Recibido: 21/10/2020 / Aceptado: 14/11/2020
doi: https://doi.org/10.26439/interfases2020.n013.4886

Comparación de técnicas de *machine learning* para detección de sitios web de *phishing*

Andres Eduardo Moncada Vargas
Universidad de Lima, Lima, Perú
20152102@aloe.ulima.edu.pe

Resumen

El *phishing* es el robo de datos personales a través de páginas web falsas. La víctima de este robo es dirigida a esta página falsa, donde se le solicita ingresar sus datos para validar su identidad. Es en ese momento que se realiza el robo, ya que al ingresar sus datos, estos son almacenados y usados por el *hacker* responsable de dicho ataque para venderlos o ingresar a las entidades y realizar robos o estafas. Para este trabajo se ha investigado sobre distintos métodos de detección de páginas web *phishing* utilizando técnicas de *machine learning*. Así, el propósito de este trabajo es realizar una comparación de dichas técnicas que han demostrado ser las más efectivas en la detección de los sitios web *phishing*. Los resultados obtenidos demuestran que los clasificadores de árboles, denominados Árbol de Decisión y Bosque Aleatorio, han alcanzado las mayores tasas de precisión y efectividad, con valores de entre 97 % y 99 % en la detección de este tipo de páginas.

PALABRAS CLAVE: *phishing* / *machine learning* / ciberseguridad / clasificador / Bosque Aleatorio / *dataset* / *grid search*

Abstract

A comparison of machine learning techniques for detection of phishing websites

Phishing is the theft of personal data through fake websites. Victims of this type of theft are directed to a fake website, where they are asked to enter their data to validate their identity. At that moment, theft is carried out, since entered data are stored and used by the hacker responsible for said attack to sell them or enter to websites and perform a fraud or scam. In order to conduct this work, we researched different methods for detecting phishing websites by using machine learning techniques. Thus, the purpose of this work is to compare machine learning techniques that have demonstrated to be the most effective methods to detect phishing websites. The results show that decision tree classifiers such as Decision Tree and Random Forest have achieved the highest accuracy and efficacy rates, with values between 97% and 99%, in detecting these types of websites.

KEYWORDS: phishing / machine learning / cybersecurity / classifier / Random Forest / dataset / grid search

1. INTRODUCCIÓN

El ataque cibernético conocido como *phishing* constituye una de las mayores amenazas en la actualidad, ya que es el medio por el cual se han realizado la mayoría de los robos y estafas cibernéticas en los últimos años. El *phishing* es comúnmente conocido como el robo de datos personales por medio de una página web falsa, denominada página *phishing*, creada para que las víctimas de este ataque ingresen sus datos para "validar" su identidad. Estas páginas *phishing* usualmente suelen imitar el diseño y configuración de páginas legítimas de distintas organizaciones, comúnmente bancos y empresas con servicios bancarios. El medio más usado para difundir estas páginas *phishing* es el correo *spam*. Sin embargo, no es el único medio por el cual se realizan ataques *phishing*, debido a que existen otros métodos como, por ejemplo, el *pharming*, que es el redireccionamiento de un usuario que se encuentra en una página legítima a una página *phishing* a través de enlaces directos implantados en dicha página legítima (Abu-Nimeh, Nappa, Wang y Nair, 2007, p. 1).

Según ESET (2017), Ecuador, Perú y México fueron los tres países latinoamericanos con más ataques de *phishing* en el 2016, teniendo un porcentaje del 20,9 %, 16,6 % y 16,1 % respectivamente. Según estudios realizados, en el 2017 hubo un incremento de ataques en un 15 % respecto del año anterior. En el año 2018 el incremento de ataques fue mayor, ya que se detectaron alrededor de 500 millones de ataques *phishing* en todo el mundo. De todos ellos, el número de ataques *phishing* financieros en el 2018 estuvo cerca del total de ataques *phishing* detectados en el 2017. Estos estudios dejan evidencias del porqué se considera al *phishing* como una de las ciberamenazas más comunes que existe en la actualidad.

Ante los ataques de *phishing*, los datos personales se encuentran en peligro, ya que vivimos en una época donde la internet es usada para la mayoría de las actividades diarias. Por este motivo, en este trabajo se han examinado varias metodologías, propuestas a lo largo de los últimos años, que han demostrado ser capaces de detectar y prevenir los ataques *phishing*. Lamentablemente, los métodos propuestos no son capaces de detectar nuevas variantes de estos ataques, debido a que las métricas más importantes para detectar páginas *phishing* derivan de experiencias humanas (Mao, Bian, Tian, Zhhu, Wei, Li y Liang, 2018, p. 2). Por dicho motivo, en la actualidad se recurre a la inteligencia artificial para poder identificar páginas *phishing* de manera dinámica y automática utilizando para ello diferentes métricas (Abu-Nimeh, 2007; Al-Janabi, 2017; Bulakh, 2016; Chen, 2010; Hota, 2018; Jain, 2016; Mao, 2018; Medvet, 2008; Mourtaji, 2017; Rajab, 2018; Sanglerdsinlapachai, 2010).

Anteriormente, Abu-Nimeh, Nappa, Wang y Nair (2007); Abdelhamid, Thabtah, y Abdel-jaber (2017) compararon diversas técnicas de *machine learning* para poder encontrar la mejor manera de detectar los ataques *phishing*. El objetivo de dichos trabajos fue

comprobar la precisión de dichas técnicas y sus resultados, y aunque la metodología de este trabajo podía ayudar a determinar a la técnica más eficiente entre las implementadas, los resultados y la forma de presentarlos no fueron suficientemente concluyentes para conocer qué métodos y bajo qué supuestos y condiciones alcanzaban los mejores resultados de manera objetiva.

La motivación de este trabajo es encontrar la técnica más eficiente para detectar los ataques *phishing*. Por ese motivo, la labor se enfoca en investigar trabajos que hayan usado técnicas de *machine learning* para la detección de *phishing*, replicarlos, revisar los criterios de calibrado y compararlos para determinar cuál de esas técnicas es la más efectiva para detectar páginas *phishing*. El análisis se conduce sobre un conjunto de características sugeridas en la literatura científica revisada en esta investigación. Las contribuciones que ofrece el presente trabajo son las siguientes:

- Realizar una revisión del uso de técnicas de *machine learning* para la detección de *phishing*. De esta forma, podemos valorar la efectividad de los algoritmos de clasificación estudiados.
- Buscar y seleccionar los *datasets* empleados por autores de investigaciones similares para contrastar los resultados de experimentación aplicados.
- Desarrollar una experimentación extendida sobre los criterios mencionados anteriormente.
- Comparar los resultados de la experimentación tomando en cuenta métricas objetivas como la exactitud, precisión, recuperación y valor F. Esto se concreta realizando una parametrización detallada de los algoritmos de clasificación a usar y se discuten los hallazgos de este trabajo en comparación con los resultados de investigaciones anteriores.

Este trabajo se compone de seis secciones; la primera es la presente introducción. En la sección 2 se presenta el estado del arte, en donde se revisa la literatura existente sobre detección de *phishing* basada en aprendizaje automático. La metodología de trabajo y la experimentación se describen en la sección 3. En la sección 4 se analizan los resultados obtenidos en esta investigación y se debate desde la comparativa con propuestas similares. Finalmente, en la sección 5 se presentan las conclusiones de este trabajo.

2. ESTADO DEL ARTE

Con el paso del tiempo se han diseñado varias herramientas como medida de seguridad para detectar el *phishing*; y, desafortunadamente, las habilidades de los *hackers*, en constante evolución, han sido un obstáculo para las herramientas tradicionales. Esto ocasionó que las antiguas técnicas de detección de *phishing* no funcionen contra los

ataques más recientes (Bulakh, 2016, p. 1). Como consecuencia de esta amenaza evolutiva, se ha decidido usar técnicas de *machine learning* que permitan encontrar nuevas páginas fraudulentas en un periodo de tiempo largo o indefinido (Hota, 2018; Medvet, 2008; Chen, 2010; Bulakh, 2016; Rajab, 2018).

Basándose en los enfoques de distintos autores para detectar correos *phishing* en el pasado, los autores Hota, Shrivas y Hota (2018) decidieron desarrollar un modelo de identificación de correos *phishing* ensamblando dos técnicas de árboles de decisiones, CART y C4.5, en un modelo de clasificación robusto con la intención de reducir el error que estos algoritmos pueden tener por separado. CART es un clasificador que construye un árbol de decisión binario dividiendo el registro de cada nodo en base a una función de un atributo. C4.5 es un árbol de decisión capaz de manejar atributos continuos y discretos, incluyendo registros con valores desconocidos en el entrenamiento de este.

Los autores Mao *et al*. (2018) analizaron varias soluciones anti-*phishing* y descubrieron que las similitudes de diseño dan un indicio alto de detección de este fraude. Lamentablemente, estas similitudes no podían ser comprendidas para detectar nuevos ataques. Decididos a superar esta dificultad, los autores crearon una herramienta capaz de determinar similitudes de diseño con base en reglas determinadas por un mecanismo de análisis de agregación. Para desarrollar esta herramienta, se evaluaron los clasificadores denominados Máquina de Soporte Vectorial y Árbol de Decisión. En el caso de Máquina de Soporte Vectorial, se configuraba el parámetro *gamma* (gama) y se comparaban los resultados obtenidos de varias pruebas realizadas con diferentes valores para este parámetro. En el caso de Árbol de Decisión, se hizo la misma comparación, pero configurando el parámetro *max depth* (máxima profundidad).

Jain y Gupta (2016) observaron que una de las soluciones más efectivas es integrar funciones de seguridad en los navegadores de internet. Por ese motivo, decidieron realizar una *whitelist* autoactualizable. De esta manera, los navegadores podrán mandar una alerta si es que se ingresa a una página web que no se encuentrara en la lista, evitando que el usuario ingrese a una página *phishing*. Además de eso, es posible detectar y agregar a la lista una página nueva gracias al sistema de detección implementado, el cual funciona con los hipervínculos de las páginas a las cuales ingresa el usuario. La ventaja de esta herramienta es que al agregarle más características de detección puede mejorar la exactitud de detectar una página *phishing*. Sin embargo, al hacer eso, el sistema de detección tomará más tiempo en correr para determinar si una página es *phishing* o no.

Las técnicas convencionales pueden ser muy útiles, pero si no están implementadas con un algoritmo que permita un actualizado automático, como el

mostrado anteriormente, entonces dichas técnicas no serán capaces de tener un funcionamiento continuo, a diferencia de las técnicas con *machine learning* que permiten su uso continuo para una detección de *phishing* a largo plazo. Por el motivo mencionado es que se propusieron varias técnicas de *machine learning*, para ser usadas en las herramientas de detección *phishing* más recientes.

Los métodos de detección de páginas *phishing* basados en similitudes visuales diseñados por Medvet, Kirda y Kruegel (2008), así como por Chen, Dick y Miller (2010) fueron planteados tras notar que las personas asociaban directamente las apariencias de las páginas *phishing* con las páginas legítimas. De este modo, las personas con poco o nulo conocimiento sobre el *phishing* pueden ser víctimas fáciles de robo de datos personales. Por ese motivo, estos métodos fueron diseñados para detectar similitudes visuales y evitar el ingreso de datos personales en páginas web fraudulentas. En el caso de Medvet *et al*. (2008), implementaron un complemento para la herramienta AntiPhish, la cual analiza si una página de ingreso de datos sensibles es segura a través del análisis de información enviada. Esta herramienta puede tener una mala clasificación si el uso de información usada en distintas cuentas es repetido. El implemento realizado por los autores ayuda a evitar la clasificación errónea de esta herramienta, ayudando a verificar si la página sospechosa es idéntica a una página legítima de una empresa o entidad bancaria. Sin embargo, este complemento no se restringe a AntiPhish, sino a toda herramienta que pueda otorgar al complemento una imagen de las páginas legítimas de empresas.

Los autores Sanglerdsinlapachai y Rungsawang (2010) encontraron que, en la primera mitad del 2009, se detectaron más de 55 000 páginas *phishing* activas. Debido a esto, los dos autores detectaron que los dos mayores vectores de ataque son los correos y las páginas *phishing*. Después de decidir enfocarse en las páginas *phishing*, ambos decidieron hacer una herramienta de ensamble de clasificadores, el cual consiste en el ensamblado de distintas técnicas de *machine learning* junto con el uso de aplicaciones heurísticas de CANTINA. Este ensamble de clasificadores utiliza AdaBoost, J48 (árbol de decisión), Naive Bayes, Red Neuronal, Bosque Aleatorio o Máquina de Soporte Vectorial. Estos clasificadores fueron escogidos después de una comparación de clasificación entre distintos algoritmos; los seis mencionados son los que obtuvieron mejor resultado que un método heurístico implementado de la herramienta de detección de *phishing* CANTINA. Al final, el ensamble de clasificadores consistió en AdaBoost, Red Neuronal y Bosque Aleatorio, obteniendo el mejor resultado entre distintos ensamblados realizados con los clasificadores mencionados.

Tras revisar y analizar varios métodos de detección de *phishing*, Bulakh y Gupta (2016) decidieron usar una perspectiva nueva, la de las marcas usadas para el

phishing, para desarrollar un nuevo método de detección de *phishing*. Esta herramienta tiene como propósito permitir a las marcas defenderse de manera proactiva contra estos ataques. Tras analizar las páginas web *phishing*, se determinaron las características que les permiten tener un gran tráfico de usuarios y se desarrolló una solución que consiste en una etapa de prefiltro, seguido de un *whitelist* el cual contiene las páginas web legítimas. Dicho método finaliza con un clasificador de *machine learning* supervisado. Esta herramienta funciona cuando se filtran las páginas con formularios HTML; luego, al verificar si la página sospechosa está dentro de la *whitelist,* el cual es el *whitelist* autoactualizable mencionado anteriormente en esta sección y, en caso no se encuentre ahí, funciona mediante la utilización de un clasificador supervisado para clasificar si esta página es *phishing* o no. El clasificador elegido para esta herramienta es Bosque Aleatorio, después de pasar por una comparación con tres algoritmos de árboles de decisiones (J48, CART y PART), Red Neuronal, Regresión Logística y Naive Bayes.

En la actualidad, las redes sociales son el medio de entretenimiento y comunicación de las personas. Sin embargo, los usuarios promedio tienen poco o nulo conocimiento sobre ciberamenazas, por lo que son víctimas fáciles de las páginas maliciosas que aparentan ser páginas regulares. Estas páginas maliciosas pueden robar datos personales de la misma computadora, inyectar virus, descargar un *software* malicioso o ingresar a una página *phishing*. En las redes sociales, al ingresar a estas páginas a veces se activan códigos que hacen que el usuario comparta automáticamente el mismo enlace sin conocimiento alguno.

Para defender a los usuarios de las redes sociales de estas amenazas, los autores Al-Janabi, Quincey y Andras (2017) han planteado un clasificador de Bosque Aleatorio que analiza todas las publicaciones realizadas en una red social que contengan direcciones URL para el redireccionamiento a páginas fuera de dicha red social. Una vez que se hayan filtrado esas publicaciones, se analizan las características de la publicación, tanto del usuario que publicó el URL como el contenido de la publicación. El algoritmo usado para este trabajo fue Bosque Aleatorio, debido a que obtuvo los mejores resultados tras compararlo con otros algoritmos de clasificación, como Naive Bayes, K Vecinos Más Próximos y Regresión Logística.

Rajab (2018) descubrió que, al momento de ingresar a una página web *phishing*, muchas características pueden ser iguales a las de la página web real, pero otras pueden ser diferentes. Las características similares y diferentes en cada página web *phishing* no son las mismas siempre, lo cual llevó al autor a plantear la técnica del análisis de características. Gracias a ella, es posible determinar un conjunto de atributos de una página web sospechosa de *phishing* y compararlas con el mismo conjunto de características de la página web real. La forma en la que trabaja esta

técnica es analizando la correlación de las características con los atributos objetivos. De esta manera se determinan las características importantes, a través de la observación de una alta correlación con los atributos objetivos y baja correlación con características menos relevantes. Tras determinar estas condiciones, se puede ver el conjunto de características adecuado a comparar.

Las herramientas desarrolladas han ayudado en la detección de *phishing* gracias a su efectividad y capacidad de funcionamiento a largo plazo. Sin embargo, como todo instrumento creado por los humanos, tienen un tiempo de obsolescencia y, en este caso, utilidad. Esto se debe a que con el tiempo los hackers van a encontrar métodos para evadir estas detecciones de *phishing* y estas páginas volverán a ser difíciles de detectar, eventualmente. A pesar de esto, es posible que estas técnicas puedan ser mejoradas, al igual que varias aplicaciones y servicios en la red, y se pueda impedir su obsolescencia. Por lo tanto, uno de los propósitos que persigue este trabajo es ofrecer una visión actualizada sobre la efectividad del aprendizaje automático en la identificación de sitios web de *phishing*.

3. METODOLOGÍA Y EXPERIMENTACIÓN

Esta investigación propone una metodología de comparación objetiva entre los diferentes métodos de *machine learning* aplicados para la detección de *phishing*. El diagrama de bloques correspondiente a las fases de procesamiento se muestra en la figura 1 y en la siguiente subsección se describe en detalle cada fase, mientras que en las siguientes subsecciones se describen los *datasets* usados y la experimentación realizada.

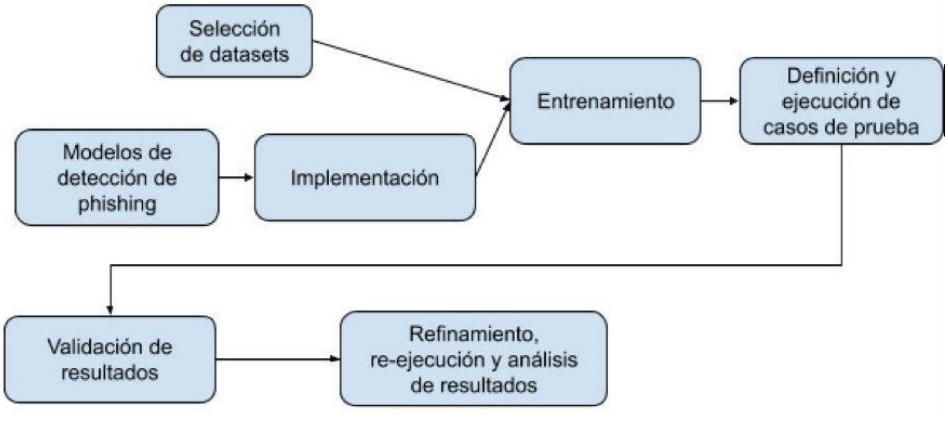

Figura 1. Diagrama de bloques de la metodología

Elaboración propia

a. Metodología

i. Modelos de detección de *phishing*

Se identifican metodologías usadas en los trabajos revisados en esta investigación para poder replicar las técnicas usadas por los autores y realizar una comparación exacta entre los resultados obtenidos en esta experimentación y la realizada por ellos.

ii. Implementación

Una vez que se determinen y se encuentre la manera de replicar las técnicas usadas o creadas por los distintos autores investigados, estas serán implementadas en ambientes de prueba.

iii. Selección de *datasets*

Al mismo tiempo en que se buscará la manera de replicar las técnicas de los autores, se realizará una búsqueda de los *datasets* usados por estos para llevar a cabo una experimentación comparativa.

iv. Entrenamiento

Tras encontrar todos los *datasets* públicos usados por los autores, estos serán divididos en dos conjuntos, de entrenamiento y de prueba, cada uno. En esta fase de la metodología se usan los conjuntos de entrenamiento con cada clasificador.

v. Definición y ejecución de casos de prueba

Después de entrenar los clasificadores con los conjuntos de entrenamiento, se definirá el caso de prueba de cada *dataset* usado en cada prueba. Posteriormente, se usarán los conjuntos de ejecución con cada clasificador implementado para la predicción y clasificación de estos.

vi. Validación de resultados

Tras ejecutar los clasificadores con los *datasets*, se observarán los resultados obtenidos. Si los resultados obtenidos son coherentes con los resultados de los autores, se procederá a su análisis. En caso contrario, se refinará la implementación de la técnica correspondiente y se volverá a ejecutar.

vii. Refinamiento, reejecución y análisis de resultados

En caso los resultados obtenidos no sean consistentes con los vistos en los trabajos investigados, se realizará un refinamiento de las técnicas en cada caso de prueba. En caso los resultados continúen siendo inconsistentes, se concluirá que los obtenidos son los resultados finales y se analizarán junto con los resultados de las demás pruebas y *datasets*. Al momento de analizar los resultados, se comparará la exactitud, precisión, recuperación y valor F obtenidos y se llegará a una conclusión respecto a los clasificadores.

b. *Datasets*

i. ISCX-URL-2016: este es un conjunto de *datasets* creado por Islam *et al*. (2016) que contienen cuatro tipos de URL maliciosos, los cuales son *malware*, *phishing*, *spam* y desfiguración. Cada tipo de URL fue agrupado en cuatro *datasets*, combinando URL maliciosas correspondientes al grupo en el que se encuentran con URL benignas en cada uno. Entre los cuatro *datasets* por cada tipo de URL malicioso, se encuentra el primer *dataset* con más de cuarenta características, que luego de una evaluación fueron reducidos por los mismos autores. Debido al alcance de este trabajo de investigación, solo se utilizarán los *datasets* específicos para *phishing* y los que tengan las características reducidas tras la evaluación de los autores, los cuales contienen alrededor de 10 000 URL *phishing* extraídas de OpenPhish, un repositorio activo de páginas *phishing*, de las cuales se usaron 7586 URL *phishing* y 7781 páginas benignas. Al ser diferentes *datasets*, estos pueden tener características en común o únicas respecto a otros de los *datasets*. Por ejemplo:

1. Phishing_BestFirst: este *dataset* contiene catorce características, entre las cuales se encuentra la característica *class* que representa si dicha instancia es benigna o *phishing*, usando los valores *phishing* y *benign* para esta clasificación. Las características analizadas para determinar dicha clasificación están descritas en la tabla 1.

Tabla 1
Descripción de características del dataset *Phishing_BestFirst*

Característica	Descripción
domain_token_count	Indica el conteo de *tokens* en el dominio.
tld	Indica el conteo de uso de un dominio de alto nivel en la URL.
urlLen	Indica el número de caracteres que tiene la URL.
domainlength	Indica el tamaño del dominio.
fileNameLen	Indica el tamaño del nombre del archivo al que corresponde la URL.
pathurlRatio	Indica la ratio de la ruta dividida por URL.
NumberofDotsinURL	Indica el número de puntos encontrados en la URL.
Query_DigitalCount	Indica el número de dígitos en la parte de consulta de la URL.
LongestPathTokenLength	Indica la longitud del *token* dentro de la ruta más larga en la URL.
delimiter_Domain	Indica el número de delimitadores de dominios que se encuentran en la URL.

(*continúa*)

(*continuación*)

delimiter_path	Indica el número de delimitadores de rutas encontrados en la URL.
SymbolCount_Domain	Indica el conteo de símbolos encontrados en el dominio.
Entropy_Domain	Indica el porcentaje de entropía encontrada en la URL.
class	Indica si la instancia es benigna o *phishing*.

Elaboración propia

2. Phishing_Infogain: estos dos *datasets* son similares entre sí, con la única diferencia de que este tiene una característica más que la última mencionada. Dicha característica diferente es avgdomaintokenlen. Las características se explican con mayor detalle en la tabla 2.

Tabla 2
Descripción de características del dataset *Phishing_Infogain*

Característica	Descripción
domain_token_count	Indica el conteo de *tokens* en el dominio.
avgdomaintokenlen	Indica la longitud promedio de los *tokens* de ruta dentro de la URL.
tld	Indica el conteo de uso de un dominio de alto nivel en la URL.
urlLen	Indica el número de caracteres que tiene la URL.
domainlength	Indica el tamaño del dominio.
fileNameLen	Indica el tamaño del nombre del archivo al que corresponde el URL.
pathurlRatio	Indica la ratio de la ruta dividida por URL.
NumberofDotsinURL	Indica el número de puntos encontrados en el URL.
Query_DigitalCount	Indica el número de dígitos en la parte de consulta del URL.
LongestPathTokenLength	Indica la longitud del *token* dentro de la ruta más larga en la URL.
delimiter_Domain	Indica el número de delimitadores de dominios que se encuentran en la URL.
delimiter_path	Indica el número de delimitadores de rutas encontrados en la URL.
SymbolCount_Domain	Indica el conteo de símbolos encontrados en el dominio.
Entropy_Domain	Indica el porcentaje de entropía encontrada en la URL.
Class	Indica si la instancia es benigna o *phishing*.

Elaboración propia

ii. Phishing Dataset for Machine Learning: *Feature Evaluation* (Phishing_Legitimate_Full): Este *dataset* contiene 48 características extraídas de cinco mil páginas *phishing* y cinco mil páginas legítimas del 2015 y 2017. El archivo está en formato .arff. Este *dataset* fue creado por Tan (2019), uno de los autores del trabajo investigado por Chiew, Tan, Wong, Yong y Tiong (2019) y tiene un gran número de características para evaluar. Las características del *dataset* se encuentran descritas en la tabla 3.

Tabla 3
Descripción de características del dataset *Phishing_Legitimate_Ful*

Característica	Descripción
NumDots	Cuenta el número de puntos en la URL.
SubdomainLevel	Cuenta el nivel de subdominio de la URL.
PathLevel	Cuenta la profundidad de la ruta de la URL.
UrlLength	Cuenta el número de caracteres que contiene la URL.
NumDash	Cuenta el número de "-" en la URL.
NumDashInHostname	Cuenta el número de "-" en la parte del *hostname* de la URL.
AtSymbol	Verifica si el símbolo "@" existe en la URL.
TildeSymbol	Verifica si el símbolo "~" existe en la URL.
NumUnderscore	Cuenta el número de "_" en la URL.
NumPercent	Cuenta el número de "%" en la URL.
NumQueryComponents	Cuenta el número de partes de consulta en la URL.
NumAmpersand	Cuenta el número de "&" en la URL.
NumHash	Cuenta el número de "#" en la URL.
NumNumericChars	Cuenta el número de caracteres numéricos en la URL.
NoHttps	Indica si hay HTTPS en la URL.
RandomString	Indica si hay un valor de cadena aleatorio en la URL.
IpAddress	Indica si la dirección IP es usada en la parte del *hostname* de la URL.
DomainInSubdomains	Indica si TLD o ccTLD es usado como parte del subdominio en la URL.
DomainInPaths	Indica si TLD o ccTLD es usado en la ruta de la URL.
HttpsInHostname	Indica si HTTPS es ofuscado en la parte de *hostname* de la URL.
HostnameLength	Cuenta el número de caracteres en la parte de *hostname* de la URL.
PathLength	Cuenta el número de caracteres en la ruta de la URL.
QueryLength	Cuenta el total de caracteres en la parte de consulta de la URL.
DoubleSlashInPath	Verifica si "//" existe en la ruta de la URL.
NumSensitiveWords	Cuenta el número de palabras sensibles en la URL.
EmbeddedBrandName	Indica si el nombre de la marca aparece en subdominios y la ruta de la URL siendo el nombre de la marca el nombre de dominio más frecuente en el contenido HTML.
PctExtHyperlinks	Cuenta el porcentaje de hipervínculos externos en el código fuente HTML.
PctExtResourceUrls	Cuenta el porcentaje de URL de recursos externos en el código fuente HTML.
ExtFavicon	Verifica si el favicon está siendo cargado desde un nombre de dominio que es diferente del nombre de dominio de la URL.
InsecureForms	Verifica si el atributo de formulario de acción contiene una URL sin protocolo HTTPS.

(*continúa*)

(*continuación*)

RelativeFormAction	Verifica si el atributo de formulario de acción contiene una URL relativa.
ExtFormAction	Verifica si el atributo de formulario de acción contiene una URL de un dominio externo.
AbnormalFormAction	Verifica si el atributo de formulario de acción contiene un "#" o "about:blank" o una cadena vacía o "javascript:true".
PctNullSelfRedirectHyperlinks	Cuenta el porcentaje de campos de hipervínculos que contienen valor vacío o un valor de autorredireccionamiento o un valor anormal.
FrequentDomainNameMismatch	Verifica si el nombre de dominio más frecuente en el código fuente HTML no concuerda con el nombre de dominio de la URL.
FakeLinkInStatusBar	Verifica si el código fuente HTML contiene un comando JavaScript onMouseOver para mostrar una URL falsa en la barra de estado.
RightClickDisabled	Verifica si el código fuente HTML contiene comando JavaScript para deshabilitar funciones de clic derecho.
PopUpWindow	Verifica si el código fuente HTML contiene comando JavaScript para lanzar ventanas emergentes.
SubmitInfoToEmail	Verifica si el código fuente HTML contiene la función HTML "mailto".
IframeOrFrame	Verifica si se usa iframe o frame en el código fuente HTML.
MissingTittle	Verifica si la etiqueta de título en el código fuente HTML está vacía.
ImagesOnlyInForm	Verifica si el alcance del formulario en el código fuente HTML contiene únicamente imágenes.
SubdomainLevelRT	Cuenta el número de puntos en la parte del *hostname* del URL.
UrlLengthRT	Cuenta el total de caracteres en el URL de la página y utiliza reglas y límites para generar el valor.
PctExtResourceUrlsRT	Cuenta el porcentaje de las URL de recursos externos en el código fuente HTML de la página.
AbnormalExtFormActionR	Verifica si el atributo de formulario de acción contiene un dominio extranjero o "about:blank" o se encuentra vacío.
ExtMetaScriptLinkRT	Cuenta el porcentaje de metaetiquetas, *scripts* y etiquetas de enlace que contienen URL externos en los atributos.
PctExtNullSelfRedirectHyperlinksRT	Calcula el porcentaje de hipervínculos en el código fuente HTML que usen un nombre de dominio diferente, empiecen con "#" o usen "JavaScript :void(0)".
CLASS_LABEL	Indica si la instancia es benigna o *phishing*.

Elaboración propia

iii. Website Phishing Data Set (PhishingData): Este *dataset* fue usado por Cuzzocrea, Martinelli y Mercaldo (2018) para comparar algoritmos de *machine learning* basados en el árbol de decisiones (por ejemplo, Árbol de Decisión y Bosque Aleatorio). Debido a que los autores obtuvieron métricas únicamente de precisión, recuperación y valor F, estos son los valores por comparar en la

sección de resultados y discusiones. Este *dataset* presenta valores 1, 0 y -1, los cuales indican si las características presentan indicios de que la página es legítima, sospechosa o *phishing,* respectivamente. Las características de este *dataset* son descritas en la tabla 4.

Tabla 4
Descripción de características del dataset *PhishingDat*

Característica	Descripción
SFH	Verifica si la información ingresada en la página se transfiere a un servidor o no, o si se transfiere a un servidor de un dominio diferente.
popUpWindow	Verifica si la página usa ventanas emergentes para redirigir a los usuarios a esta página.
SSLfinal_State	Verifica si el protocolo HTTPS de la página es de confianza o si es falso.
Request_URL	Verifica si la mayoría de los objetos en la página web son cargados desde un dominio diferente al de la URL o no.
URL_of_Anchor	Verifica si los enlaces dentro de la página web apuntan a un dominio diferente al de la URL o no.
web_traffic	Verifica si el tráfico web de la página es el de una página legítima, un *phishing* o si es sospechoso.
URL_Length	Verifica si la longitud de la URL es considerada *phishing*, sospechosa o legítima.
age_of_domain	Verifica si el tiempo de vida del dominio es de una página *phishing* o una legítima.
having_IP_Address	Verifica si la URL tiene la dirección IP en ella o no.
Result	Indica si la página es *phishing*, es sospechosa de *phishing* o si es legítima.

Elaboración propia

c. Experimentación

Siguiendo los pasos mencionados en la metodología, se ejecutaron los *datasets* mencionados anteriormente en el ambiente Spyder con las técnicas de clasificación mencionadas en los antecedentes. En los trabajos investigados se abordaron dos métodos de ejecución, el primero fue división por porcentaje (DP), el cual es la división del *dataset* en un conjunto de entrenamiento para los clasificadores y un conjunto de ejecución. El segundo es *cross-validation* de 10 pliegues (CV). La DP de los tres primeros *datasets* será explicada en la sección de resultados para cada uno. En el caso del CV, el último *dataset* fue dividido en diez grupos, usando nueve para el entrenamiento y el restante para la ejecución; luego, se repitió este proceso para que cada grupo haya sido usado para la ejecución y se obtengan resultados con todos estos.

En la fase de refinamiento, ejecución y análisis de resultados de la metodología, se realizó un refinamiento de los hiperparámentros usando *Grid Search* para la automatización de este proceso. Debido a que las ejecuciones en cada *dataset* se realizan de manera diferente y con un número de datos diferentes, cada *dataset* recibió un refinamiento único.

Los hiperparámetros de cada clasificador para cada *dataset* son indicados en las tablas 5, 6, 7, 8 y 9 en el orden en el que los clasificadores fueron mencionados anteriormente en la sección de antecedentes. Los hiperparámetros listados son los usados en el ambiente Spyder. En los casos de los clasificadores denominados Redes Neuronales y Máquina de Soporte Vectorial, debido al largo tiempo de entrenamiento y ejecución al usar *Grid Search*, se usaron los hiperparámetros por defecto de estos. Los resultados de esta experimentación se ven en la siguiente sección.

Tabla 5

Valores de los hiperparámetros modificados del clasificador Árbol de Decisión para cada caso de prueba

Hiperparámetros	Phishing_ BestFirst	Phishing_Infogain	Phishing_ Legitimate_Full	PhishingData
criterion	gini	entropy	entropy	entropy
splitter	best	best	best	best
max_depth	None	None	None	None
min_samples_split	2	2	2	2
min_samples_leaf	1	1	1	1
min_weight_fraction_leaf	0.0	0.0	0.0	0.0
max_features	None	None	None	None
random_state	None	None	None	None
max_leaf_nodes	None	None	None	None
min_impurity_decrease	0.0	0.0	0.0	0.0
class_weight	None	None	None	None
ccp_alpha	0.0	0.0	0.0	0.0

Elaboración propia

Tabla 6
Valores de los hiperparámetros modificados del clasificador Regresión Logística para cada caso de prueba

Hiperparámetros	Phishing_ BestFirst	Phishing_Infogain	Phishing_ Legitimate_Full	PhishingData
penalty	none	none	none	l1
dual	False	False	False	False
tol	0.0001	0.0001	0.0001	0.0001
C	1.0	1.0	1.0	1.0
fit_interceptor	True	True	True	True
intercept_scaling	1	1	1	1
class_weight	None	None	None	None
random_state	None	None	None	None
solver	newton-cg	newton-cg	newton-cg	liblinear
max_iter	10000	1627	1627	1627
multi_class	auto	auto	auto	auto
verbose	0	0	0	0
warm_start	False	False	False	False
n_jobs	None	None	None	None
l1_ratio	None	None	None	None

Elaboración propia

Tabla 7
Valores de los hiperparámetros modificados del clasificador Red Neuronal para cada caso de prueba

Hiperparámetros	Phishing_ BestFirst	Phishing _Infogain	Phishing_ Legitimate_Full	Phishing Data
hidden_layer_sizes	(100,)	(100,)	(100,)	(100,)
activation	relu	relu	relu	relu
solver	adam	adam	adam	adam
alpha	0.0001	0.0001	0.0001	0.0001
batch_size	auto	auto	auto	auto
learning_rate	constant	constant	constant	constant
learning_rate_init	0.0001	0.0001	0.0001	0.0001
power_t	0.5	0.5	0.5	0.5
max_iter	200	400	400	1000
shuffle	True	True	True	True
random_state	None	None	None	None

(*continúa*)

(*continuación*)

tol	0.0001	0.0001	0.0001	0.0001
verbose	False	False	False	False
warm_start	False	False	False	False
momentum	0.9	0.9	0.9	0.9
nesterovs_ momentum	True	True	True	True
early_stopping	False	False	False	False
validation_fraction	0.1	0.1	0.1	0.1
beta_1	0.9	0.9	0.9	0.9
beta_2	0.999	0.999	0.999	0.999
épsilon	1e-08	1e-08	1e-08	1e-08
n_iter_no_change	10	10	10	10
max_fun	15000	15000	15000	15000

Elaboración propia

Tabla 8

Valores de los hiperparámetros modificados del clasificador Máquina de Soporte Vectorial para cada caso de prueba

Hiperparámetros	Phishing_ BestFirst	Phishing_Infogain	Phishing_ Legitimate_Full	PhishingData
C	1.0	1.0	1.0	1.0
kernel	rbf	rbf	rbf	rbf
degree	3	3	3	3
gamma	auto	auto	auto	auto
coef0	0.0	0.0	0.0	0.0
shrinking	True	True	True	True
probability	False	False	False	False
tol	0.001	0.001	0.001	0.001
cache_size	200	200	200	200
class_weight	None	None	None	None
verbose	False	False	False	False
max_iter	-1	-1	-1	-1
decisión_function_shape	ovr	ovr	ovr	ovr
break_ties	False	False	False	False
random_state	None	None	None	None

Elaboración propia

Tabla 9
Valores de los hiperparámetros modificados del clasificador Bosque Aleatorio para cada caso de prueba

Hiperparámetros	Phishing_ BestFirst	Phishing_Infogain	Phishing_ Legitimate_Full	PhishingData
n_estimators	100	100	100	100
criterion	gini	entropy	gini	entropy
max_depth	None	None	None	None
min_samples_split	2	2	2	2
min_samples_leaf	1	1	1	1
min_weight_fraction_leaf	0.0	0.0	0.0	0.0
max_features	auto	auto	auto	auto
max_leaf_nodes	None	None	None	None
min_impurity_decrease	0.0	0.0	0.0	0.0
bootstrap	False	False	False	True
oob_score	False	False	False	False
n_jobs	None	None	None	None
random_state	Nonce	Nonce	Nonce	Nonce
verbose	0	0	0	0
warm_start	False	False	False	False
class_weight	balanced	balanced	balanced	balanced
ccp_alpha	0.0	0.0	0.0	0.0
max_samples	None	None	None	None

Elaboración propia

4. RESULTADOS Y DISCUSIONES

a. Resultados

Tras realizar la experimentación de la sección anterior, se obtuvieron los resultados vistos en las siguientes subsecciones. En cada subsección se menciona a los autores que usaron el *dataset* correspondiente. Los trabajos realizados por estos autores son mencionados como "Trabajos anteriores" en las tablas de comparación correspondientes.

Algo que se puede notar, en casi todos los casos, es que los resultados obtenidos por el clasificador Bosque Aleatorio son superiores a los demás, siendo Árbol de Decisión el siguiente clasificador con mejor resultado. El único caso en el que no ocurre esto es en el del *dataset* PhishingData que usa el método de entrenamiento CV de 10 *folds*, siendo Redes Neuronales el que obtiene los mejores resultados y Bosque Aleatorio los segundos mejores. Con base en esto podemos

confirmar lo mencionado por Chiew *et al.* (2019) acerca de Bosque Aleatorio, que es el clasificador más adecuado en comparación con los otros analizados.

i. Phishing_BestFirst

En el trabajo de detección de URL maliciosas realizado por Islam, Rathore, Lashkari, Stakhanova, y Ghorbani (2016), se creó este *dataset* junto con Phishing_Infogain. Al realizar la prueba de este *dataset*, se usó un 80 % de este para el entrenamiento y el 20 % restante para la ejecución de los clasificadores usados en su experimentación. En los resultados obtenidos, los cuales aparecen en la tabla 10, se puede observar que Bosque Aleatorio obtuvo un mejor rendimiento que los demás clasificadores con porcentajes de entre 97,69 % y 98,47 %, siendo seguido por Árbol de Decisión con una diferencia menor a 1% en cada métrica. También se observa que Regresión Logística es el clasificador menos eficiente de los cinco comparadores, con porcentajes de rendimiento entre 91,80 % y 91,81 %.

En la tabla 11 se observa una comparación entre los resultados obtenidos en este trabajo y los que obtuvieron Islam *et al*. (2016) al momento de realizar su experimentación. Ellos solo usaron dos de los clasificadores que se usaron en este trabajo, por lo que esos dos son los únicos que se podrán comparar. Se puede observar que Bosque Aleatorio es mejor que Árbol de Decisión en ambos trabajos. Sin embargo, se puede notar que los resultados anteriores de Bosque Aleatorio pueden llegar a ser mejores que los obtenidos en este trabajo, mientras que en Árbol de Decisión ocurre lo contrario. No obstante, un punto a tomar en cuenta es que los resultados anteriores no presentan porcentaje con decimales, lo cual impide conocer si los resultados fueron redondeados o no. En conclusión, Bosque Aleatorio clasifica de manera más eficiente que Árbol de Decisión en esta prueba de características con valores exactos.

Tabla 10

Resultados obtenidos para cada clasificador (porcentajes) (Phishing_BestFirst)

	Exactitud	Precisión	Recuperación	Valor F
Árbol de Decisión	97,20	97,20	97,20	97,20
Regresión Logística	91,80	91,81	91,80	91,80
Redes Neuronales	96,16	96,98	95,19	96,08
Máquina de Soporte Vectorial	96,19	95,64	96,71	96,17
Bosque Aleatorio	98,11	98,47	97,69	98,08

Elaboración propia

Tabla 11
Comparación de resultados anteriores y obtenidos (porcetantajes) (Phishing_BestFirst)

	Precisión		Recuperación	
	Islam *et al.* (2016)	Propuesta DP	Islam *et al.* (2016)	Propuesta DP
Árbol de Decisión	97	97,20	97	97,20
Bosque Aleatorio	99	98,47	99	97,69

Elaboración propia

ii. Phishing_Infogain

Este *dataset*, al igual que el anterior, fue creado por Islam *et al.* (2016). Por ese motivo, se utilizó el mismo método con el *dataset* anterior, se usó el 80 % del *dataset* para entrenamiento y el 20 % restante para la ejecución. Los resultados obtenidos se encuentran en la tabla 12. En este caso, se observa que los resultados obtenidos son similares a los resultados del *dataset* anterior, encontrándose una ligera diferencia entre ellos. En conclusión, el clasificador con el mejor rendimiento en los *datasets* creados por Islam *et al.* (2016) es Bosque Aleatorio, un resultado que se observó de igual manera en la investigación de estos autores.

En la tabla 13 se encuentra la comparación de los resultados anteriores, los mismos vistos en la tabla 11 y los resultados recientemente obtenidos. Lo observable aquí es que los resultados anteriores son mejores que los obtenidos en este trabajo. Esto da a concluir que este *dataset*, aunque tenga las mismas características y pueda ser usado para clasificación, no es el indicado para realizar una predicción formal y certera. Otra conclusión es que el mejor clasificador entre los dos vistos en esta tabla es Bosque Aleatorio, al igual que en el trabajo de Islam *et al.* (2016).

Tabla 12
Resultados obtenidos para cada clasificador (porcentajes) (Phishing_Infogain)

	Exactitud	Precisión	Recuperación	Valor F
Árbol de Decisión	97,33	97,33	97,20	97,20
Regresión Logística	91,44	91,52	91,44	91,44
Redes Neuronales	95,16	96,89	94,16	95,51
Máquina de Soporte Vectorial	95,74	96,59	94,75	95,66
Bosque Aleatorio	98,11	98,48	97,70	98,09

Elaboración propia

Tabla 13
Comparación de resultados anteriores y obtenidos (porcentajes) (Phishing_Infogain)

	Precisión		Recuperación	
	Islam *et al.* (2016)	Propuesta DP	Islam *et al.* (2016)	Propuesta DP
Árbol de Decisión	97	97,33	97	97,33
Bosque Aleatorio	99	98,48	99	97,70

Elaboración propia

iii. Phishing_Legitimate_Full

En el trabajo de Chiew *et al.* (2019), donde fue creado y usado este *dataset*, se realizó la experimentación dividiendo el *dataset* en diez particiones y realizando la prueba con cada una de ellas; luego, se promedió la exactitud obtenida por cada partición. Para cada partición se usó el 70 % del *dataset* para entrenamiento y el otro 30 % para la ejecución. En este caso se realizó la división por porcentaje de igual manera que en el trabajo realizado, pero usando el *dataset* completo, en lugar de dividirlo en 10 partes.

Como se puede ver en la tabla 14, Redes Neuronales obtuvo un mejor rendimiento que con los *datasets* anteriores, logrando obtener un porcentaje de rendimiento cercano a Árbol de Decisión, superándolo ligeramente en Recuperación. Además de eso, se ve que Regresión Logística tuvo un mejor rendimiento de igual manera, logrando que Máquina de Soporte Vectorial quede como el clasificador menos eficiente para este caso de prueba. Pero, al igual que en los *datasets* anteriores, Bosque Aleatorio obtuvo el mejor rendimiento, siento el más eficiente entre los cincos clasificadores.

En la tabla 15 vemos una comparación entre los resultados anteriores de las pruebas realizadas con este *dataset* y los resultados obtenidos actualmente con una DP similar a la realizada en el trabajo de Chiew *et al*. (2019). En el trabajo anterior solo se evaluó la exactitud, por lo que será lo único a comparar en este caso de prueba. Se puede ver que, al igual que los casos anteriores, Bosque Aleatorio obtuvo el mejor rendimiento, seguido de Árbol de Decisión y, al final, Máquina de Soporte Vectorial. Sin embargo, a pesar de haber realizado el mismo experimento, los resultados anteriores y los obtenidos en este trabajo son diferentes. Eso lleva a suponer que los resultados obtenidos tras dividir el *dataset* en 10 partes en el trabajo anterior causaron que los resultados en cada uno no fueran tan buenos como una experimentación con el *dataset* entero.

En conclusión, es posible que utilizar el *dataset* entero y realizar un entrenamiento con este pueda aumentar las probabilidades de que se pueda

clasificar con mayor exactitud; asimismo, que usar Bosque Aleatorio es la mejor opción vista tanto por otros autores como en este trabajo.

Tabla 14
Resultados obtenidos para cada clasificador (porcentajes) (Phishing_Legitimate_Full)

	Exactitud	Precisión	Recuperación	Valor F
Árbol de Decisión	97,60	97,60	97,60	97,60
Regresión Logística	95,60	95,61	95,60	95,60
Redes Neuronales	97,45	96,71	98,18	97,44
Máquina de Soporte Vectorial	92,10	91,75	92,31	92,03
Bosque Aleatorio	98,80	98,79	98,79	98,79

Elaboración propia

Tabla 15
Comparación de resultados anteriores y obtenidos (porcentajes) (Phishing_Legitimate_Full)

	Exactitud	
	Chiew *et al.* (2019) (DP)	Propuesta DP
Árbol de Decisión	94,37	97,60
Máquina de Soporte Vectorial	92,20	92,10
Bosque Aleatorio	96,17	98,80

Elaboración propia

iv. PhishingData

En la tabla 16 se puede ver la comparación de los resultados obtenidos por este *dataset* usado por Cuzzocrea *et al.* (2018). Estos autores decidieron usar únicamente CV de 10 pliegues, por lo que, a diferencias de los anteriores casos de prueba, se están comparando estos resultados. Al observar bien la tabla, se puede deducir que este *dataset* no pude ser clasificado como los anteriores por los clasificadores, lo cual llevó a que los resultados en este caso de prueba fueran menores a los vistos en las tablas anteriores. En este caso, el clasificador con mejor resultado fue Redes Neuronales, por lo que se puede concluir que en un caso de prueba distinto a los anteriores se puede conseguir un clasificador diferente a Bosque Aleatorio como el de mejor rendimiento.

Algo similar se observa en la tabla 17, que es la comparación de los resultados anteriores y los resultados recientemente obtenidos de los clasificadores en común. En este caso vemos cómo los resultados de Árbol de Decisión del trabajo anterior son mayores a los de Bosque Aleatorio. Sin embargo, los resultados de este trabajo tienen un patrón diferente. Otra observación es que los resultados del trabajo anterior son mejores que los obtenidos en esta experimentación, lo cual indica que hace falta un mejor refinamiento al momento de hacer pruebas con este *dataset*. Se concluye además que los clasificadores pueden tener un mejor rendimiento dependiendo del caso de prueba, como se ve en los resultados de este *dataset*.

Tabla 16
Resultados obtenidos para cada clasificador (porcentajes) (PhishingData)

	Exactitud	Precisión	Recuperación	Valor F
Árbol de Decisión	87,08	87,36	87,10	87,52
Regresión Logística	81,92	75,39	81,92	78,44
Redes Neuronales	89,22	90,03	90,25	89,52
Máquina de Soporte Vectorial	86,26	85,66	86,26	84,44
Bosque Aleatorio	88,56	88,60	88,56	88,56

Elaboración propia

Tabla 17
Comparación de resultados anteriores y resultados obtenidos (porcentajes) (PhishingData)

	Precisión		Recuperación		Valor F	
	Cuzzocrea *et al.* (2018) (CV)	Propuesta (CV)	Cuzzocrea *et al.* (2018) (CV)	Propuesta (CV)	Cuzzocrea *et al.* (2018) (CV)	Propuesta (CV)
Árbol de Decisión	91,35	87,36	90,40	87,10	90,85	87,52
Bosque Aleatorio	90,40	88,60	90,20	88,56	90,30	88,56

Elaboración propia

b. Discusión

Como se observa en las tablas de la subsección anterior, desde la tabla 10 a la tabla 17, los experimentos llegan a tener resultados diferentes a los resultados vistos en los trabajos realizados por Islam *et al.* (2016), Chiew *et al.* (2019) y Cuzzocrea *et al.* (2018). Estas diferencias son atribuibles al escaso nivel de detalle aportado en la implementación de ciertos métodos de aprendizaje

automático. Debido a esto, la experimentación con cada *dataset* fue optimizada hasta obtener los mejores resultados posibles, Adicionalmente, estos autores solo usaron algunos de los clasificadores utilizados en la experimentación, los cuales son Árbol de Decisión, Bosque Aleatorio y, en el caso del *dataset* Phishing_Legitimate_Full, Máquina de Soporte Vectorial, por lo que la comparación entre los resultados de estos autores y de este trabajo se limita a estos métodos en común. No obstante, la inclusión de resultados con clasificadores adicionales contribuye a un mejor análisis de los resultados. En los casos de Bosque Aleatorio en los *datasets* usados en los trabajos de Islam *et al*. (2016) y Cuzzocrea *et al*. (2018), los resultados de dichas investigaciones fueron mejores que los obtenidos en este trabajo, por lo cual se presume que los clasificadores pueden ser refinados a mayor detalle, hipótesis que será contrastada en futuras investigaciones. En otros casos, como los de los resultados del *dataset* Phishing_Legitimate_Full del trabajo de Chiew *et al*. (2019) y los casos de Árbol de Decisión de los *datasets* del trabajo de Islam *et al*. (2016), los resultados de esta experimentación fueron mejores en la mayoría de los criterios comparados.

En los *datasets* Phishing_BestFirst y Phishing_Infogain, los resultados obtenidos en el trabajo de Islam *et al*. (2016) fueron redondeados, por lo que solo se muestran valores porcentuales enteros, siendo el 97 % para la precisión y recuperación del Árbol de Decisión y 99 % para Bosque Aleatorio. Se observa que los resultados obtenidos en Árbol de Decisión en la experimentación son mayores a 97 %, lo cual supone una mejora en estos resultados, mientras que en Bosque Aleatorio se observa un resultado menor a lo requerido para ser redondeado a 99 %, lo que indica que un mejor refinamiento de los hiperparámetros es capaz de dar mejores resultados que igualan o superan los obtenidos en el trabajo de Islam *et al*. (2016). Sin embargo, al analizar las tablas 10 y 11, podemos determinar que los clasificadores de árboles, Árbol de Decisión y Bosque Aleatorio son los más efectivos para este *dataset*, siendo el que da un mejor rendimiento entre estos dos Bosque Aleatorio. Este patrón también se observa en los resultados del trabajo de Islam *et al*. (2016), donde Bosque Aleatorio es el clasificador con mejor rendimiento, seguido por Árbol de Decisión.

En el *dataset* Phishing_Legitimate_Full, solo se realizó una comparación de la exactitud, debido a que fue la única métrica obtenida del trabajo de Chiew *et al*. (2019). En este caso, gracias a que el clasificador Máquina de Soporte Vectorial fue usado tanto en dicho estudio como en este trabajo, se puede hacer una comparación más amplia y que no engloba únicamente a clasificadores de árboles. En la tabla 15 se puede ver con claridad que los resultados obtenidos han sido mejores que los obtenidos en el trabajo de Chiew *et al*. (2019), a excepción de Máquina de Soporte Vectorial, lo que significa que en este

trabajo la configuración de los clasificadores ha sido mejor refinada o que el uso del *dataset* entero para el entrenamiento y ejecución de los clasificadores dio mejores resultados que dividiendo el *dataset* en diez partes, ejecutando los clasificadores con cada una de ellas y promediando los resultados.

Viendo primero el caso de Máquina de Soporte Vectorial, la diferencia entre los resultados de la parametrización propuesta y el trabajo investigado es de 0,1 %, lo cual no indica una baja de rendimiento significativa. En el caso de Árbol de Decisión, la diferencia entre el resultado obtenido en la experimentación y el resultado del trabajo investigado es de 3,23 %, teniendo un mejor resultado que el visto en el trabajo de Chiew *et al*. (2019). En el caso de Bosque Aleatorio, la diferencia de resultados respecto a este trabajo es de 2,63 %, siendo mejor el resultado obtenido en esta experimentación. Analizando las dos tablas se puede determinar que Bosque Aleatorio es el clasificador con mejor rendimiento del grupo para este *dataset*, al igual que para los anteriores.

Finalmente, en el caso del *dataset* PhishingData, ocurre un conjunto de hechos diferentes a los vistos en los *datasets* anteriores. En este caso se comparan las métricas de precisión, recuperación y valor F de los clasificadores Árbol de Decisión y Bosque Aleatorio, además de comparar resultados de ejecución de CV. Los resultados obtenidos en este trabajo son muy diferentes a los observados en el trabajo de Cuzzocrea *et al*., (2018). En este caso de prueba los resultados de las precisiones, las recuperaciones y los valores F obtenidos son menores que los vistos en el trabajo de estos autores. Además, en este caso en particular, se ve que el clasificador con mejor rendimiento es Redes Neuronales, un patrón diferente al visto en el trabajo de Cuzzocrea *et al*. (2018).

Tras revisar todos los resultados de la experimentación y de la comparación entre estos y los resultados de trabajos pasados, podemos concluir que los clasificadores de árboles de decisión siempre han dado los mejores resultados, siendo los más eficientes para la clasificación de páginas *phishing*. De entre los dos clasificadores de árboles de decisión, Árbol de Decisión y Bosque Aleatorio, se observa que Bosque Aleatorio obtiene mejores resultados en la mayoría de los *datasets*, a excepción del último, donde Redes Neuronales logra un mejor rendimiento. Pero, a pesar de eso, Bosque Aleatorio logró ser más eficiente que Árbol de Decisión. En los trabajos de Islam *et al*. (2016) y Chiew *et al*. (2019) se puede notar que el patrón de resultados es el mismo, a pesar de presentar indicadores de precisión diferentes. En estos casos, Bosque Aleatorio da mejores resultados que Árbol de Decisión. En el caso del *dataset* Phishing_Legitimate_Full, el clasificador de Máquina de Soporte Vectorial dio resultados menores a los otros dos clasificadores mencionados, de la misma manera que en este

trabajo. El único caso donde el patrón de resultados obtenidos en este trabajo no concuerda con el patrón del trabajo anterior es en el trabajo de Cuzzocrea *et al*, (2018), donde vemos que Redes Neuronales logró un mejor rendimiento, seguido por Bosque Aleatorio y posteriormente Árbol de Decisión, un patrón totalmente diferente al del trabajo de estos autores. Con esto podemos determinar que Bosque Aleatorio es el mejor clasificador para la detección de *phishing* y Árbol de Decisión es una segunda opción muy cercana.

5. CONCLUSIONES

El *phishing* es una de las ciberamenazas más grandes actualmente, debido al alto número de ataques reportados en Latinoamérica en los últimos años y la cantidad creciente de robos y estafas realizadas como consecuencia de estos ataques cada año. Debido a este problema, se han ideado varias herramientas capaces de detectar páginas *phishing*, que evitan el ingreso a estas y, por ende, el robo de datos personales. Sin embargo, estas herramientas deben ser actualizadas debido a que las páginas *phishing* tienen un tiempo de vida corto para evitar su detección y porque las técnicas de generación de estas páginas evolucionan para evadir las herramientas de detección. Por dicho motivo, se recurrió a técnicas de *machine learning* para detectar páginas *phishing* de manera efectiva. Los clasificadores de *machine learning* tienen un rendimiento variado debido a la ejecución que tienen y a las características de las cuales aprenden y luego comparan para determinar si una instancia es etiquetada como *phishing* o legítima. Tras investigar en distintos trabajos acerca de los clasificadores de *machine learning* más efectivos aplicados a la detección de *phishing*, se identificaron cinco principales: Árbol de Decisión, Regresión Logística, Redes Neuronales, Máquina de Soporte Vectorial y Bosque Aleatorio, Tras definir una metodología que permita hacer una comparación detallada y exhaustiva, se realizó una experimentación extendida. Los resultados experimentales fueron comparados entre sí y con los resultados obtenidos en investigaciones similares cuyos métodos de aprendizaje automático fueron aplicados a los mismos *datasets* analizados en este trabajo. Como resultado de este proceso, se concluye que los clasificadores de árboles son los más eficientes para la detección de Phishing, entre los cuales están Árbol de Decisión y Bosque Aleatorio. Los resultados entre estos dos clasificadores tienen diferencias ligeras, siendo Bosque Aleatorio el de mejor rendimiento en la mayoría de los casos, tanto en trabajos investigados como en esta experimentación. Árbol de Decisión, que es el clasificador de mejor rendimiento en el último *dataset* visto, es una segunda opción muy cercana y factible al momento de realizar una herramienta de detección de *phishing* usando *machine learning*. En investigaciones futuras se buscará extender el análisis a nuevas variantes de páginas *phishing*, realizar un análisis más profundo de las mejores estrategias de calibrado de parámetros, y ampliar la muestra de *datasets* a analizar. Se buscará también contemplar la detección frente a métodos de evasión.

REFERENCIAS

Abdelhamid, N., Thabtah, F. y Abdel-jaber, H. (2017). Phishing Detection: A Recent Intelligent machine learning Comparison based on Models Content and Features. *IEEE Explorer*, *6*. doi:10.1109/ISI.2017.8004877

Abu-Nimeh, S., Nappa, D., Wang, X. y Nair, S. (2007). A Comparison of machine learning Techniques for Phishing Detection. *ACM Digital Library*, *10*. doi:10,.1145/1299015.1299021

Al-Janabi, M., De Quincey, E. y Andras, P. (2017). Using Supervised Machine Learning Algorithms to Detect Suspicious URLs in Online Social Networks. *ACM Digital Library*, *8*. doi:10.1145/3110025.3116201

Bulakh, V. y Gupta, M. (2016). Countering Phishing from Brands' Vantage Point. *ACM Digital Library*, *8*. doi:10.1145/2875475,2875478

Campo, D. (20 de noviembre de 2017). MachineLearningPhishing. *GitHub*. Recuperado de https://github.com/diegoocampoh/MachineLearningPhishing

Chen, T.-C., Dick, S. y Miller, J. (2010). Detecting Visually Similar Web Pages: Application to Phishing Detection. *ACM Digital Library*, *38*. doi:10.1145/3282373.3282422

Chiew, K. L., Tan, C. L., Wong, K. S., Yong, K. S. y Tiong, W. K. (2019). A New Hybrid Ensemble Feature Selection Framework for Machine Learning-Based Phishing Detection System. *Science Direct*, *14*. doi:10.1016/j.ins2019.01.064

Cuzzocrea, A., Martinelli, F., y Mercaldo, F. (2018). Applying Machine Learning Techniques to Detect and Analyze Web Phishing Attacks. *ACM Digital Library*, *5*. doi:10,1145/3282373,3282422

ESET Security Report Latinoamérica 2017. (2017). Recuperado de https://www.welivesecurity.com/wp-content/uploads/2017/04/eset-security-report-2017.pdf

Hota, H. S., Shrivas, A. K. y Hota, R. (2018). An Ensemble Model for Detecting Phishing Attack with Proposed Remove-Replace Feature Selection Technique. *Science Direct*, *8*. doi:10.1016/j.procs.2018.05.103

Islam Mamun, M. S., Rathore, M. A., Lashkari, A. H., Stakhanova, N. y Ghorbani, A. A. (2016). Detecting Malicious URLs Using Lexical Analysis. *Springer Link*, 16. doi:10,1007/978-3-319-46298-1_30

Jain, A. K. y Gupta, B. B. (2016). A novel Approach to Protect against Phishing Attacks at Client Side Using Auto-Updated White-List. *Springer Open*, *11*. doi:10.1186/s13635-016-0034-3

Mao, J., Bian, J., Tian, W., Zhu, S., Wei, T., Li, A., y Liang, Z, (2018), Detecting Phishing Websites via Aggregation Analysis of Page Layouts. *Science Direct*, 7, doi:10,1016/j,procs,2018,03,053

Medvet, E., Kirda, E. y Kruegel, C. (2008). Visual-Similarity-Based Phishing Detection. *ACM Digital Library*, *6*. doi:10.1145/1460877.1460905

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill Science.

Mourtaji, Y., Bouhorma, P. y Alghazzawi, P. (2017). Perception of a New Framework for Detecting Phishing Web Pages. *ACM Digital Library*, 6. doi:10.1145/3175628.3175633

Rajab, M. (2018). An Anti-Phishing Method based on Feature Analysis. *ACM Digital Library*, *7*. doi:10.1145/3184066.3184082

Sanglerdsinlapachai, N. y Rungsawang, A. (2010). Web Phishing Detection Using Classifier Ensemble. *ACM Digital Library*, *6*. doi:10.1145/1967486,1967521

Tan, C. L. (2018). Phishing Dataset for Machine Learning: Feature Evaluation. *Mendeley.* doi:10.17632/h3cgnj8hft.1

URL dataset (ISCX-URL-2016). (2016). *UNB*. Recuperado de https://www.unb.ca/cic/datasets/url-2016.html