, J. Wang, y B. Chen, ”Detecting false data injection attacks in smart grids: A semi-supervised deep learning approach,ÏEEE Transactions on Smart Grid, vol. 12, no. 1, pp. 623-634, 2020.

[9] L. Breiman, Random Forests,”Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[10] Alejandro Gil, Juan David Sánchez, *Project_ML1*, GitHub, 2025. [En línea]. Disponible en: https://github.com/AGGE82/Project_ML1