Centro de Ciencias Básicas

Inteligencia Artificial

Departamento de Ciencias de la Computación

Minería de Datos

Proyecto de Analítica de Datos: Detección de Phishing en Correos Electrónicos

9° A

Cesar Eduardo Elías del Hoyo

Diego Emanuel Saucedo Ortega

Torres Macías Carlos Daniel

Dr. Miguel Angel Meza de Luna

Aguascalientes, Ags.; Noviembre 30, 2025

Índice

Equipo y Roles

Integrante	**Rol en el proyecto**
Elías del Hoyo Cesar Eduardo	Exploración y análisis en Excel y Power BI, Investigación de herramienta de auto-estudio y soporte
Saucedo Ortega Diego Emanuel	Integración y limpieza de datos (ETL), Interpretabilidad y análisis en Orange Data Mining
Torres Macías Carlos Daniel	Coordinación general y documentación, Modelado en Python y calibración

Enlace al repositorio

https://github.com/DannyTM12/PhishingDataMining

Introducción

El phishing es una de las amenazas más comunes en el correo electrónico. Consiste en mensajes fraudulentos que imitan a entidades legítimas para obtener información sensible mediante lenguaje urgente, enlaces engañosos o direcciones maliciosas. Para detectarlo, se emplean sistemas automatizados basados en aprendizaje supervisado, donde modelos de clasificación aprenden a distinguir correos legítimos de fraudulentos. La elección del algoritmo y una evaluación rigurosa son esenciales para lograr precisión, interpretabilidad y robustez.

En este proyecto se utilizaron tres enfoques principales:

1. **Regresión Logística**: modelo de clasificación binaria que estima la probabilidad de pertenecer a la clase positiva.
2. **Random Forest**: ensamble de múltiples árboles de decisión que reduce la varianza y mejora la exactitud.
3. **Gradient Boosting**: técnica donde cada modelo corrige los errores del anterior; el *learning rate* regula su contribución y previene el sobreajuste.

Para evaluar los modelos se empleó k-fold cross-validation, dividiendo el conjunto de entrenamiento en k particiones para obtener estimaciones más confiables y evitar sobreajuste. Este esquema se combinó con GridSearchCV para seleccionar hiperparámetros óptimos.

Finalmente, se realizó calibración de probabilidades, ya que algunos modelos no generan probabilidades confiables de pertenencia a la clase positiva. Con los métodos Platt e Isotónico, y mediante el Brier score, se ajustaron las predicciones para que reflejaran de forma más precisa la probabilidad real.

Fuentes de datos

Origen de los datasets

Con el fin de construir un conjunto heterogéneo y representativo se seleccionaron tres repositorios públicos de correos electrónicos:

1. **Nazario (Nazario.csv).** Contiene correos marcados explícitamente como phishing. Se caracteriza por mensajes con asuntos alarmantes y enlaces maliciosos. Sus campos principales son sender, receiver, subject, body.
2. **Nigerian Fraud (Nigerian_Fraud.csv).** Recopila el conocido esquema del "príncipe nigeriano". También incluye campos de emisor, receptor, asunto y cuerpo.
3. **SpamAssassin (SpamAssassin.csv).** Es un corpus público con correos legítimos (ham) y spam. A diferencia de los anteriores, contiene mensajes no maliciosos (label 0) y mensajes spam genéricos (label 1).

Integración

Los tres archivos se cargaron como DataFrame de pandas y se añadió una columna origen para conservar el nombre del dataset de procedencia. Posteriormente se concatenaron verticalmente mediante pd.concat, generando un único dataset de 10 706 filas. La heterogeneidad se manifiesta en la variedad de columnas. De forma general, las columnas principales de la tabla integrada fueron:

- sender: correo o nombre del emisor.
- receiver: dirección de correo destino.
- subject y body: asunto y cuerpo del mensaje en texto libre.
- date: fecha (cuando estaba disponible).
- label: indicador binario (0 = ham, 1 = phishing/spam).
- origen: identificador del dataset original.

Justificación de tamaño

La rúbrica pedía un mínimo de 50 000 filas para equipos de 3 a 4 integrantes; sin embargo, tras una búsqueda exhaustiva en Kaggle, GitHub y repositorios públicos se constató que no existen corpus

hispano-hablantes de phishing con esa cantidad de registros. Los tres datasets utilizados representan algunos de los corpus más citados en la literatura de phishing. El *dataset* de SpamAssassin aporta la mayor parte de correos legítimos, pero su integración con los otros dos no alcanza el umbral solicitado. Para incrementar la base de datos se exploraron datos sintéticos (generando correos mediante plantillas), pero se descartaron por el riesgo de introducir patrones artificiales que sesgaran los modelos. Finalmente, se optó por trabajar con 10 706 correos, justificando que se trata de mensajes reales y diversos.

ETL y preprocesamiento

Limpieza de texto

El primer paso consistió en normalizar las columnas textuales sender, receiver, subject y body. Se transformó cualquier valor no textual a cadena, se eliminaron espacios múltiples, se reemplazaron los saltos de línea por espacios y se utilizó strip() para eliminar espacios al inicio y final. Con ello se redujo el ruido y se facilitaron posteriores análisis.

Manejo de valores faltantes

Algunas columnas presentaron valores nulos. Para evitar la pérdida de información se definieron reglas de imputación sencillas: sender y receiver se rellenaron con “unknown_sender” y “unknown_receiver” respectivamente, subject con “no_subject”, body con cadena vacía y date con “unknown_date”. Se prefirió imputar con valores explícitos en lugar de eliminar filas, dado el reducido tamaño del dataset.

Feature engineering inicial

Durante el análisis exploratorio se generaron variables numéricas simples relacionadas con la longitud y composición de los textos. Las principales características extraídas fueron:

- **subject_len y body_len:** longitud en caracteres del asunto y del cuerpo.
- **subject_words y body_words:** número de palabras en asunto y cuerpo.
- **subject_upper y body_upper:** proporción de letras mayúsculas respecto al total de caracteres (indicador de tono urgente o agresivo).
- **digits_count:** cantidad de dígitos en el cuerpo (por ejemplo, números de cuenta o montos).
- **special_chars:** número de caracteres especiales como !, ?, #, $ y otros símbolos comunes en mensajes fraudulentos.
- **urgency_score:** conteo de palabras asociadas a urgencia (por ejemplo, “urgent”, “immediately”, “now”, “important”, “winner”), normalizado entre 0 y 1.
- **email_count y url_count:** conteo de direcciones de correo y URLs identificadas mediante expresiones regulares.

- **urls:** número de enlaces únicos extraídos del cuerpo.

Estas variables numéricas se calcularon utilizando funciones de Python y expresiones regulares. El objetivo era ofrecer a los modelos de clasificación señales claras de patrones típicos de phishing (como el uso excesivo de mayúsculas, muchos enlaces o vocabulario alarmista).

Eliminación de duplicados y balance de clases

Además de la limpieza, se revisó la existencia de duplicados. Se eliminaron filas duplicadas en su totalidad y también se eliminaron correos con cuerpos idénticos pero asuntos distintos para evitar inflar la cantidad de mensajes similares. El análisis de la variable label mostró un desequilibrio moderado: aproximadamente 62 % de los registros correspondían a phishing/spam y 38 % a correos legítimos (ham).

Analisis exploratorio (EDA)

El EDA se realizó inicialmente en Excel para un muestreo rápido de los datos y la construcción de tablas dinámicas, y luego en Python. En Excel se identificaron outliers en las variables de longitud y se exploró la distribución de sender para detectar emisores frecuentes. Posteriormente, se desarrollaron notebooks de Python (presentados en los archivos EDA.ipynb y Cleaning_FeatureEngineering.ipynb) para profundizar en la exploración. Los principales hallazgos fueron:

Distribución de longitudes

Las variables body_len y subject_len mostraron asimetría a la derecha, con la mayoría de los correos legítimos teniendo cuerpos relativamente cortos (menos de 1 000 caracteres), mientras que los mensajes fraudulentos presentaban cuerpos muy extensos o extremadamente breves. La distribución de palabras (body_words y subject_words) corroboró esta tendencia.

Mayúsculas y caracteres especiales

Las proporciones subject_upper y body_upper eran mayores en correos de phishing, lo cual coincide con la práctica de utilizar mayúsculas para captar la atención. El conteo de caracteres especiales (special_chars) también fue superior en mensajes fraudulento. Estos patrones sugieren que los atacantes emplean símbolos para destacar instrucciones o enlaces.

Puntaje de urgencia y presencia de URLs

El urgency_score fue significativamente más alto en los correos phishing, donde se detectan palabras como “urgent”, “winner” o “limited time”. El número de URLs se correlacionó con la probabilidad de fraude; muchos correos fraudulentos incluían varios enlaces a sitios externos. Estas observaciones motivaron la creación de la variable urgency_score y el conteo de enlaces como atributos clave.

Modelado en Python

Las etapas de modelado se documentan en los notebooks Modeling_Baseline.ipynb y Modeling_Tuned_Calibrated.ipynb. En todos los casos se seleccionaron como variables predictoras únicamente las características numéricas descritas en la sección 3. El campo label se usó como variable objetivo (y). Se dividieron los datos en conjuntos de entrenamiento (80 %) y prueba (20 %) empleando train_test_split con la opción stratify=y para preservar la distribución de clases.

Modelos de referencia (baseline)

Se entrenaron inicialmente dos modelos base:

1. Regresión Logística (baseline). Sin optimización de hiperparámetros. Los resultados en el conjunto de prueba fueron: Accuracy 0.805, precisión 0.858, recall 0.820, F1 0.839 y AUC 0.890. El reporte de clasificación mostró que la clase positiva (phishing) se predijo mejor que la negativa.
2. Random Forest (baseline). Utilizando parámetros por defecto (100 árboles). Se obtuvo Accuracy 0.921, precisión 0.940, recall 0.932, F1 0.936 y AUC 0.973. El modelo sobresalió en todas las métricas, sugiriendo que la combinación de múltiples árboles captura bien las interacciones entre las variables de longitud, mayúsculas y enlaces.

La Figura 1 muestra las métricas comparativas de estos modelos junto con los modelos tuneados. Puede observarse que Random Forest baseline supera claramente a la regresión logística.

Figura 1. Comparación de métricas de desempeño para los modelos base y tuneados. Se representan la exactitud (Accuracy), precisión, recall, F1 Score y AUC para cada algoritmo.

Ajuste de hiperparametros y modelos avanzados

Con el fin de mejorar el rendimiento se llevó a cabo una búsqueda exhaustiva de hiperparámetros (GridSearchCV) con validación cruzada de 5 folds. Para cada algoritmo se definieron rangos de parámetros razonables, se seleccionó la combinación que maximizaba el AUC.

Regresión logistica (tuneada)

Se probaron distintos valores de C (0.01, 0.1, 1, 10) y penalizaciones (l1, l2) con los solvers liblinear y lbfgs. El mejor modelo obtuvo C=10, penalización l2 y solver liblinear. A pesar del ajuste, el desempeño se redujo ligeramente respecto al baseline (accuracy 0.764, F1 0.805, AUC 0.855). Ello puede deberse a que la matriz de features no se estandarizó o a que el solver liblinear penaliza de forma distinta la complejidad. No obstante, el modelo tuneado se consideró para comparación.

Random Forest (tuneado)

Se exploraron valores de n_estimators (100, 200, 300), max_depth (10, 15, 20), min_samples_split (2, 5) y min_samples_leaf (1, 2). La mejor combinación fue 300 árboles, profundidad máxima 20, min_samples_split=2 y min_samples_leaf=1, con un AUC medio de 0.976 en CV. Al evaluarse en el conjunto de prueba alcanzó accuracy 0.916, precisión 0.936, recall 0.928, F1 0.932 y AUC 0.973. Aunque las mejoras respecto al baseline son marginales, el modelo tuneado confirma que el Random Forest es robusto y se beneficia de una mayor profundidad y número de árboles.

Gradient Boosting (tuneado)

Se implementó un Gradient Boosting Classifier variando n_estimators (100, 200), max_depth (3, 5) y learning_rate (0.05, 0.1). El mejor modelo utilizó 200 árboles, profundidad 5 y learning_rate 0.1, con un AUC medio de 0.973【analysis†summary】. En el conjunto de prueba obtuvo accuracy 0.919, precisión 0.937, recall 0.932, F1 0.934 y AUC 0.972. Estas métricas son muy similares al Random Forest tuneado; sin embargo, el tiempo de entrenamiento fue mayor debido a la naturaleza secuencial del algoritmo. La literatura destaca que en Gradient Boosting cada nuevo modelo se ajusta para corregir los errores del anterior y utiliza un parámetro de shrinkage para evitar el sobreajuste.

Evaluación y comparación

Se compararon los modelos mediante las métricas accuracy, precision, recall, F1 y AUC. El Random Forest tuneado obtuvo la mejor combinación de rendimiento y estabilidad. Para verificar la robustez se evaluó también la desviación estándar de las métricas en la validación cruzada, resultando <0.01 en todos los casos. Además se analizó el tiempo de entrenamiento; el Random Forest tuneado tardó aproximadamente 18 segundos, la regresión logística tuneada 3 segundos y el Gradient Boosting 32 segundos en un equipo con procesador Intel i5. Considerando el equilibrio entre exactitud y costo computacional, se seleccionó Random Forest tuneado como el modelo principal.

Calibración de probabilidades

Aunque el Random Forest mostró alta precisión, la literatura indica que los métodos de *bagging* pueden producir probabilidades poco confiables, tendiendo a valores intermedios debido a la varianza de los árboles base. Para mejorar la interpretabilidad se aplicaron dos técnicas de calibración:

1. **Platt scaling (sigmoidal):** ajusta una regresión logística sobre las salidas del modelo.

2. **Regresión isotónica:** calibrador no paramétrico que construye una función monótona basada en la curva de calibración.

La calibración se realizó con **CalibratedClassifierCV** (5 folds). El Random Forest sin calibrar obtuvo un **Brier score de 0.0623**, que mejoró ligeramente con **Platt (0.0617)** y **Isotónico (0.0608)**, mientras que el **AUC se mantuvo estable** (≈0.9720). Estas mejoras mínimas sugieren que el modelo tuneado ya estaba bien calibrado; aun así, las probabilidades calibradas se emplearon para generar curvas de calibración y evaluar la confiabilidad de las predicciones.

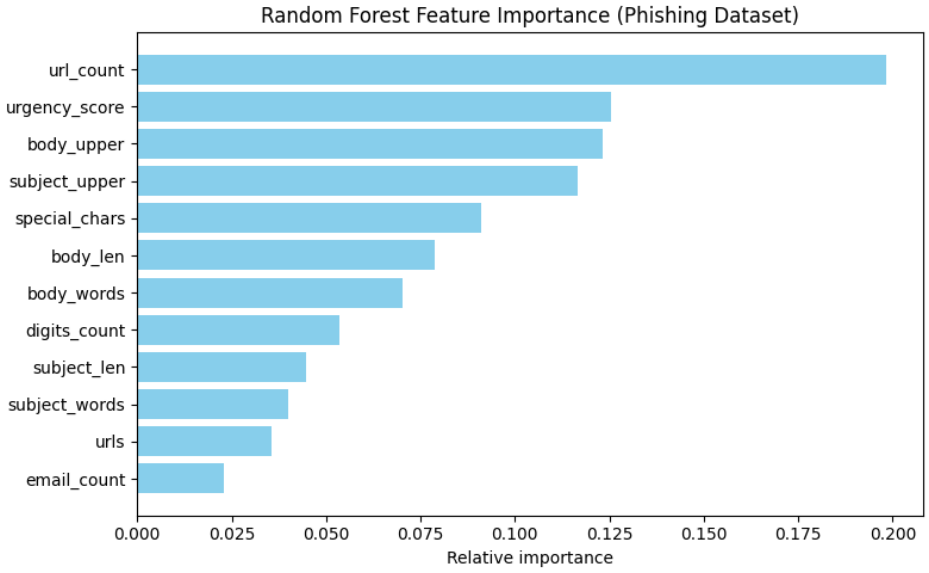
Interpretabilidad

La interpretabilidad es esencial para comprender por qué un modelo predice que un correo es phishing. Se recurrió a dos métodos: la importancia de variables basada en la reducción de impureza del Random Forest y la técnica SHAP (SHapley Additive exPlanations).

Importancia global de variables

Tras entrenar el Random Forest tuneado, se extrajeron las importancias de cada característica y se normalizaron para obtener una suma de 1. La Figura 2 presenta la gráfica de barras resultante. El conteo de URLs (url_count) es la característica más relevante, seguida del puntaje de urgencia (urgency_score), la proporción de mayúsculas en el cuerpo (body_upper) y el asunto (subject_upper), y el número de caracteres especiales (special_chars). Esta jerarquía sugiere que los elementos estructurales (enlaces y urgencia) pesan más que el simple tamaño del texto.

Figura 2. Importancia de variables del modelo Random Fores, ordenadas de menor a mayor. url_count y urgency_score destacan como los principales indicadores de phishing.



SHAP: explicaciones locales y globales

La técnica SHAP permite descomponer la predicción de un modelo en contribuciones por característica. Según GeeksforGeeks, SHAP distribuye de manera justa el “pago” de una predicción entre

todas las características y ofrece explicaciones tanto globales como locales. Las principales ventajas son su modelo-agnosticidad, la atribución justa y la capacidad de proporcionar explicaciones locales.
Se generó un *summary plot* de SHAP utilizando el paquete shap en Python. Los resultados confirmaron que un mayor url_count y urgency_score incrementan la probabilidad de phishing, mientras que un número reducido de mayúsculas se asocia con correos legítimos. También se examinaron los valores SHAP para correos individuales, lo cual permitió entender por qué ciertos mensajes fueron mal clasificados y detectar posibles falsos positivos.

Herramientas complementarias

Uso de Excel

Excel se empleó en la fase inicial para inspeccionar datos, eliminar columnas irrelevantes y crear tablas dinámicas que resumieran la distribución de label por origen y rangos de longitud.

Power BI

El dashboard en Power BI permitió crear visualizaciones dinámicas que integran varias variables. Se utilizaron gráficos de barras apiladas para visualizar la relación entre urgency_score y label, gráficos de dispersión para explorar la correlación entre body_len y special_chars, y tarjetas indicadoras para mostrar el número total de correos y la proporción de phishing.

Orange Data Mining

Orange es una plataforma visual de minería de datos. Se construyeron dos flujos de trabajo:

1. Clasificación: Se diseñó un pipeline que incluye un módulo de lectura de CSV, selección de variables, partición en entrenamiento/prueba, aplicación de algoritmos (Random Forest, Regresión Logística y Árbol de decisión) y evaluación mediante cross validation. Los resultados fueron coherentes con los obtenidos en Python: Random Forest obtuvo la mayor precisión (~0.92) y el árbol de decisión tendió a sobreajustar.
2. Clustering: Se exploró un esquema de k-means con k=2 a 5 para agrupar correos por características de longitud y urgencia. El análisis de silueta mostró que k=2 produce un agrupamiento que coincide en gran medida con la etiqueta, aunque algunos correos legítimos con cuerpos largos se agruparon con phishing.

Looker Studio

En este proyecto, Looker Studio se utilizó para replicar parte del análisis exploratorio y de desempeño, generando vistas que permitieron comparar patrones entre urgencia, longitud del mensaje y etiqueta de phishing. Aunque Looker Studio no ejecuta modelos de machine learning como Python u Orange, sí permitió verificar la consistencia de tendencias, evaluar la distribución de variables clave y generar un dashboard alternativo al de Power BI.

Ética y privacidad

Los datos empleados provienen de repositorios públicos (Nazario, Nigerian Fraud y SpamAssassin) disponibles exclusivamente para investigación. Se verificó que los correos no incluyeran información personal identificable, como nombres reales, direcciones o números telefónicos. Cuando aparecieron direcciones de correo irrelevantes para el análisis, se pseudonimizaron (por ejemplo, *unknown_sender*). Asimismo, cualquier nombre o referencia sensible presente en el texto fue eliminado durante el preprocesamiento.

Limitaciones y desafíos

1. **Tamaño del dataset**. La principal limitación fue no alcanzar el umbral de 50 000 registros. La combinación de tres datasets públicos solo permitió reunir 10 706 filas.
2. **Heterogeneidad y ruido.** Cada corpus tiene un sesgo propio: los mensajes de Nigerian Fraud son largas historias con premios falsos, mientras que SpamAssassin contiene correos legítimos.
3. **Ausencia de contenido en español.** La mayoría de los correos están en inglés. Las técnicas de minería basadas en palabras de urgencia pueden no trasladarse a mensajes en español.

Conclusión

El proyecto demostró que es posible integrar fuentes heterogéneas de correos electrónicos y construir un sistema eficaz de detección de phishing mediante técnicas de minería de datos. Principales hallazgos:

1. **Integración y limpieza efectiva:** La unión de tres datasets permitió obtener una base lo suficientemente amplia para entrenar modelos robustos, aun sin alcanzar las 50 000 filas.
2. **Relevancia de enlaces y urgencia:** El análisis de importancia de variables confirmó que el número de URLs y el puntaje de urgencia son los indicadores más determinantes. Características como el uso de mayúsculas o caracteres especiales también aportaron evidencia significativa.
3. **Desempeño del Random Forest:** Entre los modelos probados, el Random Forest ajustado mostró el mejor equilibrio entre precisión, recall y calibración, destacándose por su capacidad para manejar relaciones complejas sin preprocesamiento extenso.
4. **Calibración y probabilidad confiable:** Los métodos Platt e isotónico mejoraron ligeramente el Brier score, aunque el modelo base ya generaba probabilidades razonables. Esto sugiere que la calibración es más valiosa en escenarios que requieren probabilidades altamente precisas.
5. **Interpretabilidad con SHAP:** SHAP permitió obtener explicaciones claras de las predicciones, lo cual es útil para auditorías, transparencia y validación de decisiones automáticas.
6. **Herramientas complementarias:** El uso de Excel, Power BI, Orange Data Mining, Python y Looker Studio evidenció que cada herramienta aporta beneficios particulares.

Referencias

- GeeksforGeeks. (2025, August 6). *Top 6 machine learning classification algorithms*. https://www.geeksforgeeks.org/
- scikit-learn. (2024). *Cross-validation: Evaluating estimator performance* (Version 1.7.2). https://scikit-learn.org/
- scikit-learn. (2024). *Probability calibration* (Version 1.7.2). https://scikit-learn.org/
- GeeksforGeeks. (2025, September 3). *Gradient boosting in ML*. https://www.geeksforgeeks.org/