ColloQUIUM
Editorial - Centro de Formación

Sistema de detección de phishing basado en machine learning

Juan Elias Villegas Cubas
Oscar Efraín Capuñay Uceda
Gisella Luisa Elena Maquen Niño
Denny John Fuentes Adrianzén
Pepe Humberto Bustamante Quintana

Juan Elias Villegas Cubas
Universidad Nacional Pedro Ruiz Gallo – Lambayeque, Perú
https://orcid.org/0000-0001-7026-9767
jvillegasc@unprg.edu.pe
Ingeniero de Sistemas, Magister en Ingeniería de Sistemas, Doctor en Ciencias de la Computación y Sistemas. Docente nombrado en la Universidad Nacional Pedro Ruiz Gallo de Lambayeque – Perú, adscrito al Departamento Académico de Ingeniería de Sistemas.

Oscar Efraín Capuñay Uceda
Universidad Nacional Pedro Ruiz Gallo – Lambayeque, Perú
https://orcid.org/0000-0002-4145-6309
ocapunayu@unprg.edu.pe
Ingeniero de Sistemas, Magister en Ingeniería de Sistemas, Doctorando en Ciencias Matemáticas. Docente nombrado en la Universidad Nacional Pedro Ruiz Gallo de Lambayeque – Perú, adscrito al Departamento Académico de Ingeniería de Sistemas.

Gisella Luisa Elena Maquen Niño
Universidad Nacional Pedro Ruiz Gallo – Lambayeque, Perú
https://orcid.org/0000-0002-9224-5456
gmaquenn@unprg.edu.pe
Ingeniero en Computación e Informática, Licenciada en Educación con especialidad en Matemática y Computación. Maestra en Ciencias de la Educación con mención en tecnologías de la información e informática educativa. Doctora en Ciencias de la Educación con mención en Administración de la Educación. Doctora en Ciencias de la Computación y Sistemas. Docente nombrada en la Universidad Nacional Pedro Ruiz Gallo de Lambayeque – Perú, adscrita al Departamento académico de Computación y Electrónica.

Denny John Fuentes Adrianzén
Universidad Nacional Pedro Ruiz Gallo – Lambayeque, Perú
https://orcid.org/0000-0003-4864-1352
dfuentesad@unprg.edu.pe
Ingeniero Informático y de Sistemas, Maestro en Administración con Mención en Gerencia Empresarial, Doctor en Ciencias de la Computación y Sistemas. Además, con Estudios Concluidos en la Maestría en Gestión Pública por EUCIM Business School -USMP. Docente Nombrado en la Universidad Nacional Pedro Ruiz Gallo de Lambayeque – Perú, adscrito al Departamento Académico de Computación y Electrónica.

Pepe Humberto Bustamante Quintana
Universidad Señor de Sipán – Chiclayo, Lambayeque, Perú
https://orcid.org/0000-0001-9842-8432
bhumberto@crece.uss.edu.pe
Ingeniero de Sistemas, Licenciado en Administración, Magister en Administración de Negocios MBA – EXECUTIVE, Doctor en Ciencias de la Educación. Docente a tiempo Completo en la Universidad Señor de Sipán de Chiclayo, Lambayeque – Perú, adscrito a la Escuela de Posgrado.

Sistema de detección de phishing basado en machine learning

Juan Elias Villegas Cubas
Oscar Efraín Capuñay Uceda
Gisella Luisa Elena Maquen Niño
Denny John Fuentes Adrianzén
Pepe Humberto Bustamante Quintana

ColloQUIUM
Editorial - Centro de Formación

Juan Elias Villegas Cubas
Oscar Efraín Capuñay Uceda
Gisella Luisa Elena Maquen Niño
Denny John Fuentes Adrianzén
Pepe Humberto Bustamante Quintana

Sistema de detección de phishing
basado en machine learning

Editado por Colloquium
ISBN: **978-9942-600-33-2**
Primera edición 2022

La obra fue revisada por pares académicos antes de su proceso editorial, en caso de requerir certificación debe solicitarla a: sbores@colloquium-editorial.com.

Ecuador 2022

Índice

Introducción 6

I. La detección de phishing y machine learning 7

1.1 Realidad Problemática. 7

1.2 Trabajos Previos 10

1.3 Ciberseguridad y detección de phishing 16

1.4 Machine Learning 18

1.5 Inteligencia de Amenazas 25

II. Sistema de detección de sitios web phishing 27

2.1 Modelo de Machine Learning 27

2.2 Fases del sistema de detección de phishing 35

III. Implementación del sistema de detección de sitios web phishing 39

3.1 Fase 1. Recolección de datos. 40

3.1.1 Recolección de datos del sitio web. 40

3.1.2 Recolección de datos de inteligencia de amenazas. 44

3.2 Fase 2. Preparación de los datos. 47

3.2.1 Análisis de los datos. 47

3.2.2 Preprocesamiento de datos. 62

3.2.3 Remuestreo de los datos. 65

3.3 Fase 3. Selección de algoritmos. 66

3.4 Fase 4. Entrenamiento. 69

3.5 Fase 5. Detección de Phishing. 74

3.6 Fase 6. Evaluación del rendimiento. 75

IV. Conclusiones 78

V. Referencias 80

Introducción

Las redes informáticas se han convertido en una herramienta importante para todo tipo de organizaciones. Sin embargo, el crecimiento del uso de las redes informáticas e internet ha conllevado al crecimiento del número de ciberataques, que afectan a la intimidad e integridad de los datos confidenciales de las personas, organizaciones y gobiernos. Uno de los ciberataques más comunes que crece continuamente en el mundo es el phishing; en el Perú según la División de Investigación de Delitos de Alta Tecnología, se ha duplicado el número de denuncias de ciberdelitos en el 2021, siendo la modalidad más frecuente el phishing.

Se han desarrollado modelos de seguridad para la detección de phishing, pero son insuficientes, según reportes de seguridad de Trend Micro y Kaspersky la detección de phishing de los modelos actuales representan una menor cantidad de detección de años anteriores y además el 50% de profesionales consideran que son ineficientes las técnicas utilizadas. En este trabajo de investigación se desarrolla un sistema eficiente en la detección de phishing, basado en modelos de machine learning.

Este trabajo está estructurado en tres partes. En la primera parte se revisa las teorías relacionadas a la detección de phishing y machine learning, se analiza la realidad problemática de la detección de phishing, los trabajos previos, las teorías de la ciberseguridad, la detención de phishing, el ciclo de vida y las técnicas de machine learning; y la inteligencia de amenazas. En la segunda parte se describe el sistema de detección de phishing, se detalla el modelo de machine learning en que se sustenta el sistema y se describe las fases del desarrollo del sistema de detección de phishing. En la tercera parte se detalla la implementación del sistema de detección de phishing; utilizando Python como lenguaje de programación y siguiendo paso a paso las seis fases del sistema desde la recolección de datos, hasta la evaluación del rendimiento del sistema en la detección de sitios web phishing.

I. La detección de phishing y machine learning

1.1 Realidad Problemática.

En la actualidad muchas aspectos de nuestra vida cotidiana han cambiado con el uso de las Tecnologías de la Información y Comunicación (TIC); la mayoría de personas lleva siempre un Smartphone, utiliza el correo electrónico, envía mensajes de texto y realiza videoconferencias para comunicarse; se puede hacer transferencias de dinero, compras en línea a través de un dispositivo conectado, ha cambiado la forma de aprender, muchas personas trabajan desde su casa a través de una conexión y reciben atención médica mediante una conexión virtual.

Las redes informáticas se han convertido en una herramienta importante para todo tipo de organizaciones, como instituciones financieras, médicas, industriales, de transporte, educativas, etc. Sin embargo, el crecimiento del uso de las redes e internet ha conllevado a mejorar la protección frente a los ciberataques.

Los ciberataques afectan a la intimidad e integridad de los datos de las personas, organizaciones y gobiernos, causan cuantiosas pérdidas económicas, según (Singh & Sharma, 2019) aproximadamente 2 billones de dólares de pérdidas en el 2019 debido a los ciberataques. El ciber crimen es tema de mucha importancia tanto que el FBI (Departamento de Justicia de los Estados Unidos) lo considera al mismo nivel que el terrorismo o la de contra inteligencia (FBI, 2019).

En la literatura se encuentra varios tipos de ciberataques, según (Bendovschi, 2015) los principales ciberataques son: del hombre en el medio, de fuerza bruta, de denegación de servicio distribuido, ransomware y phishing.

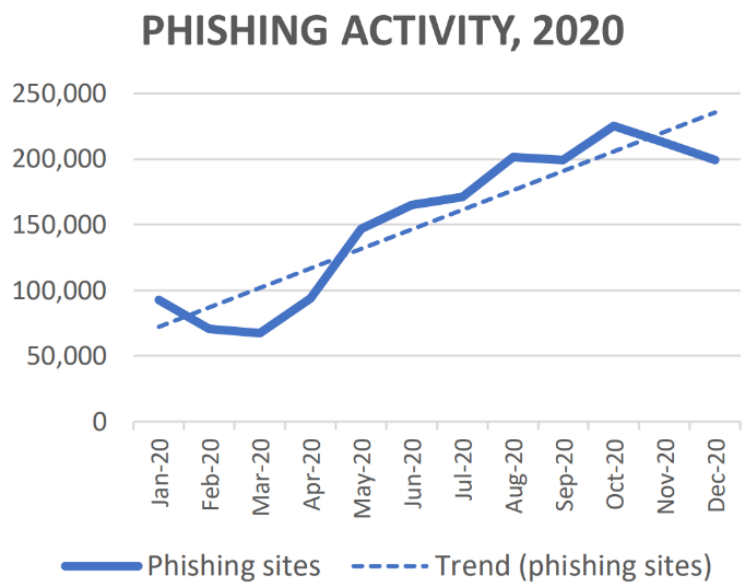
Los ciberataques han afectado a todo tipo de organizaciones para obtener información confidencial y privilegiada de sus clientes, empleados y del negocio, incluso se le asigna la muerte de una persona debido a un ciberataque ransomware (Clarin, 2020).

Los ciberataques cada día van creciendo en número, tamaño y velocidad, según (Medina & Molist, 2017) los ataques ransomware crecieron en 800%, el Phishing creció hasta el 100% en algunos meses; entre otros ataques en crecimiento son la denegación de servicio y el robo de datos personales.

Según Modi (2019) cerca de cuatro millones de ataques de denegación de servicio distribuido en seis meses, el crecimiento en número de ataques es del 39%, el tamaño en los ataques creció del 776% entre 100Gbps y 400Gbps.

Según APWG (2021) el número de ataques phishing crece continuamente, alcanzando un récord en crecimiento en el mes de octubre del 2020 con 225,304 ataques phishing, como se muestra en la figura 1.

Figura 1. Ataques phishing en el 2020. Fuente (APWG, 2021)



Entre las principales manifestaciones del problema del phishing se muestran a continuación.

- El 84% de profesionales en Estados Unidos indican que han sufrido al menos un tipo de incidente de seguridad y ponen a los ataques

phishing y ransomware, como los tipos de ataques con mayor ocurrencia. (Trend Micro, 2021)

- El 31% de empresas en Latinoamérica han percibido un aumento de los ciberataques en el 2020 y la principal amenaza de la ciberseguridad son los eventos relacionados al phishing. (Marsh & Microsoft, 2020)
- El número de ataques phishing crecieron continuamente y hasta se duplicaron a lo largo del 2020, llegando en octubre a 225,304 ataques phishing. (APWG, 2021)
- Los países de Latinoamérica que fueron más afectados con los ataques phishing durante el 2020 son las empresas de Brasil (26,4%), seguidos por las empresas de Perú (22,8%) y luego por las empresas de México (12%). (Eset, 2021)
- En el Perú hasta octubre del 2021, se eleva el número de denuncias de ciberdelitos, siendo la modalidad más frecuente el phishing, que en el 2021 se duplicaron con respecto al 2020. (Divindat , 2021)
- En el Perú, entre enero a septiembre del 2021, se han generado más de300 alertas integradas de Seguridad Digital, y en la mayoría de las alertas generadas se ha reportado a los ataques phishing. (PECERT, 2021)
- Según (Trend Micro, 2021) el 50% de los profesionales en Estados Unidos consideran ineficiente las técnicas de hacer frente al phishing y al ransomware.
- El 65% de envíos de phishing ingresan a las bandejas de los usuarios. (Trend Micro, 2021).
- Además, el 65% de los usuarios hacen clic en los enlaces phishing. (Trend Micro, 2021).
- Los sistemas de detección de phishing en el 2020 neutralizaron 434 898,635 phishing, sin embargo, representa una menor cantidad de phishing detectado en comparación con el año 2019. (Kaspersky, 2021)

1.2 Trabajos Previos

Los primeros trabajos para la detección de intrusos en las redes informáticas estaban relacionados con las auditorias de seguridad, que consistía en la revisión manual de las actividades de los usuarios.

Anderson (1972) documentó un sistema de clasificación que distingue entre ataques internos y los ataques externos, basándose en los accesos de un usuario tiene o no tiene para ingresar a un ordenador, este trabajo se considera como el primero que habla de la detección de intrusiones y que remplazaba a las auditorias de seguridad.

Denning (1987) propone un Sistema Experto de Detección de Intrusiones (IDES) en tiempo real con la capacidad de detectar robos, penetraciones y otras formas de abuso informático. Su propuesta se basa en el monitoreo de los registros de auditoría, que permite detectar patrones anormales en el uso del sistema.

Heberlein (1995) en la Universidad de California desarrollaron el "Network System Monitor" (NSM), un Sistema de Detección de Intrusos con capacidad de monitoreo de red. El funcionamiento del NSM es la base de muchos de los sistemas de detección de intrusos de red que se utilizan hoy en día.

Rami, y otros (2014) indican que se puede combatir el phishing con soluciones legales, educación y soluciones técnicas. Desarrollaron y evaluaron una red neuronal con 500 épocas, utilizaron un conjunto de datos con 1400 sitios web entre phishing y legítimos; y obtuvieron un accuracy de 92.48%.

Mohammad, y otros (2014) indican que para combatir el phishing se puede hacer mediante soluciones legales, con capacitaciones a los usuarios y con soluciones técnicas; y proponen un modelo para predecir ataques phishing basado en redes neuronales artificiales. Experimentan

con un conjunto de datos de 19 características y obtienen un rendimiento de accuracy máximo en el proceso de entrenamiento de 94.07%, utilizando 1000 épocas.

De la Hoz (2016) propone un enfoque PCA/FDR para la detección de ataques en redes, aplica el poder discriminante para la selección de características. Obtiene un alto rendimiento en la clasificación de los ataques usando el ajuste de las probabilidades de activación previa, usa las métricas de a precisión de la clasificación, o la sensibilidad.

Jain & Gupta (2017) precisan que la detección de phishing se realiza mediante la educación a los usuarios y mediante el uso de software; las técnicas basadas en software son el uso de listas negras, similitud visual, motores de búsqueda y aprendizaje automático. Proponen un modelo anti-phishing que extrae los datos solamente del lado cliente, utilizando las características de la url y el código fuente HTML, evalúan varios algoritmos de aprendizaje automático con un conjunto de datos de 2141 phishing y 1918 sitios web legítimos, obtienen un accuracy máximo con Random Fortest.

Yi, y otros (2018) presentan dos tipos de funciones para la detección de sitios web phishing, que una es la característica original y la otra es las características de interacción. Con un conjunto de datos pequeño los parámetros adecuados y Luego entrenan el modelo DBN y evalúan DBN obteniendo el 89.20% de accuracy.

Feng, y otros (2018) proponen un modelo para la clasificación de sitios web phishing aplicando redes neuronales, además evaluaron el rendimiento de los algoritmos como Naive Bayes, regresión logísticas, arboles de decisión, LSVM, RSVM y análisis de discriminante líneas (LDA). Utilizaron el conjunto de datos con acceso público en el repositorio UCI, obteniendo un resultado de accurracy de 97.71%.

Jain & Gupta (2018) proponen un sistema para detectar url phishing denominado PHISH-SAFE basado en aprendizaje automático y en las

características de la URL, el modelo es entrenado con un conjunto de datos de más de 33000 direcciones URL, con 14 características URL seleccionadas; usaron los clasificadores SVM y Naive Bayes, obtuvieron un accuracy de mas de 90% mediante el algoritmo SVM.

Niakanlahiji, y otros (2018) proponen PhishMon, un marco de aprendizaje automático con funciones para detectar páginas web de phishing. Se basa en un conjunto de datos de quince características que se pueden calcular de manera eficiente desde una página web sin requerir servicios de terceros, como motores de búsqueda o servidores de WHOIS. Estas funciones capturan varias características de las aplicaciones web legítimas, así como sus infraestructuras web subyacentes. A través de una evaluación de un conjunto de datos que consta de 4800 phishing distintos y 17,500 páginas web benignas distintas, demuestran que PhishMon puede distinguir el phishing invisible de las páginas web legítimas con un grado muy alto de precisión. En los experimentos, PhishMon logró una acurracy del 95,4%

Abutair, y otros (2018) proponen un sistema de detección de phishing de razonamiento basado en casos (CBR-PDS) que se basa en casos anteriores para detectar ataques de phishing y que puede adaptarse para detectar nuevos ataques de phishing. Evalúan al modelo propuesto con varios conjunto de datos y obtienen un rendimiento de accuracy máximo de 96.26%.

Patil, y otros (2018) utilizan tres enfoques para la detección de sitios web phishing, el primero es analizando las características de la URL, el segundo es verificando la legitimidad del sitio web y el tercero basado en la apariencia visual verificando la autenticidad del sitio web. Evalúan los datos con los algoritmos regresión logística, arboles de decisión y random forest, obteniendo un rendimiento de accuracy máximo de 96.58% con Random Forest.

Wei, y otros (2019) diseñaron un sensor para la detección de phishing aplicando técnicas de aprendizaje profundo, utilizaron un conjunto de

datos de 1’523,966 direcciones URL con sitios legítimos y sitios phishing. El modelo propuesto fue un Red Neuronal Profunda y logró tener una tasa de detección real de 86.63% de accuracy.

Kumar y otros (2019) utilizan el conjunto de datos de código abierto Kaggle, realiza la extracción de las características y trabaja con los siguientes parámetros: Longitud de la URL, dirección IP, subdominio, uso de HTTPs, tráfico del sitio web, SVM, puntos, SSL y vectores de características. El modelo propuesto evalúa diferentes algoritmos como Naive Bayes, Random Forest, KN vecino, obteniendo mejores resultados de clasificación con el algoritmo SVM.

Ubing, y otros (2019) utilizan la selección de características y se integra con voting y se compara con diferentes modelos de clasificación, incluidos el Random Forest y regresión logística. Usan a la estructura y componentes de la URL, y el conjunto de datos con acceso público en el repositorio UCI, obteniendo un accuracy de 95%.

Wang, y otros (2019) proponen un modelo denominado PDRCNN para la detección de sitios web phishing utilizando únicamente las características de la dirección URL del sitio web. Combinan dos tipos de redes neuronales y generan un conjunto de datos de casi 500,000 URL obtenidas a través de Alexa y PhishTank. Los resultados experimentales muestran que el modelo propuesto PDRCNN logra un rendimiento de accuracy de 95.79% en la detección de phishing.

Zabihimayvan & Doran (2019) aplican la teoría Fuzzy Rough Set (FRS) como una herramienta para seleccionar las características más efectivas de tres conjuntos de datos. Y se seleccionan tres clasificadores para entrenar y validar con un conjunto de datos de 14000 direcciones de sitios web y lograron obtener un accuracy máximo de 95% con el clasificador Random Forest.

Sountharrajan, y otros (2019) utilizan las características de las direcciones URL seleccionadas de forma dinámica que dependen del tipo de

aprendizaje. Evalúan los resultados aplicando las técnicas de aprendizaje profundo DBM y SAE red neuronal profunda; siendo el modelo de aprendizaje profundo la que disminuye la tasa de falsos positivos, y brindando mejores resultados.

Kulkarni & Brown (2019) indican que debido a que los humanos son tan susceptibles a ser engañados, es necesario contar con métodos automatizados para diferenciar un sitio web legítimo de uno falso o phishing. Desarrollan un sistema utilizando técnicas de aprendizaje automático como el Árbol de decisión, Naive Bayes, SVM, y una red neuronal para clasificar los sitios web en función de la URL. Probaron un conjunto de datos con 1353 direcciones URL del mundo real y lograron obtener un mejor rendimiento de acurracy de 91.5%, con los árboles de decisión.

Abdulhamit & Kremicb (2020) comparan algoritmos de machine learning en modo simple, en modo Adaboost y modo MultiBoosting, para la detección de sitios web phishing, usa el conjunto de datos con acceso público en el repositorio UCI y la evaluación da como resultado que el Adaboost con Maquinas de Vectores de Soporte brinda los mejores resultados de accuracy.

Christou, y otros (2020) consideran que incluso con la formación adecuada y una alta conciencia puede resultar difícil que un usuario pueda clasificar adecuadamente una página que visita como sitio phishing. También indican que la detección tradicional con listas de bloqueo y el análisis de contenido requiere mucho tiempo y verificación humana. Desarrollan un sistema de filtrado predictivo de los sitios web phishing, con algoritmos de Maquinas de Vectores de Soporte, Random Forest, evalúan el rendimiento del sistema con un conjunto de datos propios y obtienen un rendimiento máximo de 90% con el algoritmo Maquinas de Vectores de Soporte.

Chavan (2020) proponen utilizan los algoritmos de regresión logísticas, arboles de decisión, Random Forest, KN vecinos, SVM y redes

neuronales artificiales, para evaluar el rendimiento con el conjunto de datos disponible en kagle con 1782 registros y 19 características; y obtienen un rendimiento máximo de 96% de rendimiento con Random Forest.

Zamir, y otros (2020) realizan una comparación de enfoques de aprendizaje automático supervisado y modelado de apilamiento para detectar sitios web phishing. Propone características con PCA y apilan mecanismos de machine learning como Maquinas de Vectores de Soporte, Naive Bayes, Random Forest, KN vecinos, Bagging y redes neuronales; usa el conjunto de datos disponible en Kaggle con información de 11055 sitios web y con 32 atributos; y obtiene un rendimiento máximo de accuracy de 97.4%.

Shahrivari, y otros (2020) indican que las formas para evitar los ataques phishing es capacitando al usuario a que estén preparados para ataques phishing futuros, se trata de un método preventivo y se capacita a los usuarios a distinguir entre los sitios web phishing y los sitios web legítimos; sin embargo, los usuarios tienden a olvidarse y a cometer errores; la otra forma es mediante un sistema de detección automatizado que advierta al usuario, y desarrollan un sistema de clasificación de phishing, utilizando un conjunto de datos con 6157 sitios web legítimos y 4898 sitios web phishing y evalúan doce algoritmos de machine learning.

Alsariera, y otros (2020) proponen cuatro modelos de meta aprendizaje basados todos ellos en el algoritmo Extra Tree: los modelos evaluados son AdaBoost (ABET), Bagging (BET), Rotation Forest (RoFBET) y LogitBoost (LBET), evalúan el desempeño de los mismos y obtienen un accuracy de no menor de 97%, y un rendimiento máximo de 97.404% con el modelo BET.

Opara, y otros (2020) proponen HTMLPhish, un enfoque de clasificación de páginas web de phishing basado en datos y basado en aprendizaje profundo. Específicamente, HTMLPhish recibe el contenido del documento HTML de una página web y emplea redes neuronales

convolucionales (CNN) para aprender las dependencias semánticas en el contenido textual del HTML. Realizan realizan experimentos integrales en un conjunto de datos de más de 50.000 documentos HTML y que arroja un rendimiento superior al 93%.

1.3 Ciberseguridad y detección de phishing

El proceso de seguridad informática se considera como parte del proceso de seguridad de la información, y según la Organización Internacional de Estándares (ISO27000, 2017) tiene como propósito la protección de los riesgos, logrando que estos sean conocidos, asumidos, gestionados y minimizados por toda organización.

En los últimos años, está tomado mayor importancia al proceso de la ciberseguridad que según (ISACA, 2015) consiste en el proceso de proteger a los activos de información de una organización, cuando es procesada, almacenada y transmitida en dispositivos digitales y en las redes informáticas de las amenazas.

La Ciberseguridad contempla las diferentes normas, prácticas, herramientas y conceptos que se relacionan con la seguridad de la información y la seguridad TI operacional. Es decir, la ciberseguridad, está considerada como parte de la seguridad de la información y se orienta a la protección de los activos de información en formato digital que se transmiten a través de sistemas interconectados.

La Organización Internacional de Normalización ha publicado el estándar ISO/IEC 27032 para la ciberseguridad (ISO, 2017) . Su objetivo es de facilitar directrices para mejorar el estado de la ciberseguridad en infraestructuras críticas.

El Instituto Nacional de Estándares y Tecnología ha publicado un Marco de Trabajo de Ciberseguridad para infraestructuras críticas. El Framework NIST se compone de tres elementos principales: El marco Core, los niveles de implementación Tiers y los perfiles del marco (Profiles). El marco principal (Core) emplea cinco funciones

fundamentales de la ciberseguridad que son: Identificar, Proteger, Detectar, Responder y Recuperar (NIST, 2018).

Para el NIST la detección de intrusos es en proceso de monitorear eventos que ocurren en sistemas de computación o redes, con la finalidad de detectar signos de posibles incidentes como pueden ser violaciones, amenazas de violación de políticas de seguridad o uso de recursos en forma abusiva (Scarfone & Mell, 2007).

Se han propuesto varias definiciones de Phishing, investigadores e instituciones de ciberseguridad siguen discutiendo referido al tema que sigue evolucionado de acuerdo con la función y al contexto de aplicación.

Phishing es el proceso de engaño a los usuarios de las redes informáticas para que divulguen información confidencial para fines nefastos. Los ataques phishing se realizan comúnmente por correos masivos hasta millones de direcciones de destinatarios o ataques informáticos altamente direccionados a usuarios específicos. (Ollmann, 2004).

El phishing es una actividad fraudulenta definida como la creación de una página web falsa pero muy similar a la existente con la finalidad de engañar a un usuario para que ingrese datos personales, financieros o de contraseña (Merwe, Loock, & Dabrawski, 2005).

El phishing es una forma de ingeniería social en la que un atacante también conocido como “phisher” intenta obtener de forma fraudulenta información confidencial o sensible de usuarios legítimos, imitando de forma automatizada las comunicaciones electrónicas de una organización. (Jakobsson & Myers, 2006)

El phishing es un delito que se emplea las técnicas de ingeniería social y el engaño para robar identidad personal y las credenciales de las cuentas financieras de los usuarios. Los esquemas de ingeniería social se aprovechan de las victimas desprevenidas, engañándoles,

haciéndoles creer que están tratando con una parte legítima y de confianza. (APWG, 2021)

El phishing utiliza diversos vectores de ataque como ataques del hombre en el medio, registradores de claves y la falsificación completa de un sitio web. (Ollmann, 2004)

1.4 Machine Learning

Machine Learning, está definido según Gori (2018) como un tipo de Inteligencia artificial que proporciona a un sistema (hardware y/o software) la capacidad de aprender, sin ser explícitamente programadas.

El aprendizaje automático es un área de inteligencia artificial que permite que un sistema aprenda a partir de un conjunto de datos en lugar de a través de ser explícitamente programados (Hurwitz & Kirsch, 2018).

El aprendizaje automático según Mueller & Guido (2016) permite extraer conocimiento de los datos. Las técnicas de aprendizaje automático se han vuelto omnipresente en casi todas las actividades de nuestra vida cotidiana. Desde recomendaciones automáticas como por ejemplo qué películas elegir, qué tipo de comida consumir o qué productos adquirir, hasta reconocer a amigos en imágenes, muchas aplicaciones y dispositivos actuales utilizan algoritmos de aprendizaje automático en su funcionamiento.

Según Mueller & Guido (2016) son dos los enfoques para los problemas de aprendizaje; el aprendizaje supervisado y el aprendizaje no supervisado; sin embargo (Hurwitz & Kirsch, 2018) indica además al aprendizaje por refuerzo y al aprendizaje profundo como enfoques del aprendizaje automático.

El aprendizaje supervisado según Mueller & Guido (2016), se usa datos de entradas también llamadas características y salidas deseadas; y el algoritmo encuentra una manera de producir la salida deseada dada una

entrada especifica. El algoritmo puede crear o predecir una salida para datos de entrada sin la ayuda de una persona.

El aprendizaje supervisado según Hurwitz & Kirsch (2018) inicia con un conjunto de datos de características y una etiqueta o salida que brinda un significado (numérico o categórico) de los datos. El aprendizaje supervisado tiene como objetivo buscar y reconocer patrones en los datos analizados.

Los modelos de aprendizaje supervisado se aplican en una amplia variedad de problemas comerciales, como la detección de fraudes, detección de enfermedades, sistemas de recomendación o reconocimiento de voz.

En el aprendizaje no supervisado, el algoritmo solo recibe los datos de entrada o características y no se proporcionan las etiquetas de los datos de salida conocidos, es decir no se bridna el significado de los datos. (Mueller & Guido, 2016)

El aprendizaje no supervisado según Hurwitz & Kirsch (2018) se usa principalmente cuando el problema solo usa una gran cantidad de datos de solamente las características, sin etiquetar. Los algoritmos de aprendizaje no supervisados segmentan los conjuntos de datos en grupos de ejemplos (clústeres) o grupos de características. Los datos sin etiquetar crean los valores de los parámetros y la clasificación de los datos.

En el aprendizaje por reforzamiento según Hurwitz & Kirsch (2018) algoritmo recibe una retroalimentación por parte del usuario, para el análisis de los datos y obtener un mejor resultado; es decir, se aprende con estímulos de da un peso alto si está cerca del objetivo o un peso bajo si comete errores en el resultado.

El aprendizaje profundo o deep learning, es un tipo de aprendizaje que se basa en redes neuronales y se utiliza para problemas mas complejos como en los problemas de aprendizaje con imágenes.

Los algoritmos de aprendizaje automático supervisados se clasifican en dos tipos de problemas clasificación y regresión.

En la clasificación según Mueller & Guido (2016), el objetivo es predecir la salida del tipo categórico o de clase, que es una alternativa dentro de las posibilidades; si la lista de posibilidades es dos se denomina problemas de clasificación binaria, pero si la lista de posibilidad hay más de dos, entonces se denominada clasificación multiclase.

En la regresión según Mueller & Guido (2016), el objetivo es predecir la salida numérica, puede se continuo o de punto flotante. La predicción de los ingresos anuales de una persona a partir de algunas características como su educación, su edad y el lugar donde vive es una tarea de regresión.

El papel de los algoritmos en aprendizaje automático es muy importante; los algoritmos se definen son un conjunto de instrucciones secuenciales que debe realizar un sistema sobre cómo interactuar, manipular y transformar datos. Un algoritmo puede ser simple como realizar una operación matemática con dos números o tan complejo como reconocer un objeto en una imagen.

Los tipos de algoritmos de aprendizaje automático según (Hurwitz & Kirsch, 2018) son los Bayesianos, agrupación (clustering), arboles de decisión, reducción de la dimensionalidad, basados en instancias, regresión lineal, regularización, basado en reglas, redes neuronales y de aprendizaje profundo.

Entre los algoritmos más comunes para los problemas de aprendizaje supervisado según (Mueller & Guido, 2016) son:

- k-Nearest Neighbors. El algoritmo k-NN se basa en analizar los datos de los vecinos más cercanos para hacer una predicción de un nuevo punto de datos. El principal factor de análisis es el número de vecinos para obtener el mejor rendimiento. Se puede

implementar KNN para los problemas de clasificación y para problemas de regresión.

- Los modelos lineales. Son una clase de modelos que se utilizan ampliamente en la práctica y hacen una predicción utilizando una función lineal de las características de entrada. Los algoritmos de modelos lineales para problemas de regresión son Regresión lineal ordinario, Regresión Ridge y regresión LASSO. Y para problemas de clasificación son los algoritmos Regresión Logística y las Maquinas de vectores de soporte lineal (Lineal SVM)
- Clasificadores Naive Bayes. Los clasificadores Naive Bayes son una familia de clasificadores que son bastante similares a los modelos lineales, sin embargo, tienden a ser incluso más rápidos en el entrenamiento, pero su rendimiento generaliza y es ligeramente menor que las clasificaciones lineales.
- Arboles de decisión. Son modelos ampliamente utilizados para tareas de clasificación y regresión. Esencialmente, aprenden una jerarquía de preguntas si / si no, lo que lleva a una decisión. Por lo general, la construcción de un árbol y continuar hasta que todas las hojas sean puras conduce a modelos que son muy complejos.
- Ensamblados de árboles de decisión. Los conjuntos son métodos que combinan varios modelos de aprendizaje automático para crear modelos más potentes. Hay muchos modelos, hay dos modelos de conjunto que han demostrado ser efectivos en una amplia gama de conjuntos de datos para clasificación y regresión, los cuales usan árboles de decisión como Random Forest y árboles de decisión impulsados por gradientes.
- Kernelized Support Vector Machines. A menudo denominadas SVM, son una extensión que permite modelos más complejos que no están definidos simplemente por hiperplanos en el espacio de entrada. Hay máquinas de vectores de soporte para clasificación y regresión.

- Redes neuronales. Una familia de algoritmos conocida como "aprendizaje profundo". Los métodos más básicos son los perceptrones multicapa para clasificación y regresión, que pueden servir como punto de partida para métodos de aprendizaje profundo más complejos. Los perceptrones multicapa (MLP) también se conocen como redes neuronales de retroalimentación o, a veces, simplemente redes neuronales.

El ciclo de aprendizaje automático según (Hurwitz & Kirsch, 2018) es un proceso continuo y los pasos son los siguientes:

- Identificar los datos: Identificar las fuentes de datos relevantes es el primer paso del ciclo.
- Preparación los datos: consiste en realizar las actividades necesarias para asegúrese de que sus datos estén limpios, protegidos y gobernados. La importancia de este paso radica en que si una aplicación de aprendizaje automático aprende basado en datos con errores, la aplicación cometerá errores en las predicciones.
- Selección el algoritmo de aprendizaje automático: puede tener varios algoritmos aplicables a sus datos y en este paso consiste en seleccionar un algoritmo adecuado y que obtenga buen desempeño.
- Entrenar: Se refiere a entrenar un algoritmo con el conjunto de datos para crear el modelo.
- Evaluar: Se refiere a la evaluación de los modelos para elegir el algoritmo de brinda el mejor desempeño.
- Implementar: Consiste a implementar los algoritmos de aprendizaje automático credos.
- Predecir: después de la implementación, se pueden hacer predicciones basadas en datos de ingreso nuevos.
- Evaluar predicciones: Consiste en la evaluación de las predicciones realizadas por el modelo. La información que recopila al analizar la validez de las predicciones se retroalimenta

luego en el ciclo de aprendizaje automático para tratar de colaborar a mejorar la precisión.

Después de aplicar los algoritmos de aprendizaje automático, necesitamos medir el rendimiento o desempeño de las predicciones realizadas por el modelo. Contamos con un significativo número de métricas para medir el desempeño, Por lo tanto, para cada problema de aprendizaje automático, necesitamos utiliza métricas adecuadas para la evaluación del rendimiento.

Las métricas comunes de evaluación del desempeño de un modelo de clasificación de aprendizaje automático según (Vakili, Ghamsari, & Rezaei, 2020) y (Borja-Robalino, Monleón-Getino, & Rodellar, 2020) son la matriz de confusión, accuracy, precision, recall, f1-score, y ROC-AUC; que se describen a continuación:

- **Matriz de confusión.**

  Esta tabla de frecuencias es una de las métricas más intuitivas y descriptivas que se utilizan para encontrar la precisión y corrección de un algoritmo de aprendizaje automático. Su uso principal es en problemas de clasificación, la matriz se muestra en la tabla 1.

  Tabla 1: Matriz de confusión.

	Clases	**Resultado del clasificador**	
		Positivo	**Negativo**
Resultado real	Positivo	VP	FN
	Negativo	FP	VN

  Adaptado (Borja-Robalino, Monleón-Getino, & Rodellar, 2020)

  Dónde:

  VP: Verdaderos positivos, es el número correcto de predicciones que la instancia positiva.

  FP: Falsos Positivos, es el número incorrecto de predicciones que la instancia es positiva.

FN: Falsos Negativos, es el número incorrecto de predicciones que la instancia negativa.

VN: Verdaderos Negativos, es el número correcto de predicciones que la instancia negativa

- **Accuracy:**
  Es el más utilizado y quizás la primera opción para evaluar el desempeño de un algoritmo en problemas de clasificación. Se definirse como la relación entre elementos de datos clasificados con precisión y el número total de observaciones.
- **Classification error:**
  Es la proporción de instancias mal clasificadas sobre el conjunto total de instancias
- **Precision:**
  Muestra "qué número de elementos de datos seleccionados son relevantes". En otras palabras, de las observaciones que un algoritmo ha predicho que serán positivas, ¿cuántas de ellas son realmente positivas.
  La precisión se calcula dividiendo el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos:
- **Recall.**
  Llamado también Tasa de Verdaderos Positivos (TVP) o sensibilidad: Presenta “qué número de elementos de datos relevantes se seleccionan”. De hecho, de las observaciones que son realmente positivas, cuántas de ellas han sido predichas por el algoritmo. La sensibilidad es igual al número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos:
- **F1-Score**.

  Esta métrica, tiene en cuenta tanto la exactitud como la sensibilidad para calcular el rendimiento de un algoritmo; es la media armónica del accuracy y la recall.
- **Curva ROC y AUC.**

  La curva de características operativas del receptor o curva ROC de forma abreviada, es la curva que muestra la Tasa de Falsos Positivos (TFP) frente a la Tasa de Verdaderos Positivos (TVP). La curva ideal está cerca de 1 aparte superior izquierda, es decir que produzca un recall alta, mientras mantiene una Tasa de Falsos Positivos bajo. (Mueller & Guido, 2016)

1.5 Inteligencia de Amenazas

Gartner (2013) introduce y define el término “Threat intelligence” o Inteligencia de amenazas, que lo define como el conocimiento basado en evidencia, incluyendo su contexto, mecanismos, indicadores, implicaciones y acciones concretas, sobre la amenaza o peligro existente o emergente a los activos y que pueda ser usada para tomar decisiones informadas y acciones de respuesta por parte del afectado por la amenaza o peligro.

El tipo de información de inteligencia se clasifica en estratégica que se refiere a la inteligencia acerca de los riesgos y consecuencias asociadas a las amenazas que se usa para la toma de decisiones a alto nivel y direccionamiento de la estrategia. Puede ser táctica, que se refiere a la información de carácter técnica que normalmente contiene información específica de una dirección IP, URL, dominio, etc; y puede ser operacional que se refiere a la inteligencia que se enfoca en las técnicas, herramientas, metodologías de los adversarios.

La información de inteligencia de las amenazas y los agentes de amenazas proporcionan una comprensión suficiente para mitigar un evento dañino. Las fuentes de inteligencia pueden ser internas, se acceso libre, comerciales y organizacionales.

Actuar sobre la información de amenazas es el proceso de hacerla procesable localmente y que este conocimiento se distribuye tácitamente e informa directamente las interpretaciones futuras de la información de amenazas, lo que afecta la capacidad de las organizaciones para recibir alertas anticipadas sobre las amenazas, para contener el daño y aumentar la seguridad. Además, indican que la creación y utilización de conocimiento relevante es un proceso en el que los analistas de inteligencia de amenazas confían en gran medida y que puede ser

respaldado a través de la automatización y el aumento de procesos impulsados por software. (Ahrend, Jirotka, & Jones, 2016)

La inteligencia de amenazas como la información que utiliza una organización para comprender las amenazas que tiene, se dirigirán o se dirigen actualmente a la organización; y el propósito principal de la inteligencia de amenazas es ayudar a las organizaciones a comprender los riesgos de las amenazas externas más comunes y graves. (Cascavilla, Tamburri, & Heuvel, 2021)

La inteligencia de código libre según Cascavilla, y otros (2021) es el conocimiento obtenido a partir del procesamiento y análisis de fuentes de datos públicas, como transmisiones por televisión y radio, publicaciones en redes sociales, sitios web y plataformas informáticas. Estas fuentes proporcionan datos en formatos de texto, video, imagen y audio.

La inteligencia de amenazas es la disciplina cuya intención es proporcionar información organizada, analizada y refinada sobre ataques potenciales o actuales que amenazan a una organización, o gobiernos (Tounsi & Rais, 2018).

Una de las principales fases del ciclo de vida de la inteligencia de amenazas es el procesamiento y análisis. El ciclo de vida de la inteligencia de amenazas según Cascavilla, y otros (2021) consta de seis fases y son: dirección, recopilación, procesamiento, análisis, diseminación y retroalimentación

De lo analizado se evidencia que se utilizan modelos de la ciberseguridad, y modelos basados en las técnicas de machine learning, pero no hay suficientes referentes teóricos y aplicaciones prácticas relacionados con la detección de sitios web phishing, basadas en machine learning, teniendo en cuenta la información de las URL, la información de la inteligencia de amenazas y la lógica de machine learning.

II. Sistema de detección de sitios web phishing

El sistema de detección de sitios web phishing se basa en un modelo de machine learning que toma en cuenta la información de los sitios web y la información de la inteligencia de amenazas y las técnicas de machine learning; y se describe a continuación.

2.1 Modelo de Machine Learning

El modelo de machine learning para la detección de sitios web phishing está compuesto de 6 dimensiones: Sitio Web, Inteligencia de Amenazas, Preparación de los datos, Algoritmos de machine learning, Entrenamiento y Predicción.

2.1.1 Dimensión de sitio web.

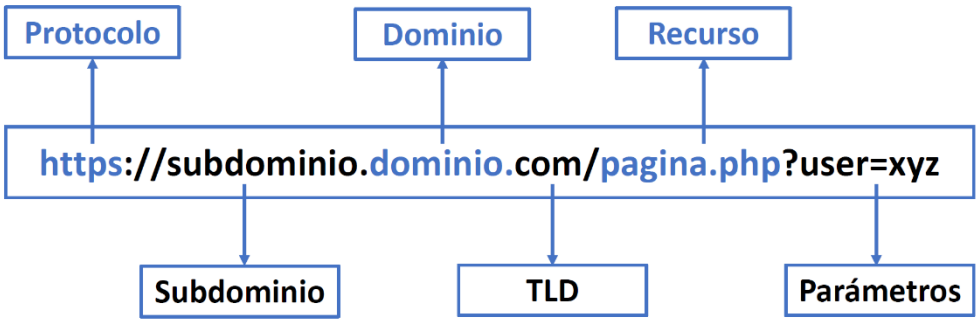
Dado que el trabajo está orientado en la detección de sitios web phishing, se tiene que analizar los sitios web y la información que proporcionan, por ello es la primera dimensión del modelo. Un sitio web se refiere al conjunto de archivos que se muestran en forma de una página web que están alojados en un dominio de internet, y que los usuarios de internet acceden a través de una URL. Además, los sitios web están escritos en un lenguaje de programación en código HTML o dinámicamente son convertidos a HTML.

La dimensión del sitio web, permitirá obtener información de los sitios web de acuerdo con sus características desde dos puntos de vista que son: Estructura de la URL, Informacion de la barra de direcciones y Código del sitio web.

Figura 2: Dimensión Sitio Web

- **Barra de direcciones URL.** La estructura de la URL brinda información del sitio web, y que está basado en la información mostrada en la barra de direcciones. La estructura de una URL se organiza con el protocolo, subdominio, dominio, TLD, recurso (archivo web) y los parámetros; como se muestra en la figura 4.

Figura 3: Estructura de una URL



- **Código fuente.** Los sitios web están desarrollados en un lenguaje de programación y desde el cliente se puede acceder al código fuente, el código fuente brinda información de que tipo de datos se está recogiendo desde la página.

2.1.2 Dimensión Inteligencia de Amenazas.

La inteligencia de amenazas brinda información importante y actualizada de las amenazas avanzadas para identificar posibles sitios web phishing. En esta dimensión se trabaja con información de inteligencia de fuentes abiertas (OSINT: Open Source Intelligence) y se recolectará información de inteligencia de los registros de dominio y de los registros de phishing disponible de forma pública para ser utilizados en un contexto de inteligencia.

Figura 4: Dimensión Inteligencia de amenazas

- **Los Registros de dominio.** Están referidos a la información de del nombre del dominio en los registros de servicios whois, y servicios de reputación por dominio.
- **Los registros phishing,** Son los registros con información de inteligencia relacionado al phishing como reportes de listas negras, reportes de tráfico, reportes y estadísticas de phishing.

La Inteligencia de amenazas, está referida a recopilar la información de los sitios web en análisis, pero además al procesamiento y organización adecuada de los datos, es por eso que la dimensión de inteligencia se relaciona directamente con la dimensión de Preparación de los datos.

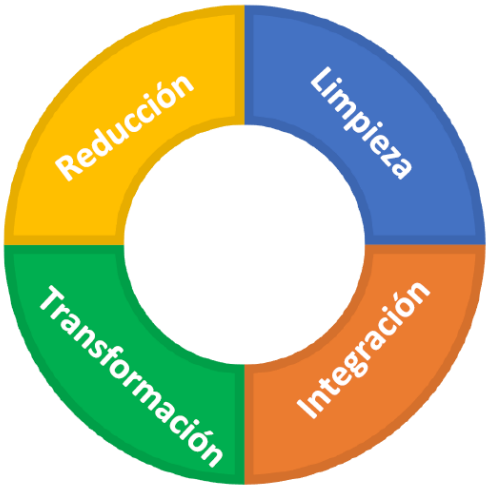
2.1.3 Dimensión Preparación de datos.

La dimensión de **Preparación de datos**. Brinda los datos de calidad para la aplicación efectiva de las técnicas de machine learning. Es el conjunto de actividades para asegúrese comprender, limpiar, transformar y controlar los datos. Las técnicas de machine learning dependen de la calidad de los datos, para un aprendizaje correcto y sin errores. Esta dimensión es una etapa previa al uso de las técnicas de machine learning, recoge los datos de las fuentes de información para realizar el análisis de los datos, el preprocesamiento, proporcionará los datos para el entrenamiento y las predicciones a través del remuestreo.

Figura 5: Dimensión Preparación de datos

- **Análisis de datos.** Es comprender los datos, realizando un análisis de los tipos de datos, una estadística descriptiva y visualización de los datos.
  - Tipos de datos. Permite describir los tipos de datos recolectados de las fuentes de información, tanto en las características y en las etiquetas de las clases.
  - Estadística descriptiva de datos. Permite entender los datos, analizar los atributos y las instancias de los datos, mediante algunas técnicas descriptivas.
  - Visualización de datos. Permite analizar de forma gráfica los datos y el comportamiento de los mismos.
- **Preprocesamiento de datos.** Consiste en dotarle calidad a los datos, a través de la limpieza, integración, transformación y reducción de los datos.

Figura 6: Preprocesamiento de datos



- Limpieza de datos. Verifica y corrige datos incompletos, datos fuera de rango o atípicos y la inconsistencia de los datos recolectados.
- Integración de los datos. Los datos son obtenidos de diferentes fuentes de información, por tanto, es importante integrar los mismos, creando un solo conjunto de datos homogéneo. Se resuelven los problemas de representación y codificación de los datos.
- Transformación de datos. Consiste en operaciones de preprocesamiento adicionales que consolidan los datos para el entrenamiento más eficiente, aplicando técnicas de transformación de datos como el escalamiento, estandarización, normalización y binarización.
- Reducción de las características. Consiste en aplicar técnicas de reducción de datos o características para obtener un conjunto de datos reducido en volumen, pero con que mantenga la integridad de los datos, siendo más eficiente con respecto a los tiempos de entrenamiento y logrando obtener los mismos o similares resultados. Las estrategias de reducción de datos incluyen la reducción de

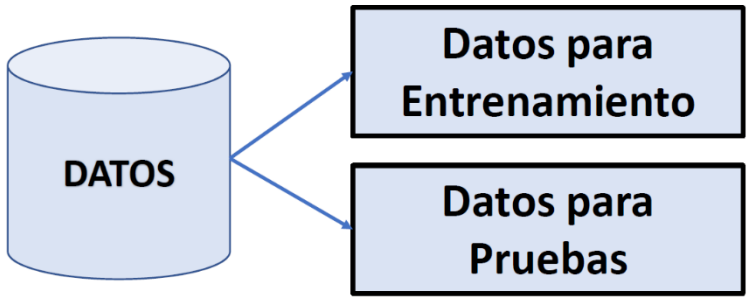
la dimensionalidad, reducción de numerosidad y comprensión de datos.

- **Remuestreo de datos.** Las técnicas de remuestreo permitirá dividir el conjunto de datos en subconjuntos para entrenamiento y un subconjunto para la predicción del modelo, además de evaluar la robustez del modelo utilizando división por porcentaje y utilizando validación cruzada.

  La división por porcentaje implica dividir los datos en un conjunto específico para el proceso de entrenamiento del modelo y otro subconjunto de datos para el proceso de evaluación del modelo. El tamaño de cada subconjunto depende del tamaño y las características del conjunto de datos en estudio y puede ser de 80% para entrenamiento y 20% para evaluación del modelo.

  La validación cruzada implica dividir el conjunto de datos en n-particiones llamados k-folds, cada conjunto se mantiene mientras el modelo se entrena en todas las demás particiones; este proceso se repite hasta que se determina el rendimiento de cada instancia en el conjunto de datos y se estima un promedio del rendimiento global del modelo.

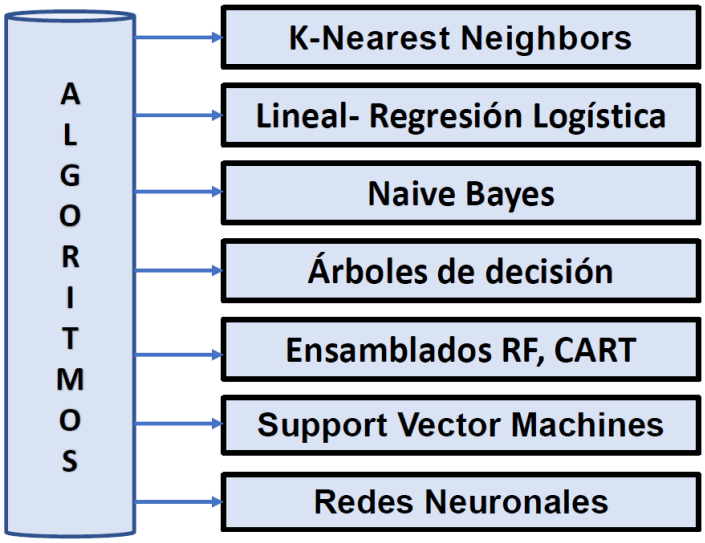
Figura 7: Remuestreo de datos



2.1.4 Dimensión Algoritmos de machine learning.

La dimensión de **Algoritmos de Machine Learning**. Consiste en la selección de los algoritmos de machine learning para realizar el entrenamiento del modelo, se realiza una evaluación preliminar de los diferentes algoritmos, seleccionando los que muestren los mejores resultados Para la evaluación de los algoritmos se debe utilizar las métricas de clasificación como el accuracy, matriz de confusión y el informe de clasificación.

Figura 8: Dimensión Algoritmos de Machine Learning



Los tipos de algoritmos de machine learning están clasificados de acuerdo a la figura anterior y se describen a continuación:

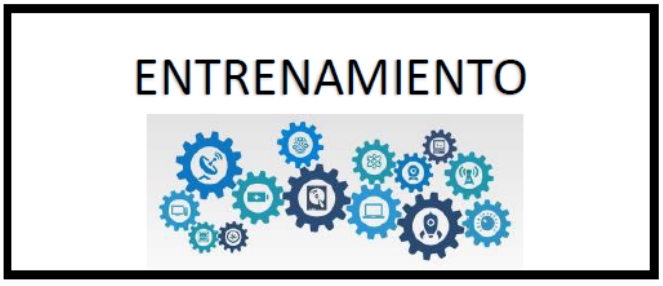
- **k-Nearest Neighbors.** El algoritmo k-NN se basa en analizar los datos de los vecinos más cercanos para hacer una predicción de un nuevo punto de datos, utiliza el valor de k vecinos para comparar.
- **Los modelos lineales.** Son una clase de modelos que se utilizan una función lineal de las características de entrada. Los algoritmos de modelos lineales son los algoritmos Regresión Logística y las Maquinas de vectores de soporte lineal (Lineal SVM).

- **Naive Bayes.** Son una familia de algoritmos basados en probabilidades y el teorema de Bayes.
- **Arboles de decisión.** Esencialmente, aprenden una jerarquía de preguntas si / si no, lo que lleva a una decisión.
- **Ensamblados.** Son métodos que combinan varios modelos de aprendizaje automático para crear modelos más potentes, hay dos modelos de conjunto que han demostrado ser efectivos y son Random Forest y árboles de decisión impulsados por gradientes.
- **Support Vector Machines.** El algoritmo Support Vector Machine denominado comúnmente como SVM,construye un hiperplano en un espacio de dimensionalidad, para lograr una separación entre una clase y otra.
- **Redes neuronales artificiales.** Las redes neuronales artificiales (RNA) pertenecen a los modelos de Deep Learning (aprendizaje profundo) que consiste en un conjunto de neuronas artificiales conectadas entre si desde unas neuronas de entrada que reciben los datos de las características, con las neuronas ocultas y las neuronas de salida que nos indica la clasificación que corresponden los datos de entrada.

2.1.5 Dimensión Entrenamiento.

Consiste en la construcción del modelo de machine learning, buscando los mejores parámetros.

Figura 9: Dimensión Entrenamiento



Se desarrollan los siguientes procesos:

- Entrenamiento base: Se refiere a entrenar un algoritmo o algoritmos seleccionados con el conjunto de datos para crear los modelos base, y seleccionar el (o los) de mejores resultados.
- Optimizar. Se refiere a la búsqueda de los mejores parámetros de cada algoritmo seleccionado que permita tener los mejores resultados. Se puede realizar mediante la búsqueda de cuadrículas o mediante búsquedas aleatorias.
- Entrenamiento Final, Se refiere a desarrollar el modelo final, con los mejores parámetros encontrados.

2.1.6 Dimensión Detección.

Es la interfaz gráfica de modelo que perimirá clasificar si un sitio web específico es phishing o es un sitio web legítimo. Además, se evalúa el rendimiento del modelo utilizando las métricas.

Figura 10: Dimensión Detección

2.2 Fases del sistema de detección de phishing

El sistema de detección de phishing está sustentado en el modelo de machine learning, integrando las características de los sitios web y la inteligencia de amenazas, y la estructura se basa en el modelo sistémico

dividiendo al sistema en subsistemas o fases, que se describen a continuación:

Fase 1. Recolección de Datos.

El objetivo de esta primera fase es la obtención de los datos para entrenamiento y para la detección.

Los datos para el entrenamiento deben tener las características del sitio web y la etiqueta si corresponde a un sitio phishing o a un sitio legítimo.

Las características de los sitios web son recolectados desde las dos dimensiones del modelo propuesto: Sitios Web (barra de direcciones y en código fuente) y de la Inteligencia de Amenazas. Las características serán catalogadas como confiable, sospechoso y riesgoso.

Características de los sitios web, basados en la barra de direcciones.

1. IP_Address.
2. URL_longitud.
3. URL_corto.
4. URL_arroba.
5. URL_slash.
6. URL_linea.
7. URL_puntos.
8. HTTPS_SSL.
9. Domain_registro.
10. Favicon.
11. Puerto.
12. Domain_https.

Características de los sitios web, basados en el código fuente.

13. Request_URL.
14. URL_ancla.
15. Tags.

16. SFH.
17. Submit_email.
18. Abnormal_URL.
19. Forwarding.
20. Barra_estado.
21. Click_derecho.
22. Pop-up.
23. IFrame.

Características de los sitios web, basados en la inteligencia de amenazas.

24. Domain_edad.
25. Registro_DNS.
26. Trafico.
27. PageRank.
28. Google_index.
29. Links.
30. Estadísticas.

Fase 2. Preparación los datos.

El objetivo de esta fase es realizar un análisis de los datos, preprocesamiento y división de los datos, para el entrenamiento y para la detección.

El análisis de los datos consiste en visualizar los tipos de datos, hacer resúmenes de los datos, filtrar los datos, agrupar los datos por clases y visualizar si hay datos perdidos o fuera de rango. Además de realizar visualizaciones de correspondencia.

El procesamiento de los datos consiste en hacer una limpieza a los datos y una transformación a los mismos aplicando técnicas de escalamiento, normalización y binarizacion

Se realiza la reducción de características y finalmente en esta fase los datos son divididos en un subconjunto de datos para el entrenamiento y

un subconjunto de datos para evaluación en la detección de phishing por el sistema.

Fase 3. Selección de algoritmos de Machine Learning.

El objetivo de esta fase es seleccionar los algoritmos para la fase de entrenamiento.

Se realiza una evaluación de todos los algoritmos de machine learning para la clasificación, con los datos de entrenamiento, se realiza una evaluación de los resultados con una validación cruzada.

Se elige los algoritmos que obtienen los mejores resultados de las métricas de clasificación, especialmente el accuracy.

Fase 4. Entrenamiento del Sistema

El objetivo de esta fase es entrenar un algoritmo que brinde los mejores resultados y guardarlo como el modelo para realizar las predicciones.

Se realiza la optimización de los algoritmos elegidos en la fase anterior, con los datos de entrenamiento, para obtener los mejores parámetros con cada algoritmo.

Con los mejores parámetros se realiza el afinamiento del modelo, eligiendo al modelo que tiene los mejores resultados y el modelo seleccionado se guarda para realizar las predicciones correspondientes a la siguiente fase.

Fase 5. Detección de Phishing

El objetivo de esta fase es verificar el funcionamiento del modelo desarrollado, en la clasificación de un sitio web como phishing o como sitio web legitimo.

Con el subconjunto de datos de validación (datos nuevos para el sistema), se verifica la capacidad predictiva del sistema, ingresando los datos originales y comparando con las predicciones del sistema, para la evaluación del rendimiento.

Fase 6. Evaluación del rendimiento

El objetivo de esta fase es evaluar el rendimiento del modelo generado, en la clasificación de un sitio web como phishing o como legitima.

Se evalúa el modelo de machine learning en la detección de sitios web falsos, utilizando la matriz de confusión y las siguientes métricas.

- Verdaderos positivos (TP). Cantidad de pruebas de sitios web phishing, clasificados por el sistema correctamente como sitio web phishing.
- Verdaderos negativos (TN). Cantidad de pruebas de sitios web legítimos, clasificados por el sistema correctamente como sitio web legítimos.
- Falsos positivos (FP). Cantidad de pruebas de sitios web legítimos, clasificados por el sistema erróneamente como sitio web phishing.
- Falsos Negativos (FN). Cantidad de pruebas sitios web phishing, clasificados por el sistema erróneamente como sitio web legítimos.
- Accuracy, es la proporción de clasificación correcta global de todas las instancias que se utilizan.
- El error en la clasificación se representa como la proporción de instancias clasificadas incorrectamente a todas las instancias.
- Recall, sensibilidad o Tasa de verdaderos positivos es proporción de instancias clasificadas correctamente como positivas.
- Especificidad o Tasa de falsos positivos, utilizando la proporción de instancias clasificadas incorrectamente como positivas.
- Precisión. es la proporción de instancias clasificadas correctamente como positivas a todas las instancias clasificadas positivamente.

III. Implementación del sistema de detección de sitios web phishing

La implementación del Sistema de Detección de Sitios Web Phishing se lleva a cabo en las fases descritas anteriormente. Se implementa en el

lenguaje de programación Python, en Jupyter Notebook en Anaconda. Se hace uso de las librerías open source.

A continuación, se describe el proceso de implementación del sistema fase por fase.

3.1 Fase 1. Recolección de datos.

La recolección de datos se realiza desde la información del sitio web y desde la información de inteligencia de amenazas.

3.1.1 Recolección de datos del sitio web.

El sitio web proporciona información en la barra de direcciones y el código fuente; y se describen a continuación.

Características de los sitios web, basados en la barra de direcciones.

1. **IP_Address.** Se refiere al uso de una dirección IP dentro de la URL. Si no se muestra la IP en la URL es un sitio confiable, de lo contrario es un sitio no confiable.
2. **URL_longitud.** Es la longitud de la dirección URL, usualmente los atacantes usan direcciones extensas para ocultar la parte fraudulenta en la barra de direcciones. Si el tamaño es corto (menor a 54) es un sitio confiable, si el tamaño es extenso (mayor a 75) es un sitio no confiable, de lo contrario es un sitio sospechoso.
3. **URL_corto**. Es el uso del servicio de acortamiento de la URL, se usa normalmente para el redireccionamiento de un sitio web mostrando un nombre corto del sitio, pero enlace a un sitio con nombre extenso. Si no se usa el servicio de acortamiento es un sitio confiable, de lo contrario no lo es.
4. **URL_arroba.** Es el uso del símbolo “@” en la URL. Se usa normalmente para llevar al navegador a ignorar todo lo que está antes del símbolo “@” y la dirección real es la que está después del símbolo “@”. Si no se encuentra el @ en la URL es un sitio confiable, de lo contrario no lo es.
5. **URL_slash.** Es el uso del “//” dentro de la URL, se usa normalmente para redirigir a otro sitio web. Los dominios inician con http:// o https://,

por lo tanto si no existe “//” después del séptimo carácter de la URL es un sitio confiable, de lo contrario no lo es.

6. **URL_linea.** El símbolo “-“ rara vez se usa en URL legítimas. No es confiable el uso de prefijos o sufijos separados por (-) al nombre de dominio. Si no se muestra una “-“ en la URL entonces es un sitio confiable, de lo contrario no lo es.
7. **URL_puntos.** El número de puntos dentro de la URL, por ejemplo, con la URL https://www.unprg.edu.pe . Un nombre de dominio incluye los dominios de nivel superior de código de país (ccTLD), que en nuestro ejemplo es "pe". La parte "edu" es una abreviatura de "educativa", el "edu.pe" combinado se denomina dominio de segundo nivel (SLD) y "unprg" es el nombre real del dominio. Si se cuenta con 3 o menos puntos en la URL es un sitio confiable, si cuenta con 4 puntos el sitio web es sospechoso y si cuenta con mas de 4 puntos en sitio web es no confiable.
8. **HTTPS_SSL**. El uso de un certificado SSL es muy importante para verificar la legitimidad de un sitio web. Existente autoridades de certificación, entre los mas importantes se incluyen: GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo, Doster y VeriSign. Si el sitio web tiene un certificado SSL entonces es confiable, de lo contrario no lo es.
9. **Domain_registro.** El tiempo de registro de un dominio es importante, los dominios legítimos normalmente tienen varios años de registro. Si el tiempo de registro del dominio es mayor a 1 año es un sitio confiable, de lo contrario no lo es.
10. **Favicon.** Es una imagen gráfica (icono) asociada a una página web específica. Muchos agentes de usuario existentes, como navegadores gráficos y lectores de noticias, muestran favicon como un recordatorio visual de la identidad del sitio web en la barra de direcciones. Si el favicon se carga desde el mismo dominio web entones es confiable, de lo contrario no lo es.

11. **Puerto.** Se verifica los puertos abiertos en el dominio en específico, si los puertos comunes están abiertos los atacantes pueden ejecutar cualquier servicio que deseen, y como resultado la información del usuario se ve amenazada. Si los puertos comunes no están abiertos es confiable, de lo contrario no lo es.
12. **Domain_https**. Se utiliza el https como parte del dominio para aparentar una página web confiable. Si no se usa https como parte del dominio es un sitio confiable, de lo contrario no lo es.

Características de los sitios web, basados en el código fuente.

13. **Request_URL.** Examina si objetos externos como imágenes o videos contenidos en una página web, cargan desde un dominio diferente. Normalmente las páginas web legítimas cargan del mismo dominio la página web y la mayoría de los objetos incrustados como imágenes y videos. Si las solicitudes son pocas (<22%) el sitio es confiables; si las solicitudes son muchas (>61%) el sitio no es confiable, de lo contrario es sospechoso.
14. **URL_ancla.** Un ancla es un elemento definido por la etiqueta <a>. Se examinan 4 tipos de anclas. <a href="#">, <a href="#content">, <a href="#skip"> y <a href="JavaScript ::void(0)">. Si hay pocas anclas (<31%) el sitio es confiable, si hay muchas anclas (<67%) el sitio no es confiable, de lo contrario el sitio es sospechoso.
15. **Tags.** Es común que los sitios web legítimos usen tags (etiquetas) <Meta> para ofrecer metadatos sobre el documento HTML; Etiquetas <Script> para crear un script del lado del cliente; y etiquetas <Link> para recuperar otros recursos web. Si hay pocas etiquetas (<17%) el sitio es confiable, si las etiquetas son muchas (>81%) el sitio no es confiable, de lo contrario el sitio es sospechoso.
16. **SFH**. Los SFH (Server Form Handler) que contienen una cadena vacía o "about: blank" se consideran no confiables. Además, si el nombre de dominio en SFH no coincide del nombre de dominio de la página es sospechosa, de lo contrario es sitio web confiable.

17. **Submit_email.** El formulario web permite a un usuario enviar su información personal que se dirige a un servidor para su procesamiento. Con ese fin, se puede utilizar un lenguaje de script del lado del servidor como la función "mail ()" en PHP. Otra función del lado del cliente que podría usarse para este propósito es la función “mailto:” Si se encuentra un mail() o mailto el sitio es no confiable, de lo contrario es confiable.
18. **Abnormal_URL**. Si el hostname está incluido en el dominio entonces es confiable, de lo contrario es un sitio no confiable.
19. **Forwarding.** Es el número de veces que se ha redirigido un sitio web. Si los sitios web legítimos han sido redirigidos una vez como máximo es confiable, si han sido redirigido más de 4 veces es un sitio no confiable, de lo contrario es sospechoso.
20. **Barra_estado**. Se puede usar JavaScript para mostrar una URL falsa en la barra de estado a los usuarios. Para extraer esta función, debemos extraer el código fuente de la página web, en particular el evento "onMouseOver", y verificar si realiza algún cambio en la barra de estado. Si no hay un cambio de estado entonces es un sitio confiable, de lo contrario no lo es.
21. **Click_derecho.** Se puede usar JavaScript para deshabilitar la función de clic derecho, de modo que los usuarios no puedan ver y guardar el código fuente de la página web. Se busca el evento "event.button == 2" en el código fuente de la página web y se verifica si el clic derecho está deshabilitado, se es así, es un sitio no confiable, de lo contrario es confiable.
22. **Pop-up.** No es común que un sitio web legítimo solicite datos personales o confidenciales a través de una ventana emergente. Si no contiene una ventana emergente (popoup) entones es confiable, de lo contrario no lo es.
23. I**Frame**. Es una etiqueta HTML que se utiliza para mostrar una página web adicional en una que se muestra actualmente. Se puede hacer uso de la etiqueta "iframe" y hacerla invisible, es decir, sin bordes de

marco. Si no se encuentra la función de IFrame entonces es confiable, de lo contrario no lo es.

Para la implementación en el código Python utilizamos las librerías que se visualizan en la figura 11.

Figura 11: Librería para recolectar datos del sitio web

```
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import datetime
import time
from dateutil.parser import parse as date_parse
```

Se recolectan los datos de un sitio web con la función definida como recolectar_datos_sitioweb, La información recogida está codificada, con los valores 1 (sitios confiables), 0 (sitios sospechosos), y -1 (sitios no confiables). La función con parte del código se muestra en la figura 12.

Figura 12: Función para recolectar datos del sitio web

```
def recolectar_datos_sitioweb(url):
    data_set_siteweb = []
    # 1. IP_address
    try:
        ipaddress.ip_address(url)
        data_set_siteweb.append(-1)
    except:
        data_set_siteweb.append(1)
    # 2.Longitud de la página
    if len(url) < 54:
        data_set_siteweb.append(1)
    elif len(url) >= 54 and len(url) <= 75:
        data_set_siteweb.append(0)
    else:
        data_set_siteweb.append(-1)
```

3.1.2 Recolección de datos de inteligencia de amenazas.

Algunos sitios web de acceso abierto, brindan información de inteligencia de amenazas, a continuación, se describen las características a recolectar.

Características de los sitios web, basados en la inteligencia de amenazas.

24. **Domain_edad.** La mayoría de los sitios web legítimos tienen un tiempo de vida largo. Se extrae de la base de datos whois y si el tiempo de vida es largo (>6 meses) es confiable, de lo contrario no lo es.
25. **Registro_DNS.** La mayoría de los sitios web falsos la base de datos WHOIS no reconoce la identidad o no tienen registros para el hostname. Si se encuentran registros DNS entones es confiable, de lo contrario no lo es.
26. **Trafico**. La popularidad del sitio web está determinando por el número de visitantes. Los sitios web con corta duración normalmente no se registran en sitios de popularidad, por tanto, lo sitios confiables es muy probable que estén registrados y sean reconocidos por la base de datos Alexa. Si no reconoce al sitio web entonces es no confiable, si lo reconoce y lo ubica entre los 100000 sitios principales es confiable, de lo contrario es sospechoso.
27. **PageRank**. PageRank tiene como objetivo medir la importancia de una página web. Cuanto mayor sea el valor de PageRank, más importante será la página web. Si el sitio web tienen un pagerank mayor a 0.2 entonces es confiable, de lo contrario no lo es.
28. **Google_index.** Esta función examina si un sitio web está en el índice de Google o no. Si el sitio web está en el Google index, entonces es confiable, de lo contrario no lo es.
29. **Links**. El número de enlaces que apuntan a la página web puede ayudarnos a identificar a un sitio web legítimo. Si el número de enlaces es mayor a 2 entonces es confiable; si no tiene enlaces no es confiable, de lo contrario es sospechoso.

30. **Estadísticas**. Varios sitios especializados como phishTank y StopBadware publican reportes de informes estadísticos de sitios web phishing. Buscamos en esos sitios web las principales direcciones ip y direcciones de dominio phishing, y si el sitio no está en los reportes entonces es un sitio confiable, de lo contrario no lo es.

Se recolectan los datos de inteligencia de amenazas en una función definida como recolectar_datos_inteligencia. La información recogida está codificada, con los valores 1 (sitios confiables), 0 (sitios sospechosos), y -1 (sitios no confiables). La función con parte del código se muestra en la figura 13.

Figura 13: Función para recolectar datos de inteligencia de amenaza

```
def recolectar_datos_inteligencia(url):
    data_set_inteligencia = []
    #24. Domain_edad
    if response == "":
        data_set_inteligencia.append(-1)
    else:
        try:
            registration_date = re.findall(r'Registration Date:</div><div
            if diff_month(date.today(), date_parse(registration_date)) >=
                data_set_inteligencia.append(-1)
            else:
                data_set_inteligencia.append(1)
        except:
            data_set_inteligencia.append(1)
    #25. Registro DNS
```

En la parte final de esta fase, se recolectan los datos de forma automatizada, con las funciones definidas previamente y se recolectan los datos con información del sitio web y de inteligencia de amenazas. En la figura 14 se muestra un ejemplo de la recolección de datos de un sitio web, los datos recogidos son codificados, con los valores 1 (sitios confiables), 0 (sitios sospechosos), y -1 (sitios no confiables).

Figura 14: Recolección automatizada de datos de un sitio web

```
recolectar_datos_sitioweb("http://www.unprg.edu.pe/")
```

```
[1, 1, 1, 1, 1, 1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, 1]
```

```
recolectar_datos_inteligencia("http://www.unprg.edu.pe/")
```

```
[1, -1, 1, -1, 1, 1]
```

Utilizando estas funciones automatizadas, se recogen los datos de 11055 sitios web, de las cuales 6157 sitios web son legítimos y 4898 son sitios web phishing; para el desarrollo y evaluación del sistema.

3.2 Fase 2. Preparación de los datos.

Luego de recoger los datos, se integran y se agregan las etiquetas (resultado: 1 si es sitio legítimo y -1 si es sitio phishing) por cada registro realizado, y se realizan 3 actividades.

3.2.1 Análisis de los datos.

Usamos la librería pandas en Python, accedemos al archivo con el conjunto de datos, cargamos los datos y sus características. En la figura 15, se visualiza la salida de las dimensiones del conjunto de datos cargados, con 11055 registros y 31 características. (23 características de sitios web, 7 características de inteligencia de amenazas y la etiqueta del sitio web).

Figura 15: Carga y visualización de los datos totales

```
import pandas as pd
filename = 'dataphishing.csv'
data = pd.read_csv(filename, names=None)
data.shape
```

```
(11055, 31)
```

Observamos los tipos de datos, por cada característica del conjunto de datos y de la etiqueta (resultado). En la figura 16 se muestra que todos los datos incluyendo las características y la etiqueta, son enteros.

Figura 16: Tipos de los datos.

```
data.dtypes
```

```
IP_Address          int64
URL_longitud        int64
URL_corto           int64
URL_arroba          int64
URL_slash           int64
URL_linea           int64
URL_puntos          int64
SSL                 int64
Domain_registro     int64
Favicon             int64
Puerto              int64
Domain_https        int64
Request_URL         int64
URL_ancla           int64
Tags                int64
SFH                 int64
Submit_email        int64
SFH                 int64
Submit_email        int64
Abnormal_URL        int64
Fordwarding         int64
Barra_estado        int64
Click_derecho       int64
PopUp               int64
Iframe              int64
Domain_edad         int64
Registro_DNS        int64
Trafico_web         int64
Page_Rank           int64
Google_Index        int64
Links               int64
Estadisticas        int64
Resultado           int64
dtype: object
```

Verificamos que no hay datos nulos o datos perdidos, en el conjunto de datos, con la función data.isnull().

Implementamos dos funciones para automatizar en análisis por característica mediante gráficos y visualizar su comportamiento. En la figura 17 se muestra la función definida como show_caracteristica para visualizar los datos de las características que cuentan con dos niveles de datos (-1 o 1); y en la figura 18 se muestra la función definida como show_caracteristica3 para visualizar los datos de las características que cuentan con tres niveles de datos (-1, 0 y 1).

Figura 17: Función para la visualización de características de 2 niveles

```
def show_caracteristica(caracteristica, etiquetas):
    fig, ax = plt.subplots()
    explode=(0.05,0.05)
    datos =data[caracteristica].value_counts()
    ax.pie(datos, explode= explode,autopct='%1.1f%%',shadow=True,startangle=90)
    plt.legend(etiquetas, bbox_to_anchor =(1.15, 1.15), ncol = 2)
    plt.show()
    print(datos)
```

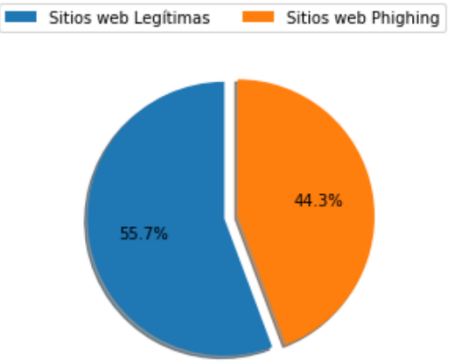
Figura 18: Función para la visualización de características de 3 niveles

```
def show_caracteristica3(caracteristica, etiquetas):
    fig, ax = plt.subplots()
    explode=(0.05, 0.05, 0.05)
    datos =data[caracteristica].value_counts()
    ax.pie(datos, explode= explode,autopct='%1.1f%%',shadow=True,startangle=90)
    plt.legend(etiquetas, bbox_to_anchor =(1.15, 1.15), ncol = 3)
    plt.show()
    print(datos)
```

En la figura 19 se muestra que el 55.7% (6157) de datos corresponden a sitios web legítimos y el 44.3% (4898) de sitios web phishing. de los datos. Las clases de los datos son relativamente equilibradas.

Figura 19: Datos de sitios web legítimos vs sitios web phishing.

```
show_caracteristica('Resultado',etiquetas)
```

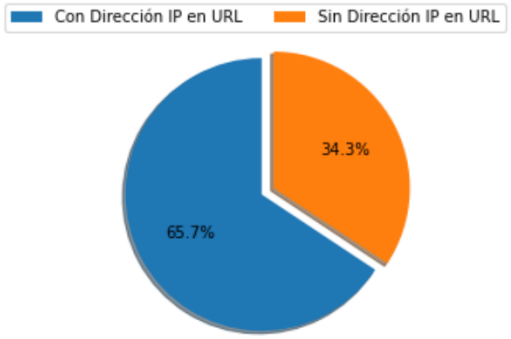



```
 1    6157
-1    4898
Name: Resultado, dtype: int64
```

En la figura 20 se muestra que el 65.7% (7262) de sitios web tienen una dirección IP en la URL y el 34.3% (3793) de los sitios web no tienen direcciones IP en la URL. Las direcciones IP dentro de las URL no es muy común en sitios web confiables,

Figura 20: Datos de sitios web con direcciones IP en la URL.

```
show_caracteristica('IP_Address',etiquetas)
```

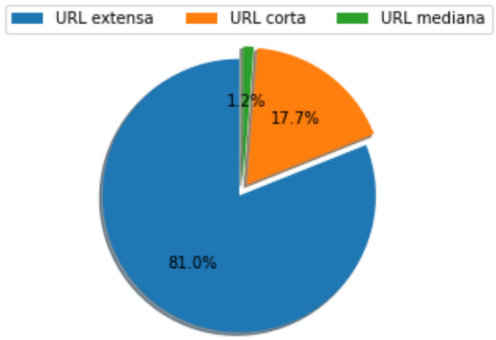



```
 1    7262
-1    3793
Name: IP_Address, dtype: int64
```

En la figura 21 se muestra que el 81% (8960) de sitios web tienen una longitud extensa de la URL (con mayor a 75 caracteres) siendo sitios no confiables, el 17.7% (1960) de los sitios web tienen una longitud corta de la URL (con menos de 54 caracteres) siendo sitios confiables, y solo el 1.2% (135) de los sitios web tienen una longitud mediana de la URL siendo sitios sospechosos. Se observa un numero alto de sitios web que tienen una longitud extensa de la URL.

Figura 21: Datos de sitios web con la longitud de la URL.

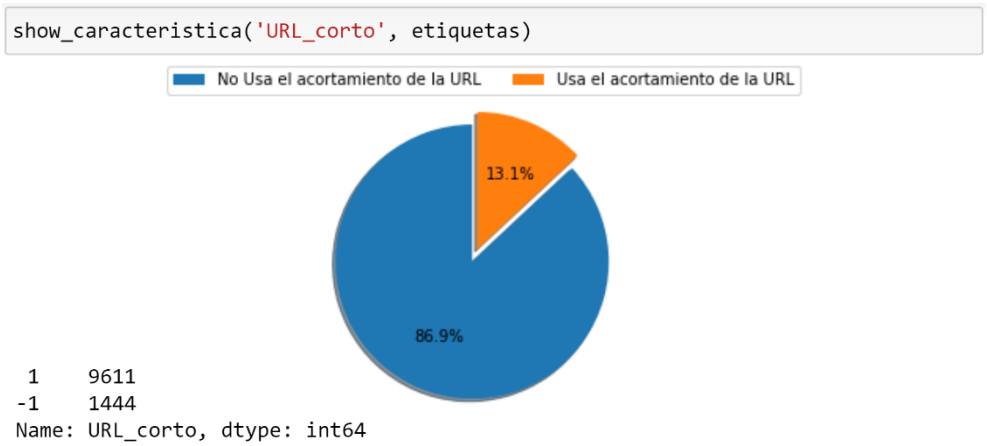
```
show_caracteristica3('URL_longitud', etiquetas)
```




```
-1    8960
 1    1960
 0     135
Name: URL_longitud, dtype: int64
```

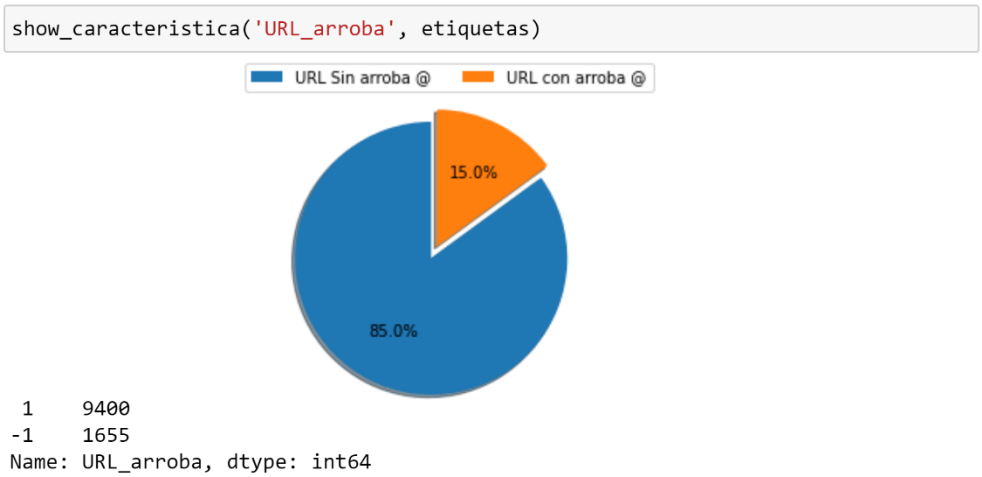
En la figura 22 se muestra que el 86.9% (9611) de sitios web no hacen uso del acortamiento de la URL y el 13.1% (1444) de los sitios web si hacen uso del servicio de acortamiento de la URL. Se visualiza que hay un uso relativamente bajo con los servicios de acortamiento URL, sin embargo, este servicio puede ser aprovechados por los sitios web falsas.

Figura 22: Datos de sitios web con el uso del acortamiento URL.



En la figura 23 se muestra que el 85.0% (9400) de sitios web no tienen una arroba en la URL y el 15.0% (1855) de los sitios web si tienen una arroba en la URL. El uso de la arroba (@) dentro de la URL, permite ignorar la dirección URL antes del arroba y que pueden ser aprovechadas por los sitios web falsos.

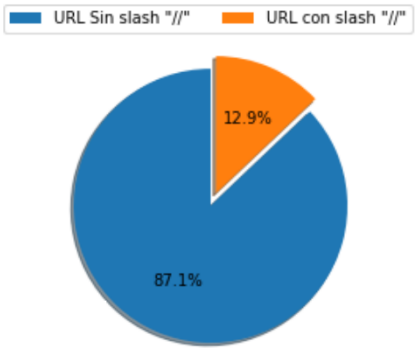
Figura 23: Datos de sitios web con el uso del arroba en la URL.



En la figura 24 se muestra que el 87.1% (9626) de sitios web usan el slash (//) después del séptimo carácter, y el 12.9% (1429) de los sitios web si usan el slash (//) en la URL. El uso del slash (//) permite redireccionar sitios web, y pueden ser aprovechadas por los atacantes para redireccionar a sitios web falsos.

Figura 24: Datos de sitios web con el uso del slash en la URL.

```
show_caracteristica('URL_slash', etiquetas)
```

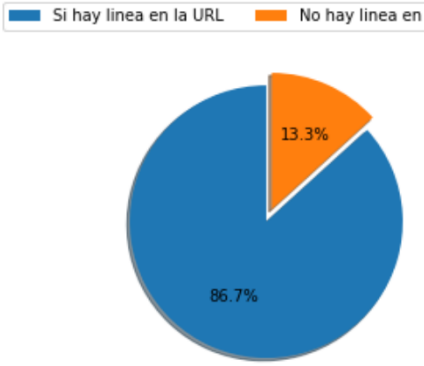



```
 1    9626
-1    1429
Name: URL_slash, dtype: int64
```

En la figura 25 se muestra que el 86.7% (9590) de sitios web en el conjunto de datos hace uso de una línea dentro de la URL, y el 13.3% (1465) de sitios web no tienen una línea en la URL y que no es confiable el uso de prefijos o sufijos separados por una linea (-) al nombre de dominio en la URL.

Figura 25: Datos de sitios web que hacen uso de líneas en la URL.

```
show_caracteristica('URL_linea', etiquetas)
```

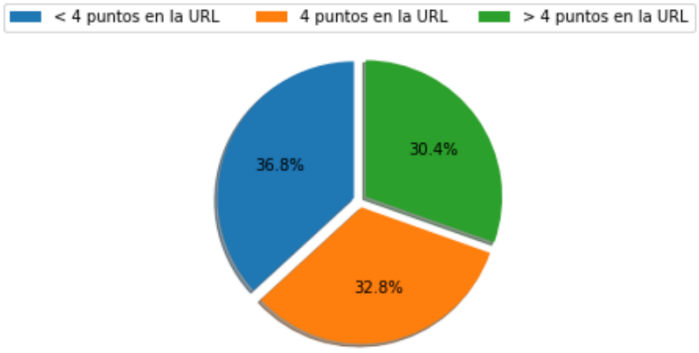



```
-1    9590
 1    1465
Name: URL_linea, dtype: int64
```

En la figura 26 se muestra que el 36.8% (4070) de sitios web del conjunto de datos tienen 3 puntos o menos en la URL que son sitios web confiables; el 32.8% (3622) de sitios web tienen 4 puntos en la URL y el 30.4% (3363) de sitios web tienen mas de 4 puntos en la URL que son sitios web no confiables.

Figura 26: Datos de sitios web que hacen uso de puntos en la URL.

```
show_caracteristica3('URL_puntos', etiquetas)
```

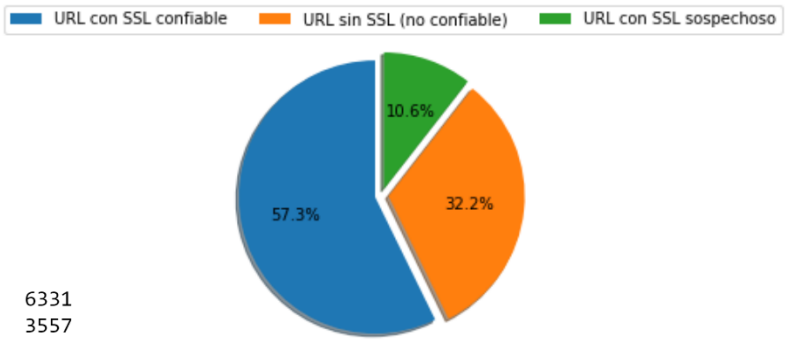



```
 1    4070
 0    3622
-1    3363
Name: URL_puntos, dtype: int64
```

En la figura 27 se muestra que el 57.3% (6331) de sitios web usan el HTTPS con el certificado SSL confiable; el 32.2% (3557) de sitios web no usan HTTPS y son no confiables, y el 10.6% (1167) de los sitios web usan certificado SSL sospechosas. Se observa que un porcentaje alto de sitios web no usan certificados SSL.

Figura 27: Datos de sitios web que hacen uso de SSL en la URL.

```
show_caracteristica3('SSL', etiquetas)
```




```
 1    6331
-1    3557
 0    1167
Name: SSL, dtype: int64
```

Implementamos dos funciones para automatizar en análisis de cada característica por clase. En la figura 28 se muestra la función definida como show_caracteristica_url para visualizar los datos por clase de las características que cuentan con dos niveles de datos (-1 o 1); y en la figura 29 se muestra la función definida como show_caracteristica_url3 para visualizar los datos por clase de las características que cuentan con tres niveles de datos (-1, 0 y 1).

Figura 28: Función para visualizar características con 2 niveles por clase

```
def show_caracteristica_url(caracteristica, caracteristica_A,caracteristica_B):
    clases = ['Phishing', 'Legitima' ]

    dataphishing = data[data['Resultado'] ==-1][caracteristica].value_counts()
    datalegitima = data[data['Resultado'] ==1][caracteristica].value_counts()
    legitima_vs_phishing = pd.DataFrame([dataphishing, datalegitima])
    legitima_vs_phishing.index = clases
    legitima_vs_phishing.columns = [caracteristica_A, caracteristica_B ]
    datostipoA = legitima_vs_phishing[caracteristica_A].tolist()
    datostipoB = legitima_vs_phishing[caracteristica_B].tolist()

    x = np.arange(len(clases))
    width = 0.25
    fig, ax = plt.subplots()

    rects1 = ax.bar(x - width/2, datostipoA, width, label=caracteristica_A)
    rects2 = ax.bar(x + width/2, datostipoB, width, label=caracteristica_B)

    ax.set_ylabel('N° de Sitios Web')
    ax.set_xticks(x)
    ax.set_xticklabels(clases)

    ax.legend(bbox_to_anchor =(0.95, 1.15), ncol = 2)
```

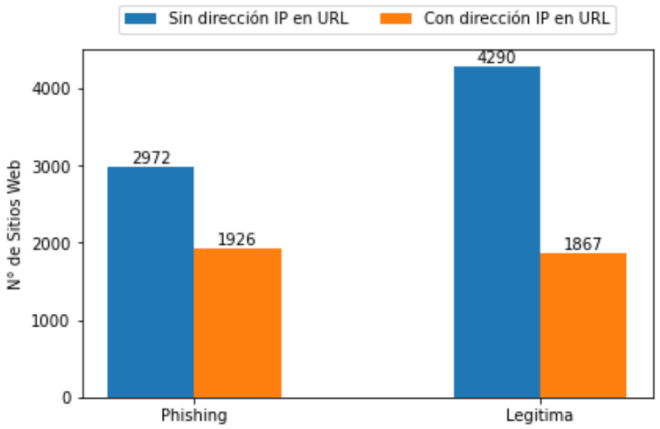
Figura 29: Función para visualizar características con 3 niveles por clase

```
def show_caracteristica_url3(caracteristica, A, B, C):
    clases = ['Phishing', 'Legitima' ]

    dataphishing = data[data['Resultado'] ==-1][caracteristica].value_counts()
    datalegitima = data[data['Resultado'] ==1][caracteristica].value_counts()
    legitima_vs_phishing = pd.DataFrame([dataphishing, datalegitima])
    legitima_vs_phishing.index = clases
    legitima_vs_phishing.columns = [A, B, C]
    datostipoA = legitima_vs_phishing[A].tolist()
    datostipoB = legitima_vs_phishing[B].tolist()
    datostipoC = legitima_vs_phishing[C].tolist()

    x = np.arange(2)
    width = 0.25
    fig, ax = plt.subplots()

    rects1 = ax.bar(x - width/2, datostipoA, width, label=caracteristica_A)
    rects2 = ax.bar(x + width/2, datostipoB, width, label=caracteristica_B)
    rects3 = ax.bar(x + 3*width/2, datostipoC, width, label=caracteristica_C)

    ax.set_ylabel('N° de Sitios Web')
    ax.set_xticks(x)
    ax.set_xticklabels(clases)
```

En la figura 30, se muestra que 1926 sitios web phishing y 1867 sitios web legítimos tienen una dirección IP en la URL. Además, se observa que 2972 sitios web phishing y 4290 sitios web legítimos no tienen una dirección IP en la URL.

Figura 30: Sitios web con direcciones IP en la URL, por clase.

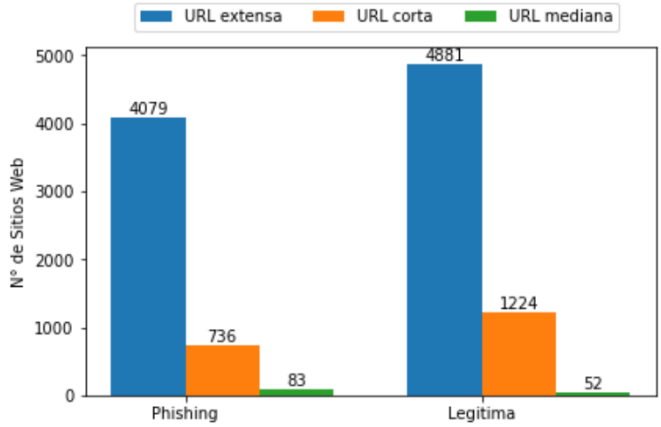
```
show_caracteristica_url('IP_Address', A, B)
```



En la figura 31, se muestra que 4079 sitios web phishing y 4881 sitios web legítimos tienen una longitud extensa de la URL. Además, se observa que 736 sitios web phishing y 1224 sitios web legítimos tienen una longitud corta de la URL; y 83 sitios web phishing y 52 sitios web legítimos tienen una longitud mediana de la URL

Figura 31: Longitud de la URL de los sitios web, por clase.

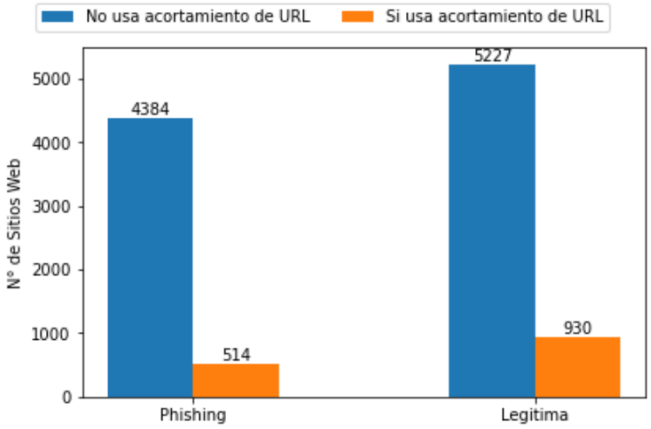
```
show_caracteristica_url3(caracteristica, A, B, C)
```



En la figura 32, se muestra que 514 sitios web phishing y 930 sitios web legítimos usan el servicio de acortamiento de la URL. Además, se observa que 4384 sitios web phishing y 5227 sitios web legítimos no usan el servicio de acortamiento de la URL. El uso del servicio de acortamiento se usa tanto en sitios web phishing y legítimos.

Figura 32: Acortamiento de la URL de los sitios web, por clase.

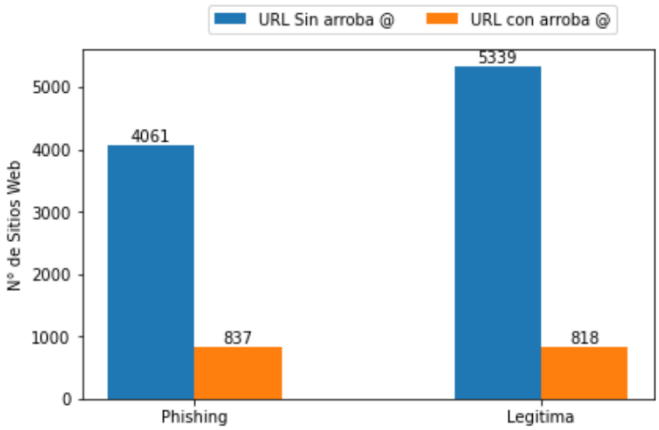
```
show_caracteristica_url(caracteristica, A, B)
```



En la figura 33, se muestra que 837 sitios web phishing y 818 sitios web legítimos usan la arroba (@) en la URL. Además, se observa que 4061 sitios web phishing y 5339 sitios web legítimos no usan la arroba (@) dentro de la URL. El uso de la arroba (@) se hace tanto en sitios web phishing y legítimos.

Figura 33: Uso de arroba en la URL de los sitios web, por clase.

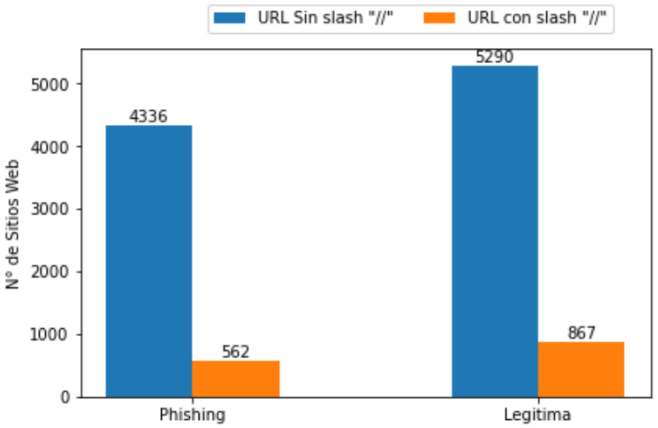
```
show_caracteristica_url(caracteristica, A, B)
```



En la figura 34, se muestra que 562 sitios web phishing y 867 sitios web legítimos usan el slash (//) después del séptimo carácter en la URL. Además, se observa que 4336 sitios web phishing y 5290 sitios web legítimos no usan el slash (//) después del séptimo carácter en la URL. El uso del slash se hace tanto en sitios web phishing y legítimos.

Figura 34: Uso de slash en la URL de los sitios web, por clase.

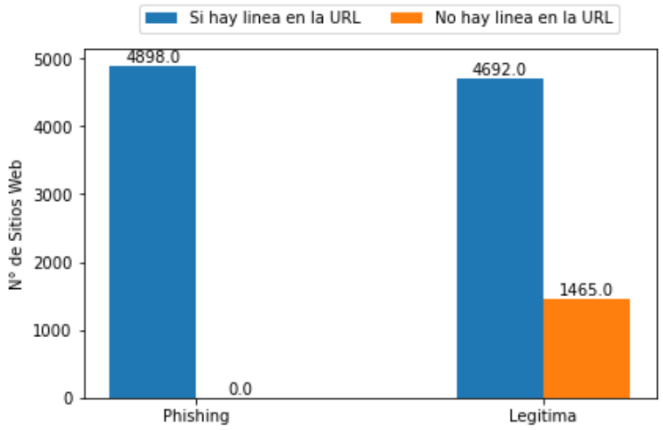
```
show_caracteristica_url(caracteristica, A, B)
```



En la figura 35, se muestra que la totalidad (4898) sitios web phishing y hacen uso de una línea (-) en la URL. Además, se observa que 4692 sitios web legítimos hacen uso de una línea (-) en la URL; y que solamente 1465 sitios web legítimos no hacen uso de una línea (-) en la dirección URL.

Figura 35: Uso de una línea en la URL de los sitios web, por clase.

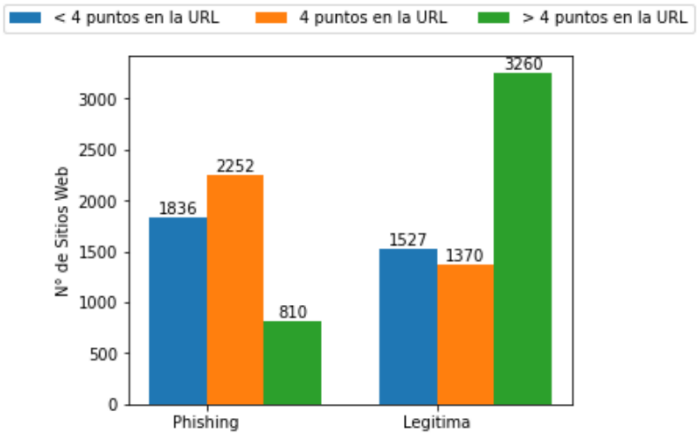
```
show_caracteristica_url(caracteristica, A, B)
```



En la figura 36, se muestra que 1836 sitios web phishing y 1527 sitios web legítimos usan menos de 4 puntos en la dirección URL; además se observa que 2252 sitios web phishing y 1370 sitios web legítimos usan 4 puntos dentro de la dirección URL; y que 810 sitios web phishing y 3260 sitios web legítimos usan mas de 4 puntos dentro de la URL del sitio web.

Figura 36: Uso de puntos en la URL de los sitios web, por clase.

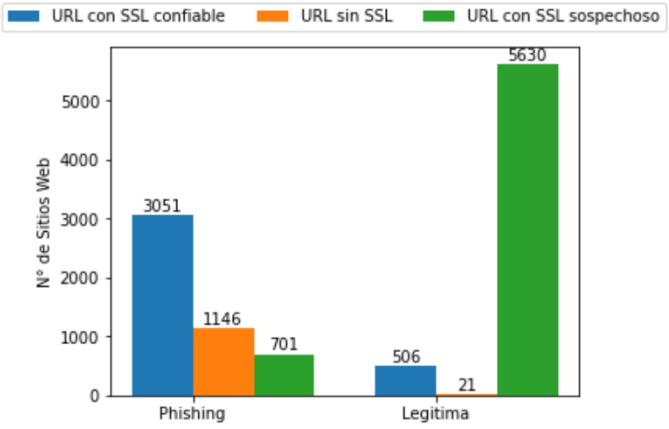
```
show_caracteristica_url3(caracteristica, A, B, C)
```



En la figura 37, se muestra que 3051 sitios web phishing y 506 sitios web legítimos usan SSL confiable; además se observa que 1146 sitios web phishing y 21 sitios web legítimos no usan un certificado SSL; y que 701 sitios web phishing y 5630 sitios web legítimos usan un SSL sospechosa. Hay muy pocos sitios web legítimos que no hacen uso de SSL.

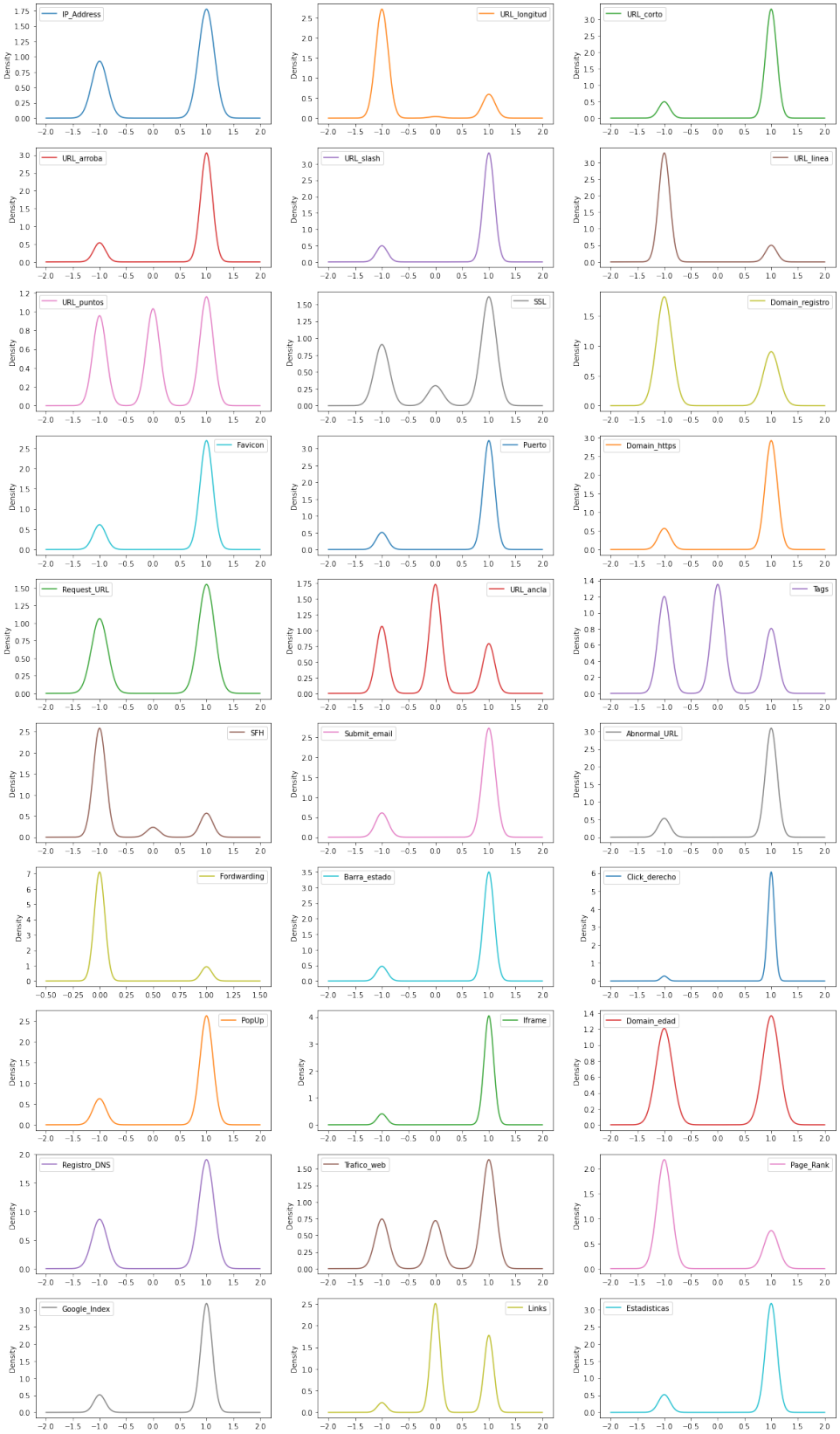
Figura 37: Uso de SSL de los sitios web, por clase.

```
show_caracteristica_url3(caracteristica, A, B, C)
```



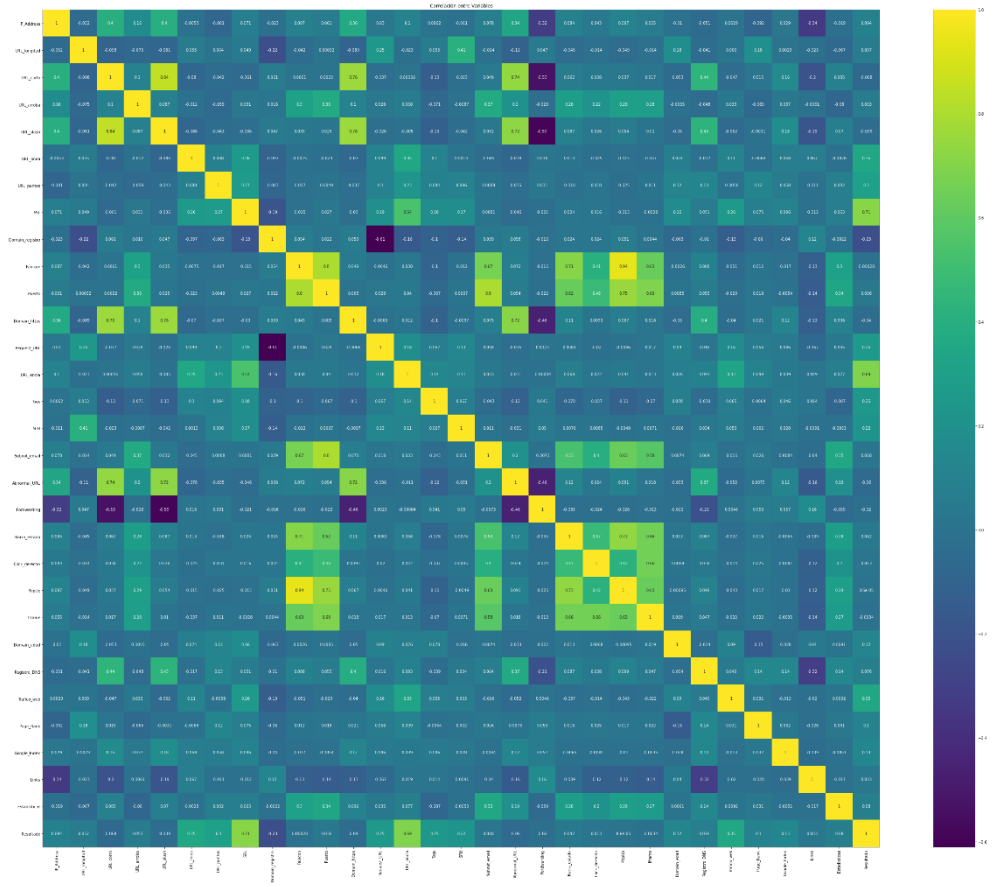
Visualizamos la densidad de todas las características del conjunto de datos y se muestra en la figura 38.

Figura 38: Densidad de las características de los sitios web



Visualizamos de forma gráfica la correlación entre las características, en la figura 39 se muestra la correlación entre variables y las correlaciones muy altas se muestran de amarillo; se muestra una correlación muy alta 0.94 entre las variables PopUP y Favicom; así mismo una correlación alta de 0.84 entre las variables URL_slash y URL_corto; una correlación de 0.80 entre las variables Submit_email y URL_corto; y una correlación de 0.75 entre las variables PopUp y Puerto. La alta correlación de estas variables nos hace suponer que debe evaluarse la omisión de algunas de estas características para las fases posteriores de entrenamiento y evaluación del sistema.

Figura 39: Correlación entre características.



Visualizamos la correlación de cada característica con respecto a la clase, para determinar la importancia de cada característica sobre la clase(sitio web phishing o sitio web legítimo); en la figura 40 se observa correlaciones positivas y negativas;

Se visualiza que las variables SSL (0.71) y URL_ancla (0.69) tienen una correlación alta y se observa que varias variables tienen una correlación muy baja con la clase.

Figura 40: Correlación de las características con la clase.

```
correlaciones = data.corr()['Resultado']
print(correlaciones)
```

```
IP_Address          0.094160
URL_longitud        0.057430
URL_corto          -0.067966
URL_arroba          0.052948
URL_slash          -0.038608
URL_linea           0.348606
URL_puntos          0.298323
SSL                 0.714741
Domain_registro    -0.225789
Favicon            -0.000280
Puerto              0.036419
Domain_https       -0.039854
Request_URL         0.253372
URL_ancla           0.692935
Tags                0.248229
SFH                 0.221419
Submit_email        0.018249
Abnormal_URL       -0.060488
Fordwarding        -0.020113
Barra_estado        0.041838
Click_derecho       0.012653
PopUp               0.000086
Iframe             -0.003394
Domain_edad         0.121496
Registro_DNS        0.075718
Trafico_web         0.346103
Page_Rank           0.104645
Google_Index        0.128950
Links               0.032574
Estadisticas        0.079857
Resultado           1.000000
Name: Resultado, dtype: float64
```

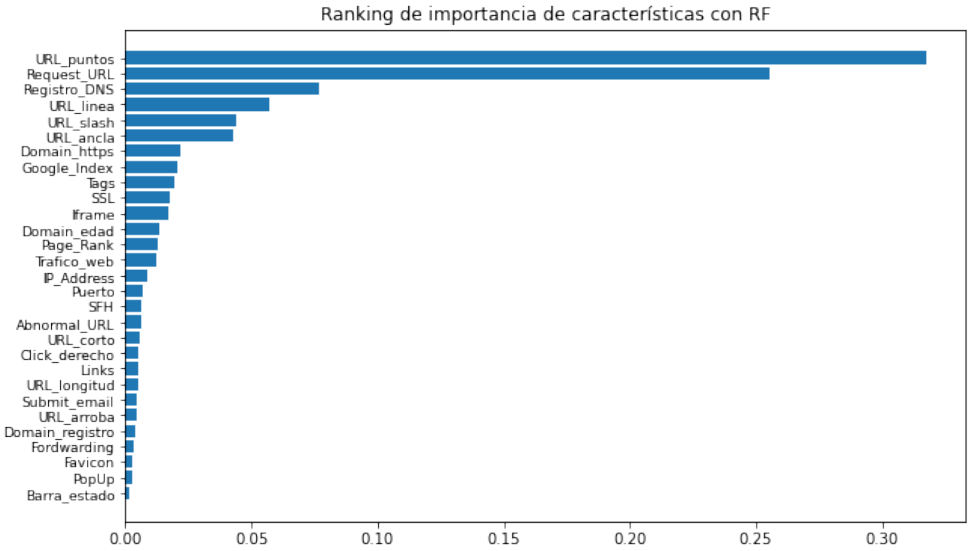
En la figura 41, se muestra las variables o características del conjunto de datos que tiene una baja correlación (menor a 0.05 en absoluto) con la salida; esta correlación hace suponer que estas variables pueden omitirse en las fases posteriores de entrenamiento y evaluación del sistema a desarrollar.

Figura 41: Características con correlación baja con la clase

```
URL_slash          0.038608
Favicon            0.000280
Puerto             0.036419
Domain_https       0.039854
Submit_email       0.018249
Fordwarding        0.020113
Barra_estado       0.041838
Click_derecho      0.012653
PopUp              0.000086
Iframe             0.003394
Links              0.032574
```

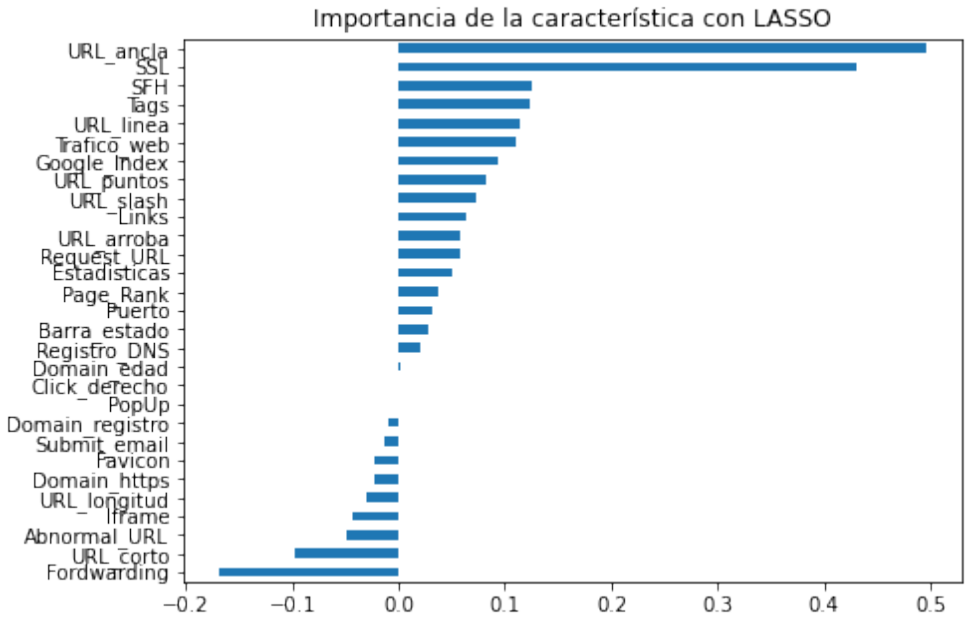
En la figura 42, se muestra las características ordenas de acuerdo con la importancia con la clase segun Random Forest; las características que tienen mayor importancia son URL_puntos y Request_URL; y las que tienen una menor importancia son Barra_estado, PopUp y Favicom.

Figura 42: Importancia de las características, con Random Forest



En la figura 43, se muestra las características ordenas de acuerdo con la importancia con la clase aplicando LASSO; las características que tienen mayor importancia son URL_ancla y SSL; y las características que tienen una menor importancia son Click_derecho y PopUp.

Figura 43: Importancia de las características, con LASSO



3.2.2 Preprocesamiento de datos.

Se observó en las figuras 40, 41, 42 y 43 que hay muchas características tienen baja importancia de correlación con respecto a la clase (Resultado), por tanto, se procede a realizar reducción de características para generar nuevos conjuntos de datos, para optimizar tiempos de entrenamiento y evaluar los resultados de estos nuevos conjuntos de datos con respecto a los datos originales.

Por tanto, trabajamos con tres conjuntos de datos. El primer conjunto de datos es el original, sin ningún cambio, el segundo conjunto de datos se genera con las características que tengan una correlación mayor a 0.05 y el tercer conjunto de datos de generan aplicando la transformación PCA (Análisis de Principales características).

En la figura 44 se observa las dimensiones del primer conjunto de datos, que son los datos originales con el nombre de data, se muestra que tiene 31 características (incluida la etiqueta o clase) y 11055 registros. Cada característica tiene completos los registros y no se observa datos perdidos o datos nulos. Este conjunto de datos queda listo para las etapas posteriores.

Figura 44: Dimensiones y conteo de los datos originales

```
data.shape
```

```
(11055, 31)
```

```
data.count()
```

```
IP_Address          11055
URL_longitud        11055
URL_corto           11055
URL_arroba          11055
URL_slash           11055
URL_linea           11055
URL_puntos          11055
SSL                 11055
Domain_registro     11055
Favicon             11055
Puerto              11055
Domain_https        11055
Request_URL         11055
URL_ancla           11055
Tags                11055
SFH                 11055
Abnormal_URL        11055
Domain_edad         11055
Registro_DNS        11055
Trafico_web         11055
Page_Rank           11055
Google_Index        11055
Estadisticas        11055
Resultado           11055
```

En la figura 45 se observa las dimensiones del segundo conjunto de datos, que han sido generados de los datos originales con solamente las características que son importantes y que tienen una correlación mayor a 0.05 con la clase. Se crea el conjunto de datos con nombre data_import y se muestra que tiene 20 características (incluida la etiqueta o clase) y 11055 registros. Cada característica tiene completos los registros y no se observa datos perdidos o datos nulos. Este conjunto de datos queda listo para las etapas posteriores.

Figura 45: Dimensiones y conteo de los datos importantes

```
data_import.shape
```

```
(11055, 20)
```

```
data_import.count()
```

```
IP_Address          11055          Tags                11055
URL_longitud        11055          SFH                 11055
URL_corto           11055          Abnormal_URL        11055
URL_arroba          11055          Domain_edad         11055
URL_linea           11055          Registro_DNS        11055
URL_puntos          11055          Trafico_web         11055
SSL                 11055          Page_Rank           11055
Domain_registro     11055          Google_Index        11055
Request_URL         11055          Estadisticas        11055
URL_ancla           11055          Resultado           11055
```

En la figura 46 se observa el código para generar el tercer conjunto de datos con solo 5 características aplicando reducción de característica con PCA, utilizando la librería sklearn.

Figura 46: Código para la reducción de características con PCA

```
# Extracción de características con PCA
from sklearn.decomposition import PCA

# PCA con k=5
k=5
pca = PCA(n_components=k)
fit = pca.fit(X)
X_transform = pca.transform(X)

C = pca.components_
print(f'Explained Variance: {fit.explained_variance_ratio_}')
print('Componentes:\n',C)

# Convertimos a dataframe
data_pca = pd.DataFrame(data = X_transform, columns=['PC1','PC2','PC3', 'PC4', '
```

En la figura 47 se muestra cinco registros del tercer conjunto de datos, con las cinco características generadas por PCA (PC1, PC2, PC3, PC4 y PC5) y la etiqueta de salida (Resultado).

Figura 47: Registros de conjunto de datos con PCA

	PC1	PC2	PC3	PC4	PC5	Resultado
0	1.477048	0.816740	2.618284	-2.606275	-0.399858	-1
1	-0.334980	1.248197	0.669342	-1.229078	0.050392	-1
2	1.029920	0.956574	1.459846	-1.241438	-1.613032	-1
3	-0.645606	-1.143744	1.994465	-0.265323	0.240228	-1
4	1.119936	0.984395	0.143043	-0.652123	0.346890	1

En la figura 48 se observa las dimensiones del tercer conjunto de datos, que han sido generados de los datos originales con PCA; se crea el conjunto de datos con nombre data_pca y se muestra que tiene 6 características (incluida la etiqueta o clase) y 11055 registros. Cada característica tiene completos los registros y no se observa datos perdidos o datos nulos. Este conjunto de datos queda listo para las etapas posteriores.

Figura 48: Dimensiones y conteo de los datos con PCA

```
data_pca.shape
```

```
(11055, 6)
```

```
data_pca.count()
```

```
PC1          11055
PC2          11055
PC3          11055
PC4          11055
PC5          11055
Resultado    11055
dtype: int64
```

Los tres conjuntos de datos no tienen datos perdidos o NAN, por lo que no es necesario realizar limpieza de los mismos. Además, los datos, están integrados y están codificados desde la recolección de estos, y

corresponden a una distribución gaussiana, por tanto, no es necesario la integración, normalización, estandarización tampoco la binarización.

3.2.3 Remuestreo de los datos.

Los conjuntos de datos se separan las características y la clase, en variables diferentes X, Y respectivamente. Luego dividimos los conjuntos de datos en dos porciones y de forma aleatoria; 80% para los datos de entrenamiento y 20% para los datos de validación del modelo.

En la figura 49, se muestra el código de la separación de las características y de la clase del conjunto de datos original, y de la división de datos del 80% para el entrenamiento del modelo y del 20% para la validación de modelo.

Figura 49: Código de división del conjunto de datos

```
array = data.values
X_data = array[:,0:30]
Y_data = array[:,30]
```

```
X_train_data, X_test_data, Y_train_data, Y_test_data = (
    train_test_split(X_data, Y_data, test_size=0.20, random_state=7))
```

De los 11055 sitios web preprocesados 8844 sitios web serán utilizados para el entrenamiento y construcción del modelo de detección y 2211 sitios web se utilizan para la validación del modelo. La división de los datos se realiza en los tres conjuntos de datos trabajado en el punto anterior: datos originales, datos con importancia y datos con PCA.

En la tabla 2 se muestran las dimensiones de los tres datos para entrenamiento y para validación. Los tres conjuntos de datos tienen 8844 registros para entrenamiento y 2211 registros para validación. Se observa que el conjunto de datos original tiene 30 características; el conjunto de datos data_import tiene 19 características y el conjunto de datos data_pca tienen 5 características.

Tabla 2: Dimensiones de los datos para entrenamiento y validación

	Data_orig	Data_Import	Data_PCA
Data Entrenamiento	(8844, 30)	(8844, 19)	(8844, 5)
Data Validación	(2211, 30)	(2211, 19)	(2211, 5)

3.3 Fase 3. Selección de algoritmos.

Implementamos el código en Python, y evaluamos el rendimiento de los algoritmos de clasificación. En esta primera evaluación utilizamos la máxima cantidad de algoritmos de clasificación y entre ellos evaluamos a los siguientes algoritmos: KNN (KN vecinos más cercano), LDA (Análisis de discriminante lineal), RL(regresión logística), NB (Naive Bayes Gaussiano), BDT(Clasificador Bagging con el estimador Decision Tree), CART(Clasificador Decision Tree), RF (Random Forest), ET(Extra Tree), AB (AdaBoost), GB(Gradient Boosting) y SVM (Support Vector Machine).

En la figura 50 se muestra el código con el listado de algoritmos a evaluar, donde se agregan a una lista denominada modelos para automatizar la evaluación de todos los algoritmos en una sola ejecución en un código posterior.

Figura 50: Código con algoritmos a evaluar el rendimiento

```
modelos=[]
modelos.append(("KNN", KNeighborsClassifier()))
modelos.append(("LDA", LinearDiscriminantAnalysis()))
modelos.append(("RL", LogisticRegression()))
modelos.append(("NB", GaussianNB()))
modelos.append(("BDT", BaggingClassifier(
    base_estimator=DecisionTreeClassifier())))
modelos.append(("CART", DecisionTreeClassifier()))
modelos.append(("RF", RandomForestClassifier()))
modelos.append(("ET", ExtraTreesClassifier()))
modelos.append(("AB", AdaBoostClassifier()))
modelos.append(("GB", GradientBoostingClassifier()))
modelos.append(("SVM", SVC()))
```

Utilizamos el conjunto de datos (originales, originados de características importantes y basados en pca) de entrenamiento y evaluamos el

rendimiento de los algoritmos con sus parámetros por defecto. Para la evaluación del rendimiento utilizamos la validación cruzada con 10 iteraciones, y obtenemos el promedio del accuracy de las 10 iteraciones y la desviación estándar de cada algoritmo. En la Tabla 3 se muestra el rendimiento promedio de todos los algoritmos evaluados con sus características por defecto, con los tres conjuntos de datos.

Tabla 3: Rendimiento promedio de los algoritmos base

	Accuracy data Original	Accuracy data Import	Accuracy Data PCA
KNN	94.131861	93.340210	91.271020
LDA	91.768643	91.666897	89.925646
RL	92.435693	92.153015	89.857811
NB	59.870836	59.418603	88.320104
BDT	96.483537	95.533655	94.177007
CART	96.189547	95.262264	93.396579
RF	97.015006	96.087609	94.663203
ET	97.037541	96.110246	94.844032
AB	93.430593	93.057596	89.303564
GB	94.606654	93.973426	90.864113
SVM	94.504934	94.233530	90.660621

En la tabla 3, se observa que los algoritmos RF y ET tienen el mejor rendimiento con la métrica accuracy en los tres conjuntos de datos, sin embargo, los algoritmos BDT y CART no podemos descartarlos debido a que tienen resultados muy cercanos a RF y ET, en los tres conjuntos de datos; y hace suponer que deben ser analizados a mayor profundidad en el problema de la detección de phishing, así mismo se visualiza que el algoritmo NB obtiene un rendimiento promedio muy bajo y que sale del contexto de los demás resultados, los algoritmos RL y LDA tienen bajos resultados en los tres conjuntos de datos y por lo que analizaremos de forma gráfica los resultados de los algoritmos sin considerar a NB, RL y LDA.

También se observa que el conjunto de datos originales muestra los mejores resultados, pero los otros conjuntos de datos tienen resultados no muy inferiores y que es necesario seguir evaluando ellos resultados con los tres conjuntos de datos.

En la figura 51, se observa que con los datos originales los algoritmos que muestran mejor rendimiento son RF y ET, y que no debemos descartar a BDT para la búsqueda de mejores parámetros. En la figura 52 con el conjunto de datos data_import se observa similares resultados.

Figura 51: Rendimiento de algoritmos base con data original

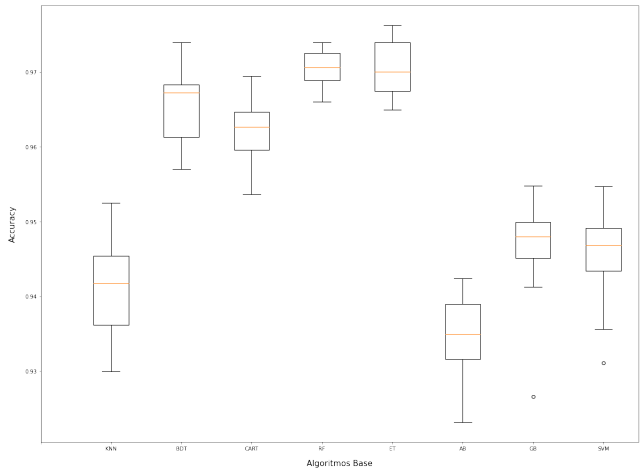
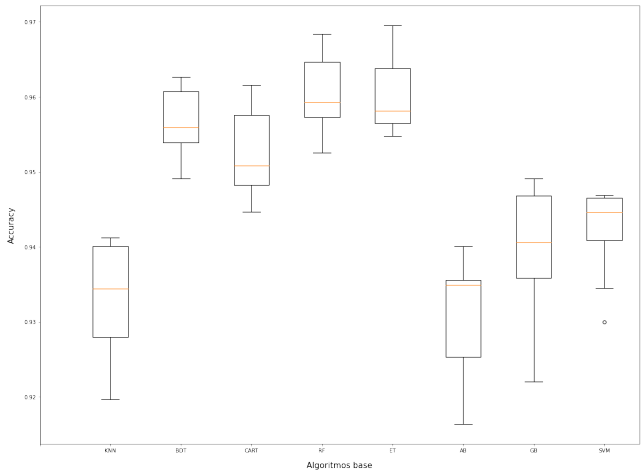


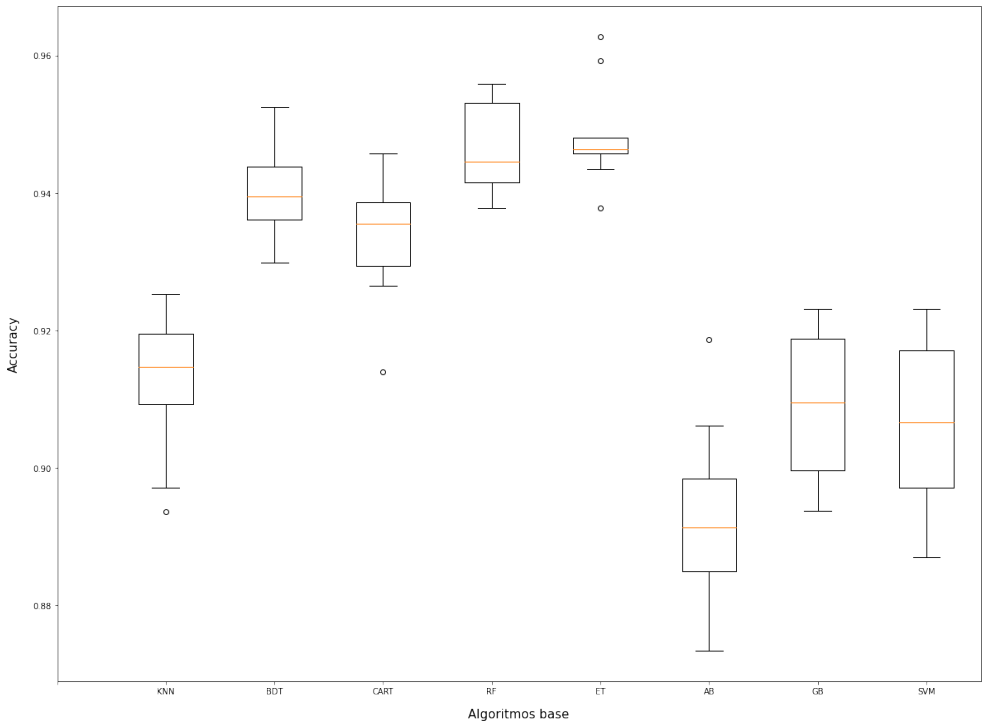
Figura 52: Rendimiento de algoritmos base con data importante



En la figura 53, se observa que con el conjunto de datos data_pca los algoritmos que muestran mejor rendimiento son de forma similar a lo mostrado con los conjuntos de datos originales; es decir los algoritmos

que brindan los mejores resultados son RF, ET y que también no debemos dar por descartado al algoritmo BDT que tiene resultados dentro de la caja de RF.

Figura 53: Rendimiento de algoritmos base con data PCA



Después de evaluar los algoritmos base con los parámetros por defecto y utilizando los tres conjuntos de datos, se visualiza en las figuras 51, 52 y 53, que el algoritmo que tiene mejores resultados y se debe seleccionar para un entrenamiento y optimización adecuado es el RF; sin embargo el algoritmo ET y BDT obtienen resultados muy cercanos y que optimizando los parámetros pueden obtener mejores resultados; por lo que en esta fase seleccionamos al los algoritmos RF (random Forest), ET (Extra Tree) y BDT (Clasificador Bagging con el estimador Decision Tree).

3.4 Fase 4. Entrenamiento.

Implementamos el código en Python, y para el entrenamiento base, con los algoritmos seleccionados en la fase anterior.

En la tabla 4, se muestra el accuracy promedio y la desviación estándar de los resultados de evaluación de los algoritmos seleccionados (BDT, RF y ET) con una validación cruzada de 10 interacciones, con los tres conjuntos de datos. Los algoritmos que son utilizados para el entrenamiento y la optimización serán BDT, RF y ET.

Tabla 4: Rendimiento de los algoritmos seleccionados

	Acc. Data	Acc. Import	Acc. PCA	Desv. Data	Desv. Import	Desv. PCA
BDT	96.460912	95.725912	94.267467	0.468608	0.471090	0.641945
RF	96.992305	96.031137	94.640489	0.312540	0.499716	0.711641
ET	97.048853	96.042386	94.923218	0.333342	0.543073	0.745204

Se muestra que con los datos originales el algoritmo Extra Tree, tiene el mayor accuracy con 97.05% que los otros algoritmos, seguido por Random Forest con el 97.00% de accuracy, seguido por BDT con 96.46%. Con el conjunto de datos importantes, Extra Tree tiene un accuracy de 96.04% y Random Forest tiene un 96.03% con muy similares resultados. Y con el conjunto de datos PCA Extra Tree tiene un accuracy de 94.92% y Random Forest tiene un accuracy de 94.64%. Con estos resultados no podemos descartar a ningún algoritmo para la optimización de los parámetros; pero si se puede visualizar que se obtienen mejores resultados con los datos de entrenamiento del conjunto de datos original, y que utilizará para la optimización de características.

Haciendo uso de Grid Search, buscamos los mejores parámetros para los algoritmos BDT, RF y ET. Con los mejores parámetros se entrena cada modelo con el conjunto de datos de entrenamiento, y para validar los resultados utilizamos validación cruzada con 10 interacciones.

En la figura 54 se muestra en código utilizando GridSearch para la búsqueda de los mejores parámetros para el algoritmo Random Forest, donde como resultado se muestra el mejor accuracy de 96.98%, con 100

como el número de estimadores, criterio entropy, mínimo simples Split 2, max_feaures 3 y Bootstrap True.

Figura 54: Código de la búsqueda de los mejores parámetros para BDT

```
#Grid Search in Random Forest
n=np.array([50,100,250,500,750])
c=np.array(['gini','entropy'])
m=np.array([2, 3])
max_features=np.array([2, 3, 4, 5, 6, 7, 8, 9, 10])
b =np.array(['True','False'])
rs = np.array([7])
param_grid=dict(n_estimators=n, criterion=c, min_samples_split=m,
                max_features=max_features, bootstrap=b, random_state=rs)
model=RandomForestClassifier()
grid=GridSearchCV(model, param_grid=param_grid, cv=5)
grid.fit(X_train_data, Y_train_data)
print(f"Mejor Accuracy: {grid.best_score_.mean()*100.0:,.2f}%")
print(f"Mejor n_estimators: {grid.best_estimator_.n_estimators}")
print(f"Mejor criterion: {grid.best_estimator_.criterion}")
print(f"Mejor min_ss: {grid.best_estimator_.min_samples_split}")
print(f"Mejor max_features: {grid.best_estimator_.max_features}")
print(f"Mejor bootstrap: {grid.best_estimator_.bootstrap}")
```

```
Mejor Accuracy: 96.98%
Mejor n_estimators: 100
Mejor criterion: entropy
Mejor min_samples_split: 2
Mejor max_features: 3
Mejor bootstrap: True
```

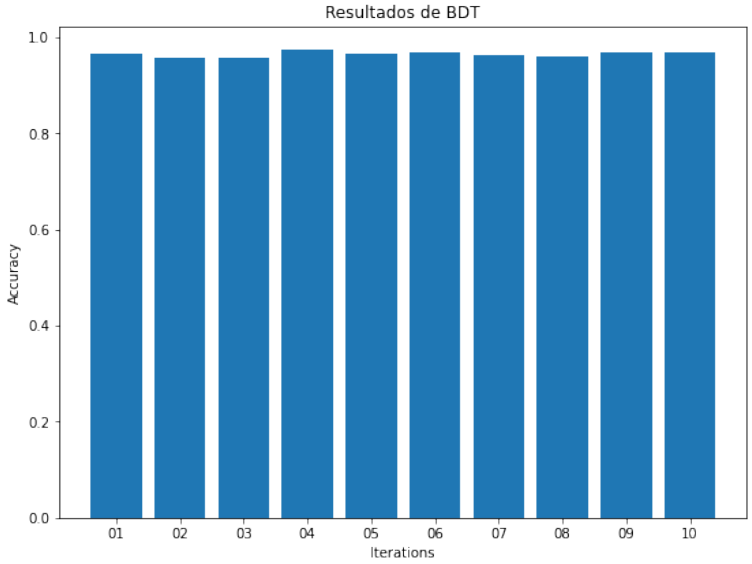
Con los mejores parámetros de cada algoritmo elegido, realizamos la evaluación de los resultados con validación cruzada de 10 iteraciones. En la tabla 5, se muestran el accuracy promedio de los algoritmos optimizados (con sus mejores parámetros) y se observa que ET tiene un accuracy de 97.09%, RF tiene un accuracy de 97.06% y BDT un accuracy de 96.72%, y con los resultados de la desviación estándar no se puede afirmar que hay un algoritmo con los mejores resultados.

Tabla 5: Rendimiento de los algoritmos optimizados

	BDT	RF	ET
Accuracy	96.720940	97.060127	97.094077
Desv_Stand	0.361184	0.307833	0.247846

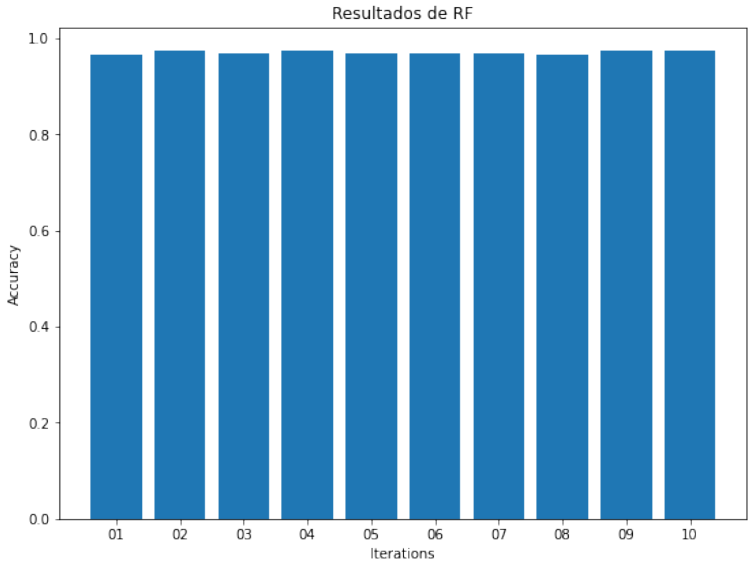
En la figura 55 se muestran los resultados del algoritmo BDT con las 10 interacciones de la validación cruzada, donde se muestra resultados muy similares en cada interacción.

Figura 55: Accuracy en el entrenamiento de BDT



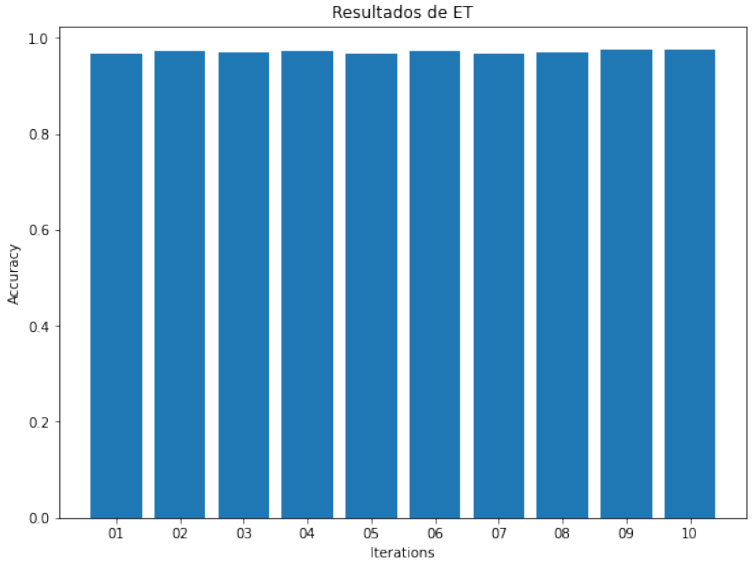
En la figura 56 se muestran los resultados del algoritmo RFcon las 10 interacciones de la validación cruzada, donde se muestra resultados muy similares en cada interacción.

Figura 56: Accuracy en el entrenamiento de RF



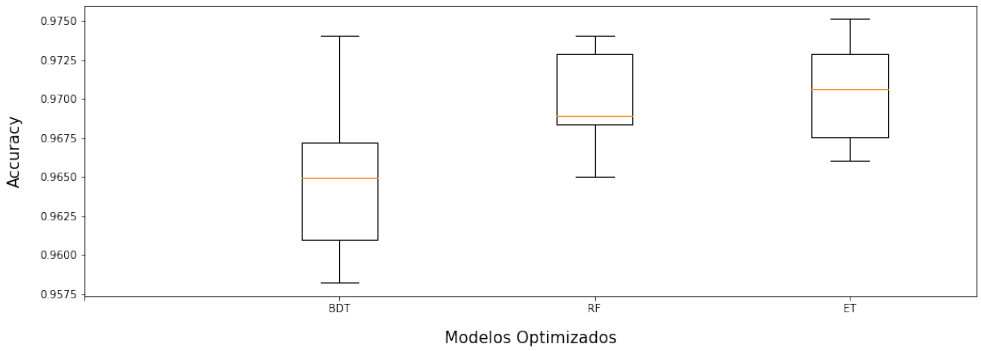
En la figura 57 se muestran los resultados del algoritmo ET con las 10 interacciones de la validación cruzada, donde se muestra resultados muy similares en cada interacción.

Figura 57: Accuracy en el entrenamiento de ET



Los resultados de los tres algoritmos optimizados se muestran en forma gráfica en la figura 58, donde se observa que los algoritmos RF y ET, tienen los mejores y muy similares resultados. Además, se visualiza los datos más centralizados del ET sobre RF. Sin embargo, en el mejor caso del algoritmo BDT puede igualar y superar al peor caso de los algoritmos RF y ET. Por lo que se guardarán los tres modelos para la evaluación con los datos de validación del modelo.

Figura 58: Resultados del Accuracy de algoritmos optimizado



En la figura 59, se muestra el código de afinamiento, generación y almacenamiento de los 3 modelos.

Figura 59: Código de afinamiento y almacenamiento de los modelos

```
model_BDT.fit(X_train_data, Y_train_data)
filename_BDT = 'Modelos/ModeloBDT.sav'
jbl.dump(model_BDT, filename_BDT)
```

```
['Modelos/ModeloBDT.sav']
```

```
model_RF.fit(X_train_data, Y_train_data)
filename_RF = 'Modelos/ModeloRF.sav'
jbl.dump(model_RF, filename_RF)
```

```
['Modelos/ModeloRF.sav']
```

```
model_ET.fit(X_train_data, Y_train_data)
filename_ET = 'Modelos/ModeloET.sav'
jbl.dump(model_ET, filename_ET)
```

```
['Modelos/ModeloET.sav']
```

3.5 Fase 5. Detección de Phishing.

La implementación de esta fase se realiza en dos formas.

1. Detección de phishing con la URL, que permite verificar en línea si un sitio web es legítimo o es sitio web phishing. En este caso el código realiza dos funciones, la función de recolectar datos del sitio web y luego realiza la predicción del sitio a que clase corresponde.
   En la figura 60 se visualiza que la función se prueba con un sitio web, y el sistema da como resultado la predicción, que se muestra en la parte inferior, en el ejemplo “El sitio web es Legítimo”.

Figura 60: Prueba de detección de un sitio web en línea

```
predecir_url("http://www.unprg.edu.pe/")
```

```
El sitio web es LEGÍTIMO
```

2. La detección de sitios web, con las características, esta funcionalidad se ha implementado con la finalidad de comprobar el funcionamiento con los sitios web phishing que ya no están en línea, pero que si se cuenta con los datos de las características.
   En la figura 61, se hace uso del modelo RF para las predicciones. Los resultados se imprimen en la parte inferior de cada detección.

Figura 61: Prueba de detección de un sitio web con datos

```
from termcolor import colored
def predecir_data(dataPredecir):
    dataP= np.append([dataPredecir], [[]], axis=1)
    prediccion = loaded_model_RF.predict(dataP[:,0:30])
    if prediccion[0] == 1:
        salida = "El sitio web es LEGÍTIMO"
        color = "green"
    else:
        salida = "El sitio web es PHISHING"
        color = "red"
    print(colored(salida, color))
```

```
data=[1,1,-1,1,1,-1,1,-1,-1,1,1,1,1,-1,-1,-1,-1,-1,0,1,1,1,1,-1,-1,-1,-1,
predecir_data(data)
```

```
El sitio web es PHISHING
```

```
data=[1,1,1,1,1,-1,0,1,1,1,1,1,-1,0,0,-1,-1,-1,0,1,1,1,1,-1,1,1,1,1,-1,-1
predecir_data(data)
```

```
El sitio web es LEGÍTIMO
```

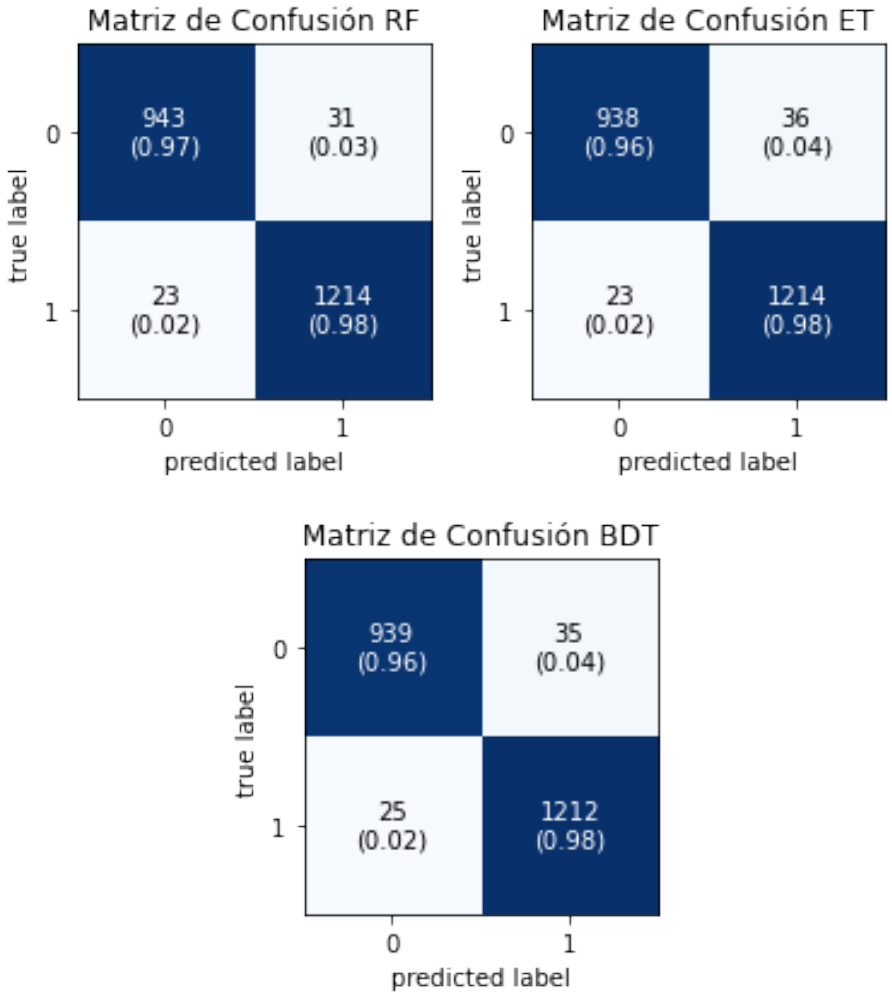
3.6 Fase 6. Evaluación del rendimiento.

Para evaluar el rendimiento del sistema en la detección de phishing, se hacen pruebas de detección, con el conjunto de datos de validación y con los modelos generados BDT, RF y ET.

En la figura 62, se muestra la matriz de confusión de cada modelo, donde se muestra que el modelo RF es la que proporciona los mejores

resultados en la detección de sitios web phishing y sitios web legítimos, igualando con ET en los Verdaderos Positivos (VP) y superando a ET y BDT en los Verdaderos Negativos (VN); además se observa que tiene menores errores de Falsos Positivos (FN) y de Falsos Positivos (FP).

Figura 62: Matriz de confusión de los modelos evaluados



En la matriz de confusión del modelo RF, se observa que 943 (97%) sitios web phishing detectados correctamente como sitios web phishing (VP); 1217 (98%) sitios web legítimos detectados correctamente como sitios web legítimos (VN), además se muestran 37 (4%) de sitios web phishing incorrectamente clasificados como sitios legítimos (FN) y 20 (2%) sitios web legítimos clasificados incorrectamente sitios web phishing.

En la figura 63, se muestra el reporte de clasificación del modelo DT, con los resultados de las métricas de accuracy al 97.56%, recall (TVP) del 96.82%, especificidad (TVN) de 98.14%, precisión de 97.62%.

Figura 63: Reporte de clasificación del modelo RF

	precision	recall	f1-score	support
-1	0.9762	0.9682	0.9722	974
1	0.9751	0.9814	0.9782	1237
accuracy			0.9756	2211
macro avg	0.9756	0.9748	0.9752	2211
weighted avg	0.9756	0.9756	0.9756	2211

En la tabla 6, se muestra la matriz de confusión del sistema construido con el algoritmo Random Forest, y las pruebas realizadas con 2211 sitios web, donde de muestran 943 verdaderos positivos, 1214 verdaderos negativos, 31 falsos negativos y 23 falsos positivos.

Tabla 6: Matriz de confusión, del sistema

	Clases	**Resultado del Sistema Propuesto**		
		Phishing	**Legítimo**	**Total**
Sitio web (Clasificación real)	Phishing	943 (VP)	31 (FN)	974
	Legítima	23 (FP)	1214 (VN)	1237

En la tabla 7 se muestra los resultados de la clasificación del sistema propuesto; donde se muestra que el 97.56% es la proporción de clasificación correcta del sistema en global, el 2.44% es el error en la clasificación errónea del sistema en global, el 96.82% de sitios web phishing clasificadas correctamente, el 98.14% de sitios web legítimos clasificados correctamente y el 97.62% es la proporción de sitios web phishing clasificados correctamente en relación a la clasificación como sitios web phishing por el sistema.

Tabla 7: Resultados del rendimiento del sistema

Métrica Evaluada	Fórmula	Resultado
Accuracy	Accuracy = (VP+VN) /(VP+VN+FP+FN)	97.56%
Classification Error	Classification Error = (FP+FN)/(VP+VN+FN+FP)	02.44%
Recall (TVP)	recall = (VP)/(VP+FN)	96.82%
Specificity (TVN)	Specificity = (VN)/(VN+FP)	98.14%
Precision	Precision = VP / (VP + FP)	97.62%

IV. Conclusiones

- Se evidenció que en la actualidad hay un crecimiento acelerado de los ataques informáticos, y en especial en los ataques phishing.
- Se realizó la caracterización del proceso de ciberseguridad y la detección de phishing, demostrando que hay insuficiencias prácticas que usen la información de los sitios web, la información de la inteligencia de amenazas y las técnicas de machine learning, para la detección de sitios web falsos.
- Se elaboró un sistema de detección de sitios web phishing, utilizando las características de los sitios web en la barra de direcciones y el código fuente, la inteligencia de amenazas y se integró con la lógica, las técnicas y los algoritmos de machine learning.
- El sistema de detección de phishing se desarrolló es seis fases: Recolección de datos, preparación de los datos, selección de algoritmos, entrenamiento, detección de phishing y evaluación del rendimiento.
- El sistema utilizó información de 11055 sitios web con 30 características, catalogados como sitios web phishing y sitios web legítimos, de los cuales 8844 sitios web se utilizaron para la fase de entrenamiento y 2211 sitios web para la fase de evaluación de rendimiento.
- Se seleccionaron los algoritmos Random Forest, Bagging y Extra Trees para la fase de entrenamiento y en la optimización el modelo de Random Forest es el que presenta los mejores resultados de rendimiento.
- En la evaluación del rendimiento del sistema de detección de phishing con 2211 sitios web, demostrando el rendimiento en de 97.56% de la detección correcta del sistema en forma global, así mismo el 96.82% de sitios web

phishing clasificadas correctamente, el 98.14% de sitios web legítimos clasificados correctamente

- Los resultados obtenidos por el sistema implementado, supera a los resultados obtenidos por los estudios previos, en la detección correcta de los sitios web phishing y los sitios web legítimos.

V. Referencias

Abdulhamit , S., & Kremicb, E. (2020). Comparison of Adaboost with MultiBoosting for Phishing Website Detection. *Procedia Computer Science*, 272–278.

Abutair, H. Y., Belghith, A., & Al-Ahmadi, S. A. (2018). CBR-PDS: a case-based reasoning phishing detection system. *Journal of Ambient Intelligence and Humanized Computing*, 2593-2606. doi: https://doi.org/10.1007/s12652-018-0736-0.

Ahrend, J. M., Jirotka, M., & Jones, K. (2016). *On the collaborative practices of cyber threat intelligence analysts to develop and utilize tacit Threat and Defence Knowledge.* pp. 1-10, doi: 10.1109/CyberSA.2016.7503279: International Conference On Cyber Situational Awareness, Data Analytics And Assessment (CyberSA).

Alexa the Web Information Company. (1996). *Alexa the Web Information Company*. Recuperado el 10 de November de 2011, de http://www.alexa.com/

Ali, W., & Ahmed, A. A. (2019). Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting. *IET Information Security*, Pages 659-669; doi: https://doi.org/10.1049/iet-ifs.2019.0006.

Aljofey, A., Jiang, Q., Qu, Q., Huang, M., & Niyigena, J.-P. (2020). An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network from URL. *Electronics* , doi: https://doi.org/10.3390/electronics9091514.

Alsariera, Y. A., Adeyemo, V. E., Balogun, A. O., & Alazzawi, A. K. (2020). AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites. *IEEE Access, vol. 8*, pp. 142532-142542, doi: 10.1109/ACCESS.2020.3013699.

Anderson, J. P. (1972). *Computer Security Technology Planning Study.* Washington: Deputy for command and Management Systems.

Anupam, S., & Kar, A. K. (2020). Phishing website detection using support vector machines and nature-inspired optimization algorithms. *Telecommunication Systems*, pages 17–32; doi https://doi.org/10.1007/s11235-020-00739-w.

APWG. (2021). *Phishing Activity Trends Report, 4th Quarter 2020.* Anti-Phishing Working Group, Inc.

Baca Urbina, G. (2016). *Introducción a la seguridad informática.* Distrito Federal, MÉXICO: Grupo Editorial Patria.

Bendovschi, A. (2015). Cyber-Attacks – Trends, Patterns and Security Countermeasures. *Procedia Economics and Finance* , 24–31.

Borja-Robalino, R., Monleón-Getino, A., & Rodellar, J. (2020). Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning. *Revista Ibérica de Sistemas e Tecnologias de Información*, 184-196.

Cascavilla, G., Tamburri, D. A., & Heuvel, W.-J. D. (2021). Cybercrime threat intelligence: A systematic multi-vocal literature review. *Computers & Security*, https://doi.org/10.1016/j.cose.2021.102258.

Chavan, S., Inamdar, A., Dorle, A., Kulkarni, S., & Wu, X.-W. (2020). Phishing Detection: Malicious and Benign Websites Classification Using Machine Learning Techniques. *Algorithms for Intelligent Systems. Series Editors: Bansal, Jagdish Chand, Deep, Kusum, Nagar, Atulya K*, 437-445.

Christou, O., Pitropakis, N., Papadopoulos, P., McKeown , S., & Buchanan, W. J. (2020). *Phishing URL Detection Through Top-Level Domain Analysis: A Descriptive Approach.* Edinburgh: School of Computing, Edinburgh Napier University.

Clarin. (18 de Setiembre de 2020). *Clarin*. Obtenido de https://www.clarin.com

De la Hoz, E. M. (2016). *Mapas auto-organizativos probabilísticos y análisis en componentes de conexiones para la deteccion de anomalías en redes de computadoras.* Granada: Universidad de Granada.

De la Hoz, E., De la Hoz, E. M., Ortíz, A., & Ortega, J. (2012). *Modelo de detección de intrusiones en sistemas de red, realizando selección de caracterísitcas con FDR y entrenamiento y clasificación con SOM.* Barranquilla: Corporación Universidad de la Costa.

Denning, D. E. (1987). *An Intrusion Detection Model.* IEEE.

Divindat . (2021). *División de Investigación de Delitos de Alta Tecnología de la Policía.* Lima: El Peruano.

Escrivá Gascó, G., Romero Serrano, R. M., & Ramada, D. J. (2013). *Seguridad Informática.* Madrid: Macmillan Iberia, S.A.

FBI. (11 de Julio de 2019). *Federal Bureau of Investigation - U.S. Department of Justice*. Obtenido de https://www.fbi.gov/investigate

Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L., & Wang, J. (2018). The application of a novel neural network in the detection of phishing websites. *Springer-Verlag GmbH Germany*.

Gartner. (2013). *Threat intelligence: What is it, and how can it protect you fron today advanced cyber-attacks.* Stamford: Gartner, Inc.

Gori, M. (2018). Machine Learning: A Constraint-Based Approach. *Morgan Kaufman*, https://doi.org/10.1016/C2015-0-00237-4.

Harinahalli, L., & BoreGowda, G. (2020). Phishing website detection based on effective machine learning approach. *Journal of Cyber Security Technology*, Pages 1-14, doi: https://doi.org/10.1080/23742917.2020.1813396.

Heberlein, T. (1995). *Network Security Monitor.* California: UIniversidad de California.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. (2014). *Metodología de la Investigación* (6ta Edición ed.). México D.F.: Mc Graw Hill Education.

Hurwitz, J., & Kirsch, D. (2018). *Machine Learning For Dummies, IBM Limited Edition.* United States of America: John Wiley & Sons, Inc.

ISACA. (2015). *Cybersecurity Fundamentals. Study Guide.* Rolling Meadown, USA: ISACA.

ISO. (1 de Agosto de 2017). *International Organization for Standardization*. Obtenido de https://www.iso.org/search/x/query/27032

ISO27000. (13 de Julio de 2017). *Sistema de Gestión de la Seguridad de la Información*. Obtenido de http://www.iso27000.es/

Jain, A. K., & Gupta, B. B. (2017). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommun Syst 68*, 687–700 doi: https://doi.org/10.1007/s11235-017-0414-0.

Jain, A. K., & Gupta, B. B. (2018). PHISH-SAFE: URL Features-Based Phishing Detection System Using Machine Learning. *Advances in Intelligent Systems and Computing, vol 729. Springer, Singapore*, Pag 467-474. https://doi.org/10.1007/978-981-10-8536-9_44.

Jakobsson, M., & Myers, S. (2006). Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. . *Wiley-Interscience*.

Kulkarni, A. D., & Brown, L. L. (2019). Phishing Websites Detection using Machine Learning. *Computer Science Faculty Publications and Presentations*, Paper 20.

Kumar, R., Gunasekaran, Nivetha, R., Sangeetha , P. K., Shanthini, G., & Vignesh., A. S. (2019). Url Phishing data analysis and detecting phishing attacks using Machine Learning in NLP. *International Journal of Engineering Applied Sciences and Technology*, 23-31.

Lakshmi, L., Reddy, M., Santhaiah , C., & Reddy , U. J. (2021). Smart Phishing Detection in Web Pages using Supervised Deep Learning Classification and Optimization Technique ADAM. *Wireless Personal Communications*, pages 3549–3564.

Larrieu-Let, E. (2015). *Ciberseguridad.* Montevideo, Uruguay: ISACA, Montevideo Chapter .

Marsh & Microsoft. (2020). *Estado del Riesgo Cibernético en Latinoamérica en tiempos de COVID-19.* Nueva York: Marsh LLC.

Martínez Puentes, J. (2011). *Sistema Inteligente de Detección de Intrusiones.* Madrid: Universidad Complutense de Madrid.

Medina, M., & Molist, M. (2017). *Ciberseguridad. Tendencias 2017.* Valencia, España.: Universidad Internacional de Valencia.

Merwe, A., Loock, M., & Dabrawski, M. (2005). Characteristics and responsibilities involved in a Phishing attack. *WISICT '05: Proceedings of the 4th international symposium on Information and communication technologies*, 249–254.

Modi, H. (2019). *Cybercrime's Innovation Machine.* California: NETSCOUT Threat Intelligence.

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Comput & Applic*, 443–458 doi: https://doi.org/10.1007/s00521-013-1490-z.

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2015). *Phishing Websites Features.* Huddersfield: School of Computing and Engineering, University of Huddersfield.

Mohammed, M., Khan, M. B., & Mohammed, E. B. (2017). *Machine Learning: Algorithms and Applications.* London: Taylor & Francis Group.

Mueller , A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python.* Sebastopol: O'Reilly Media, Inc.

Niakanlahiji, A., Chu, B.-T., & Al-Shaer, E. (2018). PhishMon: A Machine Learning Framework for Detecting Phishing Webpages. *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 220-225, doi: 10.1109/ISI.2018.8587410.

NIST. (2018). *Framework for Improving Critical Infrastructure Cybersecurity.* Gaithersburg, Maryland: National Institute of Standards and Technology.

Ollmann, G. (2004). The Phishing Guide. *NGSSoftware Insight Security Research*, 1-42.

Opara, C., Wei, B., & Chen, Y. (2020). HTMLPhish: Enabling PhishingWeb Page Detection by Applying Deep Learning Techniques on HTML Analysis. *IJCNN*, DOI:10.1109/IJCNN48605.2020.9207707.

Patil, V., Thakkar, P., Shah, C., Bhat, T., & Godse, S. P. (2018). Detection and Prevention of Phishing Websites Using Machine Learning Approach. *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697412.

PECERT. (2021). *Alerta Integrada de Seguridad Digital.* Lima: Centro Nacional de Seguridad Digital .

Rami , M. M., Fadi , T., & Lee , M. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications* , 443–458.

Real Academia Española. (5 de Julio de 2017). *Diccionario de la lengua española*. Obtenido de http://dle.rae.es

Sánchez, M. (6 de Julio de 2011). *Infraestructuras Críticas y Ciberseguridad*. Obtenido de https://manuelsanchez.com/2011/07/06/infraestructuras-criticas-y-ciberseguridad/

Sandoval, L. J. (2018). Algoritmos de Aprendizaje Automático para análisis y prediccion de datos. *Revista Tecnologica*, Pag 36-40.

Sarkar, D., Bali, R., & Sharma, T. (2018). *Practical Machine Learning with python.* New York: Spring Street. doi:https://doi.org/10.1007/978-1-4842-3207-1.

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *Computer Science*, doi: https://doi.org/10.1007/s42979-021-00592-x.

Scarfone, K., & Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94 .* Gaithersburg: National Institute of Standards and Technology.

Shahrivari, V., Darabi, M. M., & Izadi, M. (2020). *Phishing Detection Using Machine Learning Techniques.* New York: Cornell University.

Singh, R., & Sharma, T. (2019). Present Status of Distributed Denial of service (DDoS) Attacks in Internet World. *International Journal of Mathematical, Engineering and Management Sciences* , 1008-1017.

Sountharrajan, S., Nivashini, M., Shandilya, S. K., Suganya, E., Bazila , A., & Karthiga, M. (2019). Dynamic Recognition of Phishing URLs Using Deep Learning Techniques. *EAI/Springer Innovations in Communication and Computing*, 27-53.

Subasi, A. (2020). *Practical Machine Learning for data analysis using Phyton.* London: Elsevier Inc.

Tounsi, W., & Rais, H. (2018). A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, Pages 212-233, doi: https://doi.org/10.1016/j.cose.2017.09.001.

Trend Micro. (2021). How to Reduce the Risk of Phishing and Ransomware. *Osterman Research*, 1-29.

Ubing , A. A., Binti Jasmi, S. K., Azween , A., Jhanjhi, N. Z., & Supramaniam, M. (2019). Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning. *International Journal of Advanced Computer Science and Applications*.

Vakili, M., Ghamsari, M., & Rezaei, M. (2020). Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification. *Cornell University*.

Wang, W., Zhang, F., Luo, X., & Zhang, S. (2019). PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks. *Security and Communication Networks*, Paper 15.

Wei, B., Hamad, R., Yang, L., He, X., Wang, H., Gao, B., & Woo, W. (2019). *A Deep-Learning-Driven Light-Weight Phishing Detection Sensor.* Newcastle: Sensors Northumbria University.

Yang, L., Zhang, J., Wang, X., Li, Z., Li, Z., & He, Y. (2021). An improved ELM-based and data preprocessing integrated approach for phishing detection considering comprehensive features. *Expert Systems with Applications*, doi, https://doi.org/10.1016/j.eswa.2020.113863.

Yi, P., Guan, Y., Zou, F., Yao, Y., WeiWang, & Zhu, T. (2018). Web Phishing Detection Using a Deep Learning Framework. *Wireless Communications and Mobile Computing*, 9 paginas doi: https://doi.org/10.1155/2018/4678746.

Zabihimayvan, M., & Doran, D. (2019). Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection. *IEEE International Conference on Fuzzy Systems*, pag. 1-6 doi: 10.1109/FUZZ-IEEE.2019.8858884.

Zamir, A., Khan, H. U., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A., & Hamdani, M. (2020). Phishing web site detection using diverse machine learning algorithms. *Electron*, 65-80.

Colloquium
Editorial - Centro de Formación