TÍTULO

TÉCNICAS DE APRENDIZAJE AUTOMÁTICO PARA LA DETECCIÓN DE PHISHING DE URL

AUTORA

Gretel Alonso Graverán

Esta edición electrónica ha sido realizada en 2024

Tutor: Dr. D. Antonio J. Tallón Ballesteros

Instituciones: Universidad Internacional de Andalucía , Universidad de Huelva

Curso: *Máster Universitario en Economía, Finanzas y Computación (2022/23)*

Fecha documento: 2023

“TÉCNICAS DE APRENDIZAJE AUTOMÁTICO PARA LA DETECCIÓN DE PHISHING DE URL”

por

GRETEL ALONSO GRAVERÁN

Tesis presentada en conformidad con los requisitos del
Máster en Economía, Finanzas y Computación.

Universidad de Huelva & Universidad Internacional de Andalucía

Septiembre de 2023

"TÉCNICAS DE APRENDIZAJE AUTOMÁTICO PARA LA DETECCIÓN DE PHISHING DE URL"

Gretel Alonso Graverán
Máster en Economía, Finanzas y Computación

Supervisado por: **Dr. Antonio J. Tallón Ballesteros**
Universidad de Huelva

Universidad Internacional de Andalucía, 2024

Abstract

With the significant increase in the use of the Internet and online services in recent times, mainly by governments and financial institutions, the risk of phishing attacks has been constantly increasing. Consequently, protection against security threats online becomes very important. In this context, this work focuses on the detection of phishing to prevent the leakage of private information and economic losses, through the implementation of multiple supervised machine learning techniques. This study is based on the KDD methodology for the search for the most effective learning model to discriminate between legitimate and fraudulent URLs, which includes data preparation, data imbalance management, model optimization and feature engineering. and furthermore, it is complemented with an exploratory analysis of the data. Finally, it is obtained that the Extra Trees algorithm (19 attributes + hyperparameters by default), with an accuracy of 97,15 %, an AUC-ROC of 97,0 %, Sensitivity of 95,48 % and Negative Predictive Value of 96,38 % constitutes the optimal model, as it guarantees adequate discrimination between classes and offers a better balance between efficiency and precision.

Keywords : phishing, URL, supervised machine learning, classification algorithms, decision tree.

Resumen

Con el aumento significativo del uso de Internet y los servicios en línea en los últimos tiempos, principalmente por parte de gobiernos e instituciones financieras, el riesgo de ataques de *phishing* ha ido en constante aumento, por lo que la protección contra las amenazas a la seguridad en línea se vuelve de vital importancia. En este contexto, este trabajo se enfoca en la detección de *phishing* para prevenir la filtración de información privada y las pérdidas económicas, mediante la implementación de múltiples técnicas de aprendizaje automático supervisado. Este estudio se sustenta en la metodología del KDD para la búsqueda del modelo de aprendizaje más efectivo para discriminar entre URLs legítimas y fraudulentas, que incluye la preparación de datos, el manejo del desbalance de datos, la optimización de modelos y la ingeniería de características, y además, se complementa con un análisis exploratorio de los datos. Finalmente, se obtiene que el algoritmo Extra Trees (19 atributos + hiperparámetros por defecto), con una exactitud del 97,15 %, un AUC-ROC de 97,0 %, Sensibilidad del 95,48 % y Valor Predictivo Negativo de 96,38 % constituye el modelo óptimo, pues garantiza una discriminación adecuada entre clases y ofrece un mejor equilibrio entre eficiencia y precisión.

Palabras Claves: phishing, URL, aprendizaje automático supervisado, algoritmos de clasificación, árbol de decisión.

Agradecimientos

...

A la Universidad Internacional de Andalucía y la Universidad de Huelva, en particular a sus profesores, por la oportunidad de adentrarme en un nuevo campo profesional.

A la Fundación Atlantic Copper, por el soporte que ofrecen para la formación de latinoamericanos.

A mi tutor Antonio J. Tallón Ballesteros, por la guía y revisión detallada.

A mi familia, que a través de la distancia me acompaña en cada paso.

A Rey, por las nociones de ciberseguridad.

A todo el que me animó, confió y preguntó alguna vez qué tal iba el TFM.

Índice General

Índice de Figuras

1. Introducción

1.1. Fundamentación de la elección de la problemática del Phishing

Es un hecho que el Internet se ha vuelto una parte imprescindible en la vida de las personas, por lo que actualmente no se concibe el desarrollo de múltiples actividades sin esta forma de interacción, abarcando las comunicaciones, la adquisición de información, las compras, las interacciones sociales y las actividades de oficina,etc.

Esta interacción propicia que exista en la nube gran cantidad de información personal de los usuarios, como nombres, contraseñas, preguntas de seguridad, información personal y datos de tarjetas de crédito. Esta situación es aprovechada por los ciberdelincuentes, que utlilizan diversos medios ilegales para captar esta información y, posteriormente, suplantar la identidad de estos usuarios para llevar a cabo actividades ilícitas.

Este tipo de actividad fraudulenta se denomina *"phishing"*, realizando una analogía de los ataques con la "pesca" de víctimas. Por su actualidad y las consecuencias de su puesta en práctica, es muy abordada en diversos estudios en los últimos años. También resulta una técnica prometedora y atractiva para los atacantes, llamados *"phishers"* [29].

En el informe trimestral de cierre de 2022 del Grupo de Trabajo Anti-*phishing* (APWG), asociación sin fines de lucro enfocada en eliminar el robo de identidad y los fraudes de esta naturaleza, refleja que el pasado fue un año récord para el *phishing*, registrando más de 4,7 millones ataques [11]. La evolución de este fenómeno en los últimos años se puede observar en el gráfico 1.

Desde principios de 2019, el número de ataques de *phishing* ha crecido más del 150 % por año, siendo el sector financiero el más afectado representando el 27.7 % del total [11], como se constata en la figura 2.

Como resultado, varias empresas de comercio electrónico, instituciones financieras e individuos han sufrido pérdida masiva de información confidencial e importantes pérdidas financieras.

Figura 1: Tendencia de ataques Phishing (2019-2022)

Fuente: APGW, 2023 [11]

Las siguientes estadísticas, publicadas en el informe de la firma de servicios profesionales BDO publicado a inicios del actual año, ”Las 10 principales amenazas para la ciberseguridad en 2023”, ofrecen una medida del impacto económico que causa este problema. Se ha determinado que el coste promedio de una filtración de datos en Estados Unidos es de 9,4 millones de dólares (8,77 millones de euros) [3].

En el caso de los ataques de *phishing* o suplantación de identidad en la web, las pérdidas pueden alcanzar los 17.700 dólares (16.510 euros) por cada minuto que dura el ataque. Además, el estudio revela que el 25 % de las filtraciones empresariales se producen a través de correos corporativos falsos que tienen carácter interno, siendo este el origen de la mayoría de amenazas de ciberseguridad [3].

En este contexto de múltiples amenazas y riesgos relacionados con la suplantación de identidad, y con el objetivo de lograr un desarrollo rápido y sólido de Internet ha sido fundamental buscar constantemente alternativas que garanticen un entorno de red seguro. En consecuencia, la puesta en práctica de mecanismos para la detección oportuna de estas amenazas, constituye una tarea prioritaria, la cual se ha abordado desde diferentes perspectivas, en correspondencia con el entorno cambiante en el cual ocurre.

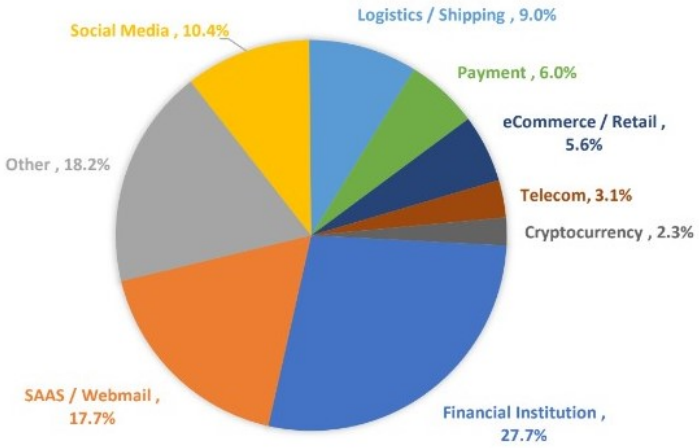


Figura 2: Ataques de Phishing por sectores

Fuente: APGW, 2023 [11]

A partir de lo expuesto anteriormente, se verifica que el enfrentamiento al *phishing* es una problemática vigente, de importancia vital en todos los ámbitos en los cuales se interactúa con Internet, y que justifican todos los esfuerzos que a nivel práctico e investigativo se realizan para su prevención y solución oportuna, siendo esta la motivación de este estudio.

Este estudio se enfoca en explorar posibles soluciones para detectar el *phishing* en las URLs utilizando diversas tecnologías. Específicamente, se pone énfasis en la aplicación de técnicas de clasificación, mediante algoritmos de aprendizaje automático, que es una tarea ampliamente utilizada en la minería de datos.

1.2. Objetivos del trabajo

En línea con lo anterior, el objetivo general de este trabajo es identificar el modelo de clasificación de aprendizaje automático que detecte, de forma más precisa y eficiente, las URLs que suplantan la identidad de sitios web y aquellas que son legítimas, guiado por la metodología del KDD y considerando múltiples características relacionadas con las URLs y el sitio web, en el conjunto de datos utilizado.

Con esa finalidad, se trazan los siguientes objetivos específicos:

1. Realizar una aproximación, desde el punto de vista teórico, a la problemática del *phishing* mediante URLs.

2. Identificar y describir los algoritmos que refieren estudios previos para tratar este problema.

3. Realizar un análisis exploratorio, identificando patrones y particularidades del conjunto de datos empleados.

4. Determinar las acciones de preprocesamiento que requiere el conjunto de datos para obtener una solución óptima.

5. Aplicar algoritmos de clasificación de aprendizaje automático y determinar, a partir de las métricas de evaluación, el modelo óptimo.

Para cumplimentar los objetivos planteados se siguió la metodología de Descubrimiento de Conocimiento en Bases de Datos (KDD, por sus siglas en inglés), enfocado en la detección de patrones y relaciones en conjuntos de datos. En este caso, y como parte de la metodología anterior, se ha puesto énfasis en la Selección de Características y en la aplicación de técnicas de Minería de Datos enfocados en herramientas de Aprendizaje Automático, específicamente métodos de Clasificación.

1.3. Estructura del trabajo

Luego de esta introducción, el resto del documento está organizado de la siguiente manera: en la sección 2 se analiza la teoría y estado del arte en torno a la utilización de la URL y el sitio web para realizar ataques de *phishing*, los diferentes métodos que se han implementado para su detección, así como los resultados y metodologías empleadas por estudios previos, relacionados con el empleo de métodos de aprendizaje automático.

En la sección 3 se proporciona una descripción general de los principales aspectos teóricos relativos al aprendizaje automático como parte del proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD), con énfasis en los algoritmos que serán puestos en práctica en la problemática que se analiza en este trabajo.

En la sección 4 se realiza un análisis exploratorio de los datos, como complemento de la implementación del KDD en el conjunto de datos empleado. En la sección 5 se describe la metodología que se seguirá para alcanzar el objetivo general de este trabajo. Los experimentos a realizar se detallan en la sección 6, mientras que en la 7, se discuten los resultados obtenidos. Por último, se presentan las conclusiones obtenidas y se señalan posibles direcciones para investigaciones futuras en la sección 8.

2. Conceptos y estado del arte sobre el phishing

La ciberseguridad protege las redes, los activos digitales y los dispositivos de accesos no autorizados o usos ilícitos. De acuerdo con los métodos y formas de los ataques a la red, los problemas de ciberseguridad se dividen principalmente en ataques de denegación de servicio (DoS), *man-in-the-middle* (MitM), inyección de SQL, explotación de día cero, tunelización de DNS, *phishing* y categorías de *malware* [34].

Estos ataques están dirigidos principalmente a las siguientes áreas: fraude, falsificación, fuerza, extorsión, piratería, bloqueo de servicios, aplicaciones de malware y contenidos digitales ilegales [15].

Centrándonos en el *phishing* mediante enlaces webs, este constituye una forma de ataque a la red que combina ingeniería social y tecnología informática. Este tipo de ciberdelincuencia consiste en un acto de engaño a los usuarios para que hagan clic en enlaces, enviados mediante correos electrónicos, otras páginas web, SMS o mensajes de redes sociales [34].

Aunque estas páginas tienen interfaces gráficas de usuario similares, tienen la particularidad de poseer localizadores uniformes de recursos (URL) diferentes de la página original, es decir, la identidad del sitio legítimo ha sido suplantada, mediante el uso de imágenes y texto similares a los oficiales, con la finalidad de obtener información confidencial del usuario y, en última instancia, usar los datos del usuario para falsificar el inicio de sesión, para robar fondos u otras acciones ilícitas [34].

En determinados casos, un usuario cuidadoso podría detectar estas páginas web maliciosas analizando minuciosamente las URL. Sin embargo, debido a la dinámica de la vida, la mayoría de las veces, los usuarios finales no investigan la dirección completa de su página web activa [29].

Por otra parte, los ataques de *phishing*, cada vez más, emplean una variedad de técnicas, como la manipulación de enlaces, la evasión de filtros, la falsificación de sitios web, la redirección encubierta y la ingeniería social, haciendo más complicada su detección [30].

Con el objetivo de lograr a través del engaño que los usuarios revelen sus credenciales de acceso o datos confidenciales, los atacantes hacen uso de técnicas de ingeniería social, generando un entorno de manipulación psicológica. Generalmente utilizan el correo electrónico u otro medio de comunicación que sugiere urgencia, provocando miedo o emociones similares en la víctima, lo que lleva a esta a no cuestionar la legitimidad del mensaje y develar información sensible, hacer clic en un enlace malicioso o abrir un archivo malicioso [13].

En la figura 3 se muestra el ciclo de vida del *phishing*.

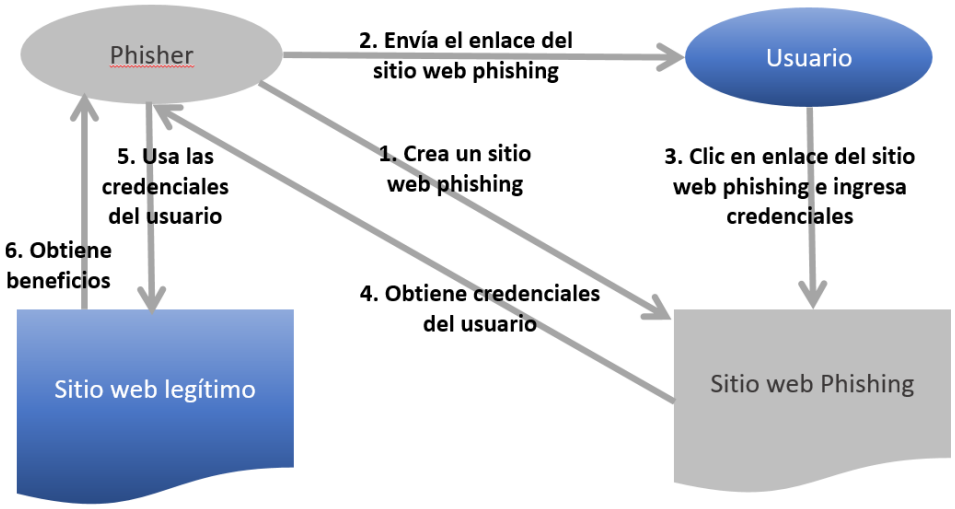


Figura 3: Ciclo de vida del Phishing

Fuente: [34]

La intención del atacante de realizar un ataque de *phishing* tiene diferentes objetivos, como son la venta de la identidad de las víctimas, obtención de un rescate, explotación de vulnerabilidades en el sistema u obtener beneficios financieros [25].

Un estudio relacionado con las experiencias de los usuarios con los ataques de *phishing* [35], se refleja que son vulnerables a este tipo de ataque debido a cinco razones principales:

- Los usuarios no tienen un conocimiento detallado sobre las URL.
- Los usuarios no saben en qué páginas *web* se puede confiar.

- Los usuarios no ven la dirección completa de la página *web*, debido a la redirección o a las URL ocultas.
- Los usuarios no tienen mucho tiempo para consultar la URL, o ingresan accidentalmente a algunas páginas *web*.
- Los usuarios no pueden distinguir las páginas *web* de *phishing* de las legítimas.

Con el uso intensivo de internet y los servicios en línea en los últimos tiempos, especialmente por parte de los gobiernos e instituciones financieras, el riesgo de ataques de *phishing* no ha dejado de aumentar.

Informes de empresas dedicadas a la ciberseguridad estiman un incremento del 47,2 % en los ataques de *phishing* en 2022 en comparación con el año anterior [5]. En el gráfico siguiente se puede constatar la estimación de la evolución del *phishing* 4.

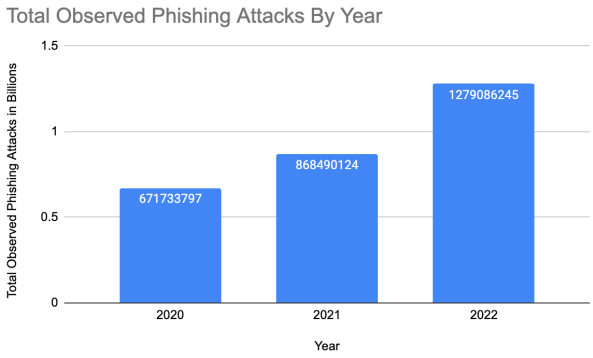


Figura 4: Total de ataques Phishing por año [5]

El mismo informe refiere que Estados Unidos, el Reino Unido, los Países Bajos, Rusia y Canadá fueron los cinco países que sumaron más ataques. Basado en el análisis de 280.000 millones de transacciones diarias y 8.000 millones de ataques diarios bloqueados. Además, concluye que existen sectores más propensos a sufrir ataques de este tipo, como se observa en la siguiente figura.

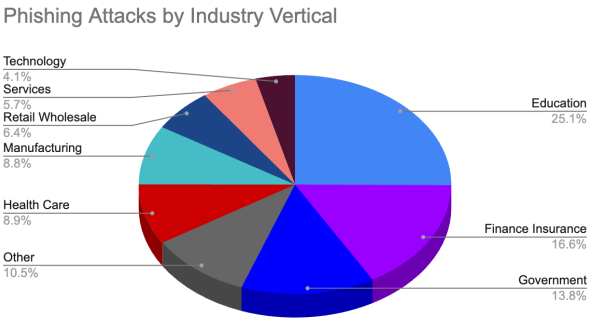


Figura 5: Ataques de Phishing por industria [5]

Al ser la seguridad de la red una cuestión de ataque y defensa, que cambia rápidamente, las variantes de *phishing* y, en consecuencia, la tecnología para su detección se están actualizando constantemente.

Debido a la evolución en la forma de interactuar en la red, se observan variaciones en los tipos de ataques de *phishing*, que van más allá de enlaces falsificados y sitios web fraudulentos. Estas alternativas utilizan *phishing* por mensajes de texto SMS (*SMiShing*), emplean la voz para atraer a las víctimas y para que abran archivos adjuntos maliciosos (*vishing*) [34], el fraude de correo electrónico dirigido a ejecutivos (*whaling*), el *phishing* a través de la redirección de los usuarios a un sitio falso (*pharming*), el *phishing* basado en el Localizador de Recursos Uniforme (URL maliciosas) y la utilización de códigos de respuesta rápida o QR (QRishing) [13].

Otra variante denominada "*spear phishing*", constituye un ataque personalizado, pues recurre a la investigación previa de las víctimas para que la estafa parezca más auténtica, por lo cual resulta ser uno de los tipos de ataque más exitosos contra los usuarios de las redes de datos [13].

2.1. Estructura de los Localizadores Uniformes de Recursos (URLs)

El presente trabajo discute el enfoque basado en el Localizador Uniforme de Recursos (Uniform Resource Locator) para la detección de *phishing*. En consecuencia, a continuación, se profundiza en la descripción de los componentes de las URLs.

Las URLs constituyen las direcciones única para acceder a sitios web a través de la World Wide Web (WWW); son creadas para localizar un recurso como páginas de hipertexto, imágenes, archivos de audio, etc. Según el estándar W3C, una URL básica presenta un formato bien definido, que se representa en la figura 6 y que consiste en el protocolo, el subdominio, el nombre de dominio, el puerto, la ruta, la consulta, los parámetros y el fragmento:

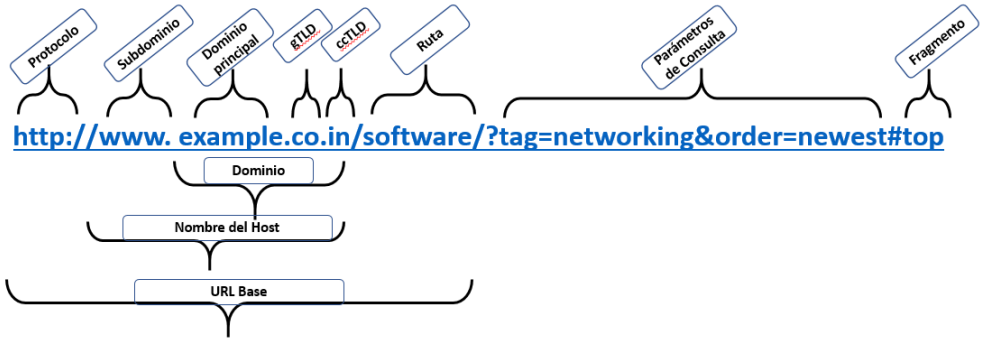


Figura 6: Estructura de la URL

El **protocolo** se utiliza para acceder al recurso y define cómo el navegador web debe comunicarse con el servidor web. Algunos de los protocolos más comunes son HTTPS (Protocolo seguro de transferencia de hipertexto), FTP (Protocolo de transferencia de archivos), POP (Protocolo de oficina postal), SMTP (Protocolo simple de transferencia de correo) e IMAP (Protocolo de acceso a mensajes de Internet) [12].

El **nombre de host** se subdivide en subdominio, dominio principal y dominio de nivel superior (TLD, por sus siglas en inglés). El dominio principal y el dominio de nivel superior juntos representan el nombre de **dominio** de la URL, que es una referencia única que identifica un sitio web en Internet [25]. El dominio de nivel superior se subdivide en TLD genéricos (gTLD) y TLD específicos de país (ccTLD). El nombre de host va seguido de un número de puerto, que es un campo opcional [25].

El número de **puerto** indica el tipo de comunicación entre cliente y servidor. No todas las URLs lo incluyen explícitamente en su estructura, solo se incluyen cuando se requiere un puerto diferente al predeterminado para un protocolo en particular.

La **ruta** se refiere a una ubicación única donde existe un archivo o directorio en un servidor web e identifica el recurso específico dentro del dominio solicitado por el usuario. La ruta va seguida de dos campos opcionales, consulta y fragmento. La **consulta** siempre va precedida de un signo de interrogación (?) y se encuentra generalmente en páginas web dinámicas [12]. El **fragmento** va precedido de una almohadilla (#) [25].

Un *phisher* puede modificar diversas partes de una URL legítima, para crear una URL fraudulenta y engañar a los usuarios, dirigiéndolos a sitios web falsos[25]:

Dominio: El *phisher* tiene la capacidad de cambiar el dominio principal de la URL, lo que proporcionaría una dirección web diferente que se parece mucho al sitio web legítimo que intentan imitar. Cualquier nombre de dominio que aún no haya sido registrado por otra persona puede ser registrado por el atacante. Sin embargo, este elemento de la URL solo se puede personalizar una vez. Para crear nuevas URL, el *phisher* puede cambiar el nombre de host y la ruta en cualquier momento; sin embargo, una vez que un dominio se marca como fraudulento, es fácil identificarlo.

Subdominio: El *phisher* podría utilizar subdominios para crear una apariencia más creíble.

Protocolo: El phisher puede modificar el protocolo utilizado en la URL, por ejemplo, cambiar 'https://' (seguro) a 'http://' (no seguro).

Ruta: Un *phisher* podría agregar rutas para simular la estructura de directorios de un sitio web legítimo o incluso crear páginas ficticias que se asemejen a las de un sitio real.

Parámetros de consulta: Algunas URL incluyen parámetros de consulta que transmiten información al servidor. Un *phisher* podría manipular estos parámetros para obtener datos sensibles de los usuarios o redirigirlos a páginas maliciosas.

Fragmento de URL: El fragmento de URL se utiliza para indicar una ubicación específica en una página web. Aunque generalmente no se utiliza para dirigir a un sitio diferente, un *phisher* podría modificarlo para hacer que la URL se vea más auténtica.

2.2. Diferentes aproximaciones para la detección de phishing en sitios web

A partir de los elementos descritos en el apartado anterior se verifica la complejidad de esta problemática, así como la necesidad de crear constantemente mecanismos de prevención, identificación y defensa, ante los múltiples riesgos de suplantación. Estas soluciones se han clasificado como convencionales, basada principalmente en el entrenamiento de los usuarios y alternativas automatizadas, que se sustentan en la creación de softwares de detección [13].

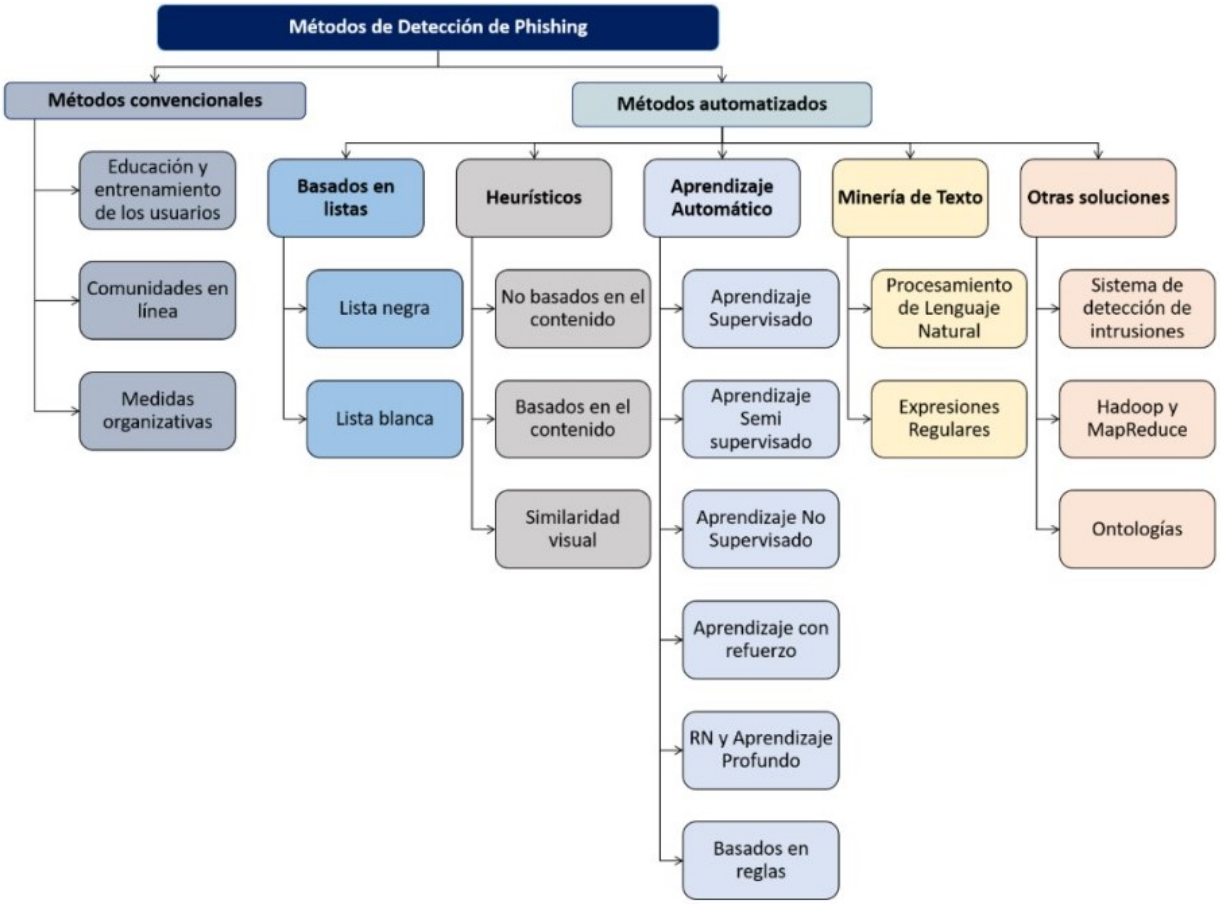


Figura 7: Mecanismos para la detección de phishing

Fuente: [13]

2.2.1. Mecanismos convencionales

Contituyen métodos enfocados en la formación de los usuarios, para detectar este tipo de ataques, utilizando para ello un entorno de entrenamiento integrado con situaciones reales (por ejemplo, SecurityIQ-PhishSim, Gophish, LUCY, etc.). Otras soluciones son basadas en la experiencia del usuario, lo que ha permitido la creación de comunidades en línea como Anti-*Phishing* (APWG, PhishTank, Millersmilesc, Symantecd, entre otros), las cuales tienen como función general monitorizar y denunciar las actividades de *phishing* recientes para los diferentes grupos de interés [13].

2.2.2. Mecanismos automatizados

Confección de listas

Es el enfoque tradicional para identificar enlaces de *phishing* que se basan en la elaboración de listas negras y listas blancas, lo cual es muy efectivo para evitar la reutilización de la misma URL de sitios web de *phishing*, lo que conlleva una reducción en el número de usuarios afectados y las pérdidas asociadas. Estas técnicas son ampliamente empleadas como medidas defensivas en tiempo real, ya que requieren muy poco tiempo computacional al utilizar algoritmos de coincidencia de una sola cadena[34], presentando un bajo porcentaje de falsos positivos. Dos de las bases de datos importantes proveedoras de listas son PhishTank y Google Safe Browsing (GSB) [8].

Dado que la vida media de la página web de *phishing* reportada es de menos de veinticuatro horas, la velocidad de sincronización y la frecuencia de las actualizaciones de la lista de denegación son factores cruciales para defenderse contra este tipo de amenaza [4].

Sin embargo, esta aproximación tiene como limitante la incapacidad de identificar nuevos enlaces fraudulentos. Por lo tanto, algunos usuarios pueden ser atacados antes de que el enlace malicioso sea agregado a una lista negra [29].

Enfoque basado en reglas

Otros mecanismos para identificar los enlaces ilegítimos se basan en la introducción de reglas, diseñadas manualmente por expertos en seguridad cibernética, en función de la estructura de la URL y/o su código fuente [34], que buscan identificar contenido engañoso, redirecciones poco usuales, mensajes de alerta falsos, solicitud de información sensible, etc.

Este enfoque presenta el inconveniente de que cuando las reglas se publican, los *phishers* las aprenden y logran sortearlas, produciendo nuevos tipos de ataques y sitios web sofisticados.

Métodos heurísticos

La solución de clasificación heurística es un software que se puede instalar en computadoras cliente o servidores y se basa en la detección de patrones y firmas de ataques conocidos para identificar posibles amenazas de seguridad. Los enfoques para detectar URLs falsas se dividen comúnmente en tres tipos: el enfoque no basado en contenido, el enfoque basado en contenido y el enfoque basado en similitud visual [13].

En el caso del enfoque basado en contenido, la página web se analiza para generar características discriminatorias que puede incluir textos, imágenes y elementos interactivos, para identificar indicios de intentos de *phishing*. Ejemplo de estas características son formularios de captura de datos, la redirección inusual, el contenido de alerta falso, verificación de la autenticidad del certificado SSL, etc. y su extracción requiere, por tanto, de muchos recursos [8].

La técnica de *phishing* basada en la similitud visual aplica técnicas de análisis visual para comparar y contrastar la apariencia de sitios web, basándose en la idea de que los sitios de *phishing* a menudo intentan imitar sitios web legítimos [34]. Con este fin se extraen características como etiquetas HTML, CSS, logotipo de imagen para encontrar la similitud. Si la similitud entre los sitios web sospechosos y los sitios web legítimos supera cierto umbral, el sitio web sospechoso se clasifica como *phishing*. La desventaja de un esquema basado en la similitud visual es que no puede detectar sitios web de *phishing* recién lanzados [12] [14].

Minería de texto

Mediante el análisis de patrones sospechosos que incluyen, entre otros, el contenido de correos electrónicos, sitios web, URL, mensajes instantáneos, entre otros, se han aplicado cuatro tipos de técnicas: la Frecuencia de Término - Frecuencia Inversa de Documento, las Expresiones Regulares, el Modelado de Temas usando Análisis Semántico Latente y el Modelo de Memoria Distribuida de Vectores de Párrafo [13].

Otras técnicas emergentes

Se identifican ontologías y Sistemas de Detección de Intrusos. Además, se ha propuesto la utilización de Hadoop, así como de las principales ventajas que proporciona la técnica de MapReduce para el procesamiento de los datos y la selección de rasgos que serán utilizados en la detección de *phishing*. [13].

Métodos de aprendizaje automático

Otra de las aproximaciones para enfrentar el problema ha sido resolver la identificación del *phishing* en sitios web por medio de tareas de clasificación. El desarrollo de la tecnología de aprendizaje automático ha posibilitado predecir si un enlace emergente es un sitio web de *phishing*, así como la mejora en la precisión de la predicción de este tipo de eventos.

Considerando que los atacantes encuentran vulnerabilidades constantemente, en un entorno muy complejo y dinámico, se recomienda utilizar modelos híbridos en lugar de un enfoque único por parte del administrador de seguridad de las redes [29].

2.3. Estudios relacionados

En esta sección se analizan estudios previos, relacionados con el empleo de métodos de aprendizaje automático para la detección de URLs fraudulentas; se describe la metodología y herramientas empleadas en estas investigaciones, así como los resultados más relevantes obtenidos.

Sahingoz et al. [29], en 2018 propusieron un sistema anti-*phishing* que se caracteriza por la independencia de idioma, uso de un gran conjunto de datos, ejecución en tiempo real e independencia de servicios de terceros. Utilizaron siete algoritmos de aprendizaje automático, y diferentes tipos y cantidades de atributos basados en Procesamiento de Lenguaje Natural, vectores de palabras y características híbridas, concluyendo que la construcción de una lista de características eficiente es una tarea crucial para aumentar la precisión del sistema de detección. Su propuesta tiene como limitante el hecho de no dectectar ataques con dominios cortos y subdominios sin ninguna ruta.

Shahrivari et al. [30], en 2020 evaluaron doce clasificadores en el conjunto de datos de sitios web de *phishing* que consta de 6157 sitios web legítimos y 4898 sitios web de *phishing*, obteniendo un buen buen desempeño con relación a la duración y la precisión del cálculo en clasificadores de ensamblaje, tales como Random Forest y XGBoost. Los autores apuntan que no hay garantía de que la combinación de múltiples clasificadores siempre funcionará mejor que el mejor clasificador individual en los clasificadores de conjunto, sin embargo, en este caso se han obtenido buenos resultados. Se plantean como desafíos futuro combinar modelos de aprendizaje automático con otras técnicas de detección de *phishing*, lo cual podría mejorar el rendimiento obtenido.

Aljofey et al. [1], en 2020 implementaron un modelo de detección de *phishing* mediante el uso de una red neuronal convolucional (CNN) a nivel de caracteres, el cual no requiere de características diseñadas manualmente y es independiente del acceso a la red. El modelo tiene tiempo de entrenamiento bastante largo y un bajo tiempo de respuesta, lo cual resulta conveniente del lado del usuario. El modelo propuesto logró una precisión del 95,02 % en el conjunto de datos utilizado y en promedio 96.42 % en conjuntos de datos de referencia. Además, presenta los inconvenientes de clasificar erróneamente si la URL es corta o contiene palabras claves como ’iniciar sesión’ o ’registrado’. Como metas se plantean la extracción de características del código y del texto de la página web.

Rao et al. [25], en 2020 crearon una aplicación, CatchPhish, que predice la legitimidad de la URL sin visitar el sitio web, que se caracteriza por utilizar características hechas a mano, basadas en código fuente, señalando como particularidad el poder manejar las descargas ocultas, limitando el

riesgo de infección por un malware y la no utilización de servicios de terceros. Utiliza el nombre de host, la URL completa, las funciones de Frecuencia de término-Frecuencia de documento inversa (TF-IDF) y palabras sugerentes de *phishing* para la clasificación, utilizando el clasificador de bosque aleatorio. El modelo propuesto logró una precisión significativa del 94,26 % en su conjunto de datos.

Sahingoz et al. [15], en 2020 propusieron el uso de 48 características extraídas sin acudir a a servicios de terceros y relacionadas con el nombre de host, dominio y ruta. Trabajaron con un conjunto de más de 126.000 instancias, donde el clasificador Random Forest obtuvo la precisión más alta.

Wei et al. [36], en 2020 proponen el uso de una red neuronal profunda con capas convolucionales, analizando únicamente el texto de la URL y afirman alcanzar cerca del 100 % de precisión. Emplean un método rápido, que detecta ataques desde el día cero y presentan una red optimizada para que pueda utilizarse incluso en dispositivos móviles sin afectar, de forma apreciable, su rendimiento.

Gandotra & Gupta [8], en 2021 utilizan 30 características de un conjunto de datos que contiene 4898 instancias etiquetadas como *phishing* y 6157 páginas web benignas y comparan diferentes algoritmos de aprendizaje automático sobre la base de su rendimiento, con y sin selección de características. Lo resultados refieren una mejor precisión del clasificador bosque aleatorio tanto antes como después de la selección de características. El empleo de técnicas de selección de características, particularmente el método de Ganancia de Información, propicia una mejora importante en el tiempo de construcción del modelo, sin comprometer su precisión.

Gupta et al. [12], en 2021 desarrollaron un enfoque de detección de *phishing* que solo necesita nueve características léxicas para detectar ataques de *phishing* de manera efectiva, teniendo en cuenta que gran cantidad de atributos requiere grandes poderes de procesamiento para detectar el *phishing*, lo que constituye una limitante para dispositivos con recursos limitados. Estas características también tienen la particularidad de no requerir servicios de terceros. Utilizaron un conjunto de datos con 11964 instancias de URL, probando con diferentes clasificadores de aprendizaje

automático y obteniendo una precisión elevada con el algoritmo de bosque aleatorio.

Butnaru et al. [4], en 2021 presentan una combinación de atributos validados por la literatura, que no utiliza recursos de terceros, así como otros atributos extraídos mediante técnicas de procesamiento de lenguaje natural y que se comparan, en términos de frecuencia, con un vocabulario sugerente. Además, incluye características que miden la similitud del dominio y el subdominio de la URL, con los dominios top legítimos del dataset, utilizando la distancia de Levenshtein. Utiliza 100.000 URL que se recopilaron en abril de 2020 y con una división de 80/20 entre datos de entrenamiento y prueba, respectivamente. El rendimiento fue evaluado a lo largo del tiempo con un conjunto de datos que consta de ataques de *phishing* activos y lo compararon con Google Safe Browsing (GSB), que es el control de seguridad predeterminado en los navegadores web más populares, obteniendo en promedio un 94 % de precisión comparación con el 45 % logrado por GSB.

Awasthi & Goel [2], en 2022 implementaron modelos utilizando varios clasificadores base y otros basados en conjuntos. Utilizaron dos conjuntos de datos, de forma independiente y fusionados, que poseen 30 características. Experimentaron con metodologías de evaluación del rendimiento sin validación cruzada y con validación cruzada, en este último caso variando la cantidad de pliegues (*folds*). Los resultados apuntan a que se logran mejores resultados con los clasificadores ensamblados y evaluando el rendimiento con validación cruzada empleando 10 *folds*. Los autores se plantean como reto futuro llevar a cabo la selección de características.

A partir de la revisión anterior se constata que la literatura incluye una amplia variedad de trabajos que emplean el aprendizaje automático para mitigar el *phishing* y en particular demostraron ser una herramienta poderosa para detectar patrones en los datos para la identificación de este tipo de ataque mediante rasgos relacionados con la URL o el sitio web.

Así mismo, se evidencia que la eficiencia en la detección de phishing va a depender de los algoritmos empleados, de los tiempos de entrenamiento de los modelos y particularmente de las características a incluir.

Para la selección de características se deberá valorar si se utilizará únicamente la URL o si se

requerirá la descarga del código fuente del sitio web junto con la URL. Este enfoque permite obtener información sobre los hipervínculos utilizados que pueden usarse como características para clasificar entre sitios web legítimos y de *phishing*. Sin embargo, esta alternativa tiene como inconveniente que algunos phishers intentan incrustar el código malicioso dentro del código, por un lado, y por otro, incrementan el tiempo de extracción de características [12].

Los estudios apuntan que la detección de la página web de *phishing* en tiempo real es esencial para la prevención de este tipo de ataques, teniendo en cuenta que construir una página web fraudulenta es una tarea rápida, fácil y económica, y que esta puede estar activa en un corto periodo (incluso por unas pocas horas). En consecuencia, el empleo de la URL únicamente aporta rapidez en el sistema de detección [29]. Este enfoque presenta como limitante la clasificación incorrecta de ataques en caso de URLs cortas, formadas únicamente por dominio y subdominio.

Los sistemas de detección también deberán estar preparados para enfrentar ataques de día cero, es decir, no deberán depender únicamente de sistemas basados en identificación mediante listas blancas y negras [29], sino que requerirán una combinación de mecanismos de detección.

Otro aspecto a evaluar al seleccionar los atributos del sistema será el nivel de dependencia de servicios de terceros para la identificación de determinadas características de la URL, dígase la utilización de registros WHOIS, listas negras/listas blancas basadas en la web, páginas de clasificación, medidas de tráfico de red, la edad del dominio, entre otras.

Por una parte, las alternativas que requieren de servicios de tercero consumen más ancho de nada, lo que sería un elemento a valorar cuando el sistema de detección se diseña para dispositivos con recursos limitados [12]. Por otra parte, el acceso a recursos de tercero implica un *trade-off* entre precisión y eficiencia del sistema de detección, en cuanto a la rapidez del mismo para detectar ataques en tiempo real [29]. En consecuencia, la elección de un modelo óptimo deberá pasar por seleccionar un conjunto apropiado de características para construir los modelos de clasificación y extraer las características en el menor tiempo posible sin comprometer la precisión.

Los algoritmos más utilizados en la experimentación [2, 4, 29, 30] son Regresión logística, Gaussian

Naïve Bayes, Árbol de decisión, Support Vector Machine, k-NN, Linear Discriminant Analysis. Bagging, Adaboost, Gradient Boosting, Voting, Extra Trees, XGBoost, Neural Network, Naive Bayes y además se ha aplicado novedosos enfoques para la detección de phishing con Redes Neuronales Convolucionales. Entre los modelos anteriores, aquellos que han destacado por su mejor rendimiento son Árbol de Decisión, Neural Network, Random Forest, Extra Trees y XGBoost.

En el apartado siguiente se detallan, desde el punto de vista teórico, una selección de los algoritmos mencionados anteriormente, para su posterior implementación en el conjunto de datos que se analizará en este trabajo.

3. El paradigma del aprendizaje automático como parte del proceso de KDD

En el apartado siguiente se resumen los principales aspectos teóricos relativos al aprendizaje automático como parte del proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD, por sus siglas en inglés), enfatizando en los algoritmos que serán puestos en práctica en la problemática que se analiza en este trabajo. Se considera pertinente mencionar algunos términos que se utilizarán como sinónimos, para una mejor comprensión del documento:

-Variables de entrada, atributos, rasgos, variables independientes, características, predictores.
-Variables de salida, respuesta, predichas o dependientes.
-Instancias, observaciones, ejemplos o tuplas.
-Variables cualitativas o categóricas.
-Variables cuantitativas o numéricas.

3.1. El proceso de Descubrimiento de Conocimiento en Bases de Datos

El Knowledge Discovery in Databases (KDD) surge en 1989 y es un proceso no trivial de revelar información novedosa, comprensible y potencialmente útil. En otras palabras, constituye una metodología multidisciplinar, cuya finalidad es descubrir patrones, relaciones y conocimientos relevantes a partir de un conjunto de datos, en el cual extraer un patrón implica el ajuste de un modelos a los datos, encontrar una estructura a partir de datos o realizar cualquier descripción de alto nivel de un conjunto de datos [7].

Este concepto ha ido evolucionando y actualmente se refiere a todo el proceso de extracción de conocimiento, almacenamiento y acceso a los datos, desarrollo de algoritmos eficientes y escalables [28], que se pueden usar para analizar conjuntos de datos masivos, e incluye interpretación y visualización de los resultados, etc.

El KDD comprende una serie de pasos interrelacionados, los cuales se ejecutan para transformar los datos sin procesar en información y conocimiento procesable y son esenciales para garantizar

el conocimiento que se deriva de los datos [7]. Es importante destacar que todo modelado comienza con la comprensión del conocimiento del dominio, lo que evidencia que es un proceso guiado por la teoría [31].

Seguidamente, se describen las etapas que lo conforman (ver figura 8):

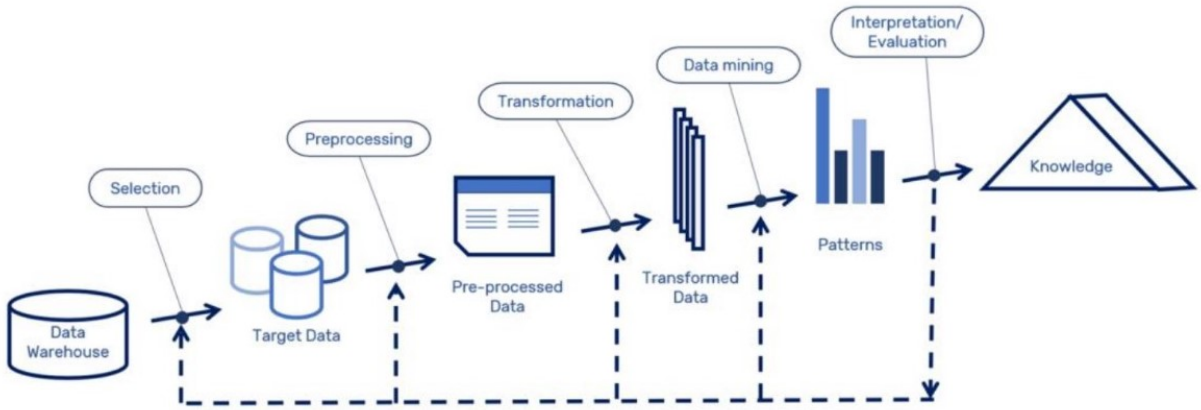


Figura 8: Arquitectura del proceso de KDD

Fuente: Fayyad et al. (1996) [7].

Selección de Datos

En esta etapa, se eligen los datos necesarios para el proceso de descubrimiento de conocimiento, en función del dominio de aplicación y del objetivo del proceso de KDD. Los datos pueden provenir de una o diversas fuentes, y generalmente se centran en un subconjunto de variables o muestras de datos [7].

Preprocesamiento de Datos

En esta etapa, los datos son sometidos acciones de preprocesamiento. Estas modificaciones pueden estar relacionadas con la integración y transformación de datos e incluir la integración de múltiples fuentes de datos, normalización, discretización, estandarización, transformación de atributos. La limpieza de datos se encarga del manejo del ruido, problemas de datos faltantes, duplicación de datos y eliminación de datos irrelevantes [7, 23].

Transformación de Datos

Constituye un campo amplio en el cual se realizan transformaciones más avanzadas para mejorar la calidad y la representación de los datos, agregando los mismos, creando características nuevas o reduciendo la dimensionalidad [7].

La reducción de datos se encarga principalmente de la selección de características, la compresión de datos, el muestreo y la selección de datos/instancias. Por otra parte, el aprendizaje de atributos se centra en el aprendizaje de relaciones de atributos (correlación y dependencias de atributos y relaciones causales entre variables), interacción de características, clasificación de características y generación de atributos [23].

Minería de Datos

Es la etapa fundamental del proceso, en la cual se implementa análisis de datos y métodos de minería de datos para revelar patrones, relaciones y tendencias en los datos. En esta etapa del proceso se seleccionan algoritmos de minería de datos en función de los objetivos del proceso, que pueden ser de verificación de una hipótesis o de descubrimiento de patrones [7].

La minería de datos integra técnicas de varios campos como estadística, matemáticas, aprendizaje automático e inteligencia artificial, bases de datos, reconocimiento de patrones, visualización de datos, optimización y economía [21, 23], nutriéndose de una variedad de algoritmos que se agrupan en clustering, clasificación, regresión, reglas de asociación, que son utilizados para análisis, inferencia y/o predicción [24].

Evaluación de Patrones

Para evaluar los patrones descubiertos en la etapa de minería se considera si los patrones son interesantes, comprensibles y si tienen sentido en el contexto del problema.

Lo anterior implica que se requiera definir medidas cuantitativas para evaluar los patrones extraídos. Por ejemplo, medidas de certeza, como la precisión, o de utilidad, como el beneficio, o de comprensibilidad que es más subjetivo y, en algunos contextos, se puede medir en términos de simplicidad [7].

Interpretación y Visualización

En esta etapa los resultados obtenidos se interpretan y se relacionan con el conocimiento previo del dominio, siendo la visualización una opción para presentar los patrones significativos de forma más comprensible. Es posible aplicar estas técnicas en diferentes etapas del proceso [23].

Utilización del Conocimiento

El proceso finaliza con la utilización del conocimiento revelado para tomar decisiones, hacer predicciones, resolver conflictos potenciales con creencias previas, generar recomendaciones, trazar futuras líneas de investigación o realizar acciones concretas para resolver problemas del mundo real [7].

Es importante destacar que el KDD no es un proceso lineal estricto, sino un ciclo iterativo e interactivo, lo cual significa que a medida que se obtiene conocimiento y se aplican acciones, pueden surgir nuevos datos y problemas, lo que lleva a nuevas iteraciones del proceso para mejorar y refinar el conocimiento descubierto [7].

En resumen, el proceso de KDD implica una serie de pasos interconectados que permiten convertir datos brutos en información valiosa y conocimiento útil que puede ser utilizado para tomar decisiones en diversos campos y aplicaciones.

3.2. Fundamentos del aprendizaje automático

El campo del aprendizaje automático se encarga del desarrollo de algoritmos y modelos estadísticos que utilizan los sistemas informáticos para realizar una tarea específica sin ser programados explícitamente [17]. Estos algoritmos se utilizan para diversos fines, como la detección

automatizada de patrones significativos en los datos, el procesamiento de imágenes, el análisis predictivo, etc.

Este paradigma tiene dos vertientes principales [31], el aprendizaje no supervisado, donde no existen variables de salida y tiene un enfoque descriptivo y el aprendizaje supervisado, que son las técnicas en las cuales se centra este trabajo y que se detallan a continuación.

El aprendizaje supervisado es la tarea de aprendizaje automático que consiste en inferir una función que mejor aproxime la relación entre un conjunto de variables de entrada con una variable respuesta o salida [31]. Este aprendizaje se realiza a partir de un conjunto de ejemplos, es decir, de datos de entrenamiento etiquetados, donde las variables de entrada y las variables de salida son conocidas [17]. La finalidad de esta tarea es hacer predicciones sobre instancias futuras, a partir de los modelos generales producidos [22].

A propósito de los tipos de datos, estos métodos pueden manejar tanto atributos numéricos como categóricos. Las variables de salida a predecir pueden ser cuantitativas y, en ese caso, nos encontramos ante un problema de regresión. Cuando las variables de salida son categóricas, entonces estamos ante una tarea de clasificación, donde los posibles valores de dichas variables determinan las clases del problema de clasificación.

Este trabajo se centra en algoritmos de clasificación, que constituye una técnica de aprendizaje automático que se utiliza para predecir la pertenencia a una clase de instancias de datos.

La clasificación se realiza en dos pasos. El primero consiste en la construcción de un modelo basado en un conjunto de datos de entrenamiento al cual se le aplica un algoritmo de clasificación. En el segundo paso, el modelo determinado anteriormente utiliza un conjunto de prueba para clasificar una tupla desconocida en una etiqueta de clase y de esta forma, medir el rendimiento y la precisión del modelo entrenado [21].

3.3. Algoritmos de aprendizaje automático supervisado para Clasificación

En el apartado siguiente se realiza una revisión, desde el punto de vista teórico, de algunas de las principales técnicas de aprendizaje automático supervisado, centradas en la tarea de Clasificación, que según los trabajos previos, han obtenido buenos resultados en la problemática estudiada, cuestiones que sentarán las bases para su comprensión y aplicación en el conjunto de datos que se analizará posteriormente en este trabajo.

3.3.1. Decision Tree

Los modelos de arboles de decisión constituyen una estructura jerárquica que toma decisiones basadas en reglas lógicas y condiciones. Cada árbol consta de nodos, ramas y hojas. Se inicia con la división del conjunto de datos de entrenamiento en dos subconjuntos, basados en la búsqueda del atributo y el punto de corte (umbral) de ese mismo atributo, que mejor separe las clases. Dicha selección se puede realizar tomando como referencia dos medidas: la Entropía, para buscar la división que proporcione la mayor ganancia de información o el Índice de Gini, que busca garantizar la menor impureza en el nodo de decisión [31].

El proceso anterior, se denomina División Binaria Recursiva, y se repite para cada nuevo subconjunto que se va formando hasta que se crea todo el árbol (la profundidad del árbol alcanza el valor máximo) o se cumple el criterio de parada establecido.

Los criterios de parada que se pueden establecer son disímiles y se relacionan con la profundidad del árbol, el número mínimo de ejemplos por hoja y por división, umbral de impureza mínima, número máximo de hojas, de nodos y de características.

Si el nodo no se divide más, se llega al resultado de clasificación, es decir, a la hoja donde se indica la clase correspondiente, que es la que aparece con mayor frecuencia entre las instancias que se agrupan en esa hoja. Finalmente, se utiliza el árbol final para hacer para predicciones, siguiendo el valor de los atributos del ejemplo a clasificar [17].

Con vista a simplificar el modelo y mejorar su capacidad de generalización, se pueden eliminar

ramas del árbol que no aportan mucha información o pueden llevar al sobreajustese (poda). En este caso la decisión se toma en función de la tasa de error en un conjunto de validación.[32].

Las versiones clásicas de este algoritmo incluyen el ID3, C4.5, C5.0, creados por Ross Quinlan. El C4.5 constituye una mejora del algoritmo ID3, que permite el manejo de datos de entrenamiento con valores faltantes de atributos, así como el empleo de características discretas y continuas [32]. El Árbol de Decisión CART (Classification and Regression Trees), creado por Breiman, Friedman, Olshen y Stone, puede utilizarse tanto para clasificación como para regresión, puede manejar atributos categóricos y numéricos, no maneja valores faltantes y es menos propenso al sobreajuste en comparación con ID3/C4.5. Principalmente diseñado para problemas de clasificación es el algortimo Árbol de Decisión CHAID (Chi-Squared Automatic Interaction Detection), que utiliza la prueba de chi-cuadrado para medir la relación entre atributos categóricos y la variable objetivo.

De forma general los algoritmos basados en árboles han ido evolucionando mediante transformaciones que mejoran la capacidad para manejar una variedad de problemas y producen modelos de más alta precisión, reduciendo el sobreajsute, como son el Árbol de Decisión Random Forest, el Extra Trees (Extreme Randomized Trees) y el XGBoost (Extreme Gradient Boosting), que se explican más adelante.

3.3.2. Support Vector Machine

El algoritmo Máquina de Soporte Vectorial (SVM, por sus siglas en inglés) constituye una buena herramienta para problemas de clasificación, especialmente en espacios de alta dimensión y cuando se busca una buena generalización [10].

Esta técnica puede emplear distintos tipos de kernel en función de de la naturaleza de los datos y el problema. El kernel es una función matemática que transforma los datos de entrada en un espacio dimensional superior y puede ser lineal y otras no lineales como: polinomial, radial (RBF), sigmoide, etc.[10].

El algoritmo identifica el hiperplano (una superficie en un espacio n-dimensional) que mejor separa las clases en el espacio transformado por el kernel, maximizando el margen entre las clases.

El margen es la distancia entre el hiperplano y los puntos de datos más cercanos de cada clase o vectores de soporte [32].

Para determinar el equilibrio entre el error y la capacidad de generalización se controla el tamaño del margen, es decir, la tolerancia a los errores de clasificación, empleando el parámetro C. Valores más altos de este parámetro penalizan más los errores, lo que puede resultar en un margen más estrecho, pero una clasificación más precisa en el conjunto de entrenamiento [32].

Por otra parte, SVM es sensible a la escala de las características, por lo que puede requerir la normalización de datos y la elección del kernel adecuado puede requerir experimentación.

3.3.3. Random Forest

El algoritmo bosque aleatorio (Random Forest) toma como base el método de árbol de decisión y se centra en reducir el problema de sobreajuste. Para ello, construye múltiples árboles de decisión a partir del conjunto de datos y combina los resultados de votación de los árboles individuales, para obtener una predicción final más precisa [6].

La construcción de cada árbol se realiza a partir de muestras del conjunto de entrenamiento original, seleccionados aleatoriamente y con reemplazo, lo que se denomina *'Bootstrap Aggregating' o 'Bagging'*.

Para dividir cada nodo de decisión se selecciona la característica a partir de un subconjunto aleatorio de características, lo cual introduce aleatoriedad en la construcción del árbol y reduce la correlación entre los árboles. De este subconjunto de características seleccionadas aleatoriamente se utiliza algún criterio como la ganancia de información o la impureza de Gini para obtener la mejor [31].

En línea con lo anterior, este método proporciona una medida de la importancia de las características, pues evalúa cuánto varía la precisión del modelo cuando se elimina una característica específica, siendo una particularidad de este algoritmo. Adicionalmente, trata valores faltantes y atípicos, maneja grandes conjuntos de datos con características complejas y no lineales, por lo cual es

considerado un método muy versátil [31].

3.3.4. Extremely Randomized Trees

El clasificador ExtraTrees (Extremely Randomized Trees) se basa en la técnica de conjuntos y constituye una variante del Random Forest. Para la construcción de cada árbol, al igual que Random Forest, toma una muestra aleatoria con reemplazo para cada árbol y considera una selección aleatoria de características [9].

A diferencia de Random Forest, en Extra Trees se agregan más niveles de aleatoriedad, pues en cada nodo de decisión, no se calcula ninguna métrica de importancia de características, sino que, tanto la característica, como su umbral se eligen de forma aleatoria [9].

Esta particularidad de ser extremadamente aleatorio hace que este clasificador presente más resistencia al sobreajuste que Random Forest. En cambio, puede requerir un mayor número de árboles para alcanzar un rendimiento similar. Así mismo, el proceso de construcción de árboles es más rápido, al no calcular la importancia de características en cada nodo de decisión. También el algoritmo deviene particularmente útil en conjuntos de datos pequeños, ruidosos o con alta dimensionalidad [9].

3.3.5. Gradient Boosting

Se basa en la idea de mejorar iterativamente el rendimiento del modelo, ajustando modelos débiles, generalmente árboles de decisión, de forma secuencial, en función de los errores cometidos por los modelos anteriores.

El algoritmo parte de la selección de un modelo base, llamado *weak learner*, y se ajusta a los datos de entrenamiento. Posteriormente se calcula el error que el modelo base no pudo capturar, comparando las predicciones del modelo base con las etiquetas reales [20].

A continuación, se ajusta un segundo modelo a los residuos calculados en el paso anterior, con el objetivo de capturar parte del error que quedó después del primer modelo. Se actualizan las predicciones del modelo totalizando las predicciones del primer modelo y las del segundo modelo,

lo cual reduce el error global. Se repite todo el proceso anterior hasta que se alcance un cierto criterio de detención. Las predicciones finales son el resultado de la suma de las predicciones de todos los modelos débiles ajustados en cada iteración [20].

Presenta la ventaja del manejo eficiente de datos ruidosos y problemas de alta dimensionalidad y la limitante de ser propenso al sobreajuste si no se controlan adecuadamente los hiperparámetros. Algunas implementaciones de Gradient Boosting son XGBoost (Extreme Gradient Boosting), LightGBM y CatBoost.

3.3.6. Extreme Gradient Boosting

XGBoost significa 'Extreme Gradient Boosting' y sigue el enfoque de Gradient Boosting que mejorar el rendimiento de los modelos al combinar múltiples modelos débiles en uno fuerte.

Su funcionamiento es similar al método Gradient Boosting, comienza con la elección, entrenamiento y predicción de un modelo base, generalmente un árbol de decisión. A continuación, se calcula el error que el modelo base no pudo capturar, comparando las predicciones iniciales del modelo base y las etiquetas reales del conjunto de entrenamiento.

Para reducir el el error global, se suman las predicciones del primer modelo y las del segundo modelo. El proceso anterior se repite iterativamente, entrenando un nuevo árbol de decisión con base en los errores calculados anteriormente. Las predicciones finales se obtienen sumando las predicciones de todos los árboles de decisión ajustados en cada iteración.

La diferencia principal radica en que el XGBoost utiliza técnicas de regularización, como la poda y la limitación de la profundidad de los árboles, para prevenir el sobreajuste.

Entre las características que distinguen a este clasificador es que puede trabajar con atributos categóricos sin necesidad de convertirlos en variables numéricas, permite el empleo funciones de pérdida personalizadas para adaptarse a diferentes tipos de problemas y puede manejar los valores faltantes de forma automática.

Adicionalmente, permite controlar la profundidad máxima de los árboles, para evitar la construcción de árboles excesivamente complejos y puede detener el entrenamiento cuando ya no mejora el rendimiento. Finalmente, aprovecha el procesamiento en paralelo en varios núcleos para entrenar modelos más rápido y calcula ganancias en la construcción de árboles, lo que mejora el rendimiento y la velocidad.

3.4. Técnicas de validación de modelos

Las técnicas de validación de modelos constituyen enfoques generales utilizados para dividir los datos, entrenar y probar el modelo, evaluar su rendimiento y estimar su capacidad de generalización. Se eligen valorando las características específicas del conjunto de datos (tamaño, balance de clases, etc.) y del modelo. A continuación se describen las técnicas principales:

El Hold-out es el enfoque más simple y consiste en dividir el conjunto de datos en dos partes, entrenamiento y prueba. Esta técnica puede ser menos precisa en la estimación del rendimiento, particularmente si el conjunto de datos es pequeño.

La técnica de validación cruzada consiste en dividir el conjunto de entrenamiento en 'k' pliegues mutuamente excluyentes. Mientras una de las partes se utiliza en el proceso de prueba, las otras se utilizan al proceso de entrenamiento. Finalmente, se promedian los resultados para obtener una estimación más robusta del rendimiento del modelo [16].

Esta técnica presenta como ventaja la utilización de todo el conjunto de datos para entrenar y evaluar el modelo, lo que ayuda a reducir la variabilidad en la estimación del rendimiento. Al realizar el proceso de entrenamiento y evaluación 'k' veces, es más costoso computacionalmente [31].

A partir de esta técnica se derivan otras. Entre ellas se encuentra la validación cruzada estratificada, en la cual los datos se dividen de forma que cada partición tenga una distribución similar de clases (una proporción similar de ejemplos de cada clase), lo cual resulta útil cuando se tienen clases desbalanceadas en el conjunto total.

El Leave-one-out (LOO) constituye una variante extrema de la validación cruzada, donde cada observación se utiliza como conjunto de prueba. Es costosa computacionalmente, y se utiliza cuando el conjunto de datos es pequeño y requiere la estimación más precisa de la tasa de error de un clasificador [16].

La validación cruzada con reemplazo (*Bootstrapping*) selecciona múltiples muestras aleatorias, con reemplazo, del conjunto de datos y se entrena y evalúa el modelo en cada una de estas muestras. Finalmente, los resultados se promedian para obtener una estimación del rendimiento.

3.5. Métricas de evaluación del rendimiento de modelos

Las métricas que se describen a continuación, constituyen medidas cuantitativas utilizadas para evaluar y comparar el desempeño de un modelo, y se obtienen comparando los valores pronosticados con las etiquetas reales en el conjunto de prueba [16].

La mayoría de las métricas empleadas parten de los resultados de la matriz de confusión, que para un problema de clasificación binaria, se estructura como sigue 9:

		Clase Real Positiva (P)	**Clase Real** Negativa (N)
Clase Predicha	Positiva (P)	Verdaderos Positivos (TP)	Falsos Positivos (FP)
Clase Predicha	Negativa (N)	Falsos Negativos (FN)	Verdaderos Negativos (TN)

Figura 9: Matriz de Confusión

donde:

- TP (True Positive) es el número de ejemplos positivos correctamente clasificados.
- TN (True Negative) es el número de ejemplos negativos correctamente clasificados.
- FP (False Positive) es el número de ejemplos negativos incorrectamente clasificados como positivos.

- FN (False Negative) es el número de ejemplos positivos incorrectamente clasificados como negativos.

A continuación, se detallan las métricas que permiten evaluar diferentes aspectos del rendimiento de un clasificador binario y pueden proporcionar información valiosa sobre cómo está funcionando el modelo en términos de capacidad de detección, exactitud y enfoques combinados.

Exactitud (Accuracy): es la relación entre las predicciones correctas y las predicciones totales. En los casos de clasificación binaria, se sabe que la selección aleatoria tiene un 50 % de precisión. En conjuntos de datos desequilibrados, no siempre una alta precisión significa que el modelo sea bueno, por lo que constituye una métrica generalmente utilizada cuando las clases están balanceadas. Un valor cercano a 1 indica que el modelo está haciendo buenas predicciones en general.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Con el fin de evaluar la capacidad del modelo para detectar correctamente, las clases, es decir los verdaderos positivos (TP) y negativos (TN), se utilizan principalmente las dos métricas siguientes:

Sensibilidad (Recall o Tasa de Verdaderos Positivos): Mide la proporción de verdaderos positivos en relación con todos los positivos reales en el conjunto de datos. Resulta más relevante cuando el costo de los falsos negativos es alto. Un alto recall indica que el modelo está capturando la mayoría de los casos positivos.

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \tag{2}$$

Especificidad o Tasa de Verdaderos Negativos (True Negative Rate): mide qué tan bueno es el modelo para predecir resultados negativos.

$$\text{Especificidad} = \frac{TN}{TN + FP} \tag{3}$$

Las dos métricas descritas a continuación se enfocan en evaluar la precisión de las predicciones cuando el modelo detecta una clase en particular, ya sea positiva o negativa:

Precisión: es el porcentaje que ha identificado correctamente el modelo como positivos, del total de datos predichos como positivos por el modelo. Resulta más relevante cuando el costo de tener falsos positivos es alto. Una alta precisión indica que hay pocos falsos positivos.

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{4}$$

Valor Predictivo Negativo : Mide la proporción de predicciones negativas que son verdaderamente negativas. Cuanto más alto sea el valor predictivo negativo, menos falsos negativos tendrá el modelo cuando predice la clase negativa.

$$\text{Valor Predictivo Negativo} = \frac{TN}{TN + FN} \tag{5}$$

Cuando, tanto el costo de los falsos positivos como el de los falsos negativos es importante, es necesario una métrica que combine esta información:

F1-Score: Calcula la media armónica ponderada de la precisión y la sensibilidad. Es una métrica particularmente útil cuando el desequilibrio entre las clases es importante.Un valor alto de F1 indica que el modelo tiene tanto una alta precisión como un alto recall, lo que significa que está clasificando correctamente tanto las instancias positivas como las negativas.

$$\text{F1-Score} = \frac{2 \cdot \text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \tag{6}$$

Por otra parte, el coeficiente kappa de Cohen (**Cohen's kappa**) calcula la concordancia entre las predicciones del modelo y las etiquetas verdaderas, teniendo en cuenta la proporción de acuerdo que podría ocurrir al azar. Es una métrica útil para evaluar el rendimiento del modelo teniendo en cuenta el acuerdo que se esperaría por casualidad. Un valor cercano a 1 indica un alto grado de concordancia más allá de lo esperado por azar.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{7}$$

donde:

kappa: Representa el símbolo de kappa.

Po: Es la proporción observada de acuerdo entre el modelo y las etiquetas verdaderas.

Pe: Es la proporción esperada de acuerdo al azar.

Otra métrica muy utilizada para evaluar el rendimiento de un modelo de clasificación binaria es la **Curva ROC** (Receiver Operating Characteristic), la cual es una representación gráfica que refleja la relación entre la Sensibilidad (Tasa de Verdaderos Positivos) en el eje Y y la medida 1-Especificidad (Tasa de Falsos Positivos) en el eje X, en función de diferentes umbrales de clasificación.

El área que se encuentra bajo la Curva ROC (**AUC-ROC**) mide la capacidad del modelo para distinguir entre clases.

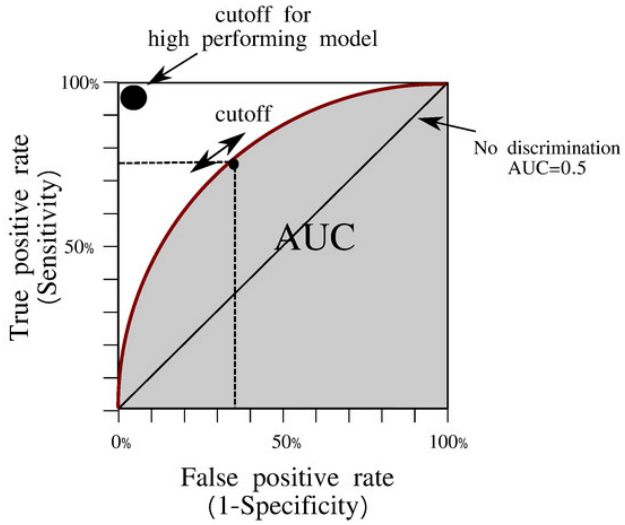


Figura 10: Área bajo la Curva ROC

Fuente: [27].

Un modelo perfecto tendría un área bajo la curva ROC (AUC-ROC) de 1, lo cual significa que es capaz de lograr una alta Sensibilidad y baja Tasa de Falsos Positivos en todas las circunstancias. Un modelo con un AUC de 0.5 sería inútil, ya que su rendimiento sería equivalente al azar. Finalmente, un modelo con AUC entre 0.5 y 1 evidencia una determinada capacidad de discriminación, donde un AUC-ROC más cercano a 1 indica un mejor rendimiento.

En resumen, en este capítulo se abordó el marco teórico que ofrece la metodología del KDD para el tratamiento de datos, profundizando en las técnicas de minería de datos que se pondrán en práctica en este trabajo, mediante la implementación de algoritmos de aprendizajes automático supervisado para hacer frente a tareas de clasificación, en correspondencia con el objetivo de este trabajo.

4. Análisis exploratorio de los datos

En este punto, se tienen como referencia el estado del arte a propósito de la detección de enlaces web ilegítimos, así como el marco teórico que ofrece la metodología KDD (Knowledge Discovery in Databases), con énfasis en la aplicación de técnicas de aprendizaje supervisado, específicamente en tareas de clasificación, descritas en los apartados anteriores. Con base en estos elementos nos encontramos en situación de ponerlos en práctica en el conjunto de datos seleccionado. En consecuencia, este apartado se centra en la realización de un análisis exploratorio del conjunto de datos.

Los datos que se utilizan pertenecen al campo de la Seguridad Cibernética, en correspondencia con la finalidad de este proyecto, que es identificar de forma más eficiente y precisa si la dirección de un sitio web está siendo utilizada para suplantar la identidad de otro.

En este apartado se describe la fuente del conjunto de datos, así como sus particularidades en cuanto a tipos de atributos, número de observaciones, análisis de correlación entre variables, aspectos que permitirán comprender mejor los experimentos a los que se someterán los datos, así como los resultados que se obtengan.

4.1. Origen del conjunto de datos

Un desafío importante ha sido identificar un conjunto de datos adecuado para las aplicaciones de aprendizaje automático. No es habitual que las empresas divulguen sus datos, por lo que es una práctica común utilizar repositorios en línea para realizar dichos análisis.

Los datos que se analizan en este trabajo provienen del repositorio en línea de aprendizaje automático de OpenML [19] que, entre otros recursos, incluye bases de datos, que utiliza la comunidad de aprendizaje automático para el análisis empírico de los algoritmos de aprendizaje automático.

4.2. Descripción del conjunto de datos

El conjunto de datos a analizar ha sido resultado de la extracción de características de URLs de sitios web, realizada a priori. Está formado por 11.055 observaciones, 30 atributos y un atributo especial binario que toma valor '1' si la instancia se corresponde con una URL legítima y '-1' en caso de phishing. Este conjunto ha sido utilizado en tareas de clasificación.

Atributos

Diversos estudios [2, 4, 13, 29, 30] han identificado un conjunto de características que pueden indicar que una URL sea considerada como legítima o *phishing*. En este conjunto de datos en particular los atributos han sido diseñados mediante reglas, que pueden tomar 2 ó 3 posibles valores y representan 3 categorías:

- 1 significa URL legítima
- 0 significa URL sospechosa
- -1 significa URL de *phishing*

Los autores [18] han agrupado las características según los siguientes criterios: características basadas en la URL, en anomalías, en HTML y JavaScript y en el dominio.

Adicionalmente, en este trabajo se han clasificado las características teniendo en cuenta si se pueden extraer únicamente de la estructura de la URL, si requieren la consulta del código fuente o el uso de recursos de terceros. Se considera que, a partir de los aportes de los estudios previos, conocer esta información es relevante para la toma de decisiones informadas sobre el diseño de un sistema de detección de *phishing*.A continuación se detallan cada uno de los atributos:

Figura 11: Características basadas en la URL

	Características	Descripción	Valores	Estructura URL	Consulta Código Fuente	Recursos de Terceros
1	having_IP_Address	Uso de la dirección IP como nombre de dominio. No uso de la dirección IP como nombre de dominio.	-1 1	x		
2	URL_Length	Verificar longitud de URL: < 54 caracteres: legítima. ≥54 y ≤75: Sospechosa > 75 caracteres: phishing	 1 0 -1	x		
3	Shortining_Service	Uso de servicios de acortamiento. No uso de servicios de acortamiento.	-1 1			x
4	having_At_Symbol	Uso del símbolo @. No uso del símbolo @.	-1 1	x		
5	double_slash_redirecting	Uso de símbolo "//" para redirigir a otro sitio web: Posición menor que 7 en la URL. Posición mayor que 7 en la URL.	 1 -1	x		
6	Prefix_Suffix	Uso de Prefijo o Sufijo separado por el símbolo guión medio (-) en nombre de dominio. No uso de Prefijo o Sufijo separado por el símbolo guión medio (-) en nombre de dominio.	-1 1	x		
7	having_Sub_Domain	Uso de subdominos múltiples: se detecta al contar los puntos (.) restantes, luego de omitir el subdominio (www.) y el dominio de nivel superior ccTLD (si existe): Cero puntos. Un punto. Más de 2 puntos.	 1 0 -1	x		
8	SSLfinal_State	No uso de https. Certificado emitido por autoridad de confianza y hace más de un año. Uso de https y el emisor no es de confianza.	-1 1 0			x
9	Favicon	Uso de favicon que se carga desde un dominio igual al que se muestra en la barra de direcciones. Uso de favicon que se carga desde un dominio diferente al que se muestra en la barra de direcciones.	1 -1		x	
10	port	Uso de puertos no estandar. Uso de puertos estandar.	-1 1		x	
11	HTTPS_token	Uso de la palabra "https" en la parte del dominio de la URL. No uso de la palabra "https" en la parte del dominio de la URL.	-1 1	x		

Figura 12: Características basadas en anomalías

	Características	Descripción	Valores	Estructura URL	Consulta Código Fuente	Recursos de Terceros
12	Request_URL	Verificar si los objetos externos contenidos en una página web se cargan desde otro dominio: <22% : Legítima entre 22 y 61%: Sospechosa ≥ 61%: Phishing	 1 0 -1		x	
13	URL_of_Anchor	Verificar si etiquetas de ancla contenidas en una página web se cargan desde otro dominio: <31% : Legítima entre 31 y 67%: Sospechosa ≥ 68%: Phishing	 1 0 -1		x	
14	Links_in_tags	Verificar si los enlaces <Meta>, <Script> y <Link> contenidos en una página web se cargan desde otro dominio: <17% : Legítima entre 17 y 81%: Sospechosa ≥82%: Phishing	 1 0 -1		x	
15	SFH	Verificar los Server Form Handler (Manejador de Formularios en el Servidor): Se dirige a un dominio diferente: Sospechosa SFH contienen una cadena vacía o "about:blank" : Phishing Caso contrario: legítima	 0 -1 1		x	
16	Submitting_to_email	Uso de funciones del lado del cliente que permiten el envío de información captada en formulario al correo electrónico "mail()\" o \"mailto:\" No usa esa funciones.	-1 1		x	
17	Abnormal_URL	Nombre del host no incluido en la URL. Nombre del host incluido en la URL.	-1 1			x

Figura 13: Características basadas en HTML y JavaScript

	Características	Descripción	Valores	Estructura URL	Consulta Código Fuente	Recursos de Terceros
18	Redirect	Verificar cuantas veces un sitio web ha sido redirigido: ≤1 : Legítima entre 2 y 3: Sospechosa ≥4: Phishing	 1 0 -1		x	
19	on_mouseover	Uso del evento "onMouseOver" para hacer cambios en la barra de estado. No uso del evento "onMouseOver" para hacer cambios en la barra de estado.	-1 1		x	
20	RightClick	Uso del evento "event.button==2" para deshabilitar clic derecho. No uso del evento "event.button==2" para deshabilitar clic derecho.	-1 1		x	
21	popUpWidnow	Uso de ventana emergente para solicitar información personal. No uso de ventana emergente para solicitar información personal.	-1 1		x	
22	Iframe	Uso de redirección de marco flotante. No uso de redirección de marco flotante.	-1 1		x	

Figura 14: Características basadas en el dominio

	Características	Descripción	Valores	Estructura URL	Consulta Código Fuente	Recursos de Terceros
23	Domain_registeration_length	Uso de dominio que expira en menos de 1 año. Uso de dominio que expira en más de 1 año.	-1 1			x
24	age_of_domain	Verificar edad de dominio: ≤6 meses : Phishing ≥ 6 meses: Legítimo	 -1 1			x
25	DNSRecord	Registro de DNS (Domain Name System) no se encuentra o está vacío. Registro de DNS (Domain Name System) adecuado.	-1 1			x
26	web_traffic	Verificar tráfico del sitio web en la base de datos de Alexa: Website Rank < 100,000 : legítima. Website Rank > 100,000: Sospechosa Dominio sin tráfico o no reconocido por la base de datos: phising	 1 0 -1			x
27	Page_Rank	PageRank (importancia de una página web en Internet) < 0,2 : phising. PageRank > =0,2 : legítima.	-1 1			x
28	Google_Index	Sitio web indexado por Google: legítima Sitio web no indexado por Google: phising	1 -1			x
29	Links_pointing_to_page	Verificar los enlaces que apuntan a la página: 0 :Phishing Entre 1 y 2: Sospechosa ≥ 3: Legítima	 -1 0 -1			x
30	Statistical_report	Verificar si el host pertenece a las principales IP de phishing o a los principales dominios de phishing en listados emitidos por terceros. Host no está en listas negras	-1 1			x

Análisis de las Clases

En la figura 15 se observa la distribución de clases en el conjunto de datos. Se verifica que el ratio entre clases URL Legítima y *phishing* es de 1.26, por lo que no se aprecia una gran diferencia en la proporción de instancias por clases.

El hecho de que el conjunto esté relativamente balanceado puede resultar beneficioso para algunos algoritmos de aprendizaje automático, como Árboles de decisión, Bosques aleatorios y Regresión logística, ya que pueden aprender de manera más equilibrada y generalizar mejor en todas las clases, realizando predicciones más precisas.

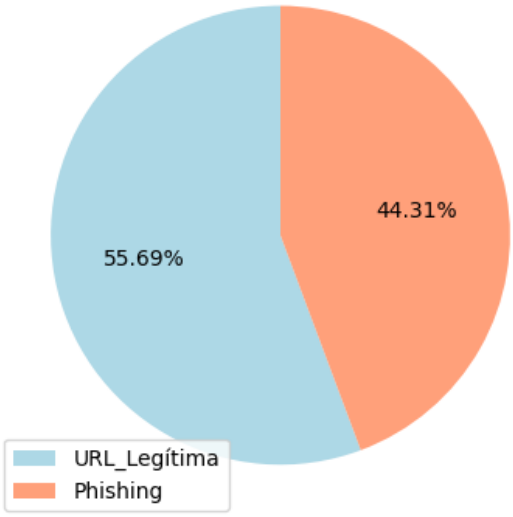


Figura 15: Distribución de Clases

4.3. Análisis de correlación

En la figura 16 se observa la correlación entre los atributos del conjunto. Es importante destacar que la correlación no necesariamente implica causalidad, sino que sugiere una posible relación (directa o inversa) en el comportamiento o características de las URLs de los sitios web.

Por otra parte, es necesario identificar los atributos que multicolineales, que proporcionan información similar, pues el descarte de uno de ellos evitaría la redundancia, facilitaría la interpretación de los modelos, reduciría la complejidad del modelo sin afectar su rendimiento y disminuiría el tiempo de entrenamiento. De ahí la importancia de realizar una selección de características efectiva.

Correlación entre características

Las correlaciones negativas son aquellas en las que una característica marca la URL como *phishing*, mientras que la otra la califica a la inversa. Sin embargo, en este conjunto de datos no se observan

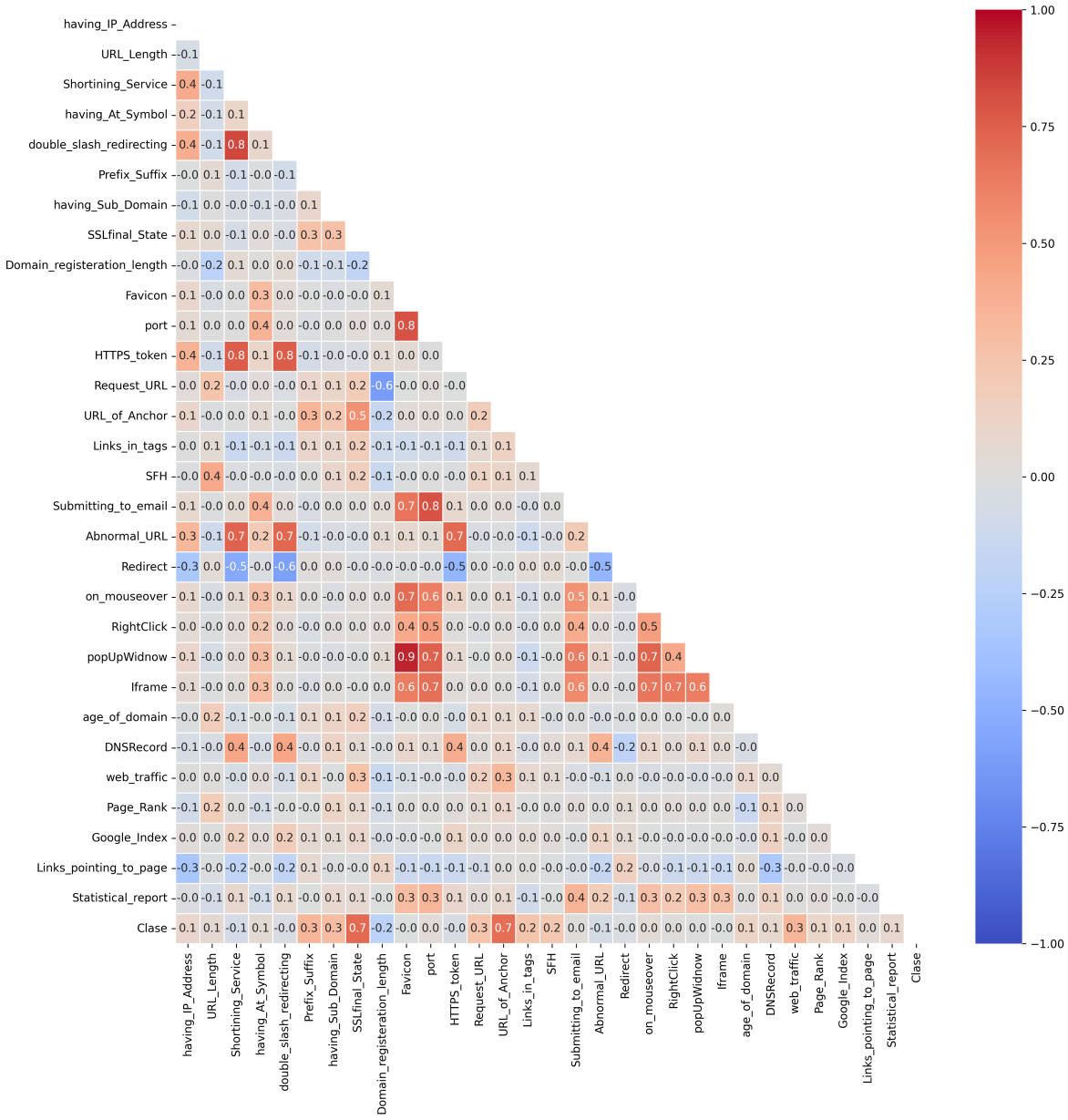


Figura 16: Matriz de Correlación

correlaciones negativas relevantes. Las correlaciones positivas entre características, en este contexto de detección de *phishing* implican que los atributos enfrentados califican en el mismo sentido una URL. A continuación se analizan aquellos atributos que presentan multicolinealidad mayor de 0.8.

Se verifica una alta correlación positiva de 0.9 entre las características 'popUpWindow' y el 'favicon', que está indicando que cuando el favicon que se carga desde un dominio diferente al que se muestra en la barra de direcciones, generalmente se está utilizando una ventana emergente para solicitar información personal, ambas situaciones indicativas de una URL ilegítima.

Por otra parte, las características 'Shortining_Service' (servicio de acortamiento de URL), 'double-slash_redirecting' (redireccionamiento de doble barra diagonal) muestran una correlación positiva de 0.8 con el atributo 'HTTPS_token' (token HTTPS). Lo anterior indica que cuando se utiliza la palabra 'HTTPS', no como protocolo de seguridad en sí, sino como parte del nombre del dominio de la URL, se están utilizando servicios de acortamiento de URL o redirecciones utilizando el doble *slash*. Todas estas constituyen prácticas sospechosa o potencialmente peligrosa, con una alta probabilidad de que se trate de una URL fraudulenta.

Finalmente, los atributos 'port' y 'Submitting_to_email' muestran una correlación de 0.8, lo cual apunta a que cuando se utilizan puertos no estándar, se están empleando funciones del lado del cliente que permiten el envío de información captada en formulario al correo electrónico 'mail()' o 'mailto:', que generalmente son estrategias utilizadas por los *phishers* para captar información confidencial.

Correlación entre las características y la clase

En la figura 17 se visualiza la correlación de los diferentes atributos con la clase. Las características que más destacan son 'SSL_final_State' (estado final SSL) y 'URL_of_Anchor' (URL del anclaje).

La implementación de certificados SSL/TLS (SSLfinalState) es indicativo de que se ha establecido una conexión segura SSL/TLS entre el navegador del usuario y el servidor del sitio web, lo cual es esencial para proteger la privacidad y la seguridad de los datos transmitidos entre el

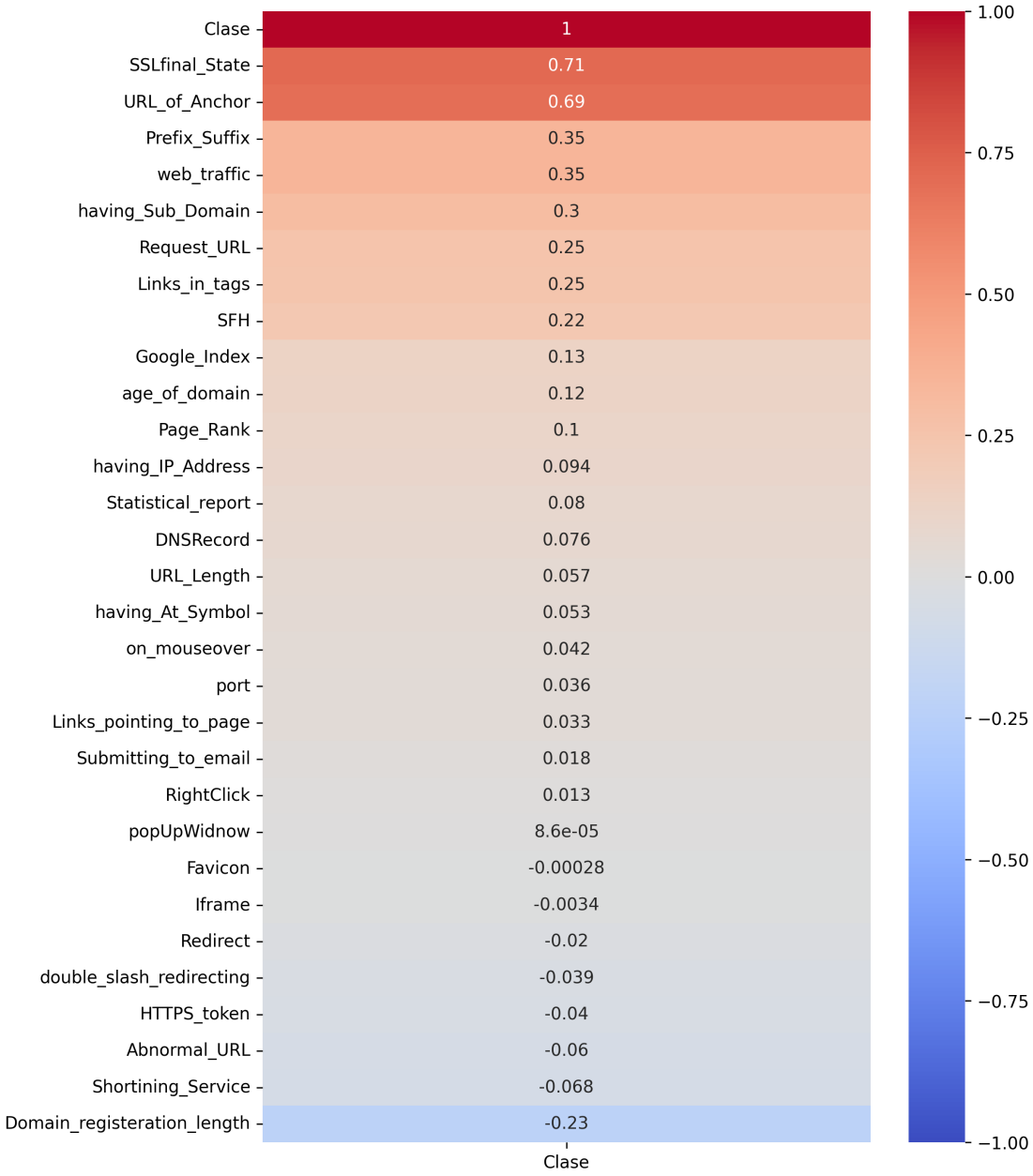


Figura 17: Correlación entre las características y la clase

navegador y el servidor. En consecuencia, la alta correlación entre la variable SSLfinalState y la Clase podría ser una indicación de que muchos sitios web legítimos, especialmente aquellos que requieren intercambio de información sensible (por ejemplo, sitios de banca en línea, comercio electrónico, etc.), están configurados para usar conexiones seguras.

Por otra parte, el atributo 'URLofAnchor' se refiere a las direcciones URL a las que enlazan los enlaces en una página web, que generalmente están diseñados para guiar a los usuarios a otras secciones del mismo sitio o a recursos relacionados, proporcionando una navegación coherente y útil. En consecuencia, la alta correlación entre la variable 'URLofAnchor' y la Clase, podría ser una indicación de que un porcentaje alto de etiquetas de ancla que se cargan de otros dominios constituye una situación potencialmente sospechosa de actividad de *phishing*.

Finalmente, el análisis detallado del conjunto de datos que se ha llevado a cabo en esta sección ha contribuido significativamente a una mejor comprensión del problema en estudio y de las complejas relaciones que existen entre las diferentes variables. Por otra parte, ha sentado las bases para determinar las acciones de preprocesamiento que requerirá el conjunto de datos, para la posterior implementación de técnicas de minería de datos.

5. Metodología

El principal objetivo que se propone este trabajo, es encontrar el algoritmo que detecte de forma más precisa y eficiente las URLs de *phishing*. En línea con esta meta, a continuación se describe la estrategia metodológica a utilizar, que incluye las tareas de preprocesamiento y transformación a realizar, el entrenamiento de clasificadores, así como el análisis de los resultados, a partir de las métricas de evaluación seleccionadas.

1. Se inicia con el preprocesamiento de datos, realizando las transformaciones básicas necesarias para el entrenamiento de clasificadores.
2. Entrenamiento y evaluación de algoritmos de aprendizaje automático supervisados que se han utilizado con frecuencia en los estudios relacionados y que han obtenido resultados relevantes.
3. Análisis del balance de clases. Comparación entre modelos iniciales y los modelos a los que se le aplicó la técnica de balanceo de clases.
4. Ajustes de hiperparámetros para encontrar una buena configuración del modelo con base en su rendimiento. Contraste con modelos iniciales para seleccionar los más eficientes en relación con el objetivo del trabajo.
5. Implementación de técnicas de selección de atributos. Comparación con modelos iniciales.
6. Utilización de técnicas de validación cruzada estratificada para evaluar el rendimiento del modelo, en todas las pruebas anteriores.
7. Evaluación final empleando conjunto de prueba no utilizado a priori en el entrenamiento y evaluación de clasificadores.
8. Análisis de las opciones de clasificadores valorando el equilibrio entre precisión y eficiencia.
9. Selección del mejor modelo.

La distribución gratuita de Python utilizada es Anaconda y dentro de esta, la plataforma Jupyter Notebook (versión 3.9). Para el tratamiento de datos se recurrió al conjunto de bibliotecas

habituales como Pandas, Numpy, Matplotlib y Seaborn. Se utilizó la biblioteca Scikit-Learn para las tareas de aprendizaje automático , pues resulta ser muy flexible al permitir la construcción de modelos personalizados y el ajuste de hiperparámetros.

Los módulos de importación necesarios, así como el código para la implementación de los algoritmos y métricas de aprendizaje automático se pueden consultar en el anexo A.1.

Los experimentos se realizaron en un ordenador con procesador AMD Ryzen 3 3250U, con una velocidad de 2.60 GHz y 8.00GB de memoria RAM instalada (5.94GB utilizable). Esta computadora está equipada con una tarjeta gráfica Radeon Graphics y trabaja con el sistema operativo Windows 10 de 64 bits.

Con el objetivo de garantizar la reproducibilidad y la consistencia de los resultados producto de la implementación de los modelos, se controló la semilla utilizada por el generador de números aleatorios de Python, fijando el parámetro *random_state* en 42, lo cual constituye una buena práctica cuando se trabaja con algoritmos que contienen aleatoriedad.

Un aspecto muy importante es la definición de las métricas en las que se centrará el análisis de resultados, para evaluar el desempeño en la clasificación realizada por el modelo. Estas métricas deben ser elegidas en función de los objetivos del trabajo, así como en función de las implicaciones de los errores del modelo.

Lo anterior requiere una comprensión detallada de la problemática que se aborda. En el contexto de detección de *phishing* mediante análisis de URLs, es fundamental entender las implicaciones de los falsos positivos y falsos negativos debido a las posibles consecuencias en términos de seguridad y experiencia del usuario.

En primer lugar, es imprescindible aclarar que se ha recodificado la clase Positiva como URL de *phishing* (”1”) y la clase negativa como URL Legítima (”0”). Un falso positivo ocurre cuando el modelo identifica incorrectamente una URL legítima como phishing. En este escenario, si el sistema etiqueta erróneamente una URL legítima como peligrosa, estaría dando una falsa alarma,

que podría llevar a que URLs legítimas sean bloqueadas, causar incomodidad y pérdida de tiempo en los usuarios y a que se produzca, en consecuencia, pérdida de confianza en el servicio.

Por otra parte, un falso negativo sucede cuando el modelo pasa por alto una URL fraudulenta y la considera legítima. Lo anterior podría exponer a los usuarios a riesgos graves, ya que podrían hacer clic en enlaces maliciosos y pueden estar expuestos a diversos riesgos, como pérdida de datos, robo de contraseñas, fraude financiero y otros tipos de ataques cibernéticos. En consecuencia, se puede intuir que la presencia de falsos negativos, en este contexto, tiene un costo superior que la de falsos positivos.

En línea con lo anterior, se requerirá monitorear con mayor énfasis las métricas que midan la capacidad del modelo para detectar correctamente los casos positivos, que serán garantizados por el valor más alto de Sensibilidad. Al mismo tiempo, es necesario vigilar el comportamiento de los falsos negativos, priorizando los valores más bajos, lo cual se puede controlar mediante la métrica Valor Predictivo Negativo que posea valores más elevados.

Para obtener una imagen completa del rendimiento del modelo, además de las métricas anteriores, se controlará la Exactitud-Accuracy (en un contexto de clases balanceadas) y el AUC-ROC (exactitud global), que constituyen medidas adecuadas generales para evaluar la capacidad del modelo de predicción y de distinción entre clases, respectivamente.

6. Experimentación

6.1. Preprocesamiento

A continuación, se describen las tareas de preprocesamiento que se realizaron inicialmente en el conjunto de datos. Como se constató a lo largo del Análisis Exploratorio de Datos realizado en la sección anterior, en este conjunto no se detectaron valores faltantes, ruidos o valores poco frecuentes, por lo que el dataset no requirió de tareas de preprocesamiento en ese sentido.

Con relación a las variables de entrada, estas representan categorías, pero se encuentran codificadas como números enteros, tomando valores de '-1', '0' y '1'. En este contexto, no fue necesario realizar normalización, ya que las variables se encontraban en la misma escala, ni estandarización, pues no se utilizaron modelos basados en distancias. En todo caso, se comprobó, mediante experimentación, que la estandarización no introduce cambios apreciables en las métricas de los modelos seleccionados.

Los algoritmos de clasificación en scikit-learn pueden manejar directamente etiquetas numéricas sin necesidad de convertirlas a categóricas, sin embargo fue necesario reemplazar los valores de las etiquetas, pues se requiere que sean enteros no negativos. Para una interpretación más intuitiva de las métricas de evaluación, la clase considerada como 'positiva' se ha etiquetado con el valor '1' y corresponde a las URLs ilegítima y la clase 'negativa' se etiqueta con el valor '0', identificando a las URLs Legítimas. Adicionalmente, se barajaron las instancias, para evitar que las evaluaciones pudiera estar influenciadas porque las observaciones tuvieran un orden inicial en particular.

Finalmente, se procedió a la división del conjunto de datos, destinándose un 80 % de las observaciones al conjunto de entrenamiento y un 20 % al conjunto de prueba. Esta separación se corresponde con la estrategia para evaluar el rendimiento del modelo mediante la técnica de validación cruzada durante la fase de entrenamiento y validación y la realización de una evaluación final en un conjunto de prueba independiente empleando el método hold-out. Se pueden verificar la codificación de las transformaciones realizadas en el Anexo A.2.

A continuación, se describen las técnicas de experimentación a aplicar con vista a determinar

el modelo que logra un mejor desempeño, permitiendo una identificación más precisa y eficiente de las URLs fraudulentas y las URLs legítimas.

Desde la Prueba 1 a la Prueba 4, todos los algoritmos se entrenaron y evaluaron según el esquema de validación cruzada estratificada k-folds, descrito en el marco teórico de este trabajo.

6.2. Prueba 1 - Entrenamiento y validación de algoritmos

En una primera etapa, y con el objetivo de tener una visión general de la capacidad explicativa de las variables en conjunto, se crearon y entrenaron 6 algoritmos que han sido identificados por estudios previos, como clasificadores eficientes para el tratamiento de este tipo de problema, que son los siguientes:

- Radom Forest
- Extra Trees
- XGBoost
- Support Vector Machine
- Decision Tree
- Gradient Boosting

En esta fase se incluyeron la totalidad de características disponibles, que en este caso son 30 y se utilizaron los hiperparámetros por defecto de los clasificadores. Este primer contraste se toma como punto de comparación para el resto de los experimentos. Para la evaluación se empleó la validación cruzada estratificada con 5 *folds*. El código utilizado se puede consultar en el anexo A.3.

6.3. Prueba 2 - Análisis de equilibrio de clases

Otro elemento a considerar fue que el conjunto de datos, no presenta un desbalance pronunciado entre las clases, al contar con una distribución de 55.7 % de URLs Legítimas y un 44.3 % de URLs de *phishing*. Para determinar si la distribución de los datos está influyendo o no en el

desempeño de los modelos se aplica la técnica de sobremuestreo (oversampling) para equilibrar las clases en el conjunto de datos.

Esta técnica consiste en la creación de ejemplos sintéticos de la clase minoritaria de forma aleatoria, duplicando ejemplos pertenecientes a esta clase hasta que se equilibre con la clase mayoritaria. Una vez entrenados los modelos con el conjunto de datos equilibrado, se compararon las métricas obtenidas con las métricas de los modelos entrenados en el conjunto de datos original. En función de los resultados obtenidos, se verifica si existe o no un problema de balance, lo cual conduce a la elección del conjunto de datos original o el balanceado para las pruebas siguientes (ver código en anexo A.4).

6.4. Prueba 3 - Optimización de los modelos

En línea con la estrategia presentada inicialmente se ajustarán los hiperparámetros de los modelos, buscando el desempeño más óptimo posible, teniendo en cuenta los recursos computacionales de los que se dispone para la ejecución de las técnicas.

Para ello, se realizará una búsqueda aleatoria en rejilla, empleando la técnica RandomizedSearchCV que consiste en la selección aleatoria de combinaciones de hiperparámetros dentro de las distribuciones definidas en la rejilla (param_grid), evaluando el rendimiento del modelo mediante validación cruzada estratificada (5 *folds*). El número total de combinaciones probadas está predeterminado por un número de iteraciones. Esta técnica no es exhaustiva, por lo que no requiere probar todas las combinaciones posibles y, por tanto, se realiza en un tiempo menor (ver código en anexo A.5).

6.5. Prueba 4 - Técnicas de selección de características

En una segunda etapa y tomando como base los resultados obtenidos en el análisis exploratorio de los datos, se implementó un proceso de selección de características, que intenta determinar si es posible reducir el número de atributos, sin que el rendimiento de la clasificación disminuya considerablemente [33]. La reducción del número de características busca reducir el costo computacional y el asociado a la recolección de datos, así como mejorar el rendimiento y la interpretabilidad de los modelos.

La técnica de selección **Step Forward Feature Selection (SFS)** se encuentra dentro del grupo de los *Wrapper Methods*. Esta técnica parte de un modelo vacío (sin características) y su objetivo es explorar incrementalmente diferentes subconjuntos de características para encontrar el que maximice una métrica de evaluación específica, que en este caso ha sido elegida la Exactitud (ver código en anexo A.7).

Este es un enfoque exploratorio que puede llevar a la selección de un subconjunto óptimo de características, pero no garantiza una mejora constante en el rendimiento en cada paso, lo cual puede darse por la presencia de multicolinealidad (alta correlación entre características), atributos irrelevantes o redundantes [26]. En este caso se utilizará la validación cruzada para evaluar el rendimiento de los diferentes conjuntos .

La técnica de **Eliminación Recursiva de Características con validación cruzada** (RFECV por sus siglas en inglés), que combina elementos de los métodos *Wrapper* y *Embedded*, selecciona las características más importantes y posteriormente se evalúa su impacto en la precisión del modelo.

La técnica RFECV consiste en la eliminación de características irrelevantes o redundantes, partiendo del entrenamiento de un modelo de aprendizaje automático que comienza con el análisis de todas las características disponibles en el conjunto de datos, al mismo tiempo que emplea la validación cruzada para evaluar el rendimiento del mismo en cada etapa.

En cada iteración, se van eliminando las características menos importantes y se calculan el rendimiento (exactitud) del modelo en el conjunto de datos reducido. La forma en que se calcula la importancia varía según el tipo de modelo utilizado. Por ejemplo, en modelos basados en árboles como Random Forest y Extra Trees, la importancia de una característica se mide por la disminución en la impureza del nodo que produce cuando se utiliza para hacer divisiones.

El proceso de eliminación se repite hasta que se alcanza un criterio de parada (número predefinido de características) o cuando la eliminación de más características no mejora significativamente la métrica de rendimiento, lo que indica que la eliminación adicional de características está afectando

negativamente la capacidad del modelo para generalizar. Este último enfoque, empleando la métrica Exactitud, fue el aplicado en este trabajo (ver código en anexo A.6).

6.6. Prueba 5 - Evaluación de modelos en conjunto de prueba independiente

Inicialmente se aparta un conjunto de datos, correspondiente al 20 % del total de instancias disponibles, que no se utiliza en el proceso de entrenamiento ni de validación de las Pruebas 1 a la 4. Posterior a esta fase, se utilizará ese conjunto de prueba para evaluar el rendimiento del modelo y obtener, de esta forma, una estimación más realista de su desempeño en datos no vistos.

7. Análisis de Resultados

En esta sección se describen los resultados obtenidos a partir de la implementación de la estrategia metodológica definida anteriormente.

Para una mejor comprensión de los resultados, es necesario aclarar que el esquema visual de la matriz de confusión utilizada en este trabajo coincide con el estándar, descrito en el marco teórico de este trabajo (ver figura 18), y no con el utilizado por la biblioteca scikit-learn. Por otra parte, los resultados más relevantes de cada métrica se muestran resaltados en **negrita**.

		Clase Real Positiva (P)	**Clase Real** Negativa (N)
Clase Predicha	Positiva (P)	Verdaderos Positivos (TP)	Falsos Positivos (FP)
Clase Predicha	Negativa (N)	Falsos Negativos (FN)	Verdaderos Negativos (TN)

Figura 18: Matriz de Confusión

7.1. Prueba 1 - Entrenamiento y validación de algoritmos

En la Figura 19 se muestran en detalle los resultados de los modelos de clasificación entrenados, en términos de varias métricas de evaluación. Con este primer acercamiento se constata que los modelos obtenidos muestran valores de exactitud superiores al 94 %, indicando que están logrando un alto porcentaje de predicciones correctas del total de predicciones, en la clasificación de las instancias en el conjunto de datos de detección de *phishing*. La validez de esta métrica para evaluar el desempeño en este conjunto de datos será comprobada más adelante.

Por otra parte, la desviación estándar de la métrica Accuracy ofrece una medida de cuán consistente es el rendimiento del modelo a través de los diferentes pliegues de la validación, que al tomar valores bajos, como en este caso, indica que los resultado de accuracy en las distintas particiones están muy cerca entre sí, lo que sugiere que el modelo tiene un rendimiento coherente.

Figura 19: Resultados prueba 1 - Contraste de Modelos

Modelos	Matriz de confusión		Exactitud (Accuracy)	Desviación Estándar (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's Kappa	Valor Predictivo Negativo
Random Forest	3741	101	0,9704	0,0039	0,9737	0,9587	0,9662	0,9692	0,9398	0,9678
	161	4841								
Extra Trees	3741	87	**0,9720**	**0,0011**	**0,9773**	0,9587	**0,9679**	**0,9706**	**0,9430**	0,9679
	161	4855								
Support Vector Machine	3601	168	0,9470	0,0061	0,9554	0,9229	0,9389	0,9444	0,8921	0,9407
	301	4774								
XGBoost	**3754**	112	0,9706	0,0041	0,9710	**0,9621**	0,9665	0,9697	0,9403	**0,9703**
	148	4830								
Decision Tree	3712	154	0,9611	0,0036	0,9602	0,9513	0,9557	0,9601	0,9210	0,9618
	190	4788								
Gradient Boosting	3638	169	0,9510	0,0051	0,9556	0,9323	0,9438	0,9491	0,9005	0,9476
	264	4773								

Adicionalmente, los valores de AUC-ROC (Área bajo la curva ROC) obtenidos en modelos analizados se encuentran por encima del 94 %, y sugieren que los modelos son capaces de clasificar correctamente la mayoría de las instancias positivas y negativas en el conjunto de prueba, por lo que tienen una alta capacidad de discriminación entre clases. Los valores de Kappa de estos modelos confirman que existe un buen acuerdo o concordancia entre las clasificaciones del modelo y las clasificaciones reales.

El alto porcentaje de aciertos obtenidos es reflejo de que las características incluidas son relevantes, suficientemente informativas, la cantidad de instancias de entrenamiento ha sido suficiente para que los modelos aprendan correctamente y los algoritmos de clasificación seleccionados son adecuados para este tipo de problema, están aprendiendo patrones útiles en los datos, son flexibles y pueden adaptarse mejor a la complejidad de los datos.

Por otra parte, los valores elevados de precisión y sensibilidad obtenidos indican que los modelos presentan, en general, una baja tasa de falsos positivos y falsos negativos respectivamente, y, como consecuencia, clasifican correctamente la gran mayoría de las instancias.

En las matrices de confusión se observa que los modelos tiene un número mayor de falsos negativos que de falsos positivos, por lo que, cuando se equivocan, tiende a hacerlo más cometiendo el error de clasificar las URLs fraudulentas como URLs legítimas, lo cual puede resultar más costoso en

este contexto de la ciberseguridad.

Mediante este primer experimento se verifica que el clasificador más relevante en términos de discriminación entre clases es el modelo Extra Trees con valores de Accuracy de 97.2 % y AUC de 97.06 % , sin embargo, el clasificador XGBoost obtiene un porcentaje más elevado de Sensibilidad (96.21 %) y de Valor Predictivo Negativo (97.03 %), que indican qué tan bueno es el modelo para predecir casos positivos y controlan la proporción de falsos negativos, respectivamente. Estos últimos aspectos se encuentran más alineados con el objetivo de detección de casos positivos (URLs de *phishing*) y de minimización de los casos Falsos Negativos (URLs de *phishing* clasificadas incorrectamente como URLs Legítimas).

7.2. Prueba 2 - Análisis de equilibrio de clases

En la figura 20 se recogen los resultados de la aplicación de la técnica de sobremuestreo.

Figura 20: Resultados prueba 2 - Sobremuestreo

Modelos	Matriz de confusión	Exactitud (Accuracy)	Desviación Estándar (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's Kappa	Valor Predictivo Negativo
Random Forest	4764 124 163 4800	0,9709	0,0043	0,9746	0,9669	0,9708	0,9709	0,9417	0,9672
Extra Trees	4771 106 156 4818	**0,9734**	0,0037	**0,9783**	0,9683	**0,9733**	**0,9734**	**0,9468**	0,9686
Support Vector Machine	4606 212 321 4712	0,9459	0,0024	0,9560	0,9348	0,9453	0,9459	0,8918	0,9362
XGBoost	4774 137 153 4787	0,9706	0,0021	0,9721	**0,9689**	0,9705	0,9706	0,9411	**0,9690**
Decision Tree	4734 161 193 4763	0,9641	**0,0020**	0,9671	0,9608	0,9640	0,9641	0,9281	0,9611
Gradient Boosting	4639 238 288 4686	0,9466	0,0049	0,9512	0,9415	0,9463	0,9466	0,8932	0,9421

El contraste de las métricas obtenidas por el proceso de sobremuestreo con el de las métricas del conjunto de datos original, evidencia que el sobremuestreo y posterior entrenamiento de clasificadores introdujo diferencias mínimas en las métricas (ver figura 21).

Además, se observa coherencia con los resultados obtenidos inicialmente en términos de mejores modelos identificados. Lo anterior confirma que el balance de clases no constituye un problema

Figura 21: Diferencia Prueba 2 - Prueba 1

Modelos	Exactitud (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's kappa	Valor Predictivo Negativo
Random Forest	0,0005	0,0009	0,0082	0,0046	0,0017	0,0019	-0,0007
Extra Trees	0,0014	0,0010	0,0096	0,0054	0,0028	0,0038	0,0007
Support Vector	-0,0011	0,0006	0,0120	0,0064	0,0015	-0,0003	-0,0045
XGBoost	0,0000	0,0011	0,0069	0,0040	0,0009	0,0008	-0,0012
Decision Tree	0,0030	0,0069	0,0095	0,0082	0,0040	0,0071	-0,0008
Gradient Boosting	-0,0044	-0,0044	0,0092	0,0025	-0,0025	-0,0072	-0,0055

para este conjunto de datos y que la métrica Accuracy puede ser empleada como medida de la capacidad predictiva de los modelos.

7.3. Prueba 3 - Optimización de los modelos

A continuación se describen los hiperparámetros ajustados para cada modelo. En el caso de **Decision Tree** se incluyeron:

- Profundidad máxima para los nodos del árbol.Un valor más alto puede llevar al sobreajuste.
- Mínimo de muestras requeridas para considerar una división en un nodo interno.
- Mínimo de muestras requeridas en un nodo hoja.

Para los clasificadores **Random Forest** y **Extra Trees** se ajustaron los hiperparámetros siguientes:

- Número de estimadores: Controla la cantidad de árboles que se construyen para la clasificación. Un valor más alto generalmente mejora la generalización del modelo, pero también aumentan el costo computacional.
- Cantidad máxima de características (max_features): Fija la cantidad máxima de características que cada árbol individual

puede utilizar para tomar decisiones. Valores más bajos pueden reducir la correlación entre los árboles y evitar el sobreajuste.

- Criterio: fija el criterio utilizado para medir la calidad de una división en los nodos del árbol de decisión. Puede ser 'gini' o 'entropy'. Ambos criterios miden la impureza de los nodos del árbol y buscan minimizarla al tomar decisiones de división.

En el caso del clasificador **SVM** se optimizaron los siguientes hiperparámetros:

- Parámetro de regularización (C): controla el margen. Un valor más bajo de C permite un margen más amplio, pero puede resultar en un mayor error de clasificación en el conjunto de entrenamiento. Un valor más alto de C reduce el margen, pero puede llevar a un ajuste más estricto a los datos de entrenamiento.
- Kernel: es una función que calcula el producto interno en un espacio de características transformado. Kernel de base radial (Radial Basis Function).
- Gamma: controla el efecto de cada punto de entrenamiento en la creación de los límites de decisión. Un valor más bajo de gamma puede llevar a un límite de decisión más suave, mientras que un valor más alto puede llevar a un modelo más ajustado a los datos.

Descripción de los hiperparámetros a ajustar para el modelo **Gradient Boosting**:

- Número de estimadores (árboles) en el conjunto.
- Tasa de aprendizaje que controla la contribución de cada árbol al modelo final. Un valor más bajo puede mejorar la generalización, pero puede requerir más estimadores.
- Profundidad máxima de los árboles en el ensemble.
- Mínimo de muestras requeridas para dividir un nodo interno en un árbol.

Para el clasificador **XGBoost** se exploraron las combinaciones de los hiperparámetros siguientes:

- Número de estimadores (árboles) en el conjunto.
- Tasa de aprendizaje que controla la contribución de cada árbol al modelo final.

- Profundidad máxima de los árboles
- Proporción de muestras utilizadas para entrenar cada árbol, que puede ayudar a controlar el sobreajuste permitiendo muestras aleatorias en cada árbol.
- Proporción de características utilizadas para entrenar cada árbol (colsample_bytree): controla la variabilidad al introducir aleatoriedad en la selección de características.

Luego de la ejecución del proceso de ajuste, mediante la funcionalidad *RandomizedSearchCV* los mejores hiperparámetros detectados para cada modelo fueron los siguientes:

- Random Forest : 'criterion': 'entropy', 'max_features': 3, 'n_estimators': 300
- Extra Trees: 'max_depth': 28, 'min_samples_split': 2, 'n_estimators': 469
- Support Vector Machine: 'C': 20, 'gamma': 0.1, 'kernel': 'rbf'
- XGBoost : 'colsample_bytree': 0.8, 'learning_rate': 0.3, 'max_depth': 7, 'n_estimators': 131, 'subsample': 1.0
- Decision Tree: 'max_depth': 16, 'min_samples_leaf': 1, 'min_samples_split': 10
- Gradient Boosting: 'learning_rate': 0.1, 'max_depth': 10, 'min_samples_split': 13, 'n_estimators': 161.

Como resultado del proceso de ajuste de hiperparámetros, se observa una mejora en 4 de los 6 modelos estudiados (ver figura 22). Los modelos más beneficiados fueron Support Vector Machine y Gradient Boosting, con incrementos del 2 % en términos de Exactitud, en ambos casos. Adicionalmente se observa que no ha sido posible mejorar el desempeño de los clasificadores Extra Trees y Decision Tree, siendo los hiperparámetros por defecto los más adecuados en estos dos casos.

Este es un proceso iterativo, que demandó capacidad computacional y tiempo para evaluar combinaciones aleatoria de hiperparámetros. Así mismo, esta prueba permitió identificar los tres modelos que mejor explican el comportamiento de los datos, obteniendo un mejor rendimiento en los clasificadores de ensamblaje, Extra Trees (hiperparámetros por defecto), XGBoost (hiperparámetros optimizados) y Random Forest (hiperparámetros optimizados).

Figura 22: Resultados prueba 3 - Optimización de Modelos

Modelos	Matriz de confusión	Exactitud (Accuracy)	Desviación Estándar (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's Kappa	Valor Predictivo Negativo
Random Forest	3749 95 153 4847	**0,9720**	0,0027	0,9753	0,9608	0,9680	0,9708	0,9430	0,9694
Extra Trees	3734 82 168 4860	0,9717	0,0015	**0,9785**	0,9569	0,9676	0,9702	0,9425	0,9666
Support Vector Machine	3725 113 177 4829	0,9672	0,0038	0,9706	0,9546	0,9625	0,9659	0,9334	0,9646
XGBoost	3754 100 148 4842	0,9720	0,0035	0,9741	**0,9621**	**0,9680**	**0,9709**	**0,9431**	**0,9703**
Decision Tree	3666 175 236 4767	0,9535	**0,0014**	0,9544	0,9395	0,9469	0,9521	0,9056	0,9528
Gradient Boosting	3748 97 154 4845	0,9716	0,0027	0,9748	0,9605	0,9676	0,9705	0,9424	0,9692

Los modelos más precisos derivan de árboles de decisión, lo que indica que se posee un conjunto de entrenamiento de alta calidad, que contiene grandes variaciones, proporcionando datos ricos y variados para que el algoritmo de árbol de decisión aprenda y derive reglas de decisión certeras [31].

7.4. Prueba 4 - Técnicas de selección de características

A continuación, se muestra el resultado de la implementación del proceso de selección de características mediante dos técnicas: Eliminación Recursiva de Características con Validación Cruzada (RFSCV) y Selección Secuencial hacia Adelante (SFS).

Las características descartadas por cada modelo, así como las métricas derivadas de la ejecución de los tres mejores algoritmos con los atributos seleccionados, se muestran a continuación:

Atributos descartados con la técnica **RFECV**:

- Random Forest (4 características) : 'double_slash_redirecting', 'port', 'RightClick', 'Iframe'
- Extra Trees (11 características): 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Favicon', 'port', 'Abnormal_URL', 'Redirect', 'on_mouseover', 'RightClick', 'Iframe', 'Statistical_report'
- XGBoost (2 características): 'having_At_Symbol', 'Statistical_report'

Atributos descartados con la técnica **SFS**:

- Random Forest (5 características) : 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Favicon', 'Statistical_report'
- Extra Trees (2 características): 'having_At_Symbol', 'popUpWidnow'
- XGBoost (6 características): 'double_slash_redirecting', 'port', 'Abnormal_URL', 'popUpWidnow', 'Iframe', 'Statistical_report'

A partir del análisis de los resultados (ver figuras 23 y 24) se obtienen los siguientes *insights*. La selección de características con ambos métodos no ha propiciado un incremento significativo del rendimiento de los modelos, los cuales se han mantenido estables.

Figura 23: Resultados prueba 4A - Selección de Características - RFSCV

Modelos	Matriz de confusión		Exactitud (Accuracy)	Desviación Estándar (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's kappa	Valor Predictivo Negativo
Random Forest (Optimizado + 26 atributos)	3749	95	0,9720	0,0027	0,9753	0,9608	0,9680	0,9708	0,9430	0,9694
	153	4847								
Extra Trees (Hiperparámetros por defecto + 19 atributos)	3741	**87**	0,9720	**0,0011**	**0,9773**	0,9587	0,9679	0,9706	0,9430	0,9679
	161	4855								
XGBoost (Optimizado + 28 atributos)	3754	100	0,9720	0,0035	0,9741	**0,9621**	**0,9680**	**0,9709**	**0,9431**	**0,9703**
	148	4842								

Figura 24: Resultados prueba 4B - Selección de Características - SFS

Modelos	Matriz de confusión		Exactitud (Accuracy)	Desviación Estándar (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's kappa	Valor Predictivo Negativo
Random Forest (Optimizado + 25 atributos)	3749	95	0,9720	0,0027	0,9753	0,9608	0,9680	0,9708	0,9430	0,9694
	153	4847								
Extra Trees (Hiperparámetros por defecto + (28 atributos)	3741	87	0,9720	0,0011	0,9773	0,9587	0,9679	0,9706	0,9430	0,9679
	161	4855								
XGBoost (Optimizado + 24 atributos)	3749	95	0,9720	0,0027	0,9753	0,9608	0,9680	0,9708	0,9430	0,9694
	153	4847								

Fuente: Elaboración propia.

Esta situación puede tener varias interpretaciones. Por una parte, se corrobora que, la mayoría de las características originales en el conjunto de datos son informativas y relevantes para el problema, por lo que la selección de características no produce un incremento del rendimiento de los modelos. Por otra parte, los clasificadores utilizados (Random Forest, Extra Trees y XGBoost), tienden a ser robustos ante características irrelevantes o ruidosas, es decir, por definición, pueden adaptarse y seleccionar automáticamente las características más importantes durante el proceso de entrenamiento.

Otro de los resultados de la aplicación de estas técnicas es que, en todos los casos, se ha logrado reducir el número de características necesarias para mantener un rendimiento adecuado. El caso más significativo se obtiene con una reducción de 11 características con el método de RFSCV y empleando el clasificador Extra Trees con los parámetros por defecto.

Por otra parte, es el clasificador XGBoost con hiperparámetros optimizados y 28 atributos, obtenido con la técnica RFSCV, es el que logra un mejor resultado en las métricas más relevantes para el tratamiento del problema analizado, que son AUC-ROC (97.09 %), Sensibilidad (96.21 %) y Valor Predictivo Negativo (97.03 %).

En consecuencia, ambos modelos, Extra Trees (Hiperarámetros por defecto + 19 atributos) y XGBoost (Optimizado + 28 atributos), pasan a la fase final de evaluación, para estimar la capacidad de generalización de los mismos.

7.5. Prueba 5 - Evaluación de modelos en conjunto de prueba independiente

Una vez que se han entrenado y validado los modelos con los mejores hiperparámetros, y se ha realizado el proceso de selección de atributos, se procede a evaluar el rendimiento en un conjunto de datos independiente (conjunto de prueba) para obtener una estimación final de su desempeño en datos no vistos.

Como se observa en la Figura 25, ambos clasificadores mantiene el rendimiento alto tras su evaluación en el conjunto de pruebas, evidenciado una buena capacidad de generalización y las

diferencias entre ambos son mínimas, en cuanto a las métricas de rendimiento.

Figura 25: Resultados prueba 5 - Evaluación de algoritmos en Conjunto de Prueba

Modelos	Matriz de confusión		Exactitud (Accuracy)	Precisión	Sensibilidad (Recall)	F1-score	AUC-ROC	Cohen's kappa	Valor Predictivo Negativo
Extra Trees (Hiperparámetros por defecto + 19 atributos)	951	18	0,9715	**0,9814**	0,9548	0,9679	0,97	0,9423	0,9638
	45	1197							
XGBoost (Optimizado + 28 atributos)	**955**	20	**0,9724**	0,9795	**0,9588**	**0,9691**	**0,9712**	**0,9442**	**0,9668**
	41	1195							

El modelo XGBoost (28 atributos + hiperparámetros optimizados) muestra un desempeño ligeramente superior en la capacidad de detección de verdaderos positivos, respaldado por un valor de Sensibilidad de 95.88 %. Por otra parte, este algoritmo es más exacto detectando los casos negativos y garantizando menos presencia de falso negativos, lo cual se evidencia mediante la métrica correspondiente al Valor Predictivo Negativo (96.68 %).

El modelo Extra Trees (19 atributos + hiperparámetros por defecto) es más exacto clasificando los casos positivos, lo cual se constata con un valor de Precisión ligeramente mayor del 98.14 %.

A continuación se muestran las Curvas ROC, correspondientes a ambos modelos (ver figuras 26 y 27), que permiten visualizar el área que abarcan, así como sus umbrales óptimos. En el caso de Extra Trees el óptimo es de 0.40 y para XGBoost es de 0.45, que son los puntos que maximizan la Sensibilidad (Tasa de Verdaderos Positivos) y minimiza la Tasa de Falsos Positivos (1-Especificidad) en cada caso.

Figura 26: Curva ROC - Modelo Extra Trees (19 atributos)

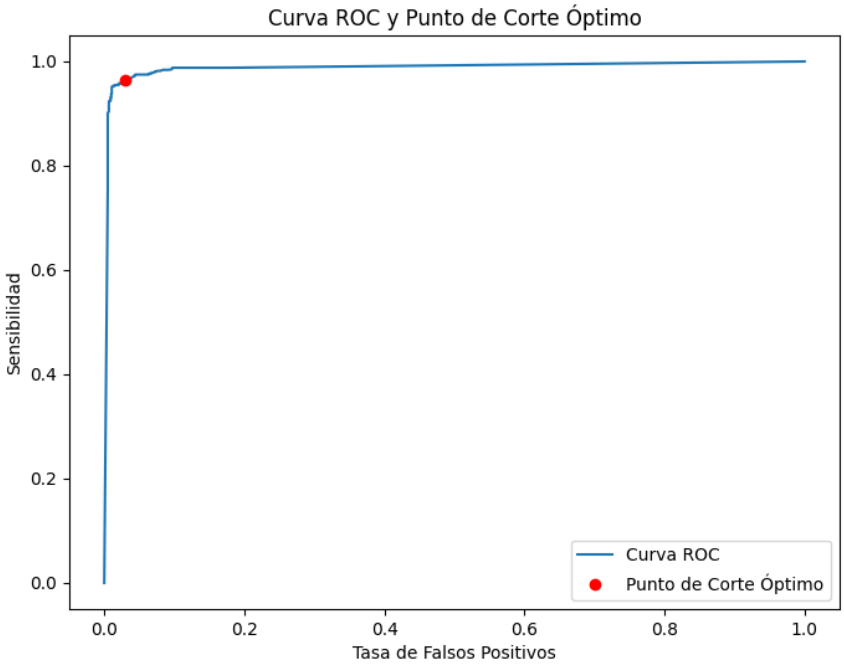
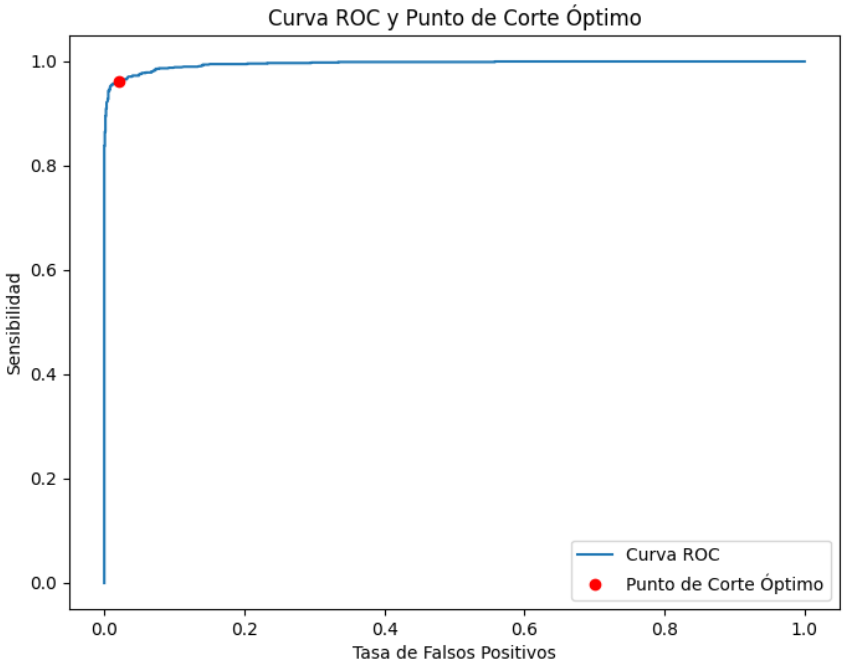


Figura 27: Curva ROC - Modelo XGBoost (28 atributos)



El hecho de que estos puntos estén cercanos evidencia que ambos modelos alcanzan su máximo rendimiento (en términos de sensibilidad y especificidad) cerca del mismo umbral de decisión, lo que significa que los clasificadores son igualmente buenos para discriminar entre las clases positiva y negativa, en el rango de umbrales que se están comparando, resultando ambos adecuados para esta tarea de clasificación.

En línea con lo anterior, la elección entre modelos con rendimientos similares se podría realizar con base en otros factores, como la complejidad de los mismos o el costo de implementación. En este caso, un factor a considerar es la cantidad de características utilizadas por cada modelo, ya que tener menos atributos puede ser más ventajoso en términos de simplicidad y eficiencia computacional.

A continuación, se listan los atributos descartados durante la Prueba 4, correspondientes a los dos modelos seleccionados:

- Extra Trees (11 características): 'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Favicon', 'port', 'Abnormal_URL', 'Redirect', 'on_mouseover', 'RightClick', 'Iframe', 'Statistical_report'
- XGBoost (2 características): 'having_At_Symbol', 'Statistical_report'

Resulta evidente que, de los dos modelos seleccionados, Extra Trees (19 atributos + hiperparámetros por defecto) es más simple en cuanto al número de atributos que utiliza. Cuando se analizan en detalle las características descartadas por este modelo se verifica que 9 de las 11 características descartadas no se extraen directamente de la URL, sino que requieren la consulta del código fuente del sitio web o la utilización de recursos de terceros.

El hecho de que el modelo Extra Trees (19 atributos + hiperparámetros por defecto) emplee características que requieran menos acceso al código fuente y menor utilización de servicios de terceros, lo hace más eficiente en términos de tiempo y recursos computacionales, pudiendo procesar un mayor número de URLs en menos tiempo y con menos recursos de hardware, por una parte, y por otra, puede ser menos vulnerable a interrupciones o problemas con esos recursos externos.

No obstante, la elección de algoritmos para la construcción de sistemas de detección de *phishing*, también debería considerar otros factores como los recursos disponibles, la capacidad de identificación de amenazas más sofisticadas, la capacidad de adaptación ante nuevas amenazas que surjan, aspectos que quedan fuera del alcance de este trabajo.

A partir de las reflexiones anteriores, se considera que el modelo **Extra Trees (19 atributos + hiperparámetros por defecto)** ofrece un mejor equilibrio entre eficiencia y precisión, siendo la opción más adecuada en este contexto, teniendo en cuenta la información disponible.

8. Conclusiones

Las estadísticas relacionadas con la temática del *phishing*, en cuanto a número de ataques, pérdidas financieras, sectores más vulnerables, etc., realizado a lo largo del trabajo han demostrado la actualidad del tema. La complejidad de esta problemática confirma la necesidad de crear constantemente mecanismos de prevención, identificación y defensa, ante los múltiples riesgos de suplantación de identidad online.

La revisión de la literatura realizada, relacionada con la temática del *phishing*, ha proporcionado información sobre la aplicación de enfoques convencionales y automatizados para detectar este tipo de ataques, destacando la importancia de adoptar enfoques híbridos en lugar de depender exclusivamente de una única estrategia de detección. Esto se debe a que los *phishers* están constantemente explorando vulnerabilidades, lo que hace necesario combinar diferentes métodos de detección para abordar de manera efectiva esta amenaza en evolución constante.

Basándonos en la revisión bibliográfica, se pudo constatar que los estudios previos abarcan una amplia gama de investigaciones que utilizan el aprendizaje automático como medio para enfrentar el *phishing*. En particular, se ha demostrado que estas técnicas constituyen una herramienta efectiva para identificar este tipo de ataque, al detectar patrones en los datos a través de características asociadas a la URL o al sitio web.

Este análisis permitió identificar los algoritmos de aprendizaje supervisado más eficaces para este tipo de problemas y constituyó una guía para el diseño de una metodología efectiva para determinar el modelo óptimo, proceso que depende principalmente de los algoritmos empleados, del tiempo de detección y de las características a incluir.

Adicionalmente, el análisis de los estudios relacionados ha permitido identificar otros factores relevantes para la selección de un modelo, señalando la importancia de la detección en tiempo real de enlaces web fraudulento. En este sentido, es crucial considerar diversas características, evaluando si se pueden obtener directamente de la URL, si es necesario acceder al código fuente del sitio web, si se deben utilizar servicios externos o si se deben combinar estos enfoques para

una detección efectiva.

En línea con lo anterior, la selección del modelo óptimo implica seleccionar un conjunto apropiado de atributos para implementar algoritmos de clasificación, garantizando que la extracción de las características se realice en el menor tiempo posible, sin comprometer la precisión del mismo.

Por otra parte, mediante el análisis del marco teórico relacionado con el KDD, se constató que el mismo ofrece una base metodológica sólida para la construcción de modelos empleando técnicas de aprendizaje automático, destacando que todo modelado debe comenzar con la comprensión del conocimiento del dominio, lo que evidencia que es un proceso guiado por la teoría.

El análisis exploratorio del conjunto de datos analizado, posibilitó adquirir una conocimiento más profundo de la estructura de los datos, las conexiones entre las diferentes variables y cómo estas se relacionan con la variable objetivo. Este proceso de exploración resultó fundamental para obtener una visión más clara del problema en estudio, lo que a su vez facilitó la definición de las métricas apropiadas para evaluar el rendimiento del modelo y la identificación de las acciones de preprocesamiento requeridas.

Las acciones de experimentación realizadas de acuerdo a la metodología trazada, dígase análisis del balance de clases, optimización de modelos, selección de atributos y evaluación en conjunto de prueba, permitieron la identificación del modelo óptimo. En este caso se selecciona el modelo Extra Trees (con 19 atributos e hiperparámetros por defecto) con una exactitud del 97.15 %, un AUC-ROC de 97.0 %, Sensibilidad del 95.48 % y Valor Predictivo Negativo de 96.38 %, garantizando el mejor equilibrio entre eficiencia y precisión, de acuerdo a la información disponible.

A partir de los resultados obtenidos, los futuros proyectos de investigación pueden enfocarse en:

- la validación de los resultados en conjuntos de datos de referencia,
- la extracción de nuevas características de la URL, en correspondencia con las nuevas técnicas en los ataques de *phishing*,
- la incorporación de técnicas de aprendizaje profundo, para mejorar la eficiencia del sistema,

- la exploración de técnicas de aprendizaje que consideren el costo asimétrico de los errores y que permitan, por tanto, asignar un peso mayor a los errores de clasificación que tienen un mayor impacto.

Referencias

[1] Ali Aljofey, Qingshan Jiang, Qiang Qu, Mingqing Huang, and Jean-Pierre Niyigena. An effective phishing detection model based on character level convolutional neural network from url. *Electronics*, 9(9):1514, 2020.

[2] Anjaneya Awasthi and Noopur Goel. Phishing website prediction using base and ensemble classifier techniques with cross-validation. *Cybersecurity*, 5(1):1–23, 2022.

[3] BDO España. Las 10 principales amenazas para la ciberseguridad de las empresas en 2023. *BDO España.Publicaciones Técnicas*, 2023. Acceso en línea.

[4] Andrei Butnaru, Alexios Mylonas, and Nikolaos Pitropakis. Towards lightweight url-based phishing detection. *Future internet*, 13(6):154, 2021.

[5] Emily Laufer Deepen Desai, Rohit Hegde. Informe de phishing de threatlabz 2023. aumento del 47,2 *Investigación de Seguridad*, 2023.

[6] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14:241–258, 2020.

[7] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996.

[8] Ekta Gandotra and Deepak Gupta. An efficient approach for phishing detection using machine learning. *Multimedia Security: Algorithm Development, Analysis and Applications*, pages 239–253, 2021.

[9] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[10] Sudhir M. Gorade, Ankit Deo, and Preetesh Purohit. A study of some data mining classification techniques. *International Research Journal of Engineering and Technology*, 4(4):3112–3115, 2017.

[11] Anti-Phishing Working Group et al. Phishing activity trends report 4th quarter 2022. *Apwg, no. May*, pages 1–12, 2023.

[12] Brij B. Gupta, Krishna Yadav, Imran Razzak, Konstantinos Psannis, Arcangelo Castiglione, and Xiaojun Chang. A novel approach for phishing urls detection using lexical based machine learning in a real-time environment. *Computer Communications*, 175:47–57, 2021.

[13] Antonio Hernández Dominguez and Walter Baluja García. Principales mecanismos para el enfrentamiento al phishing en las redes de datos. *Revista Cubana de Ciencias Informáticas*, 15:413 – 441, 00 2021.

[14] Ankit Kumar Jain and B.B. Gupta. Phish-safe: Url features-based phishing detection system using machine learning. In *Cyber Security: Proceedings of CSI 2015*, pages 467–474. Springer, 2018.

[15] Mehmet Korkmaz, Ozgur Koray Sahingoz, and Banu Diri. Detection of phishing websites by using machine learning-based url analysis. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2020.

[16] V. Krishnaiah, G. Narsimha, and N. Subhash. Survey of classification techniques in data mining. *International Journal of Computer Sciences and Engineering*, 2(9):65–74, 2014.

[17] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386, 2020.

[18] Rami Mohammad and Lee McCluskey. Phishing Websites. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C51W2X.

[19] Rami M. Mohammad, Lee McCluskey, and Fadi Thabtah. Phishing websites. 2015.

[20] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[21] Sagar S. Nikam. A comparative study of classification techniques in data mining algorithms. *Oriental Journal of Computer Science and Technology*, 8(1):13–19, 2015.

[22] F.Y. Osisanwo, J.E.T. Akinsola, O. Awodele, J.O. Hinmikaiye, O. Olakanmi, and J. Akinjobi. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3):128–138, 2017.

[23] Yi Peng, Gang Kou, Yong Shi, and Zhengxin Chen. A descriptive framework for the field of data mining and knowledge discovery. *International Journal of Information Technology & Decision Making*, 7(04):639–682, 2008.

[24] Thair Nu Phyu. Survey of classification techniques in data mining. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 727–731. Citeseer, 2009.

[25] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais. Catchphish: detection of phishing websites by inspecting urls. *Journal of Ambient Intelligence and Humanized Computing*, 11:813–825, 2020.

[26] Sebastian Raschka. *Python machine learning*. Packt publishing ltd, 2015.

[27] M.M. Rodríguez-Hernández, R.E. Pruneda, and J.M. Rodríguez-Díaz. Statistical analysis of the evolutive effects of language development in the resolution of mathematical problems in primary school education. *Mathematics*, 9(10):1081, 2021.

[28] Anna Rotondo and Fergus Quilligan. Evolution paths for knowledge discovery and data mining process models. *SN Computer Science*, 1(2):109, 2020.

[29] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2018.

[30] Vahid Shahrivari, Mohammad Mahdi Darabi, and Mohammad Izadi. Phishing detection using machine learning techniques. *arXiv:2009.11116*, 2020.

[31] Xiaoling Shu and Yiwan Ye. Knowledge discovery: Methods from data mining and machine learning. *Social Science Research*, 110:102817, 2023.

[32] Aized Amin Soofi and Arshad Awan. Classification techniques in machine learning: applications and issues. *Journal of Basic & Applied Sciences*, 13(1):459–465, 2017.

[33] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data classification: Algorithms and applications*, page 37, 2014.

[34] Lizhen Tang and Qusay H Mahmoud. A survey of machine learning-based solutions for phishing website detection. *Machine Learning and Knowledge Extraction*, 3(3):672–694, 2021.

[35] Melanie Volkamer, Karen Renaud, Benjamin Reinheimer, and Alexandra Kunz. User experiences of torpedo: Tooltip-powered phishing email detection. *Computers & Security*, 71:100–113, 2017.

[36] Wei Wei, Qiao Ke, Jakub Nowak, Marcin Korytkowski, Rafał Scherer, and Marcin Woźniak. Accurate and fast url phishing detector: a convolutional neural network approach. *Computer Networks*, 178:107275, 2020.

A. Anexos

A.1. Definición de bibliotecas de Python

```
# BIBLIOTECAS BÁSICAS
import pandas as pd
import sklearn
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from scipy.stats import randint
from scipy.stats import uniform
from openpyxl import load_workbook
## PREPROCESAMIENTO
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from imblearn.over_sampling import RandomOverSampler
##  BIBLIOTECA - SCIKIT LEARN
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.model_selection import cross_val_score, StratifiedKFold, cross_val_predict, cross_validate, KFold
# MÉTODOS DE SELECCION DE CARACTERÍSTICAS - SCIKIT LEARN
from sklearn.feature_selection import RFECV
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
# CLASIFICADORES - SCIKIT LEARN
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier, ExtraTreesClassifier
# MÉTRICAS - SCIKIT LEARN
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report, roc_auc_score, cohen_kappa_score, roc_curve
## - REPRODUCIBILIDAD
import random as rnd
seed = 42
np.random.seed(seed)
```

A.2. Código para Preprocesamiento en Python

```python
# Leyendo el archivo .csv con Pandas
df = pd.read_csv('OpenML_11000_31.csv', delimiter=',')

# Fijar la semilla aleatoria para garantizar la reproducibilidad
random_seed = 42

# Mezclar los datos de manera reproducible
df = sklearn.utils.shuffle(df, random_state=random_seed)

# Reemplazando los valores de Clase -1 por 1 (Positiva) y los valores de 1 por Cero (Negativa)
df['Result'] = df['Result'].replace({-1: 1, 1: 0})

# Dividiendo el conjunto de datos en características (X) y etiquetas (y)
X = df.drop(["Result"], axis=1)
y = df["Result"]

# Definir los datos de conjuntos de entrenamiento y prueba
# random_state=random_seed para garantizar la reproducibilidad en la división
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=random_seed)
```

A.3. Código para entrenamiento y validación de modelos

```
# Definir el modelo Random Forest con semilla para reproducibilidad
model = RandomForestClassifier(random_state=42)

# Definir la estrategia de validación cruzada con 5 folds
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Realizar validación cruzada y obtener las predicciones
y_pred_train = cross_val_predict(model, X_train, y_train, cv=cv)

# Calcular las métricas
accuracy_scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
accuracy = accuracy_scores.mean()
accuracy_std = accuracy_scores.std()
precision = precision_score(y_train, y_pred_train)
recall = recall_score(y_train, y_pred_train)
roc_auc = roc_auc_score(y_train, y_pred_train)
kappa = cohen_kappa_score(y_train, y_pred_train)
f1 = f1_score(y_train, y_pred_train)

# Crear un DataFrame para almacenar las métricas
metrics_df = pd.DataFrame(columns=["Accuracy", "Accuracy Desviación Estándar", "Precision", "Recall", "F1-score", "ROC AUC", "Cohe

# Agregar una fila con las métricas al DataFrame
metrics_df.loc[0] = [accuracy, accuracy_std, precision, recall, f1,roc_auc, kappa]

# Guardar el DataFrame en un archivo Excel
excel_filename = "Prueba1_rf.xlsx"
metrics_df.to_excel(excel_filename, index=False)

# Calcular y mostrar la matriz de confusión
conf_matrix = confusion_matrix(y_train, y_pred_train)

# Cargar el archivo Excel y agregar la matriz de confusión
book = load_workbook(excel_filename)
writer = pd.ExcelWriter(excel_filename, engine='openpyxl')
writer.book = book
confusion_df = pd.DataFrame(np.flip(conf_matrix.T), columns=['Clase 1 Real', 'Clase 0 Real'], index=['Clase 1 Predicha', 'Clase 0
confusion_df.to_excel(writer, sheet_name='Matriz de Confusión')
```

A.4. Código para remuestrear clases

```
# Leyendo el archivo .csv con Pandas
df = pd.read_csv('OpenML_11000_31.csv', delimiter=',')

# Fijar la semilla aleatoria para garantizar la reproducibilidad
random_seed = 42

# Mezclar los datos de manera reproducible
df = sklearn.utils.shuffle(df, random_state=random_seed)

# Reemplazando los valores de Clase -1 por 1 (Positiva) y los valores de 1 por Cero (Negativa)
df['Result'] = df['Result'].replace({-1: 1, 1: 0})

# Dividiendo el conjunto de datos en características (X) y etiquetas (y)
X = df.drop(["Result"], axis=1)
y = df["Result"]

# Crear un objeto de sobremuestreo
oversampler = RandomOverSampler(random_state=42)

# Aplicar el sobremuestreo al conjunto de datos
X_resampled, y_resampled = oversampler.fit_resample(X, y)

# Definir los datos de conjuntos de entrenamiento y prueba
# random_state=random_seed para garantizar la reproducibilidad en la división
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=random_seed)
```

A.5. Código para optimización de modelos

```
rf_model = RandomForestClassifier(random_state=42)

# Definir la cuadrícula de hiperparámetros para la búsqueda aleatoria

param_grid = {
    'n_estimators': randint(10, 500),   # Número de árboles en el bosque
    'max_depth': randint(1, 30),        # Profundidad máxima de los árboles
    'min_samples_split': randint(2, 20) # Mínimo de muestras requeridas para dividir un nodo interno
}

# Definir la estrategia de validación cruzada
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Realizar búsqueda aleatoria de hiperparámetros con RandomizedSearchCV
random_search = RandomizedSearchCV( rf_model,
                                    param_distributions=param_grid,
                                    n_iter=100,
                                    scoring='accuracy',
                                    cv=cv,
                                    random_state=42)

# Ajustar el modelo a los datos
random_search.fit(X_train, y_train)

# Obtener los mejores hiperparámetros y el mejor score
best_params = random_search.best_params_
best_score = random_search.best_score_

# Imprimir los resultados
print("Mejores hiperparámetros:", best_params)
print("Mejor score:", best_score)
```

A.6. Código para selección de características: RFECV

```
# Definir el clasificador
classifier = ExtraTreesClassifier(random_state=42)

# Definir la estrategia de validación cruzada
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Crear el objeto RFECV
rfecv = RFECV(estimator=classifier, step=1, cv=cv, scoring='accuracy')

# Ajustar el objeto RFECV al conjunto de datos
rfecv.fit(X, y)

# Obtener las características seleccionadas usando la máscara booleana
selected_feature_names = X.columns[rfecv.support_]

# Obtener el accuracy promedio utilizando validación cruzada
scores = cross_val_score(rfecv, X, y, cv=cv, scoring='accuracy')
mean_accuracy = scores.mean()

# Imprimir los resultados
print("Características seleccionadas:")
print(selected_feature_names)
print("Accuracy promedio:", mean_accuracy)
```

```
Características seleccionadas:
Index(['having_IP_Address', 'URL_Length', 'Prefix_Suffix', 'having_Sub_Domain',
       'SSLfinal_State', 'Domain_registeration_length', 'HTTPS_token',
       'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH',
       'Submitting_to_email', 'popUpWidnow', 'age_of_domain', 'DNSRecord',
       'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page'],
      dtype='object')
Accuracy promedio: 0.9717774762550881
```

A.7. Código para selección de características: SFS

```
# Definir el modelo RandomForest y la estrategia de validación cruzada
RandomForest_model = RandomForestClassifier(max_features= 3, criterion= 'entropy', n_estimators=300, random_state=42)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Crear el objeto SFS
sfs_rf = SFS(RandomForest_model,
             k_features=30,
             forward=True,
             floating=False,
             verbose=2,
             scoring='accuracy',
             cv=cv)

# Realizar la selección de características hacia adelante
sfs_rf = sfs_rf.fit(X_train, y_train)
```