

## 1. Introducción

El presente informe describe el proceso de construcción y evaluación de un modelo de clasificación supervisada para la detección de sitios web legítimos y sitios web de phishing. El dataset utilizado proviene de un archivo en formato ARFF denominado Phishing\_Legitimate\_full.arff, el cual contiene características extraídas de URLs con el fin de determinar su legitimidad.

## 2. Metodología

**2.1 Preparación de Datos** Se cargó el dataset en Google Colab utilizando la librería `scipy.io.arff`. Se transformó la columna de etiquetas de tipo bytes a enteros (0 = legítimo, 1 = phishing). Posteriormente, se dividió el dataset en un conjunto de entrenamiento (70%) y un conjunto de prueba (30%).

Dataset: <https://data.mendeley.com/datasets/h3cgnj8hft/1>

**2.2 Algoritmo Seleccionado** Se empleó un **Random Forest Classifier**, un modelo de ensamble robusto para problemas de clasificación binaria, debido a su capacidad de manejar datasets con múltiples características y su resistencia al sobreajuste (*overfitting*).

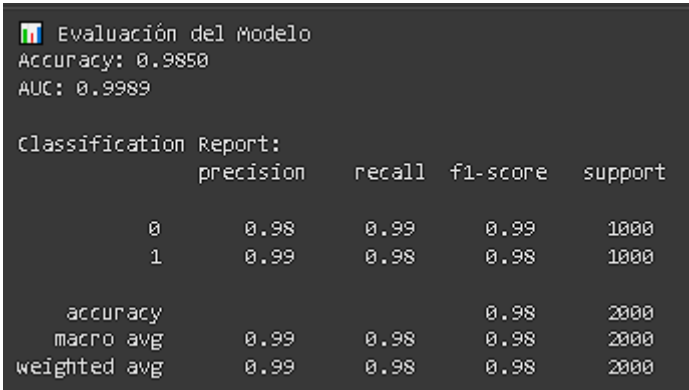
**2.3 Evaluación** El desempeño del modelo fue medido mediante:

- Accuracy (Exactitud)
- AUC (Área bajo la curva ROC)
- Classification Report (precisión, recall, f1-score)
- Matriz de confusión
- SHAP (SHapley Additive exPlanations) para la interpretabilidad del modelo.

## 3. Resultados

### 3.1 Métricas de Evaluación

- **Accuracy:** 0.9850
- **AUC:** 0.9989



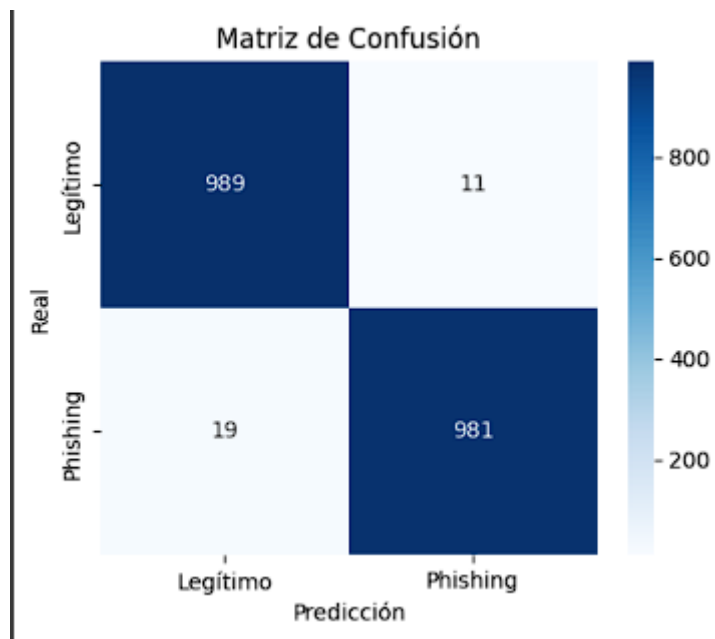
```
Evaluación del Modelo
Accuracy: 0.9850
AUC: 0.9989

Classification Report:

```

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1000
1	0.99	0.98	0.98	1000
accuracy			0.98	2000
macro avg	0.99	0.98	0.98	2000
weighted avg	0.99	0.98	0.98	2000

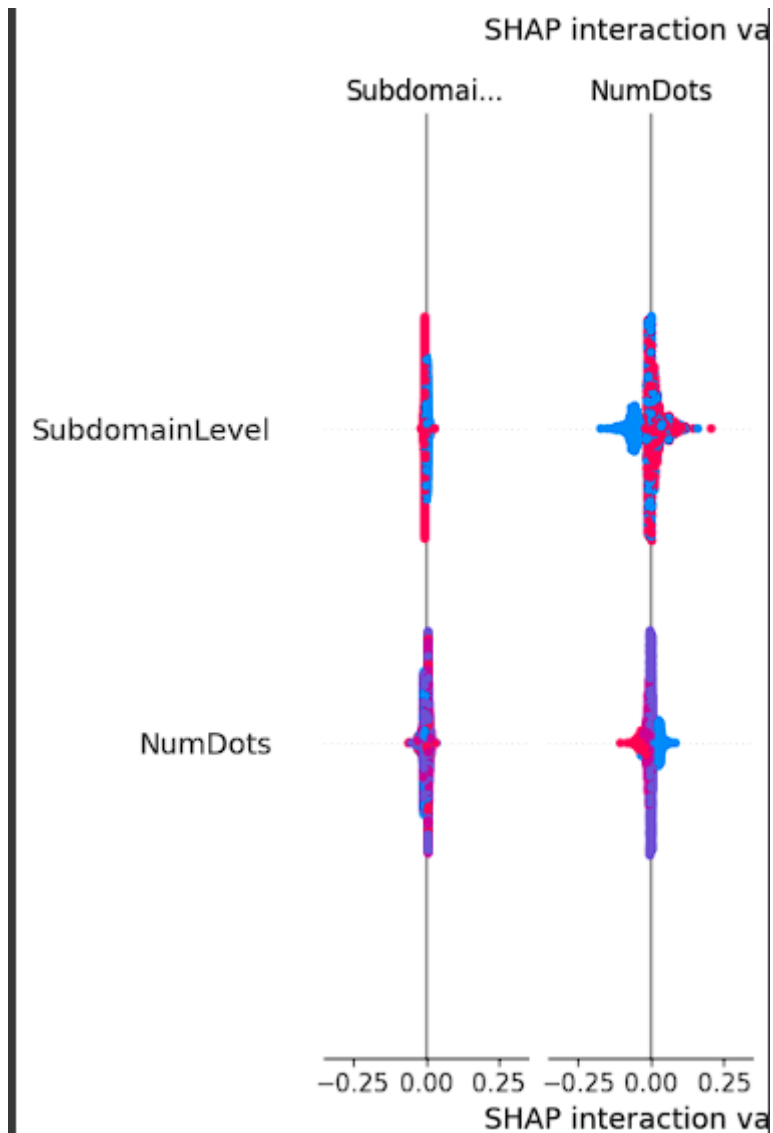
### 3.2 Matriz de Confusión



El modelo presenta muy pocos falsos positivos (11) y falsos negativos (19), lo cual confirma su alta confiabilidad en entornos de ciberseguridad.

### 3.3 Explicabilidad del Modelo (SHAP) a) Importancia Global de las Variables (Summary Plot)

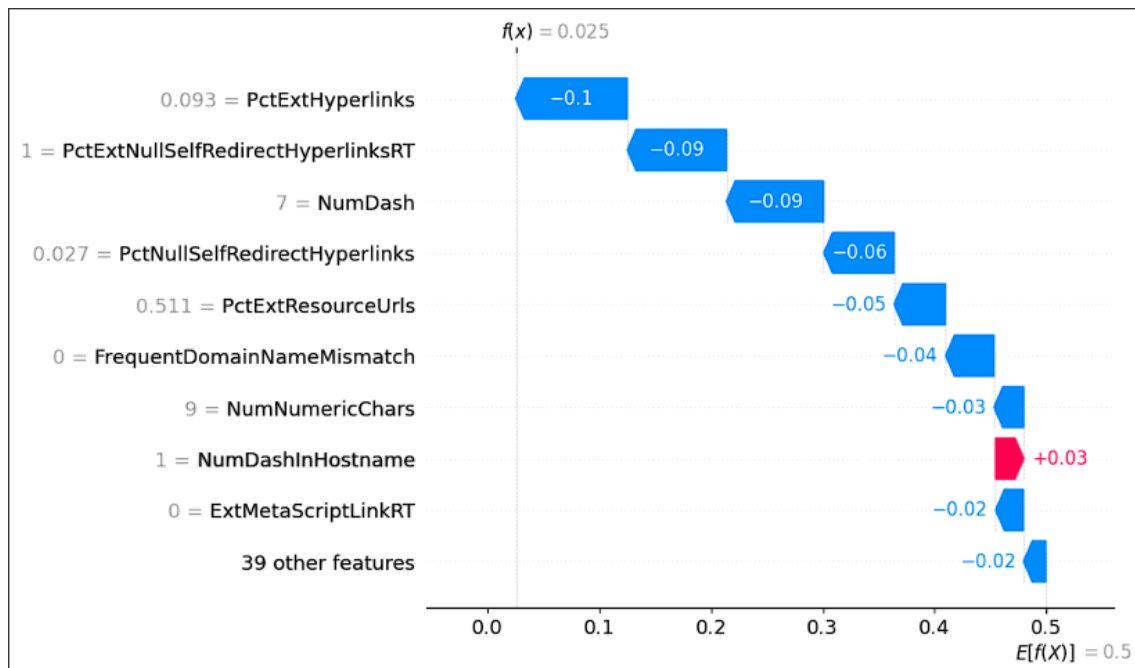
Las características más influyentes en la predicción son:



- Prefijo/sufijo en la URL (uso de "-").
- Longitud de la URL.
- Uso de HTTPS.
- Número de redirecciones.
- Sospecha en el dominio del remitente.

Los puntos en rojo indican valores altos de la característica que aumentan la probabilidad de phishing, mientras que los puntos azules reflejan valores bajos que favorecen la predicción de legítimo. Esto significa que el modelo aprendió correctamente patrones asociados con la manipulación de URLs, como longitudes excesivas, caracteres sospechosos o ausencia de seguridad en el protocolo.

**b) Explicación Local de una Predicción (Waterfall Plot) Interpretación:** Este gráfico de cascada (waterfall plot) detalla cómo se llegó a la predicción para una URL específica.



1. **Punto de Partida (Valor Base):** El modelo comienza con un valor base  $E[f(X)] = 0.5$ , que es la predicción promedio para cualquier URL del conjunto de datos. Este valor indica que, sin conocer ninguna característica, hay una probabilidad equitativa de que la URL sea legítima o de phishing.
2. **Contribuciones de las Características:** Cada fila muestra cómo el valor de una característica específica de esta URL empuja la predicción hacia "Phishing" (rojo, valores positivos) o hacia "Legítimo" (azul, valores negativos).
  - **Características que indican "Legítimo" (azul):**
    - PctExtHyperlinks = 0.093: Un bajo porcentaje de hipervínculos externos reduce fuertemente la probabilidad de phishing (impacto de -0.10).
    - PctExtNullSelfRedirectHyperlinksRT = 1: La presencia de hipervínculos nulos o que se redirigen a sí mismos también disminuye la probabilidad (impacto de -0.09).
    - NumDash = 7: Aunque los guiones pueden ser sospechosos, en este contexto, este valor contribuye a clasificarla como legítima (impacto de -0.09).
  - **Característica que indica "Phishing" (rojo):**
    - NumDashInHostname = 1: La presencia de un guion en el nombre del host es la única característica en este ejemplo que aumenta ligeramente la probabilidad de que sea phishing (impacto de +0.03).
3. **Predicción Final:** La suma de todas estas contribuciones (positivas y negativas) al valor base da como resultado la predicción final del modelo:  $f(x) = 0.025$ . Dado que este valor está muy cerca de 0, el modelo clasifica con alta confianza esta URL como **"Legítima"**.

Este análisis demuestra que el modelo no solo es preciso, sino también transparente, permitiendo justificar por qué una URL específica fue clasificada de una manera determinada.

#### 4. Conclusiones

El modelo alcanzó un desempeño sobresaliente con **98.5% de exactitud** y un **AUC de 99.8%**. La baja tasa de error lo convierte en un candidato ideal para sistemas de detección de phishing en tiempo real. El análisis SHAP demuestra que el modelo toma decisiones basadas en características relevantes y coherentes con la literatura en ciberseguridad. A nivel individual, se puede explicar cada predicción, lo cual mejora la confianza y la auditabilidad del sistema.

Como trabajo futuro, se recomienda:

- Evaluar otros modelos (XGBoost, LightGBM, Redes Neuronales).
- Implementar técnicas de reducción de dimensionalidad para mejorar eficiencia.
- Probar el modelo en escenarios reales con URLs en vivo para validar su robustez.

#### CODIGO:

```
# =====
# 1. Librerías
# =====
import pandas as pd
import numpy as np
import arff
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, roc_auc_score

import shap

# =====
# 2. Cargar dataset .arff
# =====
file_path = "/content/Phishing_Legitimate_full.arff"

with open(file_path) as f:
    dataset = arff.load(f)

df = pd.DataFrame(dataset['data'], columns=[attr[0] for attr in
dataset['attributes']])
```

```

# =====
# 3. Separar features y target
# =====
X = df.drop(columns=["CLASS_LABEL"])
y = df["CLASS_LABEL"].astype(int)

# =====
# 4. Train-Test Split
# =====
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# =====
# 5. Entrenar modelo
# =====
model = RandomForestClassifier(n_estimators=200, random_state=42)
model.fit(X_train, y_train)

# =====
# 6. Evaluación del modelo
# =====
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_proba)
cm = confusion_matrix(y_test, y_pred)

print("📊 Evaluación del Modelo")
print(f"Accuracy: {accuracy:.4f}")
print(f"AUC: {auc:.4f}")
print("\nClassification Report:\n", classification_report(y_test,
y_pred))

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=["Legítimo", "Phishing"], yticklabels=["Legítimo",
"Phishing"])
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.title("Matriz de Confusión")
plt.show()

# =====
# 7. Interpretabilidad con SHAP
# =====
explainer = shap.TreeExplainer(model)
shap_values = explainer(X_test)


```

```
# ---- 7.1 Explicación Global ----
shap.summary_plot(shap_values, X_test, plot_type="bar") # ranking
global
shap.summary_plot(shap_values, X_test)                #
distribución de importancia

# ---- 7.2 Explicación Local ----
# Ejemplo con la primera instancia de test
i = 0
print("Ejemplo de explicación local (instancia 0):")
print("Predicción:", model.predict([X_test.iloc[i]])[0],
      "| Real:", y_test.iloc[i])

# Explicación para la clase phishing (1)
shap.plots.waterfall(shap_values[i, :, 1])
```

**SALIDA:**

 Evaluación del Modelo  
Accuracy: 0.9850  
AUC: 0.9989

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1000
1	0.99	0.98	0.98	1000
accuracy			0.98	2000
macro avg	0.99	0.98	0.98	2000
weighted avg	0.99	0.98	0.98	2000

