DEEPURL: ANÁLISIS DE ENLACES PARA COMBATIR EL PHISHING CON IA

Integrantes

Juan Diego Arias Grisales

Luz Ensueño Serna Giraldo

Jhon Anderson Castaño Quintero

Leidy Milena Agudelo Lezama

Ejecutores

Frank Yesid Zapata Cataño

Natalia Betancourt Herrera

Universidad de Caldas

Talento TECH

BOOTCAMP Inteligencia Artificial

Mayo 2025

Contenido

Introducción

El phishing es una técnica de ciberataque que busca engañar a los usuarios para que revelen información confidencial, como contraseñas, datos bancarios o credenciales de acceso, haciéndose pasar por entidades confiables a través de correos electrónicos, mensajes o sitios web falsos. En las empresas, este tipo de ataque puede tener consecuencias graves: desde la filtración de datos sensibles y pérdida de información crítica, hasta fraudes financieros y daños a la reputación corporativa. Además, puede interrumpir las operaciones y generar altos costos asociados a la recuperación y a sanciones legales por incumplimiento de normativas de protección de datos.

El phishing ha tenido un impacto significativo en las empresas a nivel global, con un aumento notable en la frecuencia y sofisticación de los ataques en los últimos años

Entre las estadísticas globales sobre el phishing se puede decir que se ha dado un aumento de ataques, en el 2023, los ataques de phishing aumentaron un 58,2% en comparación con el año anterior, reflejando una creciente sofisticación y alcance de los atacantes (1); El phishing fue la segunda causa más común de filtraciones de datos, con un costo promedio de USD 4,91 millones por incidente (2); Las grandes empresas norteamericanas pierden en promedio USD 14,8 millones al año debido a ataques de phishing, lo que representa un aumento significativo respecto a años anteriores.(3); Cada empleado pierde en promedio 7 horas al año debido a estas estafas, lo que se traduce en una pérdida anual de USD 3,2 millones en productividad para una empresa mediana además El phishing fue la segunda causa más común de filtraciones de datos, con un costo promedio de USD 4,91 millones por incidente.

Hablando del impacto en América Latina según el Diario la República, el costo promedio por filtración de datos en 2024 fue de USD 2,76 millones, siendo el phishing responsable del 16% de los incidentes. Se ha dado también un aumento en clics de phishing, la tasa casi se triplicó en 2024, con más de 8 de cada 1.000 usuarios empresariales haciendo clic en estos enlaces cada mes.

El sector financiero y de seguros recibió el 27,8% de los ataques de phishing, marcando la mayor concentración entre todos los sectores, por lo que se hace necesaria una herramienta que detecte este tipo de amenazas.

Objetivo

- **General**

Desarrollar una solución inicial basada en un modelo supervisado de aprendizaje automático capaz de detectar intentos de phishing en URLs, mediante el análisis de características relevantes que permitan clasificar los enlaces como legítimos o maliciosos, contribuyendo así a mejorar la seguridad informática en Colombia.

- **Específicos**

Análisis y limpieza de datos:

Revisar la base de datos para detectar valores nulos, errores e inconsistencias, y aplicar transformaciones como normalización, codificación y tratamiento de valores atípicos.

Comprensión y selección de variables:

Clasificar las variables por tipo (numéricas, categóricas, booleanas), analizar su relevancia frente a la clase objetivo (phishing o benign), y seleccionar las más útiles para el modelo.

Preparación para el modelado:

Explorar relaciones entre variables con visualizaciones, y dividir los datos en conjuntos de entrenamiento y prueba asegurando un balance adecuado entre clases.

- **Alcance**

Crear una solución inicial usando un modelo supervisado de Machine learning cuya finalidad es identificar URLs maliciosas (phishing) a partir de características técnicas y estructurales extraídas automáticamente. Su uso está enfocado principalmente en contextos de seguridad informática y filtrado web.

Comprensión del Negocio

El phishing es la técnica más utilizada por ciberdelincuentes para robar credenciales, dinero o datos personales; Una sola URL de phishing que pase el filtro puede comprometer a decenas, cientos o miles de usuarios.

- **Objetivos de Negocio**

    - **Reducir el riesgo:** Minimizar la cantidad de URLs maliciosas a las que los empleados le dan clic entregando una herramienta que las filtre.

- **Mejorar la reputación:** Proteger la confianza de clientes o empleados en la plataforma/empresa con una herramienta que le permita filtrar las URLs y determinar si es benigna o no.

- **Criterios de Éxito**

  - Precisión alta (pocos falsos negativos, es decir, pocas URLs phishing que escapen).
  - Tasa de falsos positivos aceptable (que sitios legítimos no se bloqueen en exceso).
  - Velocidad de respuesta: Simples milisegundos por URL.

Comprensión / Preparación de los Datos

La Base de datos fue extraída del repositorio de kaggle https://www.kaggle.com/datasets/aaditeypillai/phishing-website-content-dataset

Este dataset está diseñado para entrenar modelos de detección de URLs maliciosas. Contiene características extraídas de la URL, del contenido HTML y de metadatos que permiten identificar patrones comunes en ataques de phishing.

Tipos de URLs incluidas:

Phishing (400): URLs maliciosas orientadas al engaño del usuario.

Benignas (400): URLs legítimas sin riesgo de seguridad.

Características extraídas (22 en total):

Basadas en URL: Dominio, uso de IPs, tipo de protocolo (HTTP/HTTPS).

Detalles adicionales:

Tamaño del dataset: 800 muestras balanceadas (50/50 phishing y benignas).

Licencia: MIT (uso, modificación y distribución libre para fines académicos y no comerciales).

Actualización: No especificada.

Análisis Detallado de Variables:

En la siguiente tabla (Tabla 1) se muestra la fuente de los datos, tamaño de la base de datos (Dataset), variable objetivo (Type) y las variables predictoras que corresponden a las columnas principales.

Sección	Descripción
Fuente	Archivo URL.xlsx
Tamaño del Dataset	800 filas y 22 columnas
Variable Objetivo	type: Clasificación del sitio (phishing / benign)
Variables Predictoras	Link Density - HTTP Count - HTTPS - Count External - Iframes Count - Hidden Iframes Count

Tabla 1.

Estructura de los datos:

En la siguiente tabla (Tabla 2), se muestra cada una de las 22 columnas que corresponden con las variables identificadas en la base de datos, tipo de dato (entero (INT), Decimal (FLOAT), objeto (OBJECT) o texto (STRING)

INT = 9 Columnas

FLOAT =1 Columnas

OBJECT = 12 Columnas

800 valores únicos.

Variable	Tipo de dato	Descripción
URL	object	La dirección completa del sitio web.
Domain	object	El dominio principal del sitio. Ej: "example.com".
IP Address	object	Dirección IP asociada al dominio.
Link Density	float64	Densidad de enlaces en la página (proporción de texto vs. enlaces).
External Links Count	int64	Número de enlaces salientes (externos).
Internal Links Count	int64	Número de enlaces internos.
External IP Count	int64	Cantidad de IPs externas referenciadas.
HTTP Count	int64	Número de enlaces que usan HTTP.
HTTPS Count	int64	Número de enlaces que usan HTTPS.
Non Count	int64	Número de enlaces sin protocolo (vacios o mal formateados).
External Iframes Count	int64	Número de iframes que cargan contenido externo.
Hidden Iframes Count	int64	Número de iframes ocultos (posible técnica maliciosa).
Country	object	Pais asociado a la IP del dominio.
Organization	object	Nombre de la organización que posee el dominio.
Certificate	object	Información sobre el certificado SSL.
Certificate Not Before	object	Fecha de inicio de validez del certificado SSL.
Certificate Not After	object	Fecha de expiración del certificado SSL.
If HTTPS	int64	Indicador binario (1 = sitio usa HTTPS, 0 = no).
Whois_country	object	Pais del registrante según la base de datos Whois.
whois_creation_date	object	Fech0a de creación del dominio según Whois.
whois_expiration_date	object	Fecha de expiración del dominio según Whois.
type	int64	Etiqueta de clase: probablemente 1 = phishing, 0

Tabla 2.

En la preparación de los datos se eliminaron 3 filas (registros), ya que estas generan conflicto al entrenar el modelo, ya que no tienen los datos completos quedando la base con 797 filas (registros) y 22 columnas.

En la normalización de la base de datos se convierte la columna Type a entero (INT), ya que esta columna es la elegida como etiqueta del modelo, el cual catalogamos como 'phishing': 1, 'benign': 0

Modelado

- **Modelo 1: Random Forest (sklearn)**

El modelo implementado utiliza Random Forest (Scikit-learn), un algoritmo de aprendizaje automático basado en múltiples árboles de decisión, ideal para tareas de clasificación como la detección de phishing. Se entrena con un 80% de los datos y se evalúa con el 20% restante, asegurando reproducibilidad con random_state=42.

```
# Paso 6: Entrenar el modelo
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)  # 80% train, 20% test
model = RandomForestClassifier(n_estimators=100, random_state=42)  # Crea el modelo
model.fit(X_train, y_train)  # Entrena el modelo
```

Características clave del modelo:

Algoritmo: RandomForestClassifier(n_estimators=100)

Entrenamiento: model.fit(X_train, y_train)

Datos: URLs representadas por 22 features (estructurales, contenido, SSL, WHOIS).

Ventajas:

Resistente al ruido: Ideal para datos web inconsistentes.

Evita sobreajuste: Promedia resultados entre múltiples árboles.

Alta escalabilidad: Clasifica grandes volúmenes de URLs rápidamente.

Versátil: Soporta datos categóricos y numéricos (SSL, dominios, protocolos, etc.).

Robusto ante valores faltantes: Puede predecir incluso con información incompleta (como registros WHOIS ausentes).

Eficiente en tiempo real: Clasifica URLs como phishing o benignas en milisegundos.

Aplicaciones comunes:

Detección de fraudes, clasificación de amenazas, sistemas de recomendación y predicción de comportamiento.

- **Modelo 2: XGBOOST**

Este modelo utiliza XGBoost (Extreme Gradient Boosting), una de las librerías más potentes en machine learning, especialmente eficaz en clasificación binaria con datos estructurados como URLs.

```
# Paso 6: Dividir datos en entrenamiento y prueba (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42)      # Random_state para reproducibilidad

# Crear y entrenar modelo XGBoost
model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
model.fit(X_train, y_train)                           # Entrenar modelo con datos de entrenamiento
```

Configuración clave:

Modelo: XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)

use_label_encoder=False: evita advertencias por codificadores obsoletos.

eval_metric='logloss': optimiza la pérdida logarítmica (menor es mejor).

random_state=42: garantiza resultados reproducibles.

Funcionamiento interno:

Usa Gradient Boosting, donde cada árbol nuevo corrige errores del anterior.

Los errores tienen mayor peso en las siguientes iteraciones.

Combina los árboles aditivamente, no por votación como Random Forest.

Optimiza las predicciones usando gradientes de la función de pérdida.

Ventajas:

Alta precisión: Excelente rendimiento en clasificación de URLs phishing vs benignas.

Manejo automático de valores nulos y ruido: Ideal para datos web incompletos (por ejemplo, WHOIS faltante).

Control de sobreajuste: Gracias a técnicas de regularización.

Eficiencia: Entrenamiento rápido y soporte para paralelización.

Interpretable: Permite analizar la importancia de cada característica (por ejemplo, uso de HTTPS o longitud de URL).

Escalable: Se adapta bien a grandes volúmenes de datos estructurados.

Aplicaciones típicas:

Detección de amenazas (phishing, malware)

Análisis financiero y prevención de fraude

Diagnóstico médico basado en datos tabulares

Sistemas de mantenimiento predictivo

- **Métricas de evaluación**

Para evaluar los modelos de detección de phishing, se utilizaron las siguientes métricas clave:

Accuracy (Precisión global): Porcentaje total de predicciones correctas. Útil, pero puede ser engañosa si las clases están desbalanceadas.

Precisión: Mide cuántas URLs detectadas como phishing realmente lo eran. Importante para evitar falsos positivos (bloqueo de sitios legítimos).

Recall (Sensibilidad): Indica cuántas URLs maliciosas fueron detectadas correctamente. Es la métrica más crítica en ciberseguridad, ya que reduce el riesgo de dejar pasar ataques reales (falsos negativos).

F1-Score: Balance entre precisión y recall. Muy útil cuando se busca un rendimiento equilibrado.

Matriz de confusión: Muestra los verdaderos y falsos positivos/negativos, facilitando el análisis detallado de errores del modelo.

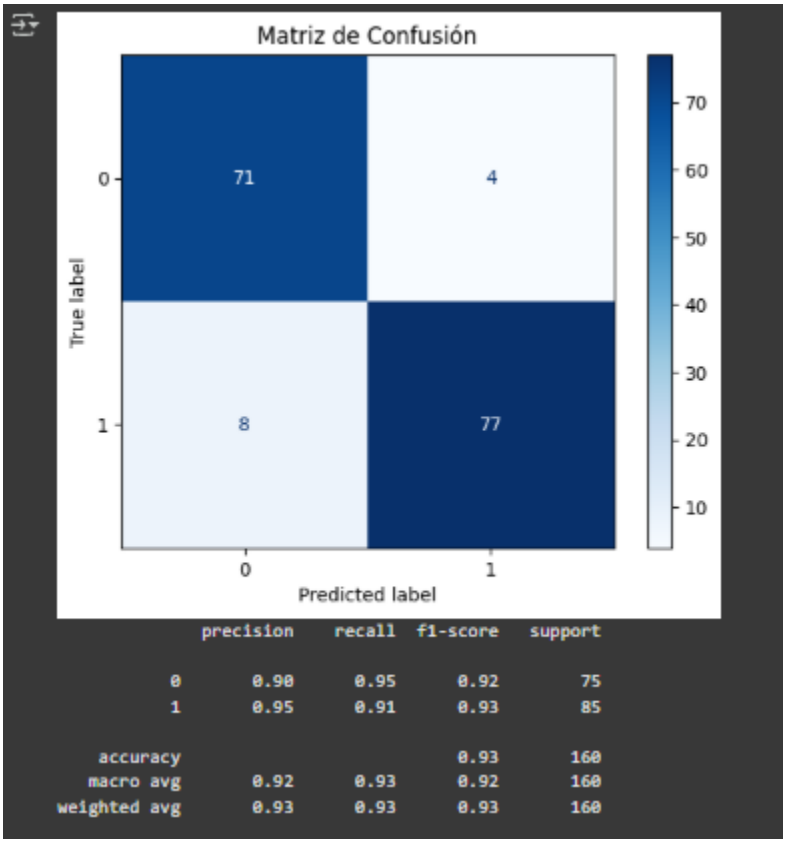
- **Comparación Random Forest (sklearn) vs. XGBOOST**

Random Forest (sklearn):

```
1 # Paso 7: Evaluar el rendimiento del modelo
2 y_pred = model.predict(X_test)
3 print("🔎 Matriz de confusión:\n", confusion_matrix(y_test, y_pred))
4 print("\n📊 Reporte de clasificación:\n", classification_report(y_test, y_pred))
```

```
🔎 Matriz de confusión:
[[71  4]
 [ 8 77]]

📊 Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.90      0.95      0.92        75
           1       0.95      0.91      0.93        85

    accuracy                           0.93       160
   macro avg       0.92      0.93      0.92       160
weighted avg       0.93      0.93      0.93       160
```

Fuente: Propia




```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92        75
           1       0.95      0.91      0.93        85

    accuracy                           0.93       160
   macro avg       0.92      0.93      0.92       160
weighted avg       0.93      0.93      0.93       160
```

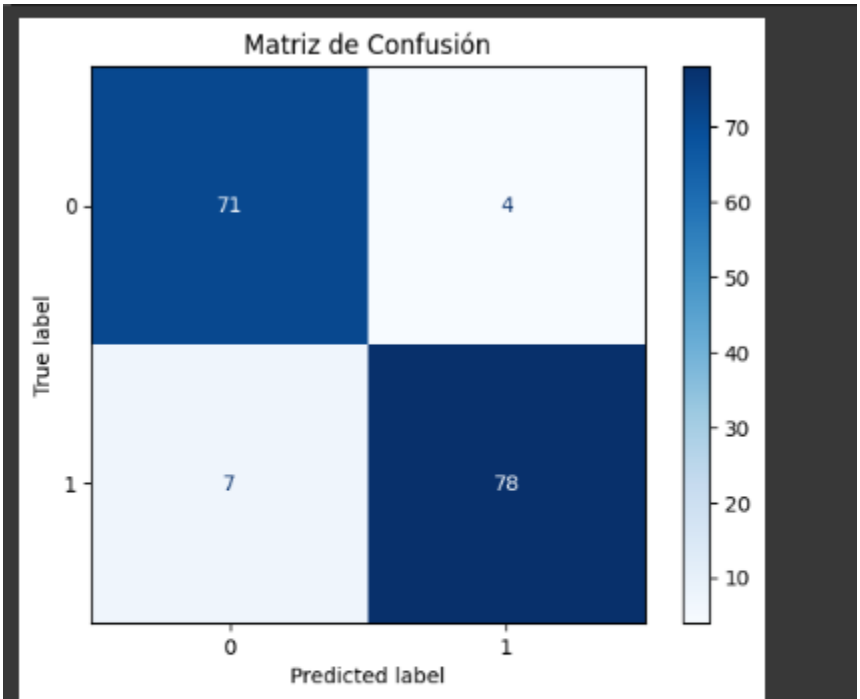
Fuente: Propia

XGBOOST:

```python
# Paso 7: Evaluar desempeño del modelo en datos de prueba
y_pred = model.predict(X_test)                          # Predecir etiquetas para el set test

print("🔍 Matriz de confusión:\n", confusion_matrix(y_test, y_pred))  # Mostrar matriz de confusión
print("\n📊 Reporte de clasificación:\n", classification_report(y_test, y_pred))  # Métricas detalladas

# Guardar el modelo entrenado y los objetos de preprocesamiento para uso futuro
joblib.dump(model, 'phishing_model_xgb.pkl')            # Modelo XGBoost
joblib.dump(imputer, 'imputer.pkl')                     # Imputador
joblib.dump(scaler, 'scaler.pkl')                       # Escalador
```

```
🔍 Matriz de confusión:
[[71  4]
 [ 7 78]]

📊 Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.91      0.95      0.93        75
           1       0.95      0.92      0.93        85

    accuracy                           0.93       160
   macro avg       0.93      0.93      0.93       160
weighted avg       0.93      0.93      0.93       160

['scaler.pkl']
```

Fuente: Propia



	precision	recall	f1-score	support
0	0.91	0.95	0.93	75
1	0.95	0.92	0.93	85
accuracy			0.93	160
macro avg	0.93	0.93	0.93	160
weighted avg	0.93	0.93	0.93	160

Fuente: Propia

Random Forest:

Recall (clase phishing - 1): 0.91

Precisión (clase phishing - 1): 0.95

Falsos negativos: 8

XGBoost:

Recall (clase phishing - 1): 0.92

Precisión (clase phishing - 1): 0.95

Falsos negativos: 7

¿Cuál es el mejor?

XGBoost es el modelo más recomendable para un sistema de detección de phishing porque logra un mejor equilibrio entre sensibilidad y precisión, lo que se traduce en mayor protección real contra amenazas con un mínimo de errores críticos.

Evaluación

Se analizaron los resultados de los modelos Random Forest y XGBoost, siendo este último el más eficaz gracias a su mayor recall en la detección de URLs phishing (0.92). Ambos modelos mostraron alta precisión (0.95) y bajo número de falsos positivos y negativos, destacando su efectividad para un sistema de seguridad informática.

La validación con expertos del Bootcamp confirmó la pertinencia del enfoque y la calidad técnica de la solución, además de proponer mejoras como el uso de datasets más amplios y la integración con herramientas de análisis dinámico.

Entre las lecciones aprendidas se resalta la importancia de una buena preparación de los datos, la necesidad de priorizar el recall en ciberseguridad y la ventaja de algoritmos como XGBoost en tareas de clasificación.

Sin embargo, se identificaron limitaciones como el tamaño reducido del dataset y la ausencia de análisis dinámico de sitios.

Implementación

Plan de despliegue: Para facilitar la interacción con el modelo y su posible adopción en entornos reales, se desarrolló una interfaz gráfica mediante Gradio, una herramienta liviana, rápida y de fácil integración. El modelo fue entrenado localmente, guardado con la librería joblib, y posteriormente cargado para permitir la predicción de URLs ingresadas manualmente o desde un archivo CSV.

Este sistema permite a los usuarios analizar enlaces sospechosos en tiempo real, mostrando si se trata de una URL benigna (0) o phishing (1), y mantiene un historial de predicciones accesible durante la sesión, para facilitar el análisis y seguimiento de casos detectados.

Herramientas utilizadas:

Python 3.10: Lenguaje de programación principal.

Pandas & NumPy: Limpieza, manipulación y análisis de datos.

Scikit-learn & XGBoost: Entrenamiento y validación de los modelos de machine learning.

Gradio: Desarrollo de la interfaz web interactiva para pruebas del modelo.

Joblib: Serialización del modelo entrenado.

Google Colab: Plataforma utilizada para el desarrollo colaborativo y pruebas del código.

Consideraciones para mantenimiento: Es importante actualizar regularmente el dataset y reentrenar el modelo para adaptarse a nuevos patrones de phishing. Si se desea escalar la solución, se recomienda implementarla mediante APIs en la nube. Además, se deben aplicar buenas prácticas de seguridad y mantener una documentación técnica clara para facilitar futuras mejoras o correcciones.

Presupuesto

La siguiente tabla (Tabla 3) presenta los costos estimados para el desarrollo del proyecto DEEPURL. La mayoría de las herramientas utilizadas, como Google Colab, librerías de Python (Scikit-learn, XGBoost, Gradio, etc.), y el diseño de la interfaz, no generaron costos al ser de uso libre. Los principales gastos corresponden al uso de infraestructura local (computadores personales), servicios básicos (internet y energía), y la estimación de sueldos para los científicos de datos. El costo total aproximado del proyecto es de $13.050.000 COP.

Concepto	Costo Aproximado (COP)	Observaciones
Plataforma de desarrollo (Google Colab)	$ 0	Plataforma gratuita (versión estándar).
Librerías y herramientas (Python, Scikit-learn, XGBoost, Gradio, Joblib)	$ 0	Todas son de código abierto y uso libre con fines académicos.
Infraestructura de prueba local (computadores personales)	$ 5.000.000	Recursos propios de los participantes.
Internet / energía (uso durante el desarrollo)	$ 50.000	Estimación de consumo doméstico por el tiempo de desarrollo.
Diseño de interfaz (Gradio)	$ 0	Herramienta gratuita y de bajo código para prototipos rápidos.
Sueldos/Data Scientist	$ 8.000.000	Sueldos de los científicos de datos
Total estimado del proyecto	**$ 13.050.000**	-

Tabla 3.

Diccionario de datos

Término	Descripción
URLS	Direcciones web que pueden ser legítimas o usadas para ataques de phishing.
KAGGLE	Plataforma de ciencia de datos donde se obtienen datasets públicos.
DATASET	Conjunto de datos utilizado para entrenar y evaluar el modelo.
WHOIS	Protocolo usado para obtener información de registro sobre dominios.
IFRAMES	Elemento HTML a veces usado maliciosamente en páginas de phishing.
INT	Números enteros (por ejemplo, longitud de la URL).
FLOAT	Números decimales (por ejemplo, tiempo de respuesta WHOIS).
OBJECT	Generalmente usado para texto o categorías en Pandas.
STRING	Cadenas de texto (usado para etiquetas o campos descriptivos).
'BENIGN'	URL legítima, no representa amenaza.
'PHISHING'	URL maliciosa diseñada para engañar.
RANDOM FOREST	Algoritmo de ensamble basado en árboles de decisión.
XGBOOST	Algoritmo de gradiente optimizado muy eficiente y potente.
RECALL	Proporción de verdaderos positivos detectados entre todos los positivos reales. Muy importante en detección de phishing.
JOBLIB	Permite guardar y cargar modelos de forma eficiente.
GRADIO	Librería Python para crear interfaces web amigables para modelos de IA.

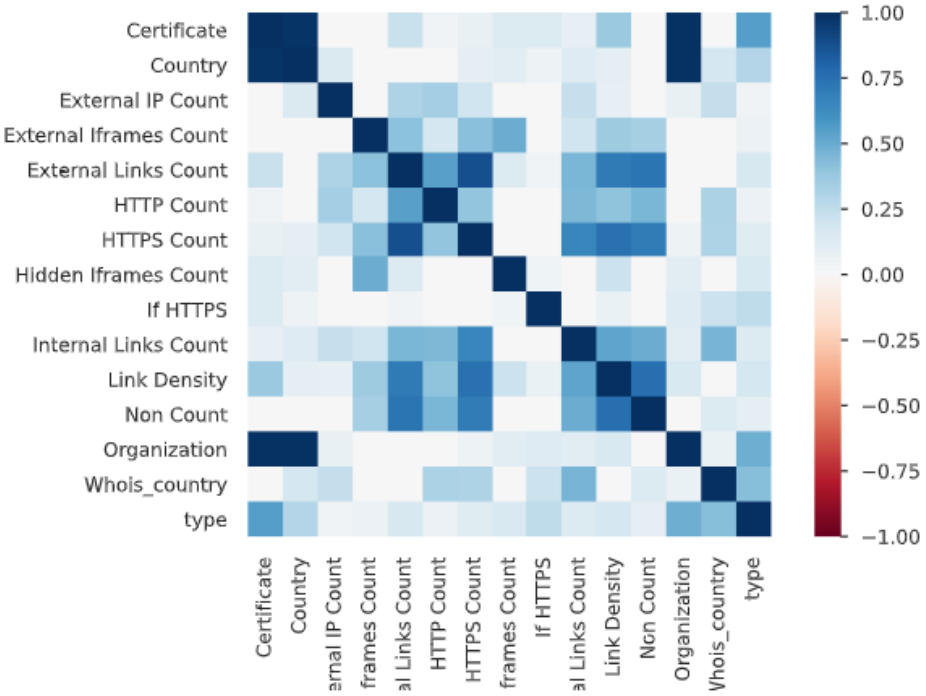
Fuente: Propia

EDA:

Este mapa de calor visualiza las correlaciones entre pares de variables. Las variables están listadas en los ejes X e Y (horizontal y vertical), y los colores representan la fuerza y dirección de la correlación entre cada par de variables.

La variable Certificate parece estar altamente correlacionada con Organization, lo cual tiene sentido ya que ambos provienen de información del certificado del sitio web.

External Links Count y Link Density tienen una correlación positiva visible, lo cual también tiene sentido, ya que más enlaces externos podrían aumentar la densidad de enlaces.(directamente proporcional).



Fuente: Propia.

Conclusiones y recomendaciones

Se realizó una limpieza exhaustiva de la base de datos, corrigiendo valores nulos, errores e inconsistencias, y se aplicaron transformaciones como normalización y codificación para mejorar la calidad y homogeneidad de los datos. Se eliminaron 3 registros incompletos, quedando 797 en total, y se ajustó la columna Type de dato de la variable objetivo a booleano.

Se clasificaron y analizaron las variables, identificando las más relevantes para detectar URLs maliciosas mediante análisis de correlación y estadísticas, lo que permitió reducir la dimensionalidad y mejorar el rendimiento del modelo.

Finalmente, se usaron visualizaciones para entender mejor las relaciones entre variables y se dividió el conjunto de datos en entrenamiento y prueba manteniendo el balance de clases, garantizando así una evaluación objetiva y un entrenamiento robusto del modelo.

Recomendaciones para el negocio o para futuras investigaciones

Se recomienda ampliar el dataset para mejorar la capacidad de generalización del modelo, incluyendo más URLs y nuevos vectores de ataque. Además, es importante reentrenar el modelo periódicamente con datos actualizados, debido a la constante evolución del phishing.

Para su implementación en entornos reales, se sugiere integrar el sistema mediante una API REST que automatice el análisis de URLs en diferentes plataformas. También se propone complementar el análisis con técnicas dinámicas como sandboxing para mejorar la detección.

Finalmente, se aconseja desarrollar una interfaz más robusta y escalable que incluya autenticación, historial persistente y control de acceso, facilitando su adopción en entornos corporativos.

Consideraciones éticas

Es fundamental proteger la privacidad de los usuarios, tratando las URLs como datos sensibles y evitando su almacenamiento sin consentimiento. El modelo debe usarse únicamente con fines de ciberseguridad y educación, no para censura o vigilancia.

Se debe promover la transparencia, explicando las decisiones del modelo cuando sea posible, especialmente si afectan a personas o empresas. Además, es crucial garantizar que el modelo no discrimine por país, idioma o sector sin una justificación técnica válida.

Webgrafía

- Pillai, A. (s.f.). Phishing website content dataset. Kaggle. https://www.kaggle.com/datasets/aaditeypillai/phishing-website-content-dataset
- (1) Zscaler. (2024, 4 de abril). Phishing attacks rise 58% year-over-year: AI & ThreatLabz 2024 phishing report. https://www.zscaler.com/mx/blogs/security-research/phishing-attacks-rise-58-year-ai-threatlabz-2024-phishing-report
- (2) IBM. (2022, 16 de agosto). Informe de IBM: Los consumidores pagan el precio de las filtraciones de datos. https://latam.newsroom.ibm.com/2022-08-16-Informe-de-IBM-Los-consumidores-pagan-el-precio-de-las-filtraciones-de-datos
- (3) Red Seguridad. (2021, 27 de agosto). El coste medio de los ataques de phishing para las empresas se ha casi cuadruplicado desde 2015. https://www.redseguridad.com/actualidad/el-coste-medio-de-los-ataques-de-phishing-para-las-empresas-se-ha-casi-cuadruplicado-desde-2015_20210827.html
- Interisle Consulting Group. (2024). Phishing landscape 2024: An annual study of the scope and distribution of phishing. https://interisle.net/insights/phishing-landscape-2024-an-annual-study-of-the-scope-and-distribution-of-phishing
- Hoxhunt. (s.f.). Phishing trends report: Key insights and data. https://hoxhunt.com/guide/phishing-trends-report

El modelo de machine learning es basado en los siguientes repositorios de GIT HUB

- • pb111. (s.f.). *Phishing URL Detection using ML* [Código fuente]. GitHub Gist. https://gist.github.com/pb111/88545fa33780928694388779af23bf58
- • dmlc. (s.f.). *XGBoost: Scalable and Flexible Gradient Boosting* [Repositorio]. GitHub. https://github.com/dmlc/xgboost

Todo el código correspondiente a los dos modelos de machine learning ha sido cargado en el repositorio de GIT HUB:

- https://github.com/jarias141/IA