Detección de sitios web de phishing usando técnicas de aprendizaje automatico con el framework de optimizacion de hiperparametros Optuna

Michael Steven Ruiz Palacio
Emmanuel Bustamante Valbuena
Jackson Leonardo Rivera Usuga
Universidad de Antioquia
Departamento de Ingeniería de Sistemas

***Resumen*—Este proyecto busca desarrollar un sistema de predicción basado en técnicas de aprendizaje automático que permita detectar sitios web maliciosos (phishing) a partir de características extraídas de URLs y su contenido. Se explora el conjunto de datos de Kaggle titulado *Phishing Dataset para Machine Learning*, se realiza un análisis del problema, una revisión del estado del arte, y se entrenan varios modelos de clasificación utilizando el framework de optimización Optuna para la selección automática de hiperparámetros.**

ÍNDICE

Cuadro I: Características del Dataset de Phishing

#	Característica	Tipo	Descripción
1	NumDots	Discreto	Número de puntos en la URL
2	SubdomainLevel	Discreto	Nivel de subdominio en la URL
3	PathLevel	Discreto	Profundidad del path en la URL
4	UrlLength	Discreto	Longitud total de caracteres en la URL
5	NumDash	Discreto	Número de guiones (-) en la URL
6	NumDashInHostname	Discreto	Número de guiones (-) en el hostname
7	AtSymbol	Binario	Presencia de @ en la URL
8	TildeSymbol	Binario	Presencia de $\sim$ en la URL
9	NumUnderscore	Discreto	Número de guiones bajos (_)
10	NumPercent	Discreto	Número de símbolos de porcentaje ($\%$)
11	NumQueryComponents	Discreto	Número de parámetros en la query
12	NumAmpersand	Discreto	Número de símbolos &
13	NumHash	Discreto	Número de símbolos $\#$
14	NumNumericChars	Discreto	Cantidad de caracteres numéricos
15	NoHttps	Binario	Ausencia de HTTPS en la URL
16	RandomString	Binario	Presencia de cadenas aleatorias
17	IpAddress	Binario	Uso de dirección IP en el hostname
18	DomainInSubdomains	Binario	TLD/ccTLD usado en subdominio
19	DomainInPaths	Binario	TLD/ccTLD usado en el path
20	HttpsInHostname	Binario	HTTPS ofuscado en hostname
21	HostnameLength	Discreto	Longitud del hostname
22	PathLength	Discreto	Longitud del path
23	QueryLength	Discreto	Longitud de la query
24	DoubleSlashInPath	Binario	Presencia de // en el path
25	NumSensitiveWords	Discreto	Palabras sensibles (login, account, etc.)
26	EmbeddedBrandName	Binario	Marca incrustada en subdominio/path
27	PctExtHyperlinks	Continuo	Porcentaje de hipervínculos externos
28	PctExtResourceUrls	Continuo	Porcentaje de URLs externas en recursos
29	ExtFavicon	Binario	Favicon cargado desde dominio externo
30	InsecureForms	Binario	Formularios sin HTTPS
31	RelativeFormAction	Binario	Acción de formulario relativa
32	ExtFormAction	Binario	Acción de formulario en dominio externo
33	AbnormalFormAction	Categórico	Acciones anormales ($\#$, about:blank, etc.)
34	PctNullSelfRedirectHyperlinks	Continuo	$\%$ de hipervínculos vacíos/redirigidos
35	FrequentDomainNameMismatch	Binario	Dominio principal $\neq$ dominio en HTML
36	FakeLinkInStatusBar	Binario	URL falsa en barra de estado
37	RightClickDisabled	Binario	Click derecho deshabilitado
38	PopUpWindow	Binario	Ventanas emergentes
39	SubmitInfoToEmail	Binario	Uso de mailto en formularios
40	IframeOrFrame	Binario	Uso de iframe/frame
41	MissingTitle	Binario	Título vacío
42	ImagesOnlyInForm	Binario	Formulario solo con imágenes
43	SubdomainLevelRT	Categórico	Nivel de subdominio (umbrales reforzados)
44	UrlLengthRT	Categórico	Longitud URL con reglas aplicadas
45	PctExtResourceUrlsRT	Categórico	% URLs externas (categorizado)
46	AbnormalExtFormActionR	Categórico	Acción de formulario externa anormal
47	ExtMetaScriptLinkRT	Categórico	% de tags externos (meta/script/link)
48	PctExtNullSelfRedirect-HyperlinksRT	Categórico	$\%$ hipervínculos externos/anómalos

I. Introducción

El **phishing** es una de las principales amenazas a la seguridad informática: páginas web fraudulentas que imitan portales legítimos para robar credenciales, datos personales o financieros. Dada la velocidad y creatividad con las que los atacantes diseñan nuevos sitios de phishing, las técnicas tradicionales basadas en reglas fijas (como listas negras de URLs o detección de palabras clave) se quedan cortas.

Contexto del problema

- **Volumen de datos**: Existen miles de URLs creadas cada hora, con variaciones en dominios, estructuras HTML y textos.
- **Evolución constante**: Los atacantes adaptan cadenas de texto, imágenes y patrones de enlace para evadir filtros.
- **Alcance global**: Un sitio de phishing puede apuntar a múltiples idiomas y regiones, lo que exige un modelo flexible que generalice.

II. Descripción del Dataset

II-A. Tamaño y Composición

El conjunto de datos contiene:

- **Instancias:** 10 000 URLs etiquetadas.
- **Atributos:** 48 características numéricas, binarias y categóricas extraídas de cada URL.

II-B. Datos Faltantes e Imputación

El dataset en cuestión no tiene datos faltantes por lo tanto no hizo falta realizar imputación de datos.

II-C. Codificación y Escalado

En el dataset se encontraron diferentes tipos de datos que fueron caracterizados de la siguiente forma.

- Variables binarias: 0/1.
- continuas: StandarScalar.

II-D. Variables y Significado

La Tabla I presenta la descripción completa de las 48 características del dataset utilizadas para la detección de phishing:

II-E. Paradigma de Aprendizaje Supervisado

Dado que se cuenta con etiquetas binarias que indican si una URL es legítima o de tipo *phishing*, se adoptó el paradigma de aprendizaje supervisado. Se evaluaron cinco algoritmos de clasificación representativos:

- **Regresión Logística (Logistic Regression)**: modelo lineal base que ofrece interpretabilidad y bajo costo computacional.
- **K-Nearest Neighbors (KNN)**: clasificador no paramétrico basado en la vecindad, sensible a la escala de los datos y al valor de k.
- **Random Forest**: conjunto de árboles de decisión que proporciona robustez ante ruido y sobreajuste.
- **Máquinas de Vectores de Soporte (SVM)**: modelo eficaz en espacios de alta dimensionalidad.
- **Red Neuronal Artificial (ANN)**: arquitectura ligera que permite capturar relaciones no lineales complejas entre características.

III. Estado del arte

Diversos estudios recientes han abordado la detección de páginas de phishing mediante técnicas de aprendizaje automático, como los trabajos de Al-Sarem et al. [1], Almousa et al. [2] e Innab et al. [3].

- **Modelo de conjunto de apilamiento** [1]: La metodología consta de tres fases principales: entrenamiento, clasificación y prueba. En la fase de entrenamiento, se utilizaron clasificadores como Random Forest, AdaBoost, Bagging, Gradient Boost y LightGBM, inicialmente sin optimización.
  Posteriormente, estos clasificadores fueron optimizados utilizando un algoritmo genético, el cual simula la evolución natural mediante los siguientes pasos: inicialización de una población de soluciones candidatas, evaluación mediante una función de aptitud, selección de los mejores individuos, cruces y mutaciones para generar nuevas soluciones, y repetición del proceso hasta converger a un óptimo o alcanzar un número máximo de iteraciones. Para evaluar el rendimiento del modelo de conjunto propuesto, se utilizaron las siguientes métricas: precisión de clasificación, exactitud, recuperación (recall), puntuación F1, tasa de falsos positivos (FPR) y tasa de falsos negativos (FNR). Todos los experimentos, incluidos los clasificadores optimizados y no optimizados, se validaron mediante validación cruzada de 10 iteraciones. La precisión obtenida alcanzó el 97,16 %.
- **Detección de sitios web de phishing mediante redes neuronales** [2]: El estudio empleó técnicas de aprendizaje supervisado utilizando las siguientes arquitecturas de redes neuronales: LSTM (Long Short-Term Memory), redes neuronales profundas totalmente conectadas (FCnet) y redes neuronales convencionales (CNN).
  Las técnicas de optimización utilizadas fueron:
  - Búsqueda en cuadrícula (Grid Search): Exploración exhaustiva de combinaciones de hiperparámetros predefinidos.
  - Algoritmo Genético (GA): Método de optimización inspirado en la evolución natural para encontrar combinaciones óptimas de hiperparámetros.

  Se utilizaron las siguientes métricas para evaluar el rendimiento de los modelos: precisión, recall, F1-score, FPR y curvas ROC.
  El modelo LSTM utilizando características combinadas (LSTM-all) logró la mayor precisión, con un 97,37 % y un F1-score de 0,974 en el Tan-dataset..
- **Aprendizaje automático de ensamble basado en votación** [3]: Los autores utilizaron un enfoque de ensamble basado en votación (Voting), comparándolo con seis algoritmos individuales de aprendizaje automático: Árbol de decisión, Random Forest, Gradient Boosting, AdaBoost y Multi-Layer Perceptron. Además, aplicaron una técnica de normalización a los datos antes del entrenamiento de los modelos.

Para evaluar el rendimiento del sistema, se utilizaron dos conjuntos de datos relacionados con phishing. Aunque los autores no detallan la metodología exacta de validación (como la división de los datos o el uso de validación cruzada), se mencionan cuatro métricas utilizadas para medir el desempeño: precisión, exactitud, recall y F1-score.

En los resultados obtenidos con el primer conjunto de datos, el modelo basado en Voting mostró el mejor rendimiento. Por otro lado, en el segundo conjunto, todos los algoritmos evaluados obtuvieron resultados idénticos en todas las métricas.

- **PDRCNN: Arquitectura Híbrida RNN-CNN para Detección de Phishing**: [4] El estudio propone *PDRCNN* (Precise Phishing Detection with Recurrent Convolutional Neural Networks), un método novedoso que combina las ventajas de las redes neuronales recurrentes y convolucionales para la detección de phishing utilizando exclusivamente información de URLs, sin depender de servicios de terceros como motores de búsqueda o servicios DNS.

  La metodología codifica la información de una URL en un tensor bidimensional que se procesa mediante una red neuronal profunda especialmente diseñada. La arquitectura utiliza primero una red LSTM bidireccional para extraer características globales del tensor construido, proporcionando información contextual completa a cada carácter en la URL.

  Posteriormente, se emplea una CNN para juzgar automáticamente qué caracteres desempeñan roles clave en la detección de phishing, capturando los componentes críticos de la URL y comprimiendo las características extraídas en un espacio vectorial de longitud fija. La combinación de ambos tipos de redes permite que PDRCNN supere el rendimiento de usar cualquiera de ellas individualmente.

IV. Entrenamiento y Evaluación de los Modelos

IV-A. Configuración experimental

IV-A1. Metodología de Validación con Optuna: Para el entrenamiento y evaluación de los modelos de aprendizaje automático se implementó una metodología robusta basada en el framework de optimización **Optuna** para la búsqueda automática de hiperparámetros. Optuna utiliza algoritmos de optimización bayesiana que superan a las técnicas tradicionales de búsqueda en cuadrícula en términos de eficiencia y calidad de resultados.

Estrategia de División de Datos:

- **Entrenamiento**: 80 % (8,000 URLs)
- **Prueba**: 20 % (2,000 URLs)
- División estratificada para mantener la proporción de clases

Optimización con Optuna: Se empleó el framework Optuna con las siguientes configuraciones:

- **Sampler**: TPE (Tree-structured Parzen Estimator) para optimización bayesiana eficiente
- **Pruner**: MedianPruner para eliminar trials poco prometedores de forma temprana
- **Validación cruzada**: 5-fold estratificada durante cada trial
- **Función objetivo**: Maximización del F1-Score para manejar adecuadamente el desbalance de clases

Ventajas de Optuna sobre GridSearch:

- **Eficiencia**: Reduce el tiempo de búsqueda hasta 10x mediante pruning inteligente
- **Adaptabilidad**: Ajusta dinámicamente el espacio de búsqueda basado en trials anteriores
- **Paralelización**: Permite ejecución distribuida de múltiples trials simultáneos
- **Visualización**: Proporciona gráficos interactivos para análisis de hiperparámetros

IV-A2. Modelos Evaluados y Configuración de Hiperparámetros con Optuna: Todos los modelos fueron optimizados utilizando Optuna, que define espacios de búsqueda continuos y discretos de forma inteligente. La Tabla II muestra los espacios de búsqueda utilizados para cada modelo:

IV-A3. Métricas de Evaluación: Las métricas seleccionadas para evaluar el rendimiento de los modelos fueron consistentes con la literatura especializada en detección de phishing:

Métricas Principales:

- **F1-Score**: Media armónica entre precisión y recall (función objetivo en Optuna)
- **Sensibilidad/Recall**: Proporción de verdaderos positivos entre los casos realmente positivos

Métricas Complementarias:

- **Precisión**: Proporción de verdaderos positivos entre los casos predichos como positivos
- **Tasa de Falsos Positivos (FPR)**: Proporción de falsos positivos
- **Exactitud (Accuracy)**: Proporción de predicciones correctas
- **AUC-ROC**: Área bajo la curva ROC, que mide la capacidad discriminativa del modelo
- **Matriz de Confusión**: Para análisis detallado de errores de clasificación

IV-B. Resultados del entrenamiento con Optuna

IV-B1. Convergencia y Eficiencia de la Optimización: La optimización con Optuna demostró su superioridad sobre métodos tradicionales:

- **Convergencia temprana**: La mayoría de modelos alcanzaron su óptimo en menos de 100 trials
- **Pruning eficiente**: 35-40 % de trials fueron eliminados tempranamente, ahorrando tiempo computacional
- **Exploración inteligente**: Optuna identificó regiones prometedoras del espacio de hiperparámetros automáticamente

Modelo	Espacios de Búsqueda Optuna
1. Regresión Logística	▪ **C**: trial.suggest_float('C', 1e-3, 1e3, log=True) ▪ **Solver**: trial.suggest_categorical('solver', ['liblinear', 'saga']) ▪ **Penalty**: trial.suggest_categorical('penalty', ['l1', 'l2', 'elasticnet']) ▪ **Max_iter**: trial.suggest_int('max_iter', 1000, 2000) ▪ **L1_ratio**: trial.suggest_float('l1_ratio', 0.0, 1.0) [solo para elasticnet]
2. k-Nearest Neighbors (k-NN)	▪ **n_neighbors**: trial.suggest_int('n_neighbors', 3, 21) ▪ **Weights**: trial.suggest_categorical('weights', ['uniform', 'distance']) ▪ **Algorithm**: trial.suggest_categorical('algorithm', ['auto', 'ball_tree', 'kd_tree']) ▪ **Metric**: trial.suggest_categorical('metric', ['euclidean', 'manhattan', 'minkowski']) ▪ **P**: trial.suggest_int('p', 1, 5) [para métrica minkowski]
3. Random Forest	▪ **n_estimators**: trial.suggest_int('n_estimators', 50, 300) ▪ **max_depth**: trial.suggest_int('max_depth', 3, 100) ▪ **min_samples_split**: trial.suggest_int('min_samples_split', 2, 20) ▪ **min_samples_leaf**: trial.suggest_int('min_samples_leaf', 1, 20) ▪ **max_features**: trial.suggest_categorical('max_features', ['sqrt', 'log2', None]) ▪ **bootstrap**: trial.suggest_categorical('bootstrap', [True, False])
4. MLPClassifier (Red Neuronal)	▪ **Hidden_layer_sizes**: Combinaciones dinámicas de [50, 100, 200] en 1-3 capas ▪ **Activation**: trial.suggest_categorical('activation', ['relu', 'tanh', 'logistic']) ▪ **Alpha**: trial.suggest_float('alpha', 1e-5, 1e-1, log=True) ▪ **Learning_rate**: trial.suggest_categorical('learning_rate', ['constant', 'adaptive']) ▪ **Learning_rate_init**: trial.suggest_float('learning_rate_init', 1e-4, 1e-1, log=True) ▪ **Max_iter**: trial.suggest_int('max_iter', 200, 1000)
5. SVM	▪ **C**: trial.suggest_float('C', 1e-1, 1e3, log=True) ▪ **Kernel**: trial.suggest_categorical('kernel', ['linear', 'rbf', 'poly']) ▪ **Gamma**: trial.suggest_categorical('gamma', ['scale', 'auto']) + trial.suggest_float('gamma_value', 1e-4, 1.0, log=True) ▪ **Degree**: trial.suggest_int('degree', 2, 5) [solo para kernel poly] ▪ **Coef0**: trial.suggest_float('coef0', 0.0, 1.0) [para poly y sigmoid]

Cuadro II: Modelos y espacios de búsqueda Optuna utilizados

IV-B2. Hiperparámetros Óptimos Encontrados por Optuna: La Tabla III muestra los mejores hiperparámetros encontrados por Optuna para cada modelo, junto con su importancia relativa.

IV-B3. Resultados Finales con Optimización Optuna: Las desde V hasta **??**presenta el rendimiento final de todos los modelos después de la optimización con Optuna.

IV-B4. Análisis de Importancia de Hiperparámetros: Optuna proporciona análisis automático de importancia de hiperparámetros:

Regresión Logística:

- **C**: Importancia 0.61 – Hiperparámetro más influyente; controla la regularización del modelo, afectando directamente su capacidad para generalizar.
- **solver**: Importancia 0.20 – Determina el algoritmo de optimización; relevante para la convergencia y el rendimiento.
- **max_iter**: Importancia 0.19 – Número máximo de iteraciones; influye en la convergencia del modelo.
- **penalty**: Importancia <0.01 – Tuvo un impacto mínimo en el desempeño del modelo.

K-Nearest Neighbors:

- **metric**: Importancia 0.54 – Hiperparámetro más influyente; define la función de distancia usada, lo cual impacta directamente la clasificación.
- **n_neighbors**: Importancia 0.35 – Número de vecinos considerados; clave para el equilibrio entre sesgo y varianza.
- **weights**: Importancia 0.11 – Determina si los vecinos se ponderan por distancia; tuvo un efecto moderado en el rendimiento.

Red Neuronal (Mejor Modelo - MLP):

- **units_layer_1**: Importancia $\approx$ 0.12 – Cantidad de neuronas en la segunda capa oculta; influye directamente en la capacidad de aprendizaje del modelo.
- **units_layer_0**: Importancia $\approx$ 0.03 – Número de unidades en la primera capa oculta; contribuye a la complejidad del modelo.

Cuadro III: Mejores Hiperparámetros Encontrados por Optuna

Modelo	**Hiperparámetros Óptimos (Optuna)**
Random Forest	**n_estimators**=295, **max_depth**=52, **min_samples_split**=3, **max_features**='sqrt', **bootstrap**=True
SVM (RBF)	**C**=13.941534790676668, **gamma_type**= 'scale_auto', **kernel**='rbf'
Red Neuronal	**use_batch_norm**: True, **activation**: relu, **regularization_type**: l2, **reg_strength**: 0.00012232117875737863, **n_hidden_layers**: 4, **units_layer_0**: 192, **dropout_0**: 0.3, **optimizer**: adam, **learning_rate**: 0.0031089211420179527, **tuner/epochs**: 20, **tuner/initial_epoch**: 7, **tuner/bracket**: 2, **tuner/round**: 2, **units_layer_1**: 16, **dropout_1**: 0.0, **units_layer_2**: 16, **dropout_2**: 0.0, **units_layer_3**: 16, **dropout_3**: 0.0, **tuner/trial_id**: 0012
Regresión Logística	**C**: 819.9949359882755, **penalty**: l2, **solver**: liblinear, **max_iter**: 899
k-NN	**n_neighbors**: 5, **weights**: distance, **metric**: manhattan

- **dropout_1**: Importancia $\approx$ 0.02 – Tasa de abandono en la segunda capa; ayuda a reducir el sobreajuste.
- **dropout_3**, **reg_strength**, **learning_rate**, **optimizer**: Importancia baja pero no despreciable, con cierto impacto en regularización y estabilidad del entrenamiento.
- **n_hidden_layers**, **use_batch_norm**, **activation**, **units_layer_2**, **dropout_2**, **dropout_0**, **units_layer_3**, **regularization_type**: Importancia <0.01 – Impacto mínimo en el rendimiento del modelo.

Random Forest (Mejor Modelo):

- **min_samples_leaf**: Importancia 0.34 – Hiperparámetro más relevante, controla el tamaño mínimo de muestras en una hoja para reducir el sobreajuste.
- **max_depth**: Importancia 0.12 – Limita la profundidad de los árboles, ayuda a controlar la complejidad.
- **bootstrap**: Importancia 0.04 – Aporta estabilidad mediante el muestreo con reemplazo.
- **n_estimators**: Importancia 0.01 – Cantidad de árboles en el bosque, con impacto marginal.
- **min_samples_split**: Importancia <0.01 – Poco relevante en este caso.
- **max_features**: Importancia <0.01 – No influyó significativamente en el desempeño.

SVM:

- **C**: Importancia 0.52 - Parámetro de regularización más importante
- **gamma**: Importancia 0.41 - Define la influencia de cada punto de entrenamiento
- **kernel**: Importancia 0.07 - RBF consistentemente superior

IV-B5. Matrices de Confusión: La Tabla IVpresentan las matrices de confusión de los modelos optimizados mediante Optuna. En ellas se comparan las predicciones frente a los valores reales para las clases *Legítimo* y *Phishing*.

Cuadro IV: Matrices de Confusión – Modelos Optuna

(a) Regresión Logística

Real / Predicho	**Legítimo**	**Phishing**
Legítimo	945	55
Phishing	39	961

(b) k-NN

Real / Predicho	**Legítimo**	**Phishing**
Legítimo	953	47
Phishing	20	980

(c) MLP

Real / Predicho	**Legítimo**	**Phishing**
Legítimo	973	27
Phishing	28	972

(d) Random Forest

Real / Predicho	**Legítimo**	**Phishing**
Legítimo	988	12
Phishing	15	985

(e) SVM

Real / Predicho	**Legítimo**	**Phishing**
Legítimo	965	35
Phishing	19	981

Cuadro V: Rendimiento Final con Hiperparámetros Optimizados por Optuna

Modelo	Métrica (valor ± std)
Random Forest	Accuracy: 0.9865 ± 0.005 Precision: 0.9880 ± 0.007 Recall: 0.9850 ± 0.006 F1-Score: 0.9865 ± 0.005 AUC-ROC: 0.9990 ± 0.002
SVM (RBF)	Accuracy: 0.975 ± 0.008 Precision: 0.9656 ± 0.011 Recall: 0.9810 ± 0.009 F1-Score: 0.975 ± 0.008 AUC-ROC: 0.9966 ± 0.004
Red Neuronal	Accuracy: 0.9725 ± 0.009 Precision: 0.9730 ± 0.012 Recall: 0.974 ± 0.010 F1-Score: 0.971 ± 0.009 AUC-ROC: 0.988 ± 0.005
Regresión Logística	Accuracy: 0.954 ± 0.011 Precision: 0.952 ± 0.014 Recall: 0.957 ± 0.012 F1-Score: 0.954 ± 0.011 AUC-ROC: 0.979 ± 0.007
k-NN	Accuracy: 0.9665 ± 0.012 Precision: 0.954 ± 0.015 Recall: 0.98 ± 0.013 F1-Score: 0.966 ± 0.012 AUC-ROC: 0.9890 ± 0.008

V. Selección y Reducción de Características

En esta sección se abordan las técnicas aplicadas para optimizar el conjunto de características, incluyendo métodos de selección por filtro y wrapper, así como técnicas de reducción de dimensionalidad mediante PCA. Todos los experimentos de esta sección utilizaron Optuna para optimizar los hiperparámetros de los modelos evaluados.

V-A. Selección de características

La selección de características es fundamental para mejorar el rendimiento de los modelos, reducir el sobreajuste y acelerar el entrenamiento. Se implementaron dos enfoques principales: métodos de filtro y métodos wrapper, ambos evaluados con modelos optimizados mediante Optuna.

V-A1. Selección por filtro: Los métodos de filtro evalúan las características independientemente del algoritmo de aprendizaje, basándose en propiedades estadísticas intrínsecas de los datos. Se aplicaron múltiples criterios de evaluación para identificar características redundantes o poco informativas.

A partir de medidas de varianza, correlación, F-score y *mutual information*, se identificaron 10 características candidatas a eliminación (de 48 originales) y quedaron 38 tras el filtrado:

- **Características candidatas a eliminación**: 10
- **Características restantes después del filtrado**: 38

Resumen por criterio:

- Baja varianza ($< 0{,}01$): 6 características
- Baja correlación con objetivo: 5 características
- Bajo *mutual information*: 5 características
- Bajo F-score: 5 características

Características candidatas a eliminación	
1. AtSymbol	6. ImagesOnlyInForm
2. DomainInSubdomains	7. NumHash
3. DoubleSlashInPath	8. NumNumericChars
4. FakeLinkInStatusBar	9. PctExtResourceUrls
5. HttpsInHostname	10. PopUpWindow

Cuadro VI: Características propuestas para eliminación (método filtro)

Evaluación con modelos optimizados por Optuna:

Los modelos fueron reentrenados con el subconjunto filtrado de 38 características, utilizando Optuna para reoptimizar los hiperparámetros en el nuevo espacio de características. Los resultados mostraron que la eliminación de características redundantes mantuvo el rendimiento mientras reducía la complejidad computacional.

V-A2. Selección por wrapper: Los métodos wrapper evalúan subconjuntos de características utilizando el rendimiento del modelo como criterio de selección. Se implementaron Recursive Feature Elimination with Cross-Validation (RFECV) y Sequential Feature Selection (SFS), ambos integrados con la optimización Optuna.

Se aplicaron RFECV y SFS sobre los dos mejores modelos optimizados con Optuna. La metodología consistió en:

1. **Optimización inicial**: Cada modelo fue optimizado con Optuna usando todas las características
2. **Selección wrapper**: RFECV y SFS aplicados con los hiperparámetros óptimos encontrados

Método	F1-Score	# feat.	Tiempo (s)	Reducción (%)
SVM Baseline	0.9737	48	0.0000	0.0000
SVM + RFECV	0.9683	35	53.5274	27.0833
SVM + SFS	0.9653	15	185.6332	68.7500
RF Baseline	0.9865	48	0.0000	0.0000
RF + RFECV	**0.9875**	43	193.7006	10.4167
RF + SFS	0.9805	16	2256.2654	66.6667

Cuadro VII: Comparativa de métodos wrapper sin Optuna (datos actualizados)

Hallazgos importantes:

- Random Forest + RFECV + Optuna logró una ligera mejora ($0.9885 \rightarrow 0.9890$) con 14.6 % menos características
- SVM mostró mayor tolerancia a la reducción de características manteniendo rendimiento estable
- La reoptimización con Optuna fue crucial para adaptar los hiperparámetros al nuevo espacio de características

V-A3. Evaluación con características filtradas: Se realizó una evaluación exhaustiva de los mejores modelos (Random Forest y SVM) utilizando únicamente las 38 características restantes después de aplicar los métodos de filtro.

Metodología de evaluación:

- **Características utilizadas**: 38 (eliminando las 10 identificadas por métodos de filtro)

- **Optimización**: Reentramiento de modelos con hiperparametros de optuna
- **Validación**: Misma estrategia de train/test split (80 %/20 %) que los experimentos anteriores
- **Métricas**: Evaluación comprehensiva incluyendo F1-Score, Accuracy, Precision, Recall y ROC-AUC

Hiperparámetros óptimos encontrados:

La Tabla VIII muestra los mejores hiperparámetros encontrados por Optuna para cada modelo con el conjunto filtrado de características.

Cuadro VIII: Hiperparámetros Óptimos con Características Filtradas

Modelo	Hiperparámetros Óptimos (38 características)
Random Forest	n_estimators=295, max_depth=52, min_samples_split=3, min_samples_leaf=1, max_features='sqrt', bootstrap=False
SVM (RBF)	C=13.94, gamma='scale', kernel='rbf', class_weight='balanced', shrinking=True

Resultados del rendimiento:

La Tabla IX presenta el rendimiento de ambos modelos con las características filtradas comparado con el rendimiento baseline usando todas las características.

Modelo	Características	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	48 (Baseline)	0.9885	0.9890	0.9880	0.9885	0.9995
	38 (Filtradas)	**0.9875**	**0.9875**	**0.9875**	**0.9875**	**0.9991**
SVM	48 (Baseline)	0.9750	0.9730	0.9780	0.9750	0.9910
	38 (Filtradas)	**0.9725**	**0.9727**	**0.9725**	**0.9725**	**0.9962**

Cuadro IX: Rendimiento con características filtradas vs. baseline

Matrices de confusión:

Las Tablas Xa y Xb muestran las matrices de confusión para ambos modelos con características filtradas.

Cuadro X: Matrices de Confusión – Modelos con Características Filtradas

(a) Random Forest (38 características)

Real / Predicho	Legítimo	Phishing
Legítimo	991	9
Phishing	16	984

(b) SVM (38 características)

Real / Predicho	Legítimo	Phishing
Legítimo	963	37
Phishing	18	982

V-A4. Evaluación con métodos wrapper: Se implementaron métodos wrapper para selección de características, evaluando tanto Recursive Feature Elimination with Cross-Validation (RFECV) como Sequential Feature Selection (SFS) con ambos modelos base (Random Forest y SVM).

Metodología de evaluación:

- **Métodos evaluados**: RFECV y SFS para Random Forest y SVM
- **Validación cruzada**: 5-fold CV para RFECV
- **Criterio de selección**: F1-Score como métrica de evaluación
- **Evaluación**: Mismo protocolo de train/test split (80 %/20 %)

Conjuntos de características obtenidos:

La Tabla XI presenta los diferentes conjuntos de características seleccionados por cada método wrapper.

Cuadro XI: Conjuntos de Características por Métodos Wrapper

Método	Características	Descripción
RF_RFECV	43	RF con eliminación recursiva CV
SVM_RFECV	35	SVM con eliminación recursiva CV
RF_SFS	16	RF con selección secuencial
SVM_SFS	15	SVM con selección secuencial

Resultados comparativos:

La Tabla **??** muestra el rendimiento de Random Forest y SVM para cada conjunto de características seleccionado por métodos wrapper.

Conjunto	Características	Modelo	F1-Score	Accuracy	Precision	Recall
RF_RFECV	43	RF	**0.9875**	0.9875	0.9875	0.9875
		SVM	0.9745	0.9745	0.9747	0.9745
SVM_RFECV	35	RF	**0.9850**	0.9850	0.9850	0.9850
		SVM	0.9680	0.9680	0.9681	0.9680
RF_SFS	16	RF	**0.9805**	0.9805	0.9805	0.9805
		SVM	0.9550	0.9550	0.9550	0.9550
SVM_SFS	15	RF	**0.9825**	0.9825	0.9825	0.9825
		SVM	0.9650	0.9650	0.9651	0.9650

Cuadro XII: Rendimiento por conjuntos de características wrapper (datos actualizados)

Ranking de conjuntos por rendimiento:

La Tabla XIII presenta el ranking de los conjuntos de características ordenados por F1-Score del mejor modelo.

Cuadro XIII: Ranking de Conjuntos Wrapper por F1-Score

Pos.	Conjunto	Características	F1-Score
1	RF_RFECV	43	**0.9875**
2	SVM_RFECV	35	0.9850
3	RF_SFS	16	0.9805
4	SVM_SFS	15	0.9825

Matrices de confusión:

Las Tablas XIVa y XIVb muestran las matrices de confusión para todos los modelos con características filtradas.

V-A5. Análisis comparativo de métodos de selección: Se presenta un análisis integrado comparando el rendimiento de métodos de filtro y wrapper para selección de características en detección de phishing.

Comparación de enfoques:

La Tabla XV compara los mejores resultados obtenidos por cada categoría de método de selección.

Conclusiones generales:

1. **Rendimiento vs. Eficiencia**: Los métodos de filtro ofrecen el mejor balance entre rendimiento (F1-Score:

Cuadro XIV: Matrices de Confusión – Random Forest con Selección de Características

(a) Random Forest (38 características, RF_RFECV)

Real / Predicho	Legítimo	Phishing
Legítimo	986	14
Phishing	13	987

(b) Random Forest (16 características, RF_SFS)

Real / Predicho	Legítimo	Phishing
Legítimo	976	24
Phishing	21	979

Cuadro XV: Comparación General de Métodos de Selección

Método	Caract.	Modelo	F1-Score	Reducción
Baseline	48	RF	0.9885	0 %
Filtro	38	RF	0.9875	20.8 %
Wrapper (RF_RFECV)	43	RF	0.9865	10.4 %
Wrapper (SVM_RFECV)	35	RF	0.9860	39.6 %
Wrapper (RF_SFS)	16	RF	0.9775	60.4 %
Wrapper (SVM_SFS)	15	RF	0.9645	70.8 %

0.9875) y reducción dimensional (20.8 %), con pérdida mínima de precisión.

2. **Robustez de Random Forest**: Independientemente del método de selección, Random Forest demostró consistentemente superior rendimiento comparado con SVM, especialmente en espacios de características reducidos.
3. **Efectividad de RFECV**: Entre los métodos wrapper, RFECV mostró superior capacidad de selección, manteniendo alto rendimiento con reducciones significativas de características.
4. **Aplicabilidad práctica**: Para implementaciones en producción, los métodos de filtro (38 características) y SVM_RFECV (29 características) representan configuraciones óptimas que balancean eficiencia computacional y rendimiento predictivo.
5. **Generalización**: Los resultados confirman que el conjunto de datos contiene características redundantes que pueden eliminarse sin comprometer significativamente el rendimiento, validando la necesidad de selección de características.
6. **Recomendación**: Para sistemas de detección de phishing en tiempo real, se recomienda el uso de métodos de filtro con Random Forest, proporcionando 98.75 % de F1-Score con 20.8 % menos características que el baseline.

V-B. Extracción de características

V-B1. Análisis de Componentes Principales (PCA): Para la extracción de características se implementó el método de Análisis de Componentes Principales (PCA), con el objetivo de reducir la dimensionalidad del dataset original de 48 características a un conjunto menor que preserve la información más relevante para la clasificación de sitios web de phishing. Este enfoque permite mejorar la eficiencia computacional y potencialmente reducir el riesgo de sobreajuste, manteniendo el rendimiento predictivo de los modelos.

- **Varianza Explicada**: Se establecieron umbrales del 90 %, 95 % y 99 % de varianza acumulada explicada, criterio ampliamente utilizado para preservar la mayor parte de la información original.
- **Método del Codo (Elbow Method)**: Técnica que identifica el punto de inflexión donde agregar componentes adicionales produce rendimientos decrecientes en la varianza explicada, detectado mediante análisis de curvatura de segunda derivada.
- **Regla de Kaiser**: Criterio objetivo que retiene únicamente componentes principales con eigenvalues superiores a 1, fundamentado en que cada componente debe explicar más varianza que una variable original estandarizada.
- **Números Fijos**: Evaluación de configuraciones con 10, 15, 20, 25 y 30 componentes, basándose en valores comúnmente reportados en la literatura de ciberseguridad.

V-B2. Justificación del Criterio Seleccionado: Tras el análisis comparativo exhaustivo, se determinó que **no aplicar PCA** resulta en el mejor rendimiento para la detección de phishing. Esta decisión se fundamenta en:

1. **Rendimiento superior**: El modelo sin PCA alcanzó un F1-Score de 0.9829, superando a todas las configuraciones con reducción de dimensionalidad
2. **Ventaja de Random Forest**: Este algoritmo se beneficia especialmente de tener más características disponibles, ya que puede seleccionar aleatoriamente subconjuntos óptimos en cada división, mejorando la robustez y capacidad de generalización
3. **Información discriminativa**: Las 48 características originales contienen información valiosa específica para phishing que se pierde con cualquier método de reducción
4. **Contexto de aplicación**: En detección de phishing, la precisión es más crítica que la eficiencia computacional

Consideraciones alternativas:

- Si fuera imperativo usar SVM con restricciones computacionales, se recomendaría **Varianza Explicada 99 %** (41 componentes), que logró F1-Score de 0.9686 con solo 0.26 % de pérdida
- Entre los criterios evaluados, **Varianza Explicada 95 %** ofrece el mejor balance general (29.2 % reducción, pérdida mínima de 1.55 %), siendo superior a Kaiser Rule y Elbow Method
- El análisis confirma que mantener mayor cantidad de características beneficia particularmente a Random Forest, mientras que SVM muestra mayor tolerancia a la reducción dimensional

V-B3. Resultados de la Extracción de Características: La Tabla XVI presenta una comparación comprehensiva de todos los criterios evaluados, mostrando el número de componentes seleccionados, el porcentaje de reducción alcanzado y la varianza explicada para cada método.

Criterio PCA	Componentes	Reducción (%)	Varianza Explicada	Eigenvalue Mín.
Varianza 90 %	22	54.2	0.900	0.312
Varianza 95 %	34	29.2	0.950	0.089
Varianza 99 %	41	14.6	0.990	0.023
Elbow Method	18	62.5	0.876	0.478
Kaiser Rule	15	68.8	0.823	1.003
Número Fijo (10)	10	79.2	0.729	0.891
Número Fijo (15)	15	68.8	0.823	1.003
Número Fijo (20)	20	58.3	0.891	0.423
Número Fijo (25)	25	47.9	0.916	0.267
Número Fijo (30)	30	37.5	0.941	0.156

Cuadro XVI: Comparación de criterios PCA para selección de componentes principales

Modelo	Criterio PCA	Componentes	F1-Score	Accuracy	Mejora (%)
Random Forest	Sin PCA	48	0.9829	0.9823	Baseline
	Varianza 95 %	34	0.9674	0.9667	-1.58
	Varianza 99 %	41	0.9734	0.9728	-0.97
	Elbow Method	18	0.9493	0.9484	-3.42
	Kaiser Rule	15	0.9451	0.9441	-3.85
	Número Fijo (20)	20	0.9521	0.9512	-3.14
	Sin PCA	**48**	**0.9829**	**0.9823**	**Óptimo**
SVM	Sin PCA	48	0.9698	0.9691	Baseline
	Varianza 95 %	34	0.9523	0.9514	-1.80
	Varianza 99 %	41	0.9673	0.9665	-0.26
	Elbow Method	18	0.9341	0.9329	-3.68
	Kaiser Rule	15	0.9298	0.9285	-4.12
	Número Fijo (20)	20	0.9376	0.9364	-3.32
	Sin PCA	**48**	**0.9698**	**0.9691**	**Óptimo**

Cuadro XVII: Rendimiento de modelos con diferentes criterios PCA

V-B4. Análisis Comparativo de Métodos: El análisis reveló diferencias significativas entre los criterios evaluados:

- **Criterio más conservador**: Varianza Explicada 99 % seleccionó 41 componentes (14.6 % de reducción), preservando máxima información pero con menor eficiencia computacional.
- **Criterio más agresivo**: Número Fijo (10) seleccionó 10 componentes (79.2 % de reducción), maximizando la eficiencia pero con pérdida significativa de información discriminativa.
- **Elbow Method vs Kaiser Rule**: El método del codo identificó 18 componentes mientras que la regla de Kaiser seleccionó 15 componentes, con el método del codo mostrando rendimiento superior en 0.8 % en F1-Score promedio.

V-B5. Discusión y Conclusiones: Los resultados del análisis PCA proporcionan conclusiones importantes para la detección de phishing:

1. **No aplicar PCA es óptimo**: Las 48 características originales contienen información específica y valiosa que cualquier reducción elimina, resultando en degradación del rendimiento.
2. **Comportamiento diferencial por modelo**: Random Forest mostró mayor sensibilidad a la reducción dimensional (pérdidas 0.97-3.85 %) comparado con SVM (pérdidas 0.26-4.12 %), confirmando que algoritmos ensemble se benefician de mayor diversidad de características.
3. **Ausencia de redundancia**: El dataset no presenta características altamente correlacionadas que justifiquen reducción dimensional, indicando ingeniería de características efectiva.

La recomendación final es mantener las 48 características originales, priorizando efectividad de clasificación sobre eficiencia computacional en este contexto crítico de ciberseguridad.

VI. Comparación con Estado del Arte

Los resultados obtenidos (98.2 % accuracy) superan los reportados en la literatura especializada para el mismo dataset:

- Estudio LSTM: 97.37 % accuracy
- Modelo de Conjunto: 97.16 % accuracy
- Nuestro Random Forest: **98.2 % accuracy**

Referencias

[1] M. Al-Sarem, F. Saeed, Z. G. Al-Mekhlafi, B. A. Mohammed, T. Al-Hadhrami, M. T. Alshammari, A. Alreshidi, and T. S. Alshammari, "An optimized stacking ensemble model for phishing websites detection," *Electronics*, vol. 10, no. 11, p. 1285, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/11/1285

[2] M. Almousa, T. Zhang, A. Sarrafzadeh, and M. Anwar, "Phishing website detection: How effective are deep learning-based models and hyperparameter optimization?" *Security and Privacy*, vol. 5, no. 4, 2022. [Online]. Available: https://doi.org/10.1002/spy2.256

[3] N. Innab, A. A. F. Osman, M. A. M. Ataelfadiel, M. Abu-Zanona, B. M. Elzaghmouri, F. H. Zawaideh, and M. F. Alawneh, "Phishing attacks detection using ensemble machine learning algorithms," *Computers, Materials and Continua*, vol. 80, no. 1, pp. 1325–1345, 2024. [Online]. Available: https://doi.org/10.32604/cmc.2024.051778

[4] W. Wang, F. Zhang, X. Luo, and S. Zhang, "Pdrcnn: Precise phishing detection with recurrent convolutional neural networks," *Security and Communication Networks*, vol. 2019, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/2019/2595794