universidad cenfotec
tecnologías digitales

MAESTRÍA EN CIBERSEGURIDAD

Documento Final de Proyecto de Investigación Aplicada

Detección de Sitios Web Falsos en Dispositivos Móviles mediante Algoritmos Supervisados de Aprendizaje Automático

---

Gómez Varela Claudio Andrés

San José, Diciembre 2019

Dedicatoria

A mi esposa Lourdes y a mis padres y hermanas por el apoyo incondicional recibido durante estos años en la maestría.

In the world of cybersecurity,
the last thing you want is to have a
target painted on you

En el mundo de ciberseguridad,
la última cosa que quieres es tener
un blanco pintado en ti

Tim Cook, CEO de Apple

Agradecimientos

A Dios, por darme la oportunidad y las fuerzas
para poder terminar esta tesis a pesar de las dificultades.

A mi esposa Lourdes, por ser mi soporte incondicional desde que te conocí,
por darme ánimos sin importar lo que pase y ser el amor de mi vida.

A mi hermana Carolina, por apoyarme y ayudarme en los diseños
de imágenes e íconos para la aplicación móvil.

A mis padres, Yolanda y Claudio, y mis hermanas, Carolina, Melissa y María Jose,
por apoyarme, creer en mi y enseñarme que siempre la familia es el mejor apoyo en tiempos difíciles.

A mi profesor tutor Msc. Juan Ignacio Zamora Mora, por la guía y ayuda en este proceso de tesis,
por creer en este proyecto y en mí.

Muy agradecido, muy agradecido, muy agradecido con mis abuelos, en especial con mi abuelo
Gonzalo, que murió mientras desarrollaba esta tesis, por enseñarme
que no importa qué, todo hay que hacerlo bien hecho y con ganas.

universidad cenfotec_
tecnologías digitales

Universidad Cenfotec
Carrera de Postgrado
Maestría en Ciberseguridad

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: **Maestría en Ciberseguridad,** requisito para optar por el título de grado de **Maestría,** para el estudiante: **Claudio Andrés Gómez Varela.**

M. Sc. Juan I. Zamora Mora
Tutor

MSEG Diego González Villachica
Lector 1

M. Sc. Ignacio Trejos Zelaya
Lector 2

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Resumen

El *phishing* afecta a millones de personas en diferentes ámbitos, en busca de obtener alguna ganancia ya sea económica, de información o simplemente generar algún daño a la persona o compañía que se está atacando. Esta investigación busca poder responder a la necesidad de las personas de tener una herramienta que les permita detectar si un sitio web que se visita o un URL que le llega al correo electrónico puede ser una amenaza o no.

En la investigación se analizan diferentes fuentes de información para entender qué se ha hecho con respecto al análisis de URL y determinar qué atributos son los necesarios para generar un conjunto de datos apropiado para un algoritmo de aprendizaje automático supervisado. Una vez que el conjunto de datos está listo, se realiza un estudio para analizar diferentes algoritmos de aprendizaje automático supervisado y determinar cuál posee un mejor porcentaje de acierto y así utilizar un modelo basado en este algoritmo para implementar el servicio web. Luego de que el servicio web se encuentre funcionando, se iniciará la implementación de una aplicación móvil que permita a los usuarios detectar si el URL estaba relacionado con *phishing* o no.

Palabras clave: *phishing*, aprendizaje automático supervisado, aplicación móvil, detección

Abstract

Phishing affects millions of people in different areas, in search of obtaining money, information or simply generating some damage to the person or company that is being attacked. This research seeks to respond to the need of people, to have a tool that allows them to detect if a website that is visited or a URL that reaches the email can be Phishing or not.

The research analyzes different sources of information to analyze what has been done with respect to URL analysis and determine which attributes are necessary to generate an appropriate data set for a supervised machine learning algorithm. Once the data set is ready, a study is carried out to analyze different supervised machine learning algorithms and determine which one has a better success rate and thus use a model based on this algorithm to implement the web service. After the web service is running, the implementation of a mobile application will begin, allowing users to detect if the URL was Phishing or not.

Keywords: Phishing, supervised machine learning, mobile application, detection

1 Introducción

1.1. Generalidades

Hoy en día, los ataques cibernéticos están creciendo a un ritmo acelerado, pero muchas personas aún desconocen que esto sucede y cómo defenderse de ellos. Para ello, la presente investigación busca encontrar una manera para ayudar a las personas que no tienen mucha relación con la tecnología a tener una herramienta que les permita detectar sitios web falsos y maliciosos (*Phishing*).

1.2. Antecedentes del problema

Los ataques por medio de sitios web maliciosos pueden ocurrir a través un redireccionamiento de una página web confiable, por medio de un enlace puesto en redes sociales o enviado por un mensaje de texto, entre otros. El usuario de dispositivos móviles puede iniciar una compra o tratar de acceder a un sitio que cree seguro, por lo que ingresará datos reales a una página maliciosa sin saberlo.

Al hacer esto el usuario puede perder dinero, o generar daños que no se puedan cuantificar. Por ello se ha decidido realizar esta investigación con el fin de darle al usuario común una manera de poder defenderse contra estos ataques.

1.3. Definición y descripción del problema

Los ataques de tipo*phishing* en sitios web son un problema de seguridad en el cual las personas creen estar en un sitio, ingresan datos relacionados a tarjetas de crédito, nombres de usuario y contraseñas, por lo que se convierten en una población vulnerable a robos de información o estafas.

El problema de la investigación radica en cómo detectar por medio de algoritmos supervisados de aprendizaje automático los sitios web falsos, y así evitar que un usuario ingrese datos sensibles y se produzca un daño importante.

1.4. Justificación

Este proyecto permite a las personas usuarias estar protegidas contra ataques de tipo *phishing*, quienes en muchos casos ni siquiera son conscientes de la existencia de este tipo de amenaza. Muchos de estos ataques llevan a personas a perder dinero o información valiosa por querer obtener lo que el sitio malicioso ofrece, ya sea ropa, accesorios, ofertas por un artículo específico, o simplemente realizar un procedimiento como lo es una transferencia bancaria.

La población se encuentra indefensa ante estos ataques, y muchos ni siquiera poseen el conocimiento básico que se necesita para detectar si un URL que se está abriendo en el explorar web es malicioso o no. Además, la cantidad de herramientas para este tipo de ataques están fuera del alcance de muchas personas, ya sea por limitaciones económicas o porque son vendidas en campos muy específicos, que el usuario ordinario no tiene acceso.

Es por ello que la presente investigación permitirá definir un algoritmo que permita identificar si un sitio web es falso o no, dando al usuario la posibilidad por medio de una aplicación móvil, ser una herramienta que le ayude en el día a día, ante el incremento de ataques tipo *phishing*.

1.5. Objetivos

Se decide implementar la taxonomía de Bloom debido a que es una de las más robustas. Lleva varias décadas funcionando y se ha comprobado que brinda una facilidad para detectar si se ha adquirido nuevas habilidades, para así comprender los niveles de la investigación desde lo más bajo hasta lo más complejo, obteniendo conocimiento en cada uno de estos niveles.

1.5.1. Objetivo general

Detectar y evaluar sitios web falsos en dispositivos móviles mediante algoritmos supervisados de aprendizaje automático por medio de un servicio web integrable con reconocimiento de voz.

1.5.2. Objetivos específicos

1.5.2.1 Identificar los atributos de un sitio web para determinar si es falso.

1.5.2.2 Identificar los algoritmos candidatos para el reconocimiento automático de sitios web falso.

1.5.2.3 Recolectar datos de sitios web falsos para la definición de un conjunto de datos de entrenamiento.

1.5.2.4 Desarrollar los modelos de entrenamiento para los algoritmos seleccionados.

1.5.2.5 Evaluación de la capacidad predictiva de los algoritmos para determinar el modelo a integrar en un servicio web.

1.5.2.6 Desarrollar un servicio web en tecnología Python para exponer el modelo predictivo.

1.5.2.7 Desarrollar una aplicación móvil para la detección de sitos web Phishing mediante el servicio web desarrollado.

1.6. Alcances y limitaciones

Al delimitar un proyecto se da la posibilidad de determinar qué será lo que se desarrollará en la investigación y evitar que existan expectativas no contempladas.

1.6.1. Alcances

1.6.1.1 Recolección y análisis de datos: Para obtener los datos de entrenamiento, se van a utilizar las bases de datos en línea de sitios web falsos llamados PhishTank y OpenPhish. Así mismo, para la obtener sitios web válidos y no maliciosos se utilizará la base de datos Alexa.

1.6.1.2 Ambiente de desarrollo: Se va a utilizar el lenguaje de programación Python y el componente Scikit-Learn para la implementación de modelos de aprendizaje automático para la detección de sitios web falsos.

1.6.1.3 Plataforma de servicios web: El servicio web se implementará con Python y Django. El tipo de servicio a desarrollar es tipo REST.

1.6.1.4 Integración con tecnología móvil: La integración con reconocimiento de voz para la identificación de sitios web falsos es dispositivos móviles se realizará con iOS y Shortcuts.

1.6.1.5 Código fuente: El prototipo final de la investigación estará disponible para el público en un repositorio de Github.

1.6.2. Limitaciones

1.6.2.1 La construcción del prototipo móvil se implementará únicamente en iOS, bajo una arquitectura compatible con otras tecnologías como Android o Windows Mobile.

1.6.2.2 La aplicación en iOS no podrá ser publicada en la tienda de aplicación de Apple, debido a que las bibliotecas utilizadas solo pueden ser accedidas en dispositivos supervisados, Apple Inc. (2019a).

1.7. Estado de la cuestión

En la presente investigación se muestra cuáles son los hallazgos científicos y técnicos que se han desarrollado desde el año 2011 en tema de detección de sitios web falsos, dado que año con año aparecen nuevos y mejores algoritmos para esta solución y desde el año 2011 se encontraron mejores resultados para satisfacer el problema.

1.7.1. Formulación de preguntas

Se trata de buscar que las preguntas abarquen el cómo identificar sitios web falsos y las reglas utilizadas para utilizar en los algoritmos. Por lo tanto las preguntas expuestas para esta investigación son:

- ¿Qué atributos son necesarios para identificar si un sitio web es falso o no?
- ¿Qué algoritmos de aprendizaje automático son los mejores para detectar sitios web falsos?

Además, las palabras clave que componen esta investigación están definidas por: algoritmo, predicción, aprendizaje automático y sitios web falsos.

1.7.2. Selección de fuentes

Se seleccionarán todas las fuentes que contengan información referente a las palabras clave antes mencionadas. Además se investigarán todos los estudios que se hayan realizado en idioma inglés, dado que en español no se encontró ningún artículo útil para la investigación. Aunado a esto, el método que se implementará para realizar la búsqueda es por medio de repositorios electrónicos en la web. Las palabras de búsqueda que se escogerán para el idioma inglés se presentan en la tabla 1.

Tabla 1: Palabras de búsqueda en el idioma inglés

prediction & model & scam & website & phishing

La lista de fuentes electrónicas utilizadas son:

1. ACM Digital Library
2. EBSCOhost Web
3. IEEE Xplore Digital Library

El mecanismo de inclusión se enfoca en el año en que el documento fue presentado. Se escogerán todos los documentos posteriores al año 2010. Además, se incluye únicamente todos los documentos relacionados al tema de detección de sitios web falsos por medio de algoritmos de aprendizaje automático. El método de exclusión se realiza por medio de la lectura y análisis del *abstract* del documento así como sus conclusiones. Tomando en cuenta que se tendrá que profundizar si los dos puntos anteriores no dan buen soporte de la teoría que se quiere investigar. Con esto, se puede observar con mayor detalle de qué trata cada documento, y analizar si de verdad se asocia con los objetivos buscados.

1.7.3. Ejecución de la revisión de fuentes

Para cada una de las fuentes se realiza una mezcla de las palabras clave referentes a la tabla 1. Para la fuente ACM Digital Library, se ejecuta la cadena de búsqueda en idioma ingles mostrada en la tabla 2.

Tabla 2: Cadena de búsqueda para fuente ACM Digital Library.

prediction & model & scam & website

Para la fuente EBSCOhost Web se aplica la cadena de búsqueda referenciada en la tabla 3.

Tabla 3: Cadena de búsqueda para fuente EBSCOhost Web.

prediction & model & phishing & website

Y finalmente, para la fuente IEEE Xplore Digital Library se aplica la cadena de búsqueda que se muestra en la tabla 4.

Tabla 4: Cadena de búsqueda para fuente IEEE Xplore Digital Library.

prediction & phishing

Para la extracción de la información se procede mediante una revisión y mapeo sistemático de los artículos encontrados para determinar la información relevante que ayudará a la investigación para tener la base de información necesaria.

1.7.4. Resultados de revisión de fuente

Hoy en día, los ataques por medio del *phishing* incrementan con los años y la manera en que los atacantes lo realizan mejora con el pasar del tiempo. Como lo muestra Apwg.org (2018), a finales del año 2018 la cantidad de sitios afectados por esta práctica fue de 138,328, recolectados por sus proveedores. Aunque cabe resaltar que se ha incrementado la dificultad de encontrar los sitios falsos dado que se han creado

mejores técnicas para ofuscar y evitar que se detecten estos sitios. Además, en este mismo reporte, se puede observar los sectores de negocio más afectados (ver figura 1), donde se muestra que el sector de pagos es el más afectado, con un 33 %; seguido del sector software y correos electrónicos, con un 29.8 % y del sector financiero, con un 14.3 %.

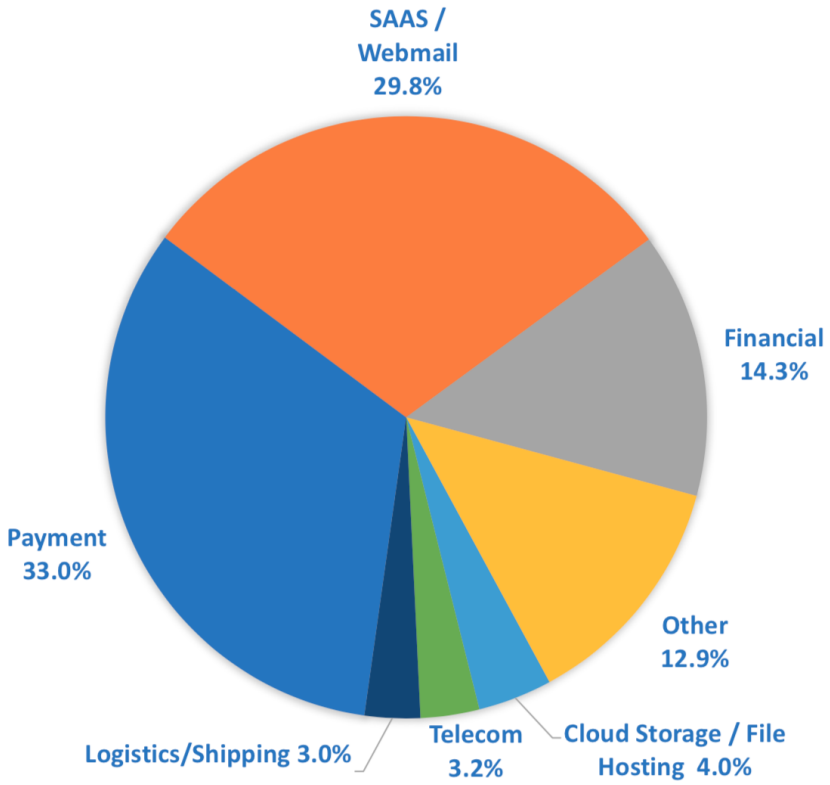

Figura 1. Sectores más perjudicados por el phishing. (Apwg.org, 2018)

Cabe destacar que cualquier persona que sufre de un ataque de *phishing* puede sufrir tanto un impacto económico como psicológico, de acuerdo con Whitty et al. (2017). Las víctimas pueden experimentar sentimientos como vergüenza, culpa, depresión e incluso pérdida de confianza. Además, tratar de capturar a los atacantes es difícil, dado que normalmente se encuentran fuera del país de la víctima y utilizan métodos difíciles de rastrear.

Los criminales siempre buscan una manera de robar información utilizando *phishing* por medio de engaños para hacerse pasar por otras entidades, enviando enlaces

web o URL, que redireccionan a los sitios falsos para que así el usuario introduzca la información requerida creyendo que está en el sitio original y usualmente los usuarios nunca le ponen atención a los URL, según Mourtaji et al. (2017).

Así también, como lo menciona Mohammad et al. (2014), el *phishing* se ha vuelto un gran problema no solo por el aumento de los ataques sino por las estrategias utilizadas para diseñar dichos sitios web que incluso personas expertas en seguridad o computación pueden ser engañadas.

El algoritmo utilizado por Mourtaji et al. (2017), consta de 4 técnicas en conjunto con un algoritmo de clasificación de aprendizaje automático. Lo que realizan los autores es extraer atributos o características de los URL basados en las técnicas que se describen a continuación:

- **Método basado en enlistado negro (Black-listed en inglés):** Cuando un URL es detectado como malicioso o de tipo *phishing*, es agregado en una base de datos. Así, cuando un usuario trata de acceder un URL, se compara contra esta base y se puede determinar facilidad si el URL es malicioso o no. Cabe destacar que ésta técnica solo funciona cuando los URL ya fueron incorporados dentro de la base de datos.

- **Método léxico:** Realiza un análisis léxico del URL por medio de una división de las diferentes partes del URL, protocolo, nombre del dominio, dirección, nombre del archivo solicitado, entre otros.

- **Método basado en el contenido:** Analiza el contenido del sitio web para buscar elementos que ayuden a determinar si la página es maliciosa o no.

- **Método basado en seguridad e identidad:** Esta técnica trata de buscar puntos de seguridad (como validación de DNS) e identidad (comparando URL utilizados para descargar imágenes dentro del sitio).

Una vez que las 13 características que Mourtaji et al. (2017) definen en la tabla 5, los autores realizan una asignación de valores para determinar si la característica está definida por sitio maligno, sitio valido o sitio sospechoso. Y con esto se logra realizar un problema de clasificación binaria, el cual es resuelto por medio de algoritmos del mismo tipo, específicamente de aprendizaje automático, regresión logística o Näives Baiyes. Los autores utilizan una biblioteca llamada Spark, la cual es de código abierto, para realizar el procesamiento de gran cantidad de datos de forma paralela.

Tabla 5: Atributos para detección de sitios web maliciosos (Mourtaji et al., 2017, p. 4)

Features Group	**Phishing Function**	**Values**	**Technical Feature identifier in the database**
Lexical based analytics method	Having IP Address	-1, 1	has_ip
	Having long url	1, 0,-1	long_url
Abnormal Based Feature (Lexical based analytic method)	Request_URL	1,-1	req_url
	Abnormal URL anchor	- 1,0,1	url_of_anchor
	Links_in_tags	1,-1,0	tag_links
Content based analytics method	Redirect	0,1	redirect
	on mouseover	1,-1	mouseover
	Iframe	1,-1	iframe
Security Black-list and identity-based method	Age of domain	-1, 0,1	domain_age
	DNSn Record	-1,1	dns_record
	Web traffic	-1,0,1	traffic
	Page Rank	-1,1	page_rank
	Google Index	1,-1	google_index
Result	target	1,-1	target

El porcentaje de éxito mostrado por Mourtaji et al. (2017) es de un 95.5 %, con un porcentaje de error del 1.4 % utilizando un conjunto de datos de 1200 pruebas.

Otro autor utiliza una técnica de las anteriores y basa todo su algoritmo en un análisis léxico de las partes del URL. Khonji, Iraqi, y Jones (2011) luego realizan una construcción de un modelo de clasificación estadística para evaluar la efectividad del algoritmo. Dentro de las observaciones realizadas por Khonji et al. (2011), hace notar que la proporción de partes del URL ya vistas pueden reaparecer en siguientes URL es más alta que se muestren partes de URL no vistas anteriormente. También

recalca que el traslape entre partes de URL vistos anteriormente de sitios maliciosos y sitios legítimos es muy pequeña.

Como parte del estudio hecho por Khonji et al. (2011), se obtiene una posición relativa de la muestra analizada, lo cual reduce el traslape entre URL legítimos y maliciosos. Es decir, que si la muestra se encuentra dentro del nombre del dominio es muy distinto que cuando existe dentro del nombre del archivo en el URL. Además incluye todas las muestras y partes del URL, no importan si son pequeñas.

Este algoritmo cuenta con un porcentaje de éxito variable dependiendo del conjunto de datos utilizado y el porcentaje de sitios web maliciosos y los legítimos utilizados, con un máximo valor de 97.31 % y un falso positivo de 2.89 %. La cantidad de datos utilizados para las pruebas es de 20.000.

Por otra parte Mohammad et al. (2014), proponen un modelo inteligente para poder predecir sitios web falsos, basado en una red neuronal artificial auto estructurada. Es decir, automatiza el proceso de estructurar la red mostrando gran aceptación en datos ruido (datos que no son necesarios), tolerancia a fallos y una muy buena predicción. Así como Mourtaji et al. (2017), Mohammad et al. (2014) tratan el problema de detectar si un sitio web es de tipo *phishing*, planteando el problema como uno de clasificación binaria.

Las redes neuronales, son un gran ejemplo de predicción no lineal, que se utiliza en ámbitos como reconocimiento de patrones, reconocimiento de voz y clasificación de archivos entre otros, de acuerdo con Mohammad et al. (2014). Además, dentro de su implementación, el autor utiliza 17 características para determinar uno de los 3 posibles valores: legítimo, malicioso, sospechoso (ver tabla 6). En algunos dependiendo de lo que los autores mencionan, el valor sospechoso puede no existir dentro de los atributos, dado que hay ciertas características que pueden definir completamente si el sitio es malicioso o no.

Tabla 6: Tabla de atributos para detección de sitios web maliciosos en red neuronal usada por Mohammad et al. (2014, p. 455)

Using IP address
Long URL
URL having @ symbol
Adding prefix and suffix
Sub-domain(s)
Misuse of HTTPs
Request URL
URL of anchor
Server form handler
Abnormal URL
Redirect page
Using pop-up window
Hiding suspicious link
DNS record
Website traffic
Age of domain
Disabling right click

Además, el algoritmo utilizado por Mohammad et al. (2014), utiliza un enfoque constructivo para la red neuronal, es decir, se inicia con un número pequeño de neuronas en la capa oculta, y se va incrementando hasta el límite establecido durante el entrenamiento de la red neuronal. El porcentaje de éxito mostrado por este algoritmo es de un 92.18 % con un conjunto de datos de 1400 pruebas.

Por su parte, Zuhair et al. (2016) proponen que un modelo que busca cuáles son las características necesarias para poder detectar si un sitio web es de tipo *phishing*

o no, a las cuales les denomina características híbridas, es decir, características de distintas categorías. En total utiliza 58 características, donde existen características o atributos del contenido del sitio web así como del URL. En la tabla 7 se puede observar los atributos utilizados por el autor basados en el contenido del sitio web y en la tabla 8, se pueden observar los atributos extraídos de los URL. Estos atributos, en un futuro próximo pueden usarse para poder obtener un mejor resultado en esta investigación.

Tabla 7: Atributos extraídos del contenido de páginas web. (Zuhair et al., 2016, p. 33)

Embedded Objects and Links Features		***Cross Site Scripting Features***	
Index	**Features**	**Index**	**Features**
F1	Number of Scripting.FileSystemObject	F25	JavaScripts scripts length
F2	Number of Excel.Application	F26	Number of functions' cealls in javascript
F3	Presence of Wscript.shell	F27	Number of script lines in javascripts
F4	Presence of Adodb.Stream	F28	Script line length in javascripts
F5	Presence of Microsoft.XMLDOM	F29	Existence of long variable names in java scripts
F6	Number of <embed>	F30	Existence of long variable names in java scripts
F7	Number of <applet>	F31	Number of fromCharCode()
F8	Number of Word.Application	F32	Number attachEvent()
F9	link length. In <embed>	F33	Number of eval()
F10	Number of <iframe>	F34	Number of escap()

F11	Number of <frame>	F35	Number of dispatchEvent()
F12	Out-of-place tags	F36	Number of SetTimeout()
F13	Number of <form>	F37	Number of exec()
F14	Number of <input>	F38	Number of pop()
F15	Number of MSXML2.XMLHTTP	F39	Number of replace code()
F16	Frequent <head>, <title>, <body>	F40	Number of onerror()
F17	<meta index.php?Sp1=>	F41	Number of onload()
F18	Codebase attirbute in <object>	F42	Number of onunload()
F19	Ciodebase attribute in <applet>	F43	Number of <script>
F20	href attribute of <link>	F44	Frequent
F21	Number of void links in <form>	F45	Number of <script>
F22	Number of out links	F46	Number of MSXML2.XMLHTTP
F23	Number of <form>in JavaScripts	F47	Number of onerror() in javascripts
F24	Number <input>in JavaScripts	F48	Number of SetInterval()

Tabla 8: Tabla de atributos extraídos de los URL. (Zuhair et al., 2016, p. 34)

Index	***Webpage URL Features***
F49	Multiple TLD
F50	Brandname in hostname

F51	Special symbols in URL
F52	Coded URL
F53	IP address instead of domain names
F54	Typos in base name
F55	Long domain name
F56	Misleading subdomains
F57	Number of dots in URL
F58	Path domain length

Luego de que el autor realiza una extracción de estos atributos, de forma experimenta, utiliza el algoritmo máquinas de vectores de soporte para realizar las pruebas basados en tres conjuntos, uno donde solo se obtienen atributos del contenido del sitio web, otro donde solo se obtienen atributos del URL, y un tercer conjunto basado en la combinación de los dos anteriores, refiriéndose a este como conjunto híbrido de atributos.

Basados en algunas deficiencias de los analizadores de texto para detectar *phishing*, Chiew, Choo, Sze, y Yong (2018), propone un método llamado Phishdentity, el cual se basa en buscar la identidad de un sitio web por medio del buscador de imágenes de Google, utilizando el ícono de la página (conocido como *favicon* en inglés). Una de las razones por las que el autor escogió Google es porque permite utilizar la imagen como parte de la consulta y además porque Google posee la mayor cantidad de sitios legítimos indexados. Además, recalca que al usar Google se evita almacenar una base de datos con la información.

El flujo sugerido por Chiew et al. (2018), radica en obtener el ícono del sitio web, y utilizar el buscador de imagénes de Google para detectar algún intento de *phishing*. El favicon se obtiene al agregar el texto *favicon.ico* al dominio del sitio web. Este nuevo URL es el utilizado en el buscador de Google.

Este método utiliza 5 características basados en los resultados de la búsqueda: el Second Level Domain (SLD), la dirección donde esta el archivo en el URL, el título y la pequeña descripción que aparece para el resultado, así como el texto resaltado. Luego, Chiew et al. (2018) calcula la frecuencia con ciertas ecuaciones matemáticas para obtener la identidad real junto con las entradas obtenidas en la búsqueda, generando una frecuencia acumulada. Además hace uso de una amplificación del nombre de dominio como la quinta característica, dado que el nombre de dominio de un sitio web legítimo es único contra otros sitios web. Por lo que los atacantes, tratan de incluir palabras clave dentro de los URL para confundir a los usuarios.

Luego, Chiew et al. (2018) mencionan que en caso de que no tenga un ícono de sitio se proponen los siguientes atributos para ayudar a la detección:

- **URL sospechoso:** Se busca por los signos @ y **-**, dado que pueden excluir partes del URL y redireccionar a otro sitio.
- **Puntos en el dominio:** Dentro de la investigación se encuentra que no es normal que un URL tenga más de 5 puntos en el dominio.
- **Edad del diominio:** Se busca la edad del dominio por medio del servicio *WHOIS*, donde normalmente los sitios web falsos tienen una vida corta. Este atributo se basa en el trabajo realizado por Carniege Mellon Anti-phishing and Network Analysis Tool (CANTINA), pero utilizando un límite más pequeño de 30 días.
- **Dirección IP:** El uso de IP en lugar de nombre del dominio.
- **Web of Trust:** Es un sitio web que muestra la reputación de otro sitio web basado en la retroalimentación de otros usuarios y páginas web.

Una vez que se obtienen estos atributos, se calcula un valor que se suma al valor anterior de los atributos generados y con esto se obtiene un tercer valor que determina si se trata de un sitio web malicioso o no. Según los resultados mostrados por Chiew

et al. (2018), dando un resultado de éxito de 96.93 % y un valor de falso positivo de 4.13 % con un conjunto de datos de 3500 pruebas.

En la misma línea de Zuhair et al. (2016) , Mourtaji et al. (2017) y Chiew et al. (2018), Gowri (2017) utiliza características basadas únicamente en el URL, pero agrega dos nuevos atributos:

1. Edad del dominio.
2. Registros del DNS.
3. Tráfico del sitio por medio del sitio web Alexa.
4. Símbolo @.
5. Inclusión de *HTTPS* en el URL.

Algunos autores encuentran problemas al momento de obtener muestras para la ejecución de sus algoritmos. Por ello, Imani y Montazer (2017) crean su propio algoritmo para poder entrenarlo, con una menor cantidad comparado con la gran cantidad que necesita una red neuronal por ejemplo. El algoritmo Weighted Feature Line Embedding (WFLE), estima matrices de dispersión utilizando pesos, para corregir la cualidad presentada por el conjunto limitado de muestras que poseen. Con ello, generan pruebas virtuales que logran clasificar en dos tipos de muestra: normal y anormal. La normal, son todas aquellas muestras que pertenecen a una misma clase. Mientras que la anormal son aquellas muestras que no son similares que pertenecen a la misma clase.

Al mezclar estas dos categorías, Imani y Montazer (2017), logran reducir el error de la clasificación. Una vez obtenido el conjunto de pruebas de entrenamiento, utiliza el algoritmo vecino cercano Nearest Neighbour (NN), junto con distancias Euclideanas, para detectar a que clase pertenece la muestra, además del algoritmo utiliza 30 atributos muchos similares con el trabajo realizado por Chiew et al. (2018), Gowri

(2017), Mourtaji et al. (2017) y Zuhair et al. (2016), según lo muestra la tabla 9. El trabajo realizado por los autores, muestra un porcentaje de éxito de un máximo de 87.12 % y una proporción de error de un 14.36 % (varía de acuerdo al tamaño de las muestras utilizadas inicialmente).

Tabla 9: Atributos usados por Imani y Montazer (2017)

No.	***Attribute***	***No.***	**Attribute**
1	having_IP_Address	16	SFH
2	URL_length	17	Submitting to email
3	Shortining Service	18	Abnormal URL
4	having_At_Symbol	19	Redirect
5	double_slash_redirecting	20	on mouseover
6	Prefix_Suffix	21	RightClick
7	having_Sub_Domain	22	popUpWidnow
8	SSLfinal_State	23	Iframe
9	Domain_registeration_length	24	age of domain
10	Favicon	25	DNSRecord
11	port	26	web traffic
12	HTTPS_token	27	Page Rank
13	Request_URL	28	Google Index
14	URL_of_Anchor	29	Links pointing to page
15	Links_in_tags	30	Statistical report

Por su parte Tyagi et al. (2018), se enfocan meramente en algoritmos de clasificación de aprendizaje automático. Utiliza SVM, Random Forest (RF), árboles de decisiones, potenciación del gradiente (*Gradient Boosting*) y regresión lineal. Realiza una extracción de 30 atributos (ver tabla 10). Una vez extraídos, se les asigna un valor de 1, 0 ó -1, si el sitio es legítimo, sospechoso o tipo phishing respectivamente.

Tabla 10: Atributos obtenidos de los URL (Tyagi et al., 2018)

No	Feature	Posible value
1	Having IP Address	1 or -1
2	Long URL Length	Length >= 75 = -1 54 >= Length <57 = 0 Length <54 = 1
3	Shortened URL	1 or -1
4	Having @ Symbol	1 or -1
5	Double Slash Redirecting	1 or -1
6	Prefix suffix	1 or -1
7	iFrame Tag	1 or -1
8	Anchor Tag	Tag % <= 31 = 1 31 >Tag % <= 67 = 0 Tag % >67 = -1
9	Disabling right click	1 or -1
10	Links in <Meta>, <Script>and <Link>tags	Tag % <= 17 = 1 Tag % >81 = -1
11	Age of domain	1 or -1
12	record	1 or -1
13	HTTPS with SSL	1 or -1
14	Domain Registration length	1 or -1
15	Website traffic	1 or -1
16	Statistical based reports feature	1 or -1
17	Using non standard port	1 or -1

18	Abnormal URL	1 or -1
19	Sub Domain and Multi Sub Domain	# of dots in the URL == 0 = 1 # of dots in the URL >1 = 0 # of dots in the URL >2 = -1
20	Favicon	1 or -1
21	Request URL	Content % >61 = -1 22 >= Content % <= 61 = 0 Content % <22 = 0
22	Server From Handler (SVF)	1 or -1
23	Submitting information to Email	1 or -1
24	Website forwarding	1 or -1
25	Status Bar Customization	1 or -1
26	Using Pop-Up Window	1 or -1
27	Page Rank	Page Rank <= 0.2 = -1 Page Rang >0.2 = 1
28	Google Index	1 or -1
29	Number of Links pointing to Page	# of Links == 0 = -1 0 ># of Links <= 2 = 0 # of Links >2 = 1
30	The existense of ”HTTPS”Token in the Domain Part of the URL	1 or -1

Luego de obtener los atributos, el autor realiza un pre-proceso de limpieza a los datos, removiendo todos aquellos atributos que tenga un factor de inflación de la varianza (Variability Inflation Factor (VIF) por sus siglas en inglés.), y luego aplicando

análisis de componentes principales y el algoritmo K-vecinos próximos (KNN) para el retoque final al conjunto de datos. Una vez aplicado este pre-proceso, se ejecutaron los algoritmos antes mencionados. Al hacer un análisis del resultado de éxito de dichos algoritmos, RF obtuvo el mejor resultado con un porcentaje de 98.40 %.

Por su parte Weiss y Khoshgoftaar (2017), utiliza un método de la rama del aprendizaje automático, llamado trasferencia de aprendizaje heterogéneo, el cual busca transformar los datos de entrada de los diferentes tipos de atributos que utiliza, a datos comunes o heterogéneos. El algoritmo utilizado por el método es llamado análisis de correlación canónica. Luego, el autor realiza una serie de pruebas con diferentes algoritmos de aprendizaje automático usando los conjuntos de datos homogéneos creados anteriormente, y consigue un porcentaje de éxito que varía del 74.3 % al 91.0 %.

Asimismo, Ahsan et al. (2018) crearon un algoritmo para aplicarlo al conjunto de datos de entrenamiento que contiene los sitios web catalogados como *phishing*, esto con el motivo de evitar un desbalance entre los conjuntos de datos con sitios legítimos. El algoritmo es llamado *Systhetic Minority Over-Sampling Technique* (SMOTE). Luego, para probar el algoritmo, utiliza 3 técnicas de aprendizaje automático XGBoost, Random Forest (RF, por sus siglas en inglés) y SVM. El autor hace un análisis basado en los atributos mostrados en la tabla 11, para determinar si un sitio web es catalogado como *phishing* o no.

Tabla 11: Atributos descritos por Ahsan et al. (2018)

Feature	***Logic***	***Result***
SFH	Blank $\vee$ empty	-1
	Diverts to different Domain	0
	Else	1

Pop-up Window	Right Click Disabled	-1
	Right Click showing Alert	0
	Else	1
SSL Final State	https $\wedge$ trusted issuer $\wedge$ age $\geq$ 2years	1
	https $\wedge$ issuer is not trusted	0
	Else	-1
Request URL	URL $\leq$ 22 %	1
	URL $\geq$ 22 % $\wedge$ <61 %	0
	Else	-1
URL anchor	URL anchor % <31 %	1
	URL anchor % $\geq$ 31 % $\wedge \leq$ 67 %	0
	Else	-1
Web Traffic	Web Traffic <150000	1
	Web Traffic <150000	0
	Else	-1
URL Length	URL length <54	1
	URL length $\geq$ 54 $\wedge \leq$ 75	0
	Else	-1
Age of Domain	Age $\leq$ 6 Months	1
	Else	-1
Having IP	IP address exists in URL	-1
	Else	1

Como Chawla, Bowyer, Hall, y Kegelmeyer (200w) lo mencionan, el algoritmo busca realizar un sobre muestreo de la clase minoritaria, por medio de ejemplos sintéticos en lugar de reemplazarlos por nuevos para equiparar el conjunto de datos. Esta creación sintética basa su creación en los atributos en lugar del ámbito del sitio

web. Según los resultados planteados por Ahsan et al. (2018), el porcentaje de éxito al momento de aplicar SMOTE varía de acuerdo al algoritmo, con valores de 97.17 %, 97.17 % y 91.13 %, para XGBoost, RF y SVM, respectivamente, utilizando un conjunto de datos de 702 pruebas.

Por su parte Aburrous et al. (2010), por medio de algoritmos de minería de datos logran obtener un valor de éxito relativamente bueno de un 88.07 % y un porcentaje de falsos positivos de 12.622 % utilizando el algoritmo MCAAR. Cabe destacar que a pesar que no es un algoritmo de aprendizaje automático, utiliza ciertos atributos similares a los usados por Mourtaji et al. (2017), que se pueden observar en la tabla 12.

Tabla 12: Indicadores de sitios falsos (Aburrous et al., 2010)

Criteria	***N***	***Phishing Indicators***
URL & Domain Identity	1	Using IP address
	2	Abnormal request URL
	3	Abnormal URL of Anchor
	4	Abnormal DNS record
	5	Abnormal URL
Security & Encryption	1	Using SSL certificate (Padlock Icon)
	2	Certificate authority
	3	Abnormal cookie
	4	Distinguished names certificate
Source Code & Java Script	1	Redirect pages
	2	Straddling attack
	3	Pharming attack
	4	OnMouseOver to hide the Link
	5	Server Form Handler (SFH)

Page Style & Content	1	Spelling errors
	2	Copying website
	3	Using forms with Submit button
	4	Using pop-ups windows
	5	Disabling right-click
Web Address Bar	1	Long URL address
	2	Replacing similar char for URL
	3	Adding prefix or suffix
	4	Using the @ symbol to confuse
	5	Using hexadecimal char codes
Social Human Factor	1	Emphasis on security
	2	Public generic salutation
	3	Buying time to access accounts

Así como Aburrous et al. (2010) utiliza minería de datos para su proceso, Zhuang et al. (2012) usa una combinación de minería de datos junto a un par de algoritmos de aprendizaje automático, como lo son Näives Baiyes y SVM. En primer lugar, el autor extrae un total de 10 atributos mencionados en la tabla 13. Luego, utiliza los algoritmos mencionados anteriormente para generar los clasificadores de los datos para ser introducidos dentro del método de clasificación de ajuste (ensemble method, en inglés), el cual provee un mejor rendimiento que el uso separado de los clasificadores mencionados anteriormente.

Tabla 13: Extracción de atributos de Zhuang et al. (2012)

ID	***Feature Name***	***Description***
1	Title	Title of the webpage.
2	H1-H6	Content in the <h1>to <h6>tags.
3	Keyword	Keyword information in the Meta tag.
4	Description	Page description in the Meta tag.
5	Copyright	Copyright info in the Meta tag.
6	Link text	Corresponding text of the link.
7	Frame	Url address of the Frame.
8	Img	Url address of the Image.
9	Alt	Description text of the image.
10	String	All the other visible string of the page.

Finalmente, utiliza el algoritmo de agrupamiento jerárquico, perteneciente a la minería de datos, por medio de la técnica Term Frequency - Inverse Document Frecuency (TF-IDF) para la asignación de etiquetas a los sitios detectados como tipo *phishing* y realizar una clasificación de los mismos. El resultado de éxito para el algoritmo propuesto por Zhuang et al. (2012) es de un máximo de 98.72 %, ya que varía un poco dependiendo del conjunto de datos utilizado. Aunque el menor obtenido es de un 97.86 %.

De la misma forma que Mourtaji et al. (2017), Abbasi et al. (2015) proponen un modelo de detección basado en los géneros de sitios web. Los géneros están definidos como los tipos de comunicación reconocidas y promulgados por los miembros de una comunidad y organización. En sitios web, los géneros incluyen la página principal, páginas de producto, boletines de noticias, páginas de soporte entre otras, en donde cada uno de estos busca cumplir con un objetivo específico. El método propuesto por Abbasi et al. (2015), utiliza un árbol tipo Kernel, con la estructura del

directorio del sitio web, y etiquetando cada uno de los nodos del árbol con el género necesario, basado en la tabla 14.

Tabla 14: Géneros de sitios web. (Abbasi et al., 2015)

Genre	***Label***	***Description***
About	A	Information about the organization, including history, background, and philosophy.
Career	E	Employment opportunities, pages with employee/workplace testimonials, and other pages with career information for potential hires.
Contact	C	Contact-related content, including locations, phone numbers, comment posting, e-mailing/speaking with representatives, e-mail sign-up, and live chat.
Event	V	Upcoming events, calendars, and lists of activities.
FAQ	Q	Frequently asked questions and other Q&A pages.
Homepage	H	The website's starting page.
Information	N	Articles, editorials, columns, and other informational resources.
Login	L	Login, logout, password retrieval, new member registration, and other account access related content.
Order	O	Order and shopping cart information, including order tracking and shipping.
Outreach	U	Community involvement, giving programs, scholarships, grant applications, and various other outreach-related activities.
Policy	P	Policies, terms, guarantees, and privacy notes.
Price	D	Fees, rates, prices, and quotes.

Product	R	Description of products, services, programs, classes, etc.
Publication	B	Magazines, newsletters, white papers, case studies, and other online publications.
Search	S	Search and navigation pages, including site maps and directories.
Social Media	M	Forums, blogs, and other social media content integrated into the website.
Support	W	Donations, volunteering, and other philanthropic opportunities.
Testimonial	T	Testimonials from customers, clients, students, patients, etc.

Una vez realizados los árboles, Abbasi et al. (2015) realizan un recorrido aleatorio de los arboles, generando sub-árboles, hasta llegar a una cantidad especificada. Una vez que se tienen los recorridos, se inicia el proceso de comparación contra los otros árboles. Se hace mediante cualesquiera dos árboles que posean la misma cantidad de nodos recorridos. Esta comparación resulta en el valor necesario para determinar si el sitio es tipo *phishing* o no. El porcentaje de éxito del algoritmo es de un 97 %.

Por su parte, Elmassry y Almogren (2016), propone un modelo que no se basa en aprendizaje automático sino que trata de validar cómo los humanos perciben las páginas y sus colores. El algoritmo que propone inicia por extraer las 3 ocurrencias máximas de colores de las imágenes en el sitio web. Luego realiza una comparación con una base de datos existente con sitios válidos. Si hay similitud con estos colores y los colores de la base de datos, inicia la comparación de los URL de las imágenes de los sitios válidos contra el sitio por analizar. Si los URL no son similares, el sitio web se declara como phishing. El resultado de éxito de este algoritmo es de un 94.99 %

contra un conjunto de datos de 670 pruebas.

Las ventajas del algoritmo de Khonji et al. (2011), radican en que es más barato procesar únicamente solo el URL que extraer toda la información y contenido de éste. Adicionalmente, ocupa menos espacio el almacenar solo el URL que toda su información.

Una de las grandes desventajas de los algoritmos basados en texto, según Chiew et al. (2018), es que dependen obviamente del texto, por lo que ataques tipo *phishing* basados contenido del texto son incompetentes contra ataques basado en imágenes, dado que los atacantes pueden reemplazar texto en imágenes para evitar ser detectados.

Parte de las limitaciones que poseen Chiew et al. (2018) en su investigación, es que algunos atacantes pueden cambiar ligeramente el contenido del *favicon*, siento similares a la vista, pero aún puede detectarse como sitio maligno. De igual forma, pueden cambiar el *favicon* con el de otro sitio web. Luego pueden existir sitios web nuevos que no han sido explorados por Google o Web of Trust, que pueden resultar en un falso positivo.

Por su parte Abbasi et al. (2015), hacen notar que existen más desventajas relacionadas con los métodos existentes. Las personas autoras señalan que muchos de los métodos toman mucho tiempo y que no son óptimos para ambientes donde se necesita el valor de forma inmediata. Además, muchos de los métodos se usan por medio de extensiones en los exploradores web, los cuales muestran un porcentaje de éxito cercano al 30 % como el autor lo menciona.

2 Marco conceptual

2.1. *Phishing*

Hoy en día, las personas hablan cada vez más de sitios web, pero muchos no conocen de qué tratan. Mourtaji et al. (2017) explican que una página web es un archivo almacenado en un servidor que puede ser accedido por su URL. Este enlace web o URL (por sus siglas en inglés) es la dirección donde se encuentra un archivo.

Los sitios web contienen cierta información necesaria para realizar análisis de detección de sitios web tipo *phishing*. Por ejemplo el ícono del sitio web o *favicon* en ingles, es el ícono anexo a la pestaña en un explorar web, o que esta a la par de la dirección electrónica o a la par del marca páginas, como se muestra en la figura 2. Éste ícono representa la identidad del sitio en un tamaño variable Chiew et al. (2018).

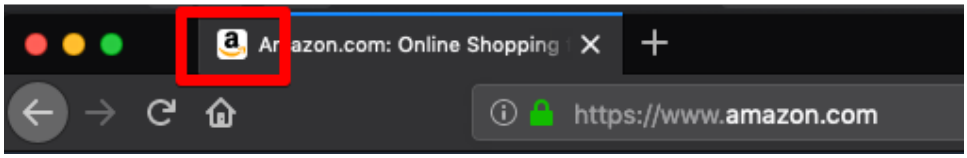

Figura 2. Ejemplo de un ícono de sitio o *favicon*.

Para dar una perspectiva más clara acerca de lo que se entiende por sitio malicioso o phishing, podemos basarnos en la definición dada por Mourtaji et al. (2017, p. 1), donde explica que son aquellos sitios web que buscan imitar sitios web válidos para realizar un robo de información personal, como tarjetas bancarias, fecha de nacimiento, número de tarjetas de crédito, entre otros. Mourtaji et al. (2017). Asimismo

Mohammad et al. (2014), lo define como el arte de hacerse pasar por otra página que es legítima para poder obtener información confidencial valiosa.

Si se explica la palabra phishing desde su taxonomía, podemos ver que proviene del término pescar (*fishing* en inglés), pues trata de poner un señuelo, hacer que el usuario caiga en el anzuelo y finalmente atraparlo Mourtaji et al. (2017). Es decir, en términos técnicos:

- El señuelo se realiza por medio de técnicas de ingeniería social para atraer a la víctima.
- El anzuelo es el sitio web falso.
- Atrapar al pez en este contexto significa obtener la información que el atacante deseaba.

2.2. Categorías del *phishing*

De acuerdo a lo expresado por Abbasi et al. (2015), existen dos categorías de phishing: el confeccionado, en donde los sitios tratan de hacerse pasar por únicos y legítimos, para hacer caer a la víctima en fraudes; y los apócrifos, en los cuales se trata de simular el contenido de otros sitios originales, tratando de atacar los clientes de dicho sitio.

2.3. Algoritmos de aprendizaje automático

Por otra parte, para poder realizar la detección de sitios web tipo *phishing*, se pueden utilizar algoritmos de aprendizaje automático, que son aquellos algoritmos donde se puede programar una máquina a reconocer patrones o que pueda aprender de lo que el algoritmo recibe, Freeman, Haigler, Schmeelk, Ellrodt, y Fields (2019).

Según Hastie, Tibshirani, y Friedman (2008), los algoritmos de aprendizaje automático, aprenden de los datos. Por ejemplo, se espera una medición de resultados, ya sea cuantitativo (un valor exacto esperado) o categórico (tratar de definir la categoría de un objeto), que se desea predecir por medio de un conjunto de características (*features*). Luego se utiliza un conjunto de entrenamiento de datos, el cual posee el resultado esperado, junto con las características que posee el objeto. Luego usando estos datos se genera un modelo de predicción, el cual permite predecir el resultado de objetos que no se conocen.

2.3.1. Supervisado vs no supervisado

Esto es lo que se conoce como aprendizaje automático Hastie et al. (2008), y se le llama supervisado porque los valores en el conjunto de entrenamiento de datos poseen el resultado (también conocido como etiquetas) para las características dadas. En caso de que un algoritmo sea no supervisado, quiere decir que este resultado no se encuentra dentro del conjunto de entrenamiento. El conjunto de **características** que se utiliza para realizar la predicción, se les puede llamar **entradas**, basado en la literatura estadística. Y el resultado esperado se le puede llamar **salidas**, **respuestas** o **variables dependientes**. Otra categoría que Brownlee (2018) menciona son los algoritmos semi-supervisados, en donde no todos los datos de entrenamiento están etiquetados, dado que es muy caro etiquetarlos todos y se necesitan expertos del dominio para llevar a cabo este proceso.

En los algoritmos de aprendizaje automático supervisado un algoritmo aprende un modelo de los datos de entrenamiento, de acuerdo a Brownlee (2018). El propósito es estimar de la mejor manera la función de mapeo para las variables de salida dato ciertos datos de entrada. Esta función de mapeo se le conoce como función objetivo (*target function*).

2.3.2. Errores de predicción

El error de predicción que los algoritmos tienen puede ser dividido en 3 partes: error de sesgo (*Bias error*), error de varianza (*Variance error*) y error irreducible (*Irreducible error*).

El error irreducible no puede ser reducido sin importar el algoritmo que se esté usando según Brownlee (2018), dado que es el error introducido por el problema y es causado por factores como variables indefinidas que influyen en el mapeo de las variables de entrada a las variables de salida.

Por otra parte Brownlee (2018) menciona que el error de sesgo, es generado por las suposiciones hechas por el modelo para hacer la función objetivo más fácil de aprender. Es decir, que un sesgo bajo significa que hay menos suposiciones sugeridas acerca de la forma de función objetivo, y por otro lado, un mayor sesgo significa que existen más suposiciones acerca de la forma de la función objetivo.

Y finalmente, Brownlee (2018) define el error de varianza como la cantidad que la estimación de la función objetivo cambiará si se utilizan diferentes datos de entrenamiento. Idealmente, este valor debe no cambiar mucho de un conjunto de datos de entrenamiento a otro, lo que se significa que el algoritmo está analizando de forma correcta los datos. Es decir, que una varianza baja sugiere cambios pequeños en el estimado de la función objetivo con cambios en el conjunto de datos de entrenamiento, mientras una varianza alta, sugiere grandes cambios en ese estimado.

2.3.3. Entradas vs salidas

Como se mencionó, según Hastie et al. (2008), las salidas pueden perteneces a dos categorías, ya sea una medida cuantitativa, donde se espera que el valor sea mayor o menos a otro, o cualitativo, donde se espera que el valor se encuentre dentro de un conjunto finito de valores, los cuales se suelen expresar por medio de etiquetas.

Asimismo, los autores mencionan que dada la diferencia que pueden tener las sa-

lidas, hace que proceso de predicción se llame distinto. Para aquellas salidas donde se espera un valor cuantitativo (como el peso de una persona o el valor de un objeto) el proceso de predicción se conoce como regresión (*regression*) y clasificación (*classification*) para aquellas salidas que sean cualitativas (cuando sean categorías, como blanco o negro).

2.3.4. Ejemplos de algoritmos de aprendizaje automático supervisado

Dentro de los algoritmos de aprendizaje supervisado más comunes, según Géron (2017) son:

- K-vecino más cercano (*K-Nearest Neighbors*)
- Regresión lineal (*Linear Regression*)
- Regresión logística (*Logistic Regression*)
- Máquinas de vectores de soporte (*Support Vector Machines*)
- Árboles de decisión y bosques aleatorios (*Decision Trees and Random Forest*)
- Redes neuronales (*Neural Networks*)

Usualmente las variables de entrada suelen representarse con la letra X y las variables de salida suelen ser representadas por la letra Y, según Brownlee (2018). Entonces, el proceso de predicción está dado por la evaluación de las entradas por medio del algoritmo generando una salida, representada por la fórmula $Y = f(X)$, es decir se genera una función de mapeo dadas las entradas representadas por X y generando las salidas Y. Además, el autor menciona que existe una diferencia entre el algoritmo y el modelo, siento el modelo la representación específica aprendida de los datos y el algoritmo como el proceso de aprendizaje. Esta relación se puede representar de la siguiente forma, $Modelo = Algoritmo(Datos)$.

2.3.5. Paramétricos vs no paramétricos

Brownlee (2018) menciona que hay dos tipos de algoritmos de aprendizaje automático: paramétricos y no paramétricos (*parametric* y *nonparametric*). En los algoritmos paramétricos o lineales, el modelo de aprendizaje resume los datos con un conjunto de parámetros de tamaño fijo, independiente de la cantidad de ejemplos de entrenamiento, es decir, hace muchas suposiciones acerca del mapeo de las variables de entrada y salida. Las ventajas de este tipo de algoritmos es que son más simples dado que son fáciles de entender y de interpretar resultados, además son rápidos en aprender de los datos y no necesitan tantos datos de entrenamiento. Además, Brownlee (2018) apunta que los algoritmos paramétricos poseen un alto sesgo, pero baja varianza.

Por otro lado los algoritmos de aprendizaje automático de tipo no paramétricos, son aquellos algoritmos que no necesitan hacer grandes suposiciones acerca de la forma de la función de mapeo, es decir, estos algoritmos son buenos cuando se tienen muchos datos, y no se tiene previo conocimiento, buscando ajustar de la mejor manera los datos de entrenamiento mientras se construye la función de mapeo. Estos tipos de algoritmos poseen un mejor rendimiento al generar modelos de predicción, y al no tener realizar suposiciones son libres de aprender de cualquier forma funcional de los datos de entrenamiento, Brownlee (2018). Además el autor menciona, que estos algoritmos poseen un sesgo bajo y una alta varianza.

La parametrización de los algoritmos de aprendizaje automático, es la principal tarea por balancear los errores de varianza y sesgo según Brownlee (2018) para así poder obtener el mejor resultado en la función objetivo.

2.3.6. Aprendizaje en línea vs aprendizaje por lote

Otra forma de categorizar los algoritmos de aprendizaje automático es basado si el sistema puede o no continuar aprendiendo incrementalmente Géron (2017). Es-

tas categorías son aprendizaje por lote (*batch learning*), y aprendizaje en línea (*online learning*). En la primera categoría de aprendizaje por lote, el sistema es incapaz de aprender incrementalmente, es decir, se entrena el sistema utilizando todos los datos disponibles. Por lo que una vez que ya se encuentra entrenado, se procede a lanzarlo a producción y se ejecuta aplicando únicamente lo que ha aprendido. En caso de que se necesite que el sistema aprenda de nuevos datos, se debe entrenar una nueva versión y luego volverlo a lanzar a producción.

En el aprendizaje en línea Géron (2017), el sistema se entrena incrementalmente, alimentándolo con secuencias de datos. Cada uno de estos pasos de aprendizaje, se espera que sea rápido y barato, permitiendo al sistema aprender sobre la marcha mientras los datos van llegando. En este tipo de aprendizaje se tiene un parámetro muy importante llamado tasa de aprendizaje (*learning rate*), que busca determinar qué tan rápido el sistema se debe adaptar para realizar el cambio de datos. Es decir, si se escoge un valor alto de tasa de aprendizaje, el sistema se adaptará a nuevos datos más rápido, pero olvidará más rápido los datos antiguos.

2.3.7. Aprendizaje basado en instancia vs aprendizaje basado en modelos

Géron (2017) añade otras dos categorías por generalización en las que se pueden clasificar los algoritmos de aprendizaje automático. La primera se llama aprendizaje basado en instancia (*Instance-based learning*), en donde compara las instancias de nuevos objetos directamente con las instancias vistas en el entrenamiento del sistema. Además se le añade un medida de similitud dependiendo de lo que el sistema necesite.

El término ”generalización”, según Brownlee (2018) se refiere a qué tan bien un concepto es aprendido por un modelo de aprendizaje automático aplicado a ejemplos específicos no vistos por el modelo cuando estaba aprendiendo. Por lo que el

principal objetivo para un modelo de aprendizaje automático es la buena generalización de los datos de entrenamiento a cualquier dato del dominio del problema.

La otra clasificación que Géron (2017) menciona es el aprendizaje basado en modelos (*Model-based learning*), en donde de un conjunto de ejemplos se crea un modelo, el cual es utilizado para realizar una predicción. Para esta categoría se debe realizar la selección de un modelo, el cual incluye configurar unos argumentos para así obtener la predicción deseada. Estos argumentos deben ser escogidos basado en una medida de rendimiento, la cual puede definirse ya sea por medio de una función de utilidad (*fitness function* o *utility function*) que mide que tan bueno es el modelo, o por una función de costo (*cost function*), que mide qué tan malo el modelo es.

2.3.8. Retos

Dentro de los principales retos que existen cuando se utilizan algoritmos de aprendizaje automático, Géron (2017), es la escogencia del algoritmo y al escogencia de datos a utilizar. Al momento de escoger datos, se debe tratar de buscar datos que representen los nuevos objetos que se quieren generalizar. En caso de que los datos usados para el entrenamiento sean muy escasos se tendrán datos que no representan los casos generales provocando ruido de muestra (*sampling noise*). Pero también puede existir el caso donde haya gran cantidad de datos pero el método para obtener la muestra resulta que tiene fallas de igual forma se puede tener datos no representativo (llamado *sampling bias*).

Como parte de los retos que Géron (2017) menciona, se encuentra el hecho de que los datos de entrenamiento tengan muchos errores, valores no comunes o mucho ruido, generando que el sistema no detecte los patrones de los datos, por lo tanto el sistema no funcionará de la mejor manera. De igual forma el autor menciona que en algunos casos las características utilizadas no son suficientemente relevantes, por lo que se procede a realizar un proceso conocido como la ingeniería de características

(*Feature engineering*).

La ingeniería de características involucra, de acuerdo con Géron (2017), en primer lugar la selección de características (*feature selection*), donde se busca seleccionar las características más significativas. Luego se tiene la extracción de características (*feature extraction*), donde se trata de combinar características existentes, para así producir nuevas que sean más útiles. Y finalmente, se tiene la creación de nuevas características cuando se obtienen más datos.

Otro de los retos mencionados por Géron (2017), es el problema de sobre-ajuste (*overfitting*), en donde el modelo funciona bien con los datos de entrenamiento pero no generaliza bien los datos no conocidos. Este tipo de problemas puede resolverse por medio de la regularización (*regulatization*), en donde el modelo se restringe al eliminar o a agregar datos que ayuden al modelo a funcionar mejor. Además Brownlee (2018) menciona que los algoritmos no paramétricos son los más comunes en presenter el problema de sobre-ajuste.

Por otro lado, se tiene el problema de falta de ajuste (*underfitting*), en donde el modelo es demasiado simple para aprender la estructura interna de los datos, según Géron (2017). Las soluciones que brinda el autor son la selección de un modelo más potente que incluya más parámetros, además de tratar de alimentar el algoritmo con mejores características, o finalmente, reducir las restricciones impuestas al modelo.

2.3.9. Mejora y validación

Brownlee (2018) añade que hay un par de técnicas utilizadas para poder encontrar los mejores valores y no caer en problemas de sobre-ajuste o falta de ajuste. La primera técnica mencionada son los métodos de remuestreo (*resampling methods* traducidos al inglés) en donde se trata de estimar la exactitud del modelo. La otra técnica es tener un conjunto de datos para validación que es evaluado una vez el algoritmo es configurado de la mejor manera para los datos del entrenamiento.

El proceso de validación para determinar si el modelo está generalizando de la mejor manera es utilizar parte de conjunto de datos y separarlo en conjunto de entrenamiento y un conjunto de pruebas, según Géron (2017). La tasa de error en los nuevos casos generado por el conjunto de pruebas se le llama error generalizado (*generalization error* o *out-of-sample error*), el cual indica qué tan bien funciona el modelo en instancias que nunca ha visto. Es decir, si la tasa de error en el entrenamiento es baja pero el error de generalización es alto, quiere decir que el modelo está realizando un sobre-ajuste de los datos del entrenamiento.

Como parte de la validación para determinar que algoritmo utilizar, Géron (2017) menciona otra técnica llamada validación cruzada de datos (*cross-validation*), en donde el conjunto de entrenamiento se divide en subconjuntos complementarios, por lo que cada modelo se entrena utilizando una combinación diferente de estos subconjuntos y luego es validado por los subconjuntos restantes. Uno de los métodos más utilizados de la validación cruzada, según Brownlee (2018), es la validación cruzada de K-iteraciones (k-*fold cross-validation*), en donde el conjunto de datos se subdivide en subconjuntos de pruebas y validación, luego se realiza el entrenamiento de cada subconjunto y se realiza un promedio del valor de predicción por cada subconjunto para así determinar el rendimiento de final del algoritmo.

La optimización es gran parte de un algoritmo de aprendizaje automático de acuerdo a Brownlee (2018). Uno de estos algoritmos es llamado descenso de gradiente (*gradient descent*), en donde se busca los valores de los parámetros o coeficientes de la función f que minimice la función de costo. El proceso del algoritmo inicia por configurar los valores de los coeficientes con valor de 0 o algún valor aleatorio que sea pequeño.

Luego el costo de los coeficientes es evaluado junto a la función para calcular el costo, el cual puede ser representado como $costo = evaluar(f(coeficiente))$, donde este costo es calculado por todo el conjunto de datos de entrenamiento. Luego,

Brownlee (2018), explica que el derivado del costo es calculado, es decir el calculo de la pendiente de la función para así determinar la dirección o signo para mover los valores de los coeficientes, representado por $delta = derivado(costo)$. Esta dirección se refiere a si en la próxima iteración el valor debe ser positivo o negativo.

Luego, Brownlee (2018) aclara, que una vez que la pendiente es calculada, se debe utilizar un valor denominado *alpha* el cual controla que tanto los valores coeficientes debe cambiar por cada iteración. Y con esto se determina el valor del coeficiente, representado por $coeficiente = coeficiente - (alpha\ x\ delta)$. Este procedimiento se debe llevar a cabo hasta el que el costo de los coeficientes sea 0 o que no hayan mayores mejoras en el costo. Cada una de estas iteraciones es llamado un lote (*batch*) y esta forma de descenso de gradiente se le llama descenso de gradiente por lote (*Batch gradient descent*).

Otra modificación del algoritmo descenso de gradiente, según Brownlee (2018) explica, es el algoritmo descenso de gradiente estocástico (*Stochastic gradient descent*), en donde la actualización del valor del coeficiente es llevado a cabo por cada instancia de entrenamiento en lugar de hacerlo al final de cada lote.

2.4. Algoritmos de aprendizaje automático supervisado

2.4.1. Regresión logística

Dentro de los algoritmos de aprendizaje supervisado se tiene la regresión logística (*logistic regression*), en donde la base del algoritmo utiliza la función logística o función sigmoidea, la cual puede tomar cualquier valor real y realizar un mapeo de ese valor a valores entre 0 y 1, sin llegar a ellos en ningún momento, según lo explica Brownlee (2018). Además el autor menciona, que el algoritmo modela la probabili-

dad de la clase por defecto, y explica por ejemplo que si se tienen dos clases A y B, el algoritmo modela la probabilidad de que un valor sea A. Así, una vez aplicado la función logística a esta probabilidad, si el valor es muy cercano a 1 quiere decir que es muy probable que el objeto sea A, y si es muy cercano a 0 quiere decir que es muy probable que el objeto sea B.

2.4.2. Árboles de decisión

Otro de los algoritmos que es más utilizado y conocido, de acuerdo con Brownlee (2018), es el árbol de decisión (*decision tree*) o árboles de clasificación y regresión (*Classification and Regression Trees*, o *CART* por sus siglas en inglés). Este algoritmo utiliza un árbol binario, en donde cada uno de los nodos representa una variable de entrada x y los nodos hojas (nodos sin bifurcaciones), contienen una variable de salida y que se utilizan para hacer la predicción. Este algoritmo utiliza un punto de división, el cual es un valor de uno de los atributos definidos, que servirá como raíz del árbol de decisión.

2.4.3. Bosque aleatorio

Uno de los algoritmos más poderosos del aprendizaje automático es el llamado bosque aleatorio (*random forest*), según Brownlee (2018). Para poder entender cómo funciona este algoritmo se debe conocer un poco más acerca de donde proviene. Inicialmente, existía el método de arranque (*bootstrap method*), el cual es un método estadístico para estimar la cantidad de una muestra de datos.

La idea del método de bootstrap o arranque, como lo muestra Brownlee (2018), es crear varias sub-muestras aleatorias del conjunto de datos inicial con la posibilidad de duplicar muestras en los subconjuntos. Luego se calcula la media de cada sub-muestra y se realiza un promedio de estas medias. Este método es utilizado por el algoritmo agregación de bootstrap (*bootstrap agreggation*) o también llamado empa-

quetado (*bagging*). Este es un algoritmo de ensamble, dado que utiliza predicciones de múltiples algoritmos de aprendizaje automático para dar mejores predicciones. Este algoritmo se utiliza para reducir la varianza en aquellos algoritmos que poseen una alta varianza como CART.

Ahora, una vez que se tienen claras las definiciones anteriores, los bosques aleatorios son una mejora del algoritmo CART por medio del empaquetado. En la implementación usual de CART, cuando se selecciona el punto de división, el algoritmo se le permite ir a todos los atributos y a todos los valores de los atributos para poder seleccionar el mejor valor. En el caso de bosque aleatorio, este selección cambia, dado que al algoritmo se le limita a una muestra aleatoria de los datos para realizar la búsqueda. Esta cantidad de atributos que pueden ser buscados en cada punto de división debe ser especificado como parámetro del algoritmo, es representado por la letra m, y este es calculado por la fórmula $m = \sqrt{p}$, donde p es la cantidad de variables de entrada.

2.4.4. K-Vecino más cercano

El algoritmo K-Vecino más Cercano (K-*Nearest Neighbor*), es otro de los algoritmos más utilizados para algoritmos de aprendizaje automático supervisado. En este caso, como Brownlee (2018) apunta, este algoritmo utiliza todo el conjunto de datos como el modelo para realizar las predicciones, por lo que no requiere realizar ningún proceso de aprendizaje. Al momento de realizar predicciones, el algoritmo busca la cantidad k de muestras similares a las entradas de la nueva predicción. Para algoritmos de clasificación, como es el caso de esta investigación, se utiliza la clase del valor moda del conjunto de similitudes, el cual es el valor más frecuente en el conjunto.

2.4.5. Máquinas de vectores de soporte

Los algoritmos de máquinas de vectores de soporte (*Support Vector Machine* y *SVM* por sus siglas en inglés), son de los más populares y usados hoy en día, según Brownlee (2018). Este algoritmo utiliza un clasificador de margen máximo (*Maximal-Margin Classifier*), el cual forma un espacio dimensional de tamaña n con la cantidad de variables x de entrada. Además contiene un hiperplano (*hyperplane*), que divide el espacio de variables antes mencionado a la mitad.

En el caso de SVM, este hiperplano es seleccionado para separar los puntos en el espacio de variables para cada una de las clases o categorías que tiene el problema, Brownlee (2018). El autor ejemplifica que, en un caso donde hay dos categorías A y B, este hiperplano sería una linea, donde por encima de la línea pertence a la clase A y por debajo de la línea pertenece a la clase B.

2.4.6. Redes neuronales

Finalmente, otro algoritmo de aprendizaje supervisado que es muy utilizado son las redes neuronales, esta es una técnica de clasificación, que trata de asemejarse al cerebro humano y sistema nervioso en general, que está compuesto unidades de procesamiento independiente que se llaman neuronas. El enlace entre cada neurona tiene un peso que significa la importancia de cada entrada con la neurona, estas conexiones son llamadas conexiones de pesos. Según Mohammad et al. (2014), estas conexiones son ajustadas durante el entrenamiento de la red neuronal hasta alcanzar una solución aceptable.

Algunas ventajas de las redes neuronales son de acuerdo a Mohammad et al. (2014):

- No lineal: las redes neuronales son efectivas en modelar problemas de clasificación donde el valor esperado es muy distinto al utilizado como entrada.

- Adaptación: Posee la habilidad de cambiar los pesos de las conexiones basado en los cambios que suceden.
- Generalización: Capaz de encontrar la respuesta correcta a los problemas no vistos.
- Tolerancia a fallos: El rendimiento de la red neuronal no se ve afectada bajo circunstancias difíciles, como pérdida de conexión entre neuronas o datos no válidos.

Además, las redes neuronales pueden tener dos tipos de arquitectura, según Mohammad et al. (2014). Alimentación hacia adelante, en donde los datos se propagan únicamente en una dirección desde el punto de inicio hasta la salida. El otro tipo es la red neuronal recurrente, donde también existen conexiones de retroalimentación de las capas previas, por lo que se puede observar que el resultado de la red neuronal no sólo depende de las entradas sino también del estado previo de la red neuronal.

Asimismo, Mohammad et al. (2014) menciona que las redes neuronales poseen una topología, es decir, que se puede componer de distintas capas, así como el número de neuronas y las conexiones de pesos entre estas. Los tipos de capas están definidos por la capa de entrada, la capa oculta y la capa de salida. En la primera capa, se reciben todos los valores externos que se van procesar, las neuronas se denominan neuronas de entrada. La siguiente capa se denomina capa oculta, y sus neuronas se llaman neuronas ocultas. Estas neuronas reciben los valores de la capa de entrada y producen combinaciones no lineales para luego pasarlas a la siguiente capa para continuar el procesamiento. Si existe uno o más capas ocultas se le llama perceptrón multicapa, la cuál ayuda a que el número de neuronas puede ser cambiado para adaptarse a los cambios entre las capas de entrada y salida. Finalmente se tiene la capa de salida, donde a las neuronas se les llama neuronas de salida que representan la salida o respuesta de la red neuronal.

Luego de que la arquitecta y topología de la red neuronal artificial está lista, se debe proceder con el entrenamiento, el cual busca reducir el error en la capa de salida por medio del ajuste en las conexiones de pesos. Además, se pueden realizar dos tipos de entrenamiento el supervisado y no supervisado. El supervisado provee datos junto con el resultado esperado para cada uno de esos datos. En el segundo, solamente se proveen los datos sin ningún resultado, (Mohammad et al., 2014).

2.5. Otros

2.5.1. CANTINA

Existe otro algoritmo llamado CANTINA, el cual es una técnica de detección de sitios web falsos por medio de la frecuencia inversa de un documento basado en frecuencia de términos que aparece(*TF-IDF*), en donde se verifica el contenido del sitio web. TF-IDF, genera pesos por la importancia de cada palabra contando su frecuencia, según Mohammad et al. (2014).

2.5.2. Minería de datos

Además del aprendizaje automático, existen otro tipo de algoritmos llamados de minería de datos, que tratan de realizar búsquedas en grandes cantidades de datos para poder extraer información Aburrous et al. (2010).

3 Marco metodológico

3.1. Tipo de investigación

La presente investigación utilizará un enfoque constructivista, es decir, se irán construyendo entregables para así obtener un resultado final por medio de la metodología ciencia del diseño aplicado al contexto de sitios web en internet que son candidatos a ser de tipo *phishing*.

Esta metodología busca realizar la investigación por medio de la construcción y evaluación de artefactos, diseñados para que cumplan con las necesidades requeridas, Hevner, March, Park, y Ram (2004). Estos artefactos se pueden definir como los posibles entregables que esta investigación desarrollará.

Se compone de 7 pasos, así como Hevner et al. (2004) lo menciona. En primer lugar, se busca definir los artefactos de la investigación, luego como segundo paso se define la utilidad del artefacto, es decir, por qué es necesario para la investigación. En tercer lugar, se procede con la evaluación de los artefactos, es decir, qué métricas se utilizan para definir que un artefacto es valioso.

Como cuarto paso, según Hevner et al. (2004), se busca determinar porque los artefactos son relevantes así como la investigación, y qué otras contribuciones aporta la investigación al medio. El quinto paso define cómo se construye cada artefacto, brindando una representación formal y coherente de los mismos. El sexto paso, busca identificar las posibles soluciones para el problema planteado para la investigación.

Y finalmente, el séptimo paso busca expresar de manera coloquial los resultados de la investigación para que cualquier audiencia los pueda entender y hacer uso de ellos.

3.2. Ciencia del diseño aplicado

En las siguientes secciones, se describirán cada uno de los pasos mencionados anteriormente, ya enfocados en la presente investigación, con el fin de determinar los artefactos necesarios.

3.2.1. Paso 1: diseño del artefacto

Por la naturaleza de esta investigación, en donde se pretende utilizar algoritmos de aprendizaje automático para detección de sitios web *phishing*, y además que incluirá el desarrollo de un servicio web y una aplicación móvil, los siguientes son los artefactos a definir.

3.2.1.1. Atributos

Se definirán los atributos a utilizar para analizar los URLs, los cuales serán basados en la literatura utilizada. Además se tratarán de mejorar mientras se vaya realizando la investigación para que el modelo de predicción tenga mejor porcentaje de acierto.

3.2.1.2. Conjunto de datos

Recopilación de información para generar un conjunto de datos que se dividirá en un conjunto de datos de entrenamiento y un conjunto de datos de validación para realizar las pruebas necesarias a los algoritmos de aprendizaje automático supervisado que se investigarán. Este conjunto de datos de entrenamiento y validación utilizará el método de validación cruzada.

3.2.1.3. Algoritmo de aprendizaje automático supervisado

Se usará un algoritmo de aprendizaje automático supervisado para generar un modelo de predicción que determine si un URL es de tipo *phishing* o no. Este algoritmo será seleccionado de la lista de algoritmos escogidos para realizar las pruebas y poder determinar cuál es el que brinde el mayor porcentaje de acierto.

3.2.1.4. Modelo de predicción

Basado en el artefacto anterior, se utilizará el modelo generado por el algoritmo para poder predecir los URL que se alimenten vía el servicio web.

3.2.1.5. Diseño e implementación de un servicio web

Se diseñará e implementará un servicio web en el lenguaje de programación Python, el cual utilizará el artefacto mencionado anteriormente (modelo de predicción) para determinar si un URL es o no de tipo *phishing*.

3.2.1.6. Aplicación móvil

Se creará una aplicación móvil, que por medio del servicio web mencionado anteriormente, los usuarios tendrán fácil acceso para determinar si los URLs que se visitan pueden ser de tipo *phishing* o no. Esta aplicación móvil implementará un análisis del tráfico de red para determinar los URL que el usuario pretende abrir en su dispositivo móvil. Esto proporcionará al usuario un mecanismo de aviso cuando está visitando sitios web.

3.2.2. Paso 2: relevancia del problema

En primer lugar, la definición de los atributos a utilizar en los URLs, determina cuáles propiedades importantes son necesarias para poder determinar que un URL

sea tipo *phishing* o no. El conjunto de características o atributos a seleccionar serán basados en la revisión literaria que se ha hecho. Luego de que estas características sean definidas, se procederá con la creación de un conjunto de datos que será lo que alimentará el algoritmo de aprendizaje automático para así poder generar un modelo que realice las predicciones y determine si el URL analizado es tipo *phishing*.

Este conjunto de datos se generará por medio de un pequeño código automatizado que extraiga las características de cada URL en el conjunto de datos, así también como incluir el valor de la etiqueta esperado, dado que lo que se busca es utilizar un algoritmo de aprendizaje automático supervisado. Una vez que se tiene el conjunto de datos a utilizar, por medio de validación cruzada se generará un conjunto de datos de entrenamiento y un conjunto de datos de validación para poder comprender qué tan bien se comporta el modelo al momento de realizar la predicción. Cuando los conjuntos de datos de entrenamiento y validación ya estén creados, se procederán a realizar pruebas sobre los diferentes algoritmos que se van a escoger para así determinar cuál de todos posee el mejor rendimiento y valor de predicción mayor.

Cuando el algoritmo sea seleccionado se generará un modelo de predicción, el cual será utilizado para la implementación de un servicio web que permita recibir un URL y que devuelva un resultado ya sea positivo o negativo del mismo. Cuando el servicio web sea implementado, se desarrollará una aplicación móvil que utilice el servicio web para que sea de fácil acceso a los usuarios determinar si el URL es malicioso o no. Esta aplicación es la razón principal de la investigación en donde se pretende proteger a los usuarios de sitios web maliciosos que traten de abusar de la confianza de las personas para hacer robo de información personal y sensible, por medio del *phishing*.

3.2.3. Paso 3: evaluación del diseño

Para poder determinar el éxito de cada uno de los artefactos definidos, se determinarán ciertas métricas que se utilizarán de apoyo para definir si la función del artefacto es la más óptima, de acuerdo a las metodologías descritas por Hevner et al. (2004).

La escogencia de características ayudará a la creación de un conjunto de datos como se mencionó, pero para definir si serán los mejores atributos, las métricas a utilizar se dividirán en dos. La primera es utilizando el método experimental, Hevner et al. (2004), y se deberá esperar hasta que se inicie el proceso de pruebas de los algoritmos, para poder determinar si los atributos son determinantes. Además, parte de este método experimental será basado en la literatura leída para obtener los atributos más determinantes.

Cuando se tenga el conjunto de características definidas, se creará el conjunto de datos de entrenamiento y de validación. El determinar si el conjunto de datos de entrenamiento es el mejor lo determinará de igual forma los atributos escogidos en el paso anterior y las pruebas sobre los algoritmos. En este caso de igual forma se aplica el método experimental.

Una vez generados los conjuntos de entrenamiento y validación, se iniciará con las pruebas para determinar qué algoritmo y su respectivo modelo serán escogidos. Este proceso puede ser el más extenso dado que es un proceso de prueba y error, en donde iniciará probando un algoritmo a la vez, y observar los resultados que arroje. Se tomará en cuenta los atributos definidos en el conjunto de datos, así como los coeficientes utilizados por el algoritmo y poder observar el porcentaje de predicciones acertadas. Este proceso influye tanto en las métodos experimental y de pruebas basados en lo que Hevner et al. (2004) mencionan.

Este porcentaje junto con el rendimiento de cada algoritmo para tener el menor tiempo posible serán los factores claves en la escogencia del algoritmo. Cabe destacar

que en caso de que el porcentaje de acierto tenga muy buenos resultados comparado con otro que sea más veloz y tenga menor rendimiento, se decidirá utilizar el de mejores porcentajes de aciertos dado que lo que se busca es tener mejor seguridad y confianza en la predicción brindada por el algoritmo.

El servicio web debe tener la seguridad necesaria para que la aplicación móvil sea el único servicio que pueda conectarse y evitar que exista algún ataque como hombre en el medio que pueda influir en el resultado brindado por el servicio web. Para este artefacto, se utilizará el método analítico, mencionado por Hevner et al. (2004), por medio de la optimización del servicio web para que sea lo más rápido y seguro posible.

De igual forma la aplicación móvil debe poder enviar URLs constantes al servicio web sin ningún problema para poder asegurar el funcionamiento correcto de la misma. La aplicación debe contener alguna forma de que un usuario pueda ingresar un URL y este pueda ser iniciado. De igual forma, la aplicación debe contener un mecanismo de análisis continuo que permita analizar el flujo internet del dispositivo del usuario para evitar que ingrese a sitios de tipo *phishing* por medio de alguna alerta. Este artefacto incluirá tanto los métodos analíticos y de pruebas, de acuerdo con Hevner et al. (2004), para determinar que la aplicación funciona de la forma correcta, y esté optimizada para el uso eficiente de los usuarios.

3.2.4. Paso 4: contribuciones de la investigación

La investigación y sus productos finales tienen la intención de ayudar a las personas usuarias de la aplicación a poder navegar en el mundo del internet de una forma más segura tratando de evitar que ingresen a sitios de tipo Phising. Existen hoy en día diferentes algoritmos y modelos definidos para poder determinar si un sitio es de tipo *phishing*.

Esta investigación busca dar las herramientas necesarias para que usuarios que

no tienen conocimiento en temas de desarrollo o incluso de seguridad tengan fácil acceso a la seguridad y confianza que da el navegar de forma segura. Al desarrollar una aplicación móvil, ayudará a que los usuarios puedan comprender que existen peligros mientras se navega en el Internet, ya sea por medio de redes sociales, o simplemente por en URL que llego en el correo electrónico y que tengan la posibilidad de estar protegidos mientras lo utilizan su dispositivo móvil.

3.2.5. Paso 5: rigor en la investigación

El conjunto de atributos que se deciden extraer de los URLs a analizar y que serán las entradas de los algoritmos a probar, son obtenidos de la literatura leída que han demostrado que son eficientes y determinantes en la predicción de sitios web tipo *phishing*. Además se incluirá una fase de pruebas para así determinar en este caso específico que atributos pueden ser determinantes y cuáles pueden estar no contribuyendo a los conjunto de datos hecho, por lo que ocasionaría ruido en el conjunto.

Cuando el conjunto de datos esté creado, se procederá a realizar una validación cruzada de K-iteraciones. Como se mencionó anteriormente, este proceso busca dividir el conjunto de datos en un número especifico de subconjuntos de entrenamiento y de pruebas, los cuales se ejecutarán por varias iteraciones cambiando estos subconjuntos. Con esto se promedia el valor de predicción para determinar el rendimiento y porcentaje de acierto del mismo.

Una vez se tengan los conjuntos de datos de entrenamiento y validación se iniciará el proceso de pruebas con cada uno de los algoritmos de aprendizaje automático supervisado escogidos (la definición de estos algoritmos está establecida en la siguiente sección). Este será un proceso de prueba y error en donde se empezará a probar uno por uno los algoritmos, modificando los coeficientes y variables de cada algoritmo para así determinar el mayor porcentaje de predicción posible por cada uno de estos algoritmos. Una vez que se tengan estos valores, se escogerá el algoritmo y modelo

generado que tenga el mayor porcentaje de predicción.

El servicio web será creado usando el lenguaje de programación Python, junto con la biblioteca Flask. El servicio web recibirá como único parámetro el URL a analizar. Luego de que el servicio web por medio del modelo generado anteriormente prediga si el URL es tipo *phishing* o no, retornará una respuesta con el valor proporcionado.

Finalmente, la aplicación móvil, será desarrollada en el ecosistema Apple por medio del sistema operativo iOS. Utilizará una biblioteca llamada *Network Extension* para poder hacer un análisis del tráfico de red y poder consumir el servicio web mencionado anteriormente.

3.2.6. Paso 6: diseño como parte del proceso de búsqueda

Esta investigación presenta el problema de poder determinar si un URL dado puede ser tipo *phishing* o no. Al tratarse de un problema de categorización de URLs, en el aprendizaje automático se trata de un problema de clasificación. Basado en la literatura leída, se procederá a probar los siguientes algoritmos de aprendizaje automático supervisado:

- Regresión logística
- Árbol de decisión
- Bosque aleatorio
- K-Vecinos cercanos
- Máquinas de vectores de soporte
- Redes neuronales

Por cada uno de los algoritmos listados anteriormente, se hará el proceso de entrenamiento correspondiente para así generar el modelo de predicción y poder determinar aquel que arroje el mejor resultado de acierto. Cabe destacar, como se mencionó

anteriormente, que se escogerá el algoritmo con el mejor porcentaje de acierto, dado que lo importante es la confianza que este pueda dar con el tipo de conjuntos de datos y atributos extraídos.

3.2.7. Paso 7: comunicación de la investigación

Durante el desarrollo de esta investigación se describirá de la forma más sencilla posible cada uno de los pasos que se tomó para llevar a cabo los productos finales. Para la definición de cada uno de los atributos a utilizar se explicará porqué este es un atributo importante para determinar si un URL puede ser *phishing* o no.

Para la definición del conjunto de datos únicamente se usará la validación cruzada para tener el conjunto de entrenamiento a utilizar. Para el algoritmo a utilizar ya la investigación habrá mencionado cómo funciona, y se brindarán los resultados que se vieron al momento de hacer las pruebas para determinar su porcentaje de acierto con los atributos utilizados.

Tanto para el servicio web como para la aplicación móvil, se adjuntará el código fuente para su análisis y lectura en caso de ser necesario. Además el servicio web estará expuesto en la nube que será el utilizado por la aplicación móvil para poder brindar la funcionalidad de detectar si el URL es *phishing* o no.

Así que por medio de la aplicación móvil, cada uno de los artefactos mencionados estarán unidos en un solo lugar. Por medio de los atributos seleccionados que serán extraídos de los URL, y la creación de un conjunto de datos, se podrá utilizar un algoritmo que genere un modelo de predicción. Este será utilizado por el servicio web para así determinar la categoría del URL que finalmente se desplegará de forma amigable al usuario para mostrarle que un sitio web puede ser malicioso.

4 Desarrollo y resultados

En esta investigación se definieron ciertos artefactos por medio de la metodología de ciencia del diseño. Estos artefactos se van a presentar a continuación, donde explicará cómo se consiguieron o se desarrollaron para finalmente mostrar los resultados finales de estos.

4.1. Atributos

Como primer artefacto se definió la creación de atributos que se deben extraer de los URL para realizar un conjunto de datos necesario para el entrenamiento, cada uno de estos atributos será definido por tres valores. El -1 identificará que el atributo identificó el URL como un URL no malicioso. El valor 0 identificará que el URL es sospechoso, es decir, puede o no puede ser de tipo phishing. Y finalmente, el 1 determina que el URL es de tipo *phishing*.

Una vez que se definan cada uno de los atributos se creará un conjunto de características. Utilizadas en conjunto con las URL, se desarrollará finalmente un conjunto de datos a utilizar para entrenar el algoritmo de aprendizaje automático. A continuación, se describirá cada uno de ellos.

1. El primer atributo a identificar es si el URL es una dirección electrónica o está definido como una IP. Cuando el URL contiene una IP dentro de sí mismo, quiere decir que la IP es más propensa a ser de tipo *phishing* dado que el ata-

cante trata siempre de generar un sitio falso y por corto tiempo. El valor es de 1 si el URL contiene un URL, -1 de otra forma.

2. Se obtiene el largo el del URL. Usualmente, los atacantes según Weiss y Khoshgoftaar (2017) tratan de tener URL muy largos para confundir a sus víctimas, por lo que, si el URL posee un largo de más de 75 caracteres, el valor será 1. Si el URL es menor o igual a 75 y mayor o igual a 54 el valor asignado será 0. Y finalmente si el valor es menor a 54 el valor será -1.

3. Si un URL utiliza un servicio para acortarlo por ejemplo utiliza el servicio de Google para acortar el URL e inicia como https://goo.gl. Este tipo de servicios ayuda a los atacantes a evitar mostrar completamente un URL, de acuerdo a Weiss y Khoshgoftaar (2017). Si el URL posee este servicio de alguno de los proveedores más conocidos obtendrá un valor de 1, en caso contrario tendrá un valor de -1. La lista de proveedores que se tendrán en cuenta son: goo.gl, bit.ly, tinyurl.com, lc.cha, is.gd, soo.gd, s2r.co, t.co, youtu.be, ow.ly, adf.ly, bit.do, cutt.ly, buff.ly, mcaf.ee, su.pr, bud.url y moourl.com.

4. Cuando un URL contiene el símbolo @, el explorador web ignorará todo lo que está antes del símbolo. Por lo que, según Mohammad y Thabtah (2012), los atacantes tratarán de ocultar el URL del tipo *phishing* después de este símbolo. Si este está el valor del atributo será 1, sino será -1.

5. En caso de que un URL contenga el símbolo "//" el valor será de 1 si está y de -1 si no lo está. Este símbolo es utilizado por los atacantes para realizar un redireccionamieto hacia otro sitio, Weiss y Khoshgoftaar (2017).

6. Muchos atacantes utilizan el guión (-), según Mohammad et al. (2014), como sufijos o prefijos para hacer creer a la víctima que son los sitios válidos. Si existe el símbolo, el valor asignado es -1 y en caso contrario 1.

7. Usualmente los URL poseen un solo subdominio, Weiss y Khoshgoftaar (2017). Por ejemplo, puede ser "www". Por lo que si el URL que solo tiene un subdominio, será asignado un -1, en caso de que tenga dos subdominios será considerado sospechoso y se le asignará un 0 y si posee más de dos será considerado malicioso y se le asignará 1.

8. Al utilizar SSL para realizar conexiones seguras un sitio usualmente estará protegido, y se asume que se está visitando un URL legítimo, Weiss y Khoshgoftaar (2017). Si se cuenta con "https", el valor definido es -1, en caso contrario 1.

9. Normalmente los URL poseen un puerto ya definido y no es especificado. En caso de que es especificado, el atacante pretende redirigir a otra parte de la misma dirección que pudo ser comprometida. En este caso si el URL posee un puerto será considerado 1, y -1 en caso contrario.

10. Según Weiss y Khoshgoftaar (2017), páginas legítimas no utilizan la función *mailto* o *mail()* para enviar información. Por lo que si el contenido de un URL tiene esa función se les asignará 1, y -1 en caso contrario.

11. Al analizar el contenido del URL, si está vacío, según Weiss y Khoshgoftaar (2017), el URL será caracterizado como malicioso con un valor de 1, y -1 en caso contrario.

12. Si existe redirección dentro del URL y si esta es menor o igual a 1, el valor será -1. En caso de que la cantidad de redirecciones es mayor a 1 y menor o igual 4 el valor será sospechoso con un valor de 0. Y, finalmente, si el valor es mayor a 4 es considerado phishing con un valor de 1.

13. Dentro de la mayoría de las páginas web utilizan JavaScript, pero los atacantes lo utilizan para ocultar los URL falsos a imágenes o texto dentro del contenido

web, según Mohammad et al. (2014). En caso de que se encuentre este método dentro del código fuente del sitio web el valor asignado es 1, en caso contrario -1.

14. De acuerdo con Mohammad et al. (2014), algunos de los atacantes utilizan JavaScript para deshabilitar la función del clic derecho, para que así los usuarios no puedan acceder el código fuente del sitio. En caso de que se encuentre deshabilitado se le asignará un valor de 1 y en caso contrario se le asignará un valor de -1.

15. No es muy usual que los sitios web presenten alertas por credenciales directamente, por lo que este comportamiento hace que un URL se le asigna un valor de 1, y en caso de contrario 1.

16. Los atributos *iFrame* no son tan comunes y los atacantes tratan de utilizarlos para engañar a los usuarios. Si estos atributos se encuentran en el contenido web se les asigna el valor 1, y en caso contrario -1.

17. Las páginas web contienen las etiquetas de anclaje (*anchor tag*), las cuales apuntan a otro recurso que está dentro del mismo dominio. Si existen etiquetas de anclaje que apuntan a otro dominio, según Mohammad et al. (2014), se considera de tipo phishing. Por lo que, si no existen etiquetas que apuntan a otros dominios distintos al original, se le asigna un valor de -1, si esta cantidad es menor o igual a dos se considera sospechoso y si es mayor a dos se le asignará un valor de 1.

18. Un tag de anclaje en HTML es especificado por $<a>$. Según Weiss y Khoshgoftaar (2017), el porcentaje de que un anclaje apunte a un dominio distinto al dominio del sitio web define que tan probable es que el sitio web sea de tipo phishing o no. Si el valor de este porcentaje es menor a 31 % se le asignará -1.

Si el porcentaje es mayor o igual a 31 % o menor o igual a 67 % se le asignará 0. Finalmente, si el porcentaje es mayor a 67 % se le asignará 1.

19. De acuerdo con Weiss y Khoshgoftaar (2017), los sitios legítimos solicitan imágenes desde el mismo dominio del sitio web. Si el porcentaje de solicitudes son hechas de diferentes dominios es menor a 22 %, se le asignará un valor de -1. Si el porcentaje es mayor o igual a 22 % o menor o igual a 61 % se le asignará un valor de 0. Finalmente, si el porcentaje es mayor a 61 % tendrá un valor de 1.

20. Según Weiss y Khoshgoftaar (2017), el sitio web de tipo *phishing*, tiene un rango de vida pequeño por lo que el nombre de dominio es registrado por un mínimo de tiempo. Por lo que si el nombre de dominio expira en menos de un año se le asignará un valor de 1, en caso contrario -1.

21. En caso de que el dominio es menor a 6 meses se le asignará un valor de 1, según Weiss y Khoshgoftaar (2017) dado que los sitios web son creados recientemente. En caso contrario se le asignará un -1.

22. Un sitio web genera poco tráfico de acuerdo a Weiss y Khoshgoftaar (2017). Si el tráfico del sitio se encuentra en los primeros 100000 sitios, se le asignará un valor de -1. Si el sitio no está dentro de los primeros 100000, pero igual está dentro del ranking se le asignará un valor de 0. Y en caso de que no se encuentre dentro del ranking se le asignará un valor de 1.

23. La mayoría de los sitios web se encuentran indexados por Google, de acuerdo con Weiss y Khoshgoftaar (2017). Si un sitio web está indexado, se le asignará un valor de -1, en caso contrario se asignará un 1.

24. Según Mohammad et al. (2014), si el nombre de dominio contiene registros de DNS se le asignará un valor de -1. En caso contrario se le asignará 1.

4.2. Conjunto de datos

Una vez definidos los atributos que se utilizarán para analizar los URL, se inicia la creación de un conjunto de datos basado en URLs. Es decir, se ejecutará el análisis de atributos en cada uno de los URLs obtenidos.

Para obtener URL maliciosos, se utiliza el sitio OpenPhish, el cual posee un servicio web que permite descargar la base de datos más reciente de sitios maliciosos que han sido reportados por usuarios. En total se descargaron 12572 registros para extraer las características y obtener el conjunto de datos de sitios tipo *phishing*.

Para obtener los URL válidos o legítimos se obtuvieron del proyecto basado en el trabajo de CSIRT Gadgets (2018, [accedido el 06-14-2019]), el cual posee un conjunto de datos ya definido. Pero dado que los sitios web maliciosos duran poco tiempo vivos como se explicó en la sección anterior, únicamente se decidió tomar los sitios válidos dado que estos se mantienen aún en la nube. Se extrajeron en total 15055 registros de URL legítimos para extraer los atributos.

Una vez extraídos los atributos de URL maliciosos y legítimos, se procede a mezclarlos de forma aleatoria para que la muestra no se encuentre segmentada. Una vez que se obtienen los datos, se inicia con el proceso de pruebas de algoritmos. En total se realizaron pruebas en 6 algoritmos de aprendizaje automático supervisado y se analizaron los resultados de los mismos. Cada uno de estos algoritmos fue creado por medio de la biblioteca *scikit-learn* del lenguage de programación Python.

4.3. Algoritmo de aprendizaje automático supervisado

En la tabla 15 se pueden encontrar las pruebas realizadas a los diferentes algoritmos que se probaron, así como los porcentajes de acierto de los datos de validación,

dado que este es utilizado por los algoritmos para determinar que tan buen resultado tiene con datos no conocidos.

Tabla 15: Pruebas de algoritmos de aprendizaje automático supervisado. Fuente: creación propia (2019)

	Ejecución 1	*Ejecución 2*	*Ejecución 3*	
Algoritmo	**% Acierto**	**% Acierto**	**% Acierto**	**Promedio**
Árbol de Decisión	89,93	89,87	89,95	**89,71**
Bosque Aleatorio	90,36	90,08	89,75	**90,06**
Regresión Logística	83,65	83,28	83,88	**83,60**
K-Vecino más Cercano	88,47	87,91	88,59	**88,32**
Máquinas de Vectores de Soporte	43,11	42,41	43,22	**42,91**
Redes Neuronales - Perceptrón Multicapa	90,30	90,42	90,74	**90,49**

4.4. Modelo de predicción

Dado que cada vez que se ejecutan las predicciones por cada modelo se genera un nuevo conjunto de datos de entrenamiento y validación, se decide realizar tres ejecuciones para determinar el porcentaje más alto. Basado en esta condición, se puede notar que el algoritmo de red neuronal - Perceptrón Multicapa, posee el mayor promedio para el porcentaje de acierto de los algoritmos probados con un total de 90,49 %, con lo que se procede a generar el modelo.

Para la generación del modelo se utilizó la biblioteca *joblib* perteneciente al lenguaje Python, que genera un archivo para el modelo del algoritmo escogido, el cual puede ser utilizado en el futuro para realizar predicciones cuando sea necesario.

4.5. Diseño e implementación de un servicio web

Al tener el modelo de predicción generado se procede con la creación del servicio web. El lenguaje de programación a utilizar es Python, por medio de la biblioteca

Flask la cual permite crear servicios web de forma fácil y sencilla. En una tapa inicial se procede a crear un servicio web local en la computadora personal para poder realizar todas las pruebas necesarias antes de publicarlo en la nube.

Una vez finalizada la etapa de desarrollo se procede a crear un ambiente en la nube por medio de Amazon Web Services (AWS, por sus siglas en inglés) y la tecnología *Elastic Beanstalk*, la cual permite crear ambientes para servicios web de forma fácil, sin tener que preocuparse tanto por la infraestructura interna del servidor. Una vez, subido el código fuente de la aplicación Flask, junto con el modelo y el código fuente para la extracción de atributos, el sitio se accede por medio de Gomez (2019b).

La aplicación en Flask, contiene dos posibles rutas. La primera se relaciona al inicio de página donde se muestra el nombre del sitio, creador, y la razón por la que fue creado. Esta ruta puede ser accedida con solo acceder al URL mencionado anteriormente en cualquier explorador web. La segunda ruta corresponde a la ruta */predict*, el cual utiliza el método **POST** de un servicio web. Esta ruta acepta un cuerpo de mensaje en lenguaje JSON mostrado en el listado 1. Además se deben enviar dos encabezados específicos para consumir el servicio web en formato JSON. El código fuente puede ser encontrado en Gomez (2019a).

Listado 1: Cuerpo de mensaje

```
{
    "url":"http://www.google.com"
}
```

El primer encabezado corresponde a *Content-Type* con el valor *application/json*. Esto quiere decir que se le está enviando al servicio web aceptará dentro del cuerpo del mensaje un mensaje en formato JSON. El segundo encabezado, corresponde a *Accept*, con valor *application/json*. Este encabezado hace saber al servicio web que el valor esperado de retorno debe estar en formato JSON. Este encabezado no es tan

necesario, dado que la aplicación web ya está retornando el valor en JSON, pero es mejor configurarlo para evitar algún problema al retornar la respuesta.

Finalmente, una vez que se tiene configurado la solicitud al servidor y se envía el URL que se quiere analizar, el servicio web retorna en caso de éxito una respuesta similar mostrada en el listado 2.

Listado 2: Ejemplo de mensaje de éxito

```
{
    "prediction": "-1"
}
```

4.6. Aplicación móvil

Una vez listo el servicio web, se procede con la creación de la aplicación móvil. Esta aplicación será creada en el ambiente de Apple, en la plataforma iOS por medio del lenguage de programación Swift. La aplicación esta compuesta por 4 partes que se puede encontrar en el repositorio Gomez (2019a). La primera parte es la aplicación como tal, la cual contiene el código fuente y el diseño de la misma. La aplicación será llamada ***CatchPhish***. Las vistas están conformadas por una vista que permite ingresar un URL para determinar si es o no de tipo *phishing*, la vista se puede observar en la imagen 3.

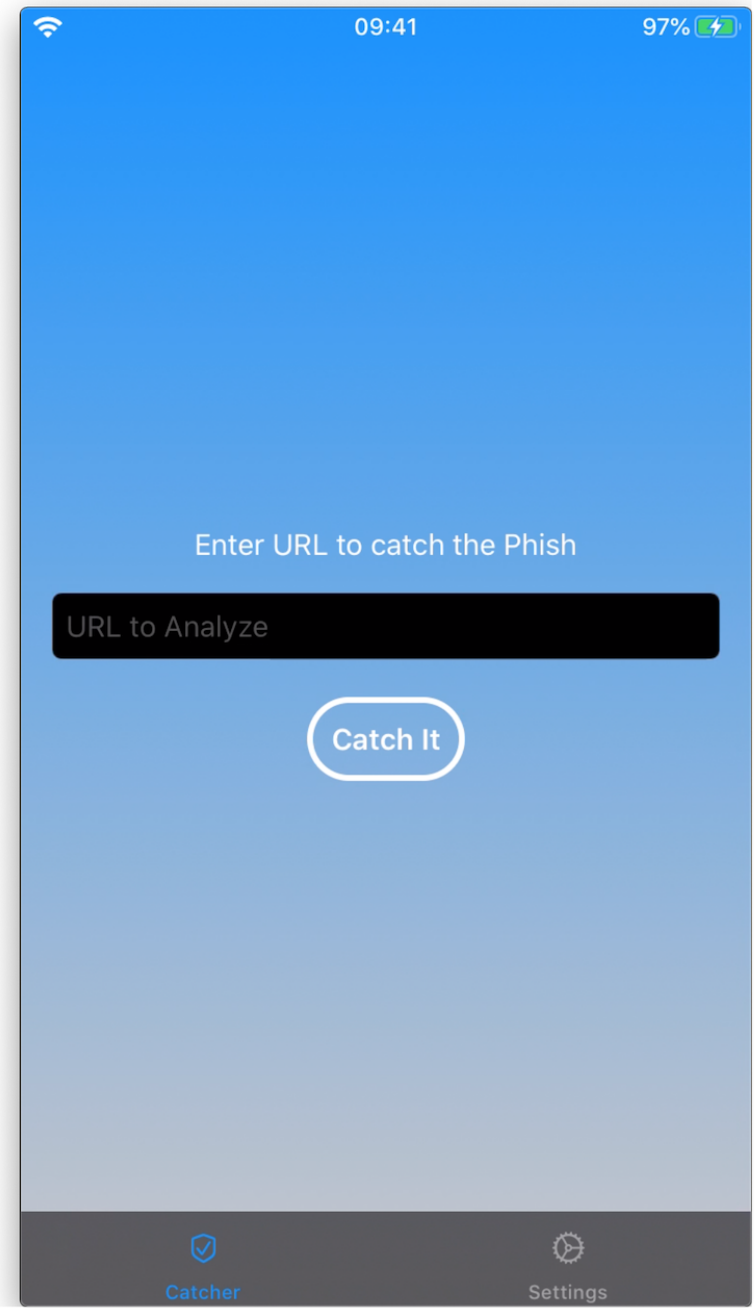

Figura 3. Vista para determinar si URL es *phishing* o no

La segunda vista tiene una configuración que permite a la aplicación tomar el control del tráfico del dispositivo, es decir, que permite analizar el tráfico que el dispositivo genera. Este análisis será realizado para que cada URL que se está tratando

de contactar sea enviado al servicio web creado anteriormente para así determinar si el URL es de tipo *phishing* o no. La vista se puede observar en la figura 4.

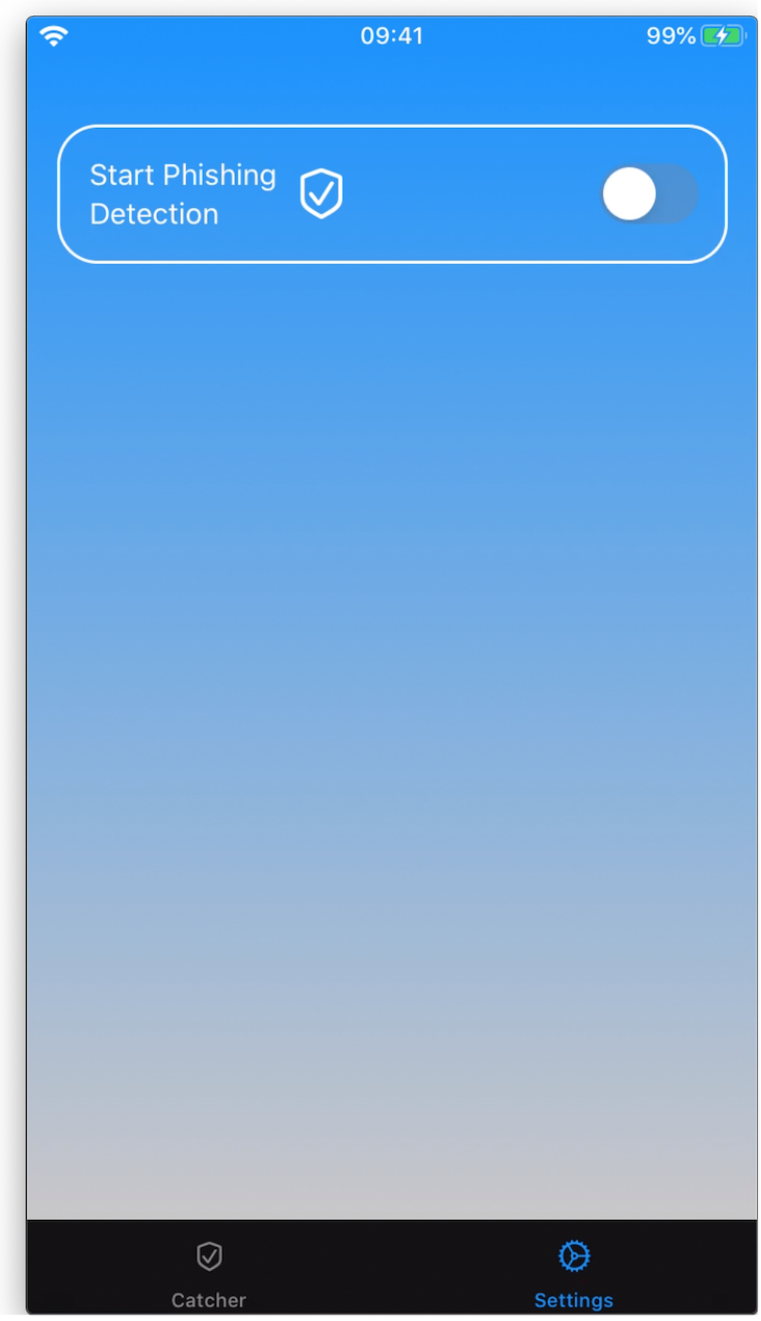

Figura 4. Vista para habilitar la detección de *phishing*

La segunda y tercera parte de la aplicación están compuestas por el análisis y fil-

trado de tráfico, el cual se realizará por medio de la biblioteca *Network Extension*, Apple Inc. (2019, [accedido el 06-09-2019]). Esta biblioteca contiene las clases y funciones necesarias para poder analizar y filtrar el tráfico entre otras funcionalidades. Estas clases crean un objeto de tipo *NEFilterFlow* el cual corresponde a una conexión abierta por una aplicación en el dispositivo. Este objeto contiene el URL que se está tratando de contactar, el cual va a ser utilizado para enviarlo al servicio web para su análisis.

Una vez que se obtiene la respuesta de si el sitio web es legítimo o de tipo *phishing*, se le notificará al usuario por medio de una notificación en el dispositivo para explicarle que el sitio que se está navegando es maligno y no se recomienda continuar visitándolo. En caso de que el URL sea legítimo no se mostrará ningún mensaje para evitar molestar al usuario con muchas notificaciones.

Finalmente, como una característica extra, es la activación y desactivación por medio de la voz del análisis de tráfico. Esta característica se realiza por medio de la biblioteca *SiriKit*, que permite agregar la posibilidad de que ciertas características puedan ser accedidas por medio de la voz, Apple Inc. (2019, [accedido el 07-09-2019]). Los comandos a utilizar están el idioma inglés, por lo que para activar el análisis de tráfico se debe decir *start phishing detection* y para poder desactivar el análisis de tráfico se debe decir *stop phishing detection*.

Cabe resaltar, que dado que la palabra *phishing* es muy difícil que sea detectada por el reconocimiento de voz de Apple, se tuvo que utilizar la palabra *fishing* (que significa pesca, y como se sabe el término *phishing* proviene del mismo). Así que cuando el texto sea identificado será mostrado como *start fishing detection*.

Se debe tomar en cuenta que la aplicación únicamente trabajará en dispositivos supervisados, como se especifico en las limitaciones de esta investigación (1.6.2). Es decir, son aquellos dispositivos que son administrados por alguna escuela o negocio que desee controlar y filtrar el contenido de estos, Apple Inc. (2019b). Ya sea

por ejemplo limitar a los usuarios instalar cierto tipo de aplicaciones o simplemente filtrar el contenido web del dispositivo. En este caso se está teniendo ventaja del filtrado del contenido para obtener el tráfico del dispositivo.

Se debe anotar que la aplicación puede ser utilizada en modo de desarrollo para probar de que funciona, pero si se desea dar al público para que la pueda descargar, no es posible hasta que Apple decida que la biblioteca *Network Extension* puede ser utilizada en dispositivos que no son supervisados.

5 Conclusiones

- Se lograron definir un total de 24 atributos basados tanto en análisis estático del URL, análisis contextual del contenido del sitio web, así como en referencias basadas en agentes externos, como por ejemplo para determinar si el sitio ha sido indexado por Google y que se muestran en más detalle en la sección 4.1.

- Se recolectaron un total de 27627 registros para la creación del conjunto de datos, a los cuales corresponde 15055 como registros de URL válidos y 12572 como registros de URL maliciosos. Este conjunto de datos luego de ser generado se mezcló para que la muestra sea aleatoria y no se encuentre segmentada.

- Se analizó la capacidad predictiva de varios algoritmos (se puede ver la tabla 15 como referencia a los algoritmos utilizados) con el conjunto de datos para determinar el de mejor porcentaje de acierto, concluyendo que el algoritmo de Red Neuronal - Perceptrón Multicapa fue el mejor con un promedio de acierto de 90,49 %.

- Se desarrolló un servicio web en el lenguaje de programación Python, por medio de la biblioteca *Flask*, que está localizado en la nube utilizando tecnología *Elastic Beanstalk* de AWS. El URL del servicio web está dado por Gomez (2019b).

- Se creó una aplicación móvil que permite realizar un análisis de URL para determinar si este es de tipo Phishing o no, el cual puede ser por medio de la vista para análisis individual de un URL específico, o ya sea activando el análisis de

tráfico en el dispositivo móvil. Además, se permite la activación y desactivación del análisis de tráfico por medio de la voz mediante los comandos *start phishing detection* y *stop phishing detection*, respectivamente.

6 Recomendaciones

- Creación de una aplicación en el ambiente de desarrollo Android, para que los usuarios puedan obtener los beneficios de detección de Phishing en una gama más amplia de dispositivos móviles presentes en el mercado actual.
- Utilización de algoritmos de aprendizaje automático no supervisado para la comprobación del aumento en el porcentaje de acierto al utilizar estos algoritmos, así como otros métodos de clasificación como la programación dinámica, que ampliarían los soluciones para que el porcentaje de acierto sea mayor al adquirido en este trabajo de investigación.
- Al momento en que Apple autorice la utilización de la biblioteca *NetworkExtension* en dispositivos no supervisados, liberar la aplicación al público para hacer un análisis y monitoreo de resultados a mayor escala, así como el rendimiento de los servicios implementados.

Lista de acrónimos

IEEE: Institute of Electrical and Electronics Engineers.

URL: Uniform Resource Locator.

SaaS: Software as a service.

DNS: Domain Name System.

TF-IFM: Term Frequency - Inverse Document Frecuency.

CANTINA: Carniege Mellon Anti-phishing and Network Analysis Tool.

SLD: Second Level Domain.

WFLE: Weighted Feature Line Embedding.

NN: Nearest Neighbour.

SMOTE: Systhetic Minority Over-Sampling Technique.

RF: Random Forest.

SVM: Support Vector Machines.

VIF: Variability Inflation Factor.

CART: Classification and Regression Trees.

Referencias

Abbasi, A., Zahedi, F. M., Zeng, D., Chen, Y., Chen, H., y Nunamaker, J. F. (2015). Enhancing predictive analytics for anti-phishing by exploiting website genre information. *Journal of Management Information Systems*, *31*(4), 109–157. doi: 10.1080/07421222.2014.1001260

Aburrous, M., Hossain, M. A., Dahal, K., y Thabtah, F. (2010). Associative classification techniques for predicting e-banking phishing websites. *MCIT'2010: International Conference on Multimedia Computing and Information Technology*, 9–12. doi: 10.1109/MCIT.2010.5444840

Ahsan, M., Gomes, R., y Denton, A. (2018). SMOTE Implementation on Phishing Data to Enhance Cybersecurity. *IEEE International Conference on Electro Information Technology*, 531–536. doi: 10.1109/EIT.2018.8500086

Amazon Web Services. (2019, [accedido el 01-10-2019]). *Deploying a flask application to elastic beanstalk.* Descargado de `https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html`

Apple Inc. (2019a). *Content filter providers.* Descargado de `https://developer.apple.com/documentation/networkextension/content_filter_providers`

Apple Inc. (2019b). *Get started with a supervised iphone, ipad, or ipod touch.* Descargado de `https://support.apple.com/en-us/HT202837`

Apple Inc. (2019, [accedido el 06-09-2019]). *Networkextension.* Descargado de https://developer.apple.com/documentation/networkextension

Apple Inc. (2019, [accedido el 07-09-2019]). *Sirikit.* Descargado de https://developer.apple.com/documentation/sirikit

Apwg.org. (2018). Phishing Activity Trends Report 4th Quarter. *Most*, 1–12. Descargado de http://docs.apwg.org/reports/apwg{_}trends{_}report{_}q2{_}2014.pdf

Brownlee, J. (2018). *Master Machine Learning Algorithms* (.13 ed.). Melbourne. (eBook)

Chawla, N., Bowyer, K., Hall, L., y Kegelmeyer, P. (200w). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 321–357. doi: 10.1613/jair.953

Chiew, K. L., Choo, J. S.-F., Sze, S. N., y Yong, K. S. C. (2018). Leverage Website Favicon to Detect Phishing Websites. *Security and Communication Networks*, *2018*, 1–11. doi: 10.1155/2018/7251750

CSIRT Gadgets. (2018, [accedido el 06-14-2019]). *Deep learning using tensor flow.* Descargado de https://www.github.com/csirtgadgets/tf-phishing-example

Desarrolladores de scikit-learn. (2007-2019, [accedido el 01-09-2019]). *scikit-learn.* Descargado de https://scikit-learn.org/stable/

Elmassry, M. Y., y Almogren, A. S. (2016). A mobile sensing method to counteract social media website impersonation. *International Journal of Distributed Sensor Networks*, *12*(10). doi: 10.1177/1550147716671265

Freeman, I., Haigler, A., Schmeelk, S., Ellrodt, L., y Fields, T. (2019). What are they Researching? Examining Industry-Based Doctoral Dissertation Research through the Lens of Machine Learning. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, 1338–1340. doi:

10.1109/ICMLA.2018.00217

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media. doi: 10.3389/fninf.2014.00014

Gomez, C. (2019a). *Código fuente.* Descargado de `https://github.com/cuyogomez/thesis-catch-phish`

Gomez, C. (2019b). *Url a servicio web implementado.* Descargado de `http://tesis-env.qhprpekfhi.us-east-2.elasticbeanstalk.com`

Gowri, R. (2017). An Efficient Algorithm To Identify Phishing Sites Using Url Domain Features. *International Journal of Advanced Research in Computer Science*, *8*(7), 508–510. doi: 10.26483/ijarcs.v8i7.4309

Hastie, T., Tibshirani, R., y Friedman, J. (2008). *The Elements of Statistical Learning Data* (2.ª ed.). California: Springer.

Hevner, A., March, S., Park, J., y Ram, S. (2004). Design Science in Information System Research. *MIS Quaterly*, *28*(1), 75–105. doi: 10.1021/jm991076c

Imani, M., y Montazer, G. A. (2017). Phishing Website Detection Using Weighted Feature Line Embedding. *Electrical Engineering*, *9*(2), 49–61.

Khonji, M., Iraqi, Y., y Jones, A. (2011). Lexical URL analysis for discriminating phishing and legitimate websites. En *2011 International Conference for Internet Technology and Secured Transactions.* Abu Dhabi. doi: 10.1145/2030376.2030389

Mohammad, R., y Thabtah, F. (2012). An assessment of features related to phishing websites using an automated technique. En *2012 International Conference for Internet Technology and Secured Transactions* (pp. 492–497). IEEE.

Mohammad, R., Thabtah, F., y McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, *25*(2), 443–458. doi: 10.1007/s00521-013-1490-z

Mourtaji, Y., Bouhorma, M., y Alghazzawi. (2017). Perception of a new framework for detecting phishing web pages. En *SCAMS '17 Proceedings of the*

Mediterranean Symposium on Smart City Application (pp. 1–6). Tangier. doi: 10.1145/3175628.3175633

OpenPhish. (2019, [Web; accedido el 06-14-2019]). *Conjunto de datos de urls tipo phishing*. Descargado de `https://www.openphish.com`

Tyagi, I., Shad, J., Sharma, S., Gaur, S., y Kaur, G. (2018). A Novel Machine Learning Approach to Detect Phishing Websites. *2018 5th International Conference on Signal Processing and Integrated Networks, SPIN 2018*, 425–430. doi: 10.1109/SPIN.2018.8474040

Weiss, K. R., y Khoshgoftaar, T. M. (2017). Detection of Phishing Webpages Using Heterogeneous Transfer Learning. *Proceedings - 2017 IEEE 3rd International Conference on Collaboration and Internet Computing, CIC 2017*, *2017-Janua*, 190–197. doi: 10.1109/CIC.2017.00034

Whitty, M., Edwards, M., Levi, M., Peersman, C., Rashid, A., Sasse, A., … Stringhini, G. (2017). Ethical and Social Challenges with developing Automated Methods to Detect and Warn potential victims of Mass-marketing Fraud (MMF). En *WWW '17 Companion Proceedings of the 26th International Conference on World Wide Web Companion* (pp. 1311–1314). Perth. doi: 10.1145/3041021.3053891

Zhuang, W., Jiang, Q., y Xiong, T. (2012). An intelligent anti-phishing strategy model for phishing website detection. *Proceedings - 32nd IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, 51–56. doi: 10.1109/ICDCSW.2012.66

Zuhair, H., Selamat, A., y Salleh, M. (2016). New Hybrid Features for Phish Website Prediction. *International Journal of Advances in Soft Computing & Its Applications*, *8*(1), 28–43.