Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN

Rodrigo Pineda Arévalo

Asesorado por el Ing. Erick Gabriel Maldonado Ramón

Guatemala, noviembre de 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN.

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

RODRIGO PINEDA ARÉVALO
ASESORADO POR EL ING. ERICK GABRIEL MALDONADO RAMÓN

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Óscar Alejando Paz Campos
EXAMINADORA	Inga. Mirna Ivonne Aldana
EXAMINADOR	Ing. José Alfredo Gonzáles
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN.

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 2 de mayo de 2019.

Rodrigo Pineda Arévalo

Guatemala, 13 de agosto de 2019

Ingeniero
Carlos Alfredo Azurdia
Coordinador de Privados y Revisor de Trabajos de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

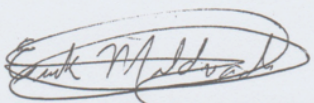
Ingeniero Azurdia:

Tengo el agrado de dirigirme a usted para informarle que he revisado el trabajo de graduación **"COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN.",** realizado por el estudiante universitario **RODRIGO PINEDA ARÉVALO** con carné **201020540** y CUI **2060 13132 0101**, quien contó con asesoría del suscrito.

Considero que el trabajo de graduación realizado por el estudiante cumple con los objetivos bajo las cuales fue planteado y cumple satisfactoriamente cada una de las actividades planificadas, por lo que procedo a aprobarlo.

Agradeciendo la atención dada a la presente.

Atentamente,



Erick Gabriel Maldonado Ramón
Ing. en Ciencias y Sistemas
Colegiado No. 13739

Ing. Erick Gabriel Maldonado Ramón
Asesor
Colegiado 13739

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 4 de septiembre de 2019

Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **RODRIGO PINEDA ARÉVALO** con carné **201020540** y **CUI 2060 13132 0101** titulado **"COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo aprobado.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

USAC · FACULTAD DE INGENIERÍA EN CIENCIAS Y SISTEMAS · COORDINACION TESIS

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN",*** *realizado por el estudiante, RODRIGO PINEDA ARÉVALO aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENS

Msc. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 12 de octubre de 2021

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

DTG. 634.2021

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **COMPARACIÓN EN TÉRMINOS DE PRECISIÓN PREDICTIVA DE TRES ALGORITMOS DE APRENDIZAJE SUPERVISADO APLICADO A LA DETECCIÓN DE SITIOS PHISHING HACIENDO USO DE LA LIBRERÍA SCIKIT-LEARN,** presentado por el estudiante universitario: **Rodrigo Pineda Arévalo,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Inga. Anabela Cordova Estrada
Decana

Guatemala, noviembre de 2021

AACE/asga

ACTO QUE DEDICO A:

Mis padres	Heidi Arévalo y Jaime Pineda, por siempre brindarme motivación, apoyo y sobre todo amor para poder alcanzar este logro.
Mi hermano	Oswaldo Pineda, por siempre creer en mí y estar presente en todo momento de mi vida.
Mis amigos	Pablo Calderón, Alejandro Enríquez y Carlos Lizama, por acompañarme incondicionalmente a lo largo de la carrera.
Mis compañeros	A todos los compañeros con los que en su momento compartimos aula, apuntes, retos, compañerismo y conocimientos.
Mis catedráticos	A todos mis catedráticos que compartieron sus conocimientos conmigo.

AGRADECIMIENTOS A:

Mis padres	Por el amor, el apoyo, los consejos y sabiduría a lo largo de toda mi vida y mi carrera universitaria.
Mi hermano	Por su apoyo incondicional, su ejemplo y modelo a seguir en toda mi vida.
Ingeniero Erick Maldonado	Por su valiosa colaboración en la asesoría y revisión del presente trabajo.
Universidad de San Carlos de Guatemala y Facultad de Ingeniería	Por abrirme las puertas a tan prestigiosa casa de estudios y albergarme en mi formación profesional.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
@	Arroba
%	Porcentaje

GLOSARIO

Algoritmo Grupo finito de operaciones organizadas de manera lógica y ordenada que permite solucionar un determinado problema.

Archivo CSV CSV (*comma-separated values*) es un archivo de texto que almacena los datos en forma de columnas, separadas por coma y las filas se distinguen por saltos de línea. Es una forma muy sencilla de representar la información.

Arista En teoría de grafos, una arista corresponde a una relación entre dos vértices de un grafo.

Arreglo Grupo o colección finita, homogénea y ordenada de elementos de una o más dimensiones.

Barra de dirección web Elemento del navegador web. Es la barra más característica del navegador, en donde figura la dirección de la página que se está mostrando en ese momento.

Base de datos Conjunto de información perteneciente a un mismo contexto, ordenada de modo sistemático para su posterior recuperación, análisis y/o transmisión.

Cibercriminal	Persona que se dedica a cometer delitos informáticos, utiliza la computadora o tecnología como herramienta u objetivo para causar daños a otras personas o sistemas o en busca de algún beneficio.
Código fuente	Conjunto de líneas de textos, que son las directrices que debe seguir la computadora para realizar dicho programa.
Data set	Un *data set* corresponde a los contenidos de una única tabla de base de datos o una única matriz de datos estadística, donde cada columna de la tabla representa una variable en particular, y cada fila representa a un miembro determinado del conjunto de datos en cuestión.
Dirección IP	Es una etiqueta numérica que identifica, de manera lógica y jerárquica, a un interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (*Internet Protocol*).
Dirección web	Es el nombre completo de las páginas de Internet que se utiliza para encontrar un servidor, los directorios o subdirectorios donde se encuentra la información que se está buscando.

DNS	Es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada. Este sistema asocia información variada con nombre de dominio asignado a cada uno de los participantes.
E-commerce	*E-commerce* o comercio electrónico es un método de compraventa de bienes, productos o servicios valiéndose de Internet como medio.
Estructura de datos	Es una forma particular de organizar datos en una computadora para que puedan ser utilizados de manera eficiente.
Grafo	Es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas, que permiten representar relaciones binarias entre elementos de un conjunto.
Grafo dirigido	Un grafo dirigido o digrafo es un tipo de grafo en el cual las aristas tienen un sentido de dirección definido.
Heurística	Método basado en la experiencia que puede utilizarse como ayuda para resolver problemas.
Host	El término *host* o anfitrión se usa en informática para referirse a las computadoras u otros dispositivos

	conectados a una red que provee y utiliza servicios de ella.
HTML	Es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a *HyperText Markup Language*, es decir, Lenguaje de Marcas de Hipertexto.
HTTP	El Protocolo de transferencia de hipertexto (en inglés: *Hypertext Transfer Protocol* o HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.
HTTPS	El Protocolo seguro de transferencia de hipertexto (en inglés: *Hypertext Transfer Protocol Secure* o HTTPS), es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.
Información sensible	Es el nombre que recibe la información personal privada de un individuo, por ejemplo, ciertos datos personales y bancarios, contraseñas de correo electrónico e incluso el domicilio en algunos casos.
Instancia	Corresponde a un registro en una tabla de una base de datos o bien a un registro en un *data set.*

JavaScript	Es un lenguaje de programación interpretado. Utilizado principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.
Lenguaje de programación	Es un lenguaje formal que proporciona una serie de instrucciones que permiten a un programador escribir secuencias de órdenes y algoritmos a modo de controlar el comportamiento físico y lógico de una computadora, con el objetivo de que produzca diversas clases de datos.
Librería de software	Es una colección o conjunto de subprogramas usados para desarrollar software.
Minería de datos	Es un campo de la estadística y las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos.
Motor de búsqueda	Son sistemas informáticos que trabajan buscando todos los archivos almacenados en Internet a través de arañas o *crawlers* para elaborar páginas de resultados que sirvan para que los usuarios accedan a sitios web tras realizar una búsqueda.

Navegador web Es un software, aplicación o programa que permite el acceso a la web, interpretando la información de distintos tipos de archivos y sitios web para que estos puedan ser visualizados.

Nodo Un vértice o nodo es la unidad fundamental de la que están formados los grafos.

Nodo hoja En un grafo en forma de árbol, un nodo hoja es aquel que no tiene nodos hijos. También llamado nodo terminal.

Nombre de dominio Es un nombre fácil de recordar asociado a una dirección IP física de Internet. Se trata de un nombre único que se muestra después del signo @ en las direcciones de correo electrónico y después de www. en las direcciones web.

Python Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Recursividad Es una técnica de programación que se utiliza para realizar una llamada a una función desde ella misma, de allí su nombre.

Tabla de base de datos En las bases de datos, se refiere al tipo de modelado de datos donde se guardan los datos recogidos por un programa. Su estructura general se asemeja a la vista general de un programa de hoja de cálculo. Se compone de columnas o campos y filas o registros.

URL Es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en Internet. URL significa *Uniform Resource Locator*, o bien, Localizador Uniforme de Recursos.

Ventana PopUp También conocida como ventana emergente. Muestra contenido que aparece de forma repentina en un navegador web o en la pantalla del ordenador.

RESUMEN

Internet se ha convertido en un componente esencial en las actividades sociales y financieras de muchas personas. Sin embargo, los usuarios de Internet son vulnerables a diferentes tipos de amenazas que pueden causar daños financieros, robo de identidad, pérdida de información privada, daño a la reputación de una marca, daño o pérdida de la confianza de los clientes de sitios de *e-commerce* y plataformas bancarias. El *phishing* es una amenaza en la web que se define como el arte de imitar un sitio web de una empresa honesta con el objetivo de obtener información confidencial como nombres de usuario, contraseñas, información de tarjetas de crédito, entre otros.

El presente trabajo consiste en una comparación en términos de eficiencia predictiva de los algoritmos de aprendizaje supervisados, Árbol de Decisión C4.5, Clasificador Naive Bayes y redes neuronales, implementados utilizando la librería Scikit-Learn para el lenguaje de programación Python y aplicados al problema de clasificar correctamente sitios web en las categorías de "*phishing*" y "legítimo".

Partiendo de la hipótesis de que algunos algoritmos de clasificación son más eficientes para ciertas aplicaciones en problemas de clasificación, se busca determinar cuál de los tres algoritmos sometidos al estudio es el más eficiente para la detección de sitios web *phishing* y hacer la recomendación para que se tome en cuenta en la implementación de una herramienta de detección de sitios web fraudulentos.

OBJETIVOS

General

Calcular las medidas de rendimiento de los modelos de predicción Árbol de Decisión C4.5, Clasificador Naive Bayes y red neuronal, y determinar cuál es el más preciso en la clasificación de sitios web *phishing* utilizando un *data set* con más de dos mil cuatrocientos registros.

Específicos

1. Definir la base teórica para justificar el conjunto de características efectivas para entrenar un modelo de machine learning capaz de clasificar sitios web phishing.

2. Determinar las medidas de rendimiento del modelo predictivo Árbol de Decisión C4.5 para la clasificación de sitios web phishing.

3. Determinar las medidas de rendimiento del modelo predictivo clasificador Naive Bayes para la clasificación de sitios web phishing.

4. Determinar las medidas de rendimiento de una red neuronal para la clasificación de sitios web phishing.

5. Hacer un análisis comparativo de las medidas de rendimiento de cada algoritmo sometido al estudio y determinar cuál es el más preciso para la clasificación de sitios web phishing.

INTRODUCCIÓN

La detección de sitios web *phishing* puede abordarse como un problema de clasificación. Determinando un conjunto de características de sitios web fraudulentos y aplicando reglas inteligentes a estas características se permite crear bases de datos con una cantidad suficientemente grande de entradas de sitios web *phishing* legítimos ya conocidos. El subcampo de las ciencias de la computación conocido como *machine learning* provee técnicas que permiten a las computadoras el aprendizaje utilizando estas bases de datos mediante la generación de modelos predictivos.

En este estudio se busca comparar la eficiencia de tres algoritmos de *machine learning*: C4.5, Naive Bayes y redes neuronales. Cada uno de estos algoritmos será implementado utilizando la librería Scikit-Learn para el lenguaje de programación Python. Cada algoritmo será alimentado con el mismo conjunto de datos y se obtendrá de cada uno un modelo predictivo capaz de clasificar un sitio web *phishing* en función de las características definidas.

De los tres modelos predictivos se obtendrán cinco medidas de eficiencia, tanto para los datos de entrenamiento como para los datos de prueba, y se buscará determinar cuál es el mejor modelo a implementar para la detección de sitios web *phishing.*

1. PLANTEAMIENTO DEL PROBLEMA

1.1. Antecedentes del problema

Internet se ha convertido en un componente esencial en las actividades sociales y financieras de muchas personas. Sin embargo, los usuarios de Internet son vulnerables a diferentes tipos de amenazas que pueden causar daños financieros, robo de identidad, pérdida de información privada, daño a la reputación de una marca, daño o pérdida de la confianza de los clientes de sitios de *e-commerce* y plataformas bancarias. El *phishing* es una amenaza en la web que se define como el arte de imitar un sitio web de una empresa honesta con el objetivo de obtener información confidencial como nombres de usuario, contraseñas y números de seguro social, entre otros.

El crecimiento en el uso de dispositivos móviles para realizar transacciones financieras y compras en línea ha incrementado los riesgos. Los cibercriminales y otros usuarios maliciosos utilizan Internet para cometer crímenes como el *phishing.* El robo de números de cuentas bancarias y la clonación de tarjetas de crédito y débito son los objetivos más comunes de los ataques *phishing.*

1.2. Enfoques para la detección de sitios web *phishing*

Existen varios enfoques para la detección de sitios web *phishing*. Cada uno de ellos difiere en la tecnología utilizada para su implementación y la efectividad en la detección.

1.2.1. Detección de sitios web *phishing* utilizando una lista negra

Consiste en la creación y actualización constante de una lista de sitios web *phishing* conocidos. La URL que el usuario visita se compara con esta lista y, en el caso de que la URL se encuentre en ella, el navegador web puede bloquear el acceso al sitio o advertir al usuario de que está ingresando a un sitio web *phishing.*

El principal inconveniente con este enfoque es que las listas negras usualmente no cubren todos los sitios *phishing*, dado que se espera que en cuestión de segundos sea lanzado un nuevo sitio de este tipo.

1.2.2. Detección de sitios web *phishing* mediante el uso de heurísticas

Una heurística es un algoritmo para distinguir sitios *phishing* de otros basados en la experiencia del usuario, es decir, una heurística verifica si un sitio parece ser un sitio *phishing.* En una solución basada en heurísticas varias características del sitio son colectadas para clasificarlo en sitio *phishing* o auténtico. En contraste con el enfoque basado en una lista negra, este enfoque puede reconocer sitios *phishing* creados recientemente y en tiempo real.

Estudios anteriores han determinado que el *machine learning* puede ser muy útil para predecir sitios *phishing* y puede brindar respuestas acerca de las características e indicadores de un sitio *phishing* y cómo se relacionan entre ellas.

1.2.3. Clasificación de sitios web *phishing* basados en reglas inteligentes

Una de las técnicas prometedoras para ser empleadas en la predicción de ataques *phishing* está basada en minería de datos, particularmente en la inducción de reglas de clasificación. La exactitud de un método de detección basado en heurísticas depende en la elección de un conjunto de características discriminativas que podrían ayudar a distinguir la clase del sitio web. La manera en que las características son procesadas también juega un rol importante

Las reglas inteligentes son una representación común de datos ya que son entendidas fácilmente por humanos. Normalmente, la regla toma la forma de una cláusula SI-ENTONCES, por ejemplo, SI condición A ENTONCES clase β, donde A contiene los valores de las características y β es la clase predicha. El proceso de detectar conocimiento no visto antes en conjuntos de datos es representado en términos de reglas y es conocido como inducción de reglas.

La inducción de reglas facilita la toma de decisión, ya que genera conocimiento que asegura la exactitud, confiabilidad y completitud y reduce el tiempo para lograr conocimiento.

1.2.4. Aplicación de *machine learning* para la detección de sitios web *phishing*

La detección de sitios *phishing* es considerada un problema de clasificación binario dado que solamente existen dos posibles valores: "*phishing*" o "legítimo". Es por esto que, mediante el entrenamiento de modelos predictivos, *machine learning* es una técnica eficiente para realizar dicha detección.

Este enfoque funciona eficientemente si se cuenta con una base de datos de sitios legítimos y *phishing* lo suficientemente grande y describiendo las características de estos sitios mediante los resultados de la aplicación de reglas inteligentes. Contar con estos datos permite obtener datos de entrenamiento y datos de prueba que serán utilizados para entrenar y validar modelos predictivos utilizando las técnicas de *machine learning* apropiadas.

2. MARCO TEÓRICO

2.1. Machine learning

Machine learning es el estudio científico de algoritmos y modelos estadísticos que sistemas computacionales utilizan para realizar una tarea específica sin utilizar instrucciones explícitas.

Arthur Samuel definió *machine learning* en 1969 de la siguiente manera: "*machine learning* es el campo de estudio que brinda a las computadoras la habilidad de aprender sin ser explícitamente programadas." [1]

Tom Mitchel definió el aprendizaje en 1997 de la siguiente manera: "se dice que un programa de computadora aprende de una experiencia E con respecto a una tarea T y una medida de desempeño D si este desempeño de T, medido por D, mejora con la experiencia E."[2]

2.1.1. Tipos de algoritmos de *machine learning*

Los algoritmos de *machine learning* se clasifican según el tipo de aprendizaje: supervisado, supervisado, semisupervisado y por refuerzo.

[1] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras and Tensor Flow.* 2da edición. Consulta: 2019..

[2] Ibíd.

2.1.1.1. Aprendizaje supervisado

Un algoritmo de aprendizaje supervisado es aquel que produce una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema.

Los datos con los que se entrena el algoritmo incluyen las soluciones esperadas, las cuales se denominan etiquetas. Los algoritmos de aprendizaje supervisado son útiles para la resolución de problemas de clasificación, que es el tipo de problema al que corresponde la detección de sitios web *phishing* basado en características discriminativas.

2.1.1.2. Aprendizaje no supervisado

Un algoritmo de aprendizaje no supervisado es aquel que crea un modelo a partir del entrenamiento con datos no etiquetados. Dado que no se tiene categorías específicas en los datos, estos algoritmos son útiles para el reconocimiento de patrones e identificación de categorías en donde no son evidentes.

2.1.1.3. Aprendizaje semisupervisado

Un algoritmo de aprendizaje semisupervisado es aquel capaz de aprender a partir de datos parcialmente etiquetados. Comúnmente una gran cantidad de datos no está etiquetada. La mayoría de los algoritmos de aprendizaje semisupervisado son combinaciones de algoritmos de aprendizaje supervisado y no supervisado.

2.1.1.4. Aprendizaje por refuerzo

El aprendizaje por refuerzo difiere mucho de los tres tipos de aprendizaje anteriores. En el contexto del aprendizaje por refuerzo el sistema es sensible al ambiente con el que interactúa, selecciona y ejecuta acciones en función de las cuales obtiene una retroalimentación a manera de recompensas. Dichas recompensas pueden ser positivas o negativas. El sistema debe aprender por sí mismo cuál es la mejor estrategia para obtener la mejor recompensa a lo largo del tiempo.

2.1.2. Técnicas de clasificación

En machine learning y en estadística la clasificación es el problema de identificar a cuál categoría, dado un conjunto de ellas, pertenece una nueva observación, con base en datos de entrenamiento que contienen observaciones o instancias cuya categorización es conocida. Es decir que los datos de entrenamiento deben estar etiquetados y el problema de clasificación puede resolverse por un algoritmo de aprendizaje supervisado.

En un problema de clasificación la salida generada por el sistema, dada una entrada, es un valor booleano. Se obtiene un “sí” o un “no” como respuesta a la pregunta de si una observación X pertenece a la categoría A.

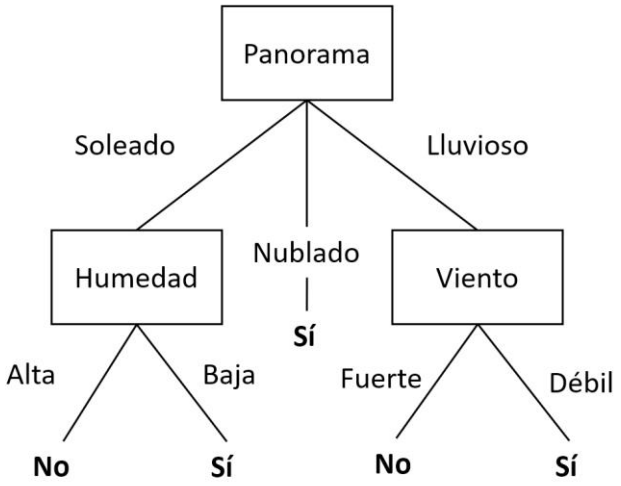
2.1.2.1. Árbol de decisión

Un árbol de decisión es una estructura de datos jerárquica que implementa la estrategia divide-y-conquista. Es un método eficiente para la clasificación.

Es un modelo jerárquico para aprendizaje supervisado compuesto por nodos de decisión internos y hojas terminales. Cada nodo de decisión m implementa una función de prueba f(x) con salidas discretas que etiquetan las ramas. Dada una entrada, en cada nodo una prueba es aplicada y una de las ramas es elegida dependiendo de la salida. Este proceso comienza en el nodo raíz y es repetido recursivamente hasta que un nodo hoja es alcanzado, en ese punto el valor obtenido por el nodo hoja constituye la salida del algoritmo, es decir, la clase a la que pertenece la entrada.

En la figura 1 se ejemplifica un árbol de decisión para predecir si va a llover basado en una base de datos del clima. Cada instancia en la base de datos tiene tres atributos: panorama, humedad y viento. Para panorama se tienen tres posibles valores: soleado, nublado y lluvioso. Para humedad se tienen dos posibles valores: alta y baja. Y para viento se tienen dos posibles valores: fuerte y débil. Los únicos dos posibles resultados son: “sí” y “no”.

Figura 1. **Árbol de decisión para la predicción de lluvia basada en el estado del tiempo**



Fuente: elaboración propia.

2.1.2.2. Algoritmo C4.5

El algoritmo C4.5 fue desarrollado por J. R. Quinlan en 1993 como una mejora del algoritmo ID3 en 1986. Este algoritmo genera un árbol de decisión a partir de datos de entrenamiento utilizando el concepto de ganancia de la información.

La ganancia de información se define como la reducción de la medida de ruido o desorden en los datos (entropía) causada al partir un conjunto de datos de entrenamiento S, con respecto a un atributo A. Es decir, la ganancia de información es la medida que indica cuánta información brinda este atributo acerca de la clase a la que pertenece una instancia del conjunto de datos. El atributo con mayor ganancia de información se coloca en la raíz del árbol.

A partir de este atributo se parten los datos con respecto a cada uno de los valores posibles del atributo para identificar el siguiente con menor entropía y crear los nodos hijos. El proceso se repite recursivamente hasta crear nodos por cada atributo.

2.1.2.3. Clasificador Naive Bayes

El clasificador Naive Bayes es una técnica de clasificación y predicción supervisada que construye modelos que predicen la probabilidad de posibles resultados. Este clasificador está basado en el teorema de Bayes, también conocido como teorema de la probabilidad condicionada. Dicho teorema indica que la probabilidad de que ocurra un suceso A habiendo sucedido otro suceso B, se define con la siguiente fórmula:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Este método de clasificación provee una medida probabilística de la importancia de las variables en el problema. Esta es quizá una de las diferencias fundamentes que ofrecen los clasificadores bayesianos con respecto a otros métodos como los árboles de decisión y las redes neuronales.

El algoritmo Naive Bayes consiste en calcular la probabilidad de que una instancia del conjunto de datos pertenezca a cada clase y se toma el resultado que emita mayor probabilidad.

Dado que es una clasificación supervisada, dichos ejemplos deben estar etiquetados, es decir, se debe definir a qué clase pertenecen.

Dada una instancia x representada por k valores, el clasificador Naive Bayes se basa en encontrar la función de hipótesis más probable que describa a esa instancia. Si la descripción de esa instancia viene dada por los valores $< a_1, a_2, \dots, a_n >$, la hipótesis más probable será aquella que cumpla:

$$v_{MAP} = argmax_{v_j} \in V^{P(v_j|a_1,\dots a_n)}$$

Es decir, la probabilidad de que conocidos los valores que describen a ese ejemplo, este pertenezca a la clase v_j (donde v_j es el valor de la función de clasificación $f(x)$ en el conjunto finito de clasificaciones V). Por el teorema de Bayes:

$$v_{MAP} = argmax_{v_j} \in V \frac{P(a_1, \dots a_n|v_j)p(v_j)}{P(a_1, \dots a_n)} = argmax_{v_j} \in V^{P(a_1, \dots a_n|v_j)p(v_j)}$$

2.1.2.4. Redes neuronales artificiales

Una red neuronal artificial es un modelo computacional inspirado por la estructura de las redes neuronales del cerebro. En modelos simplificados del cerebro, la red neuronal consiste en un gran número de dispositivos informáticos básicos (neuronas) que están conectadas entre sí en una compleja red de comunicación, a través de la cual el cerebro es capaz de llevar a cabo cálculos altamente complejos. Las redes neuronales artificiales son constructos computacionales formales que son modelados a partir de este modelo computacional.

Una red neuronal puede ser descrita como un grafo dirigido cuyos nodos corresponden a neuronas y las aristas a los enlaces entre ellas. Cada neurona recibe como entrada una suma ponderada de las salidas de las neuronas conectadas a las aristas entrantes.

Una red neuronal es descrita por un grafo dirigido acíclico, $G = (V, E)$, y una función de pesos sobre las aristas, $w : E \rightarrow \mathbb{R}$. Los nodos del grafo corresponden a las neuronas. Cada neurona es modelada como una simple función escalar, $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. Cada arista en el grafo enlaza la salida de alguna neurona a la entrada de otra. La entrada de una neurona es obtenida tomando la suma ponderada de las salidas de todas las neuronas conectadas a ella, donde los pesos corresponden a w.

Para simplificar la descripción del cálculo realizado por la red se asumirá que la red está organizada por capas. Es decir, el conjunto de nodos V se puede descomponer en una unión de subconjuntos no vacíos, de tal manera que cada arista en E conecta algún nodo en V_{t-1} a algún nodo en V_t, para algún $t \in [T]$.

Se refiere a T como el número de capas en una red (excluyendo a V_0), o la profundad de la red. El tamaño de la red es $|V|$. El ancho de la red es $max_t|V_t|$.

La primera capa, V_0, es llamada capa de entrada. Ella contiene $n+1$ neuronas, donde n es la dimensionalidad del espacio de entrada. Para cada $i \in [n]$, la salida de la neurona i en V_0 es simplemente x_i. La última neurona en V_0 es la "constante", cuya salida siempre es 1. Se denota como $v_{t,i}$ como la $i$ésima neurona de la capa t y como $o_{t,i}(x)$ la salida de $v_{t,i}$ cuando la red es alimentada con el vector de entrada X. Por lo tanto, para cada $i \in [n]$ se tiene $o_{o,i}(x) = x_i$, y para $i = n+1$ se tiene $o_{0,i}(x) = 1$. Se procede luego al cálculo capa por capa. Suponiendo que se han calculado las salidas de la capa t se pueden calcular las salidas de la capa $t+1$ y así sucesivamente. Tomando alguna $v_{t+1,j} \in V_{t+1}$. Se denota como $a_{t+1}(x)$ la entrada de $v_{t+1,j}$ cuando la red es alimentada con el vector de entrada X. Por tanto,

$$a_{t+1,j}(x) = \sum_{r:(v_t,r,v_{t+1,j})\in E} w\left(\left(v_{t,r}, v_{t+1,j}\right)\right) o_{t,r}(x)$$

Y

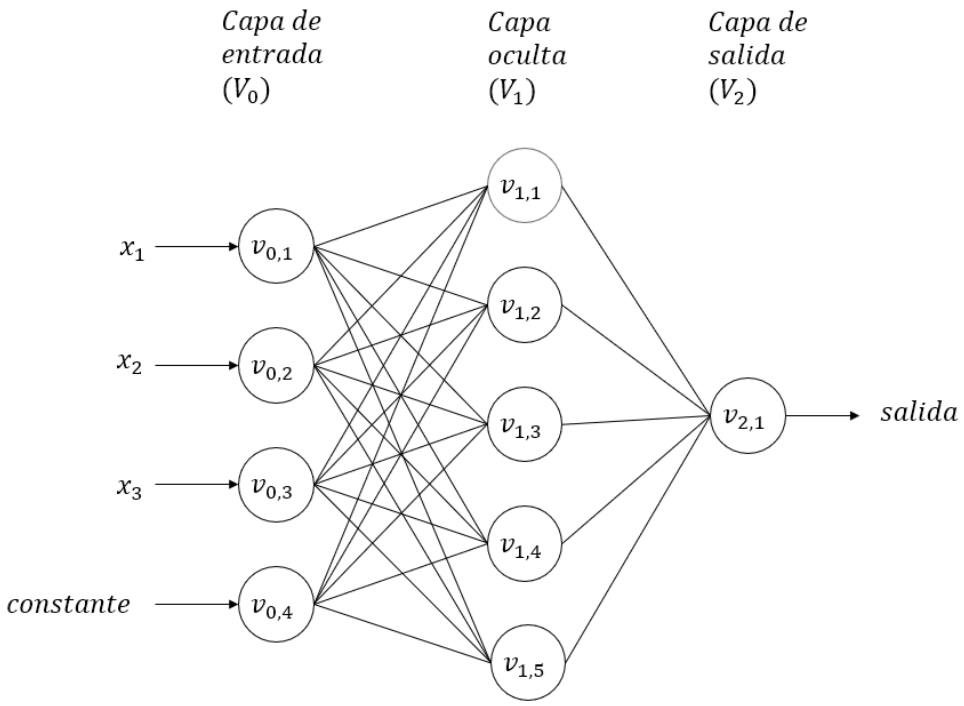
$$o_{t+1,j}(x) = \sigma(a_{t+1,j}(x)).$$

Es decir, la entrada a $v_{t+1,j}$ es una suma ponderada de las salidas de las neuronas en V_t que están conectadas a $v_{t+1,j}$, donde la ponderación es acorde a w, y la salida de $v_{t+1,j}$ es simplemente la salida de la función de activación σ en su entrada.

Las capas $V_1, \dots, V_{T-1}$ comúnmente se llaman capas ocultas. La última capa, V_T, es llamada capa de salida. En problemas de predicción simple la capa

de salida contiene una única neurona cuya salida es la salida de toda la red neuronal.

Figura 2. **Diagrama de una red neuronal**



Fuente: elaboración propia.

2.2. Entrenamiento de modelos de *machine learning*

El entrenamiento de modelos de *machine learning* involucra una serie de requerimientos y pasos que se describen a continuación.

2.2.1. Data set

Un *data set* es un grupo de datos estructurados obtenible en su totalidad a través de un solo enlace a una dirección web o una única instrucción que

descarga los datos de Internet. Los datos deben estar estructurados, por tanto, la relación entre los elementos debe estar clara e inequívocamente determinada. Comúnmente un *data set* corresponde a los contenidos de una sola tabla de base de datos, o una sola matriz de datos estadísticos, en donde cada columna de la tabla representa una variable en particular, y cada fila corresponde a un miembro del *data set* en cuestión.

Dentro del contexto de *machine learning*, el *data set* es el grupo de datos a partir del cual el algoritmo genera un modelo que realiza predicciones con instancias totalmente nuevas que tienen la misma estructura del *data set.*

En el proceso de creación de un sistema de *machine learning* este *data set* se dividirá comúnmente en dos conjuntos de datos: el *training set* y el *test set.*

2.2.2. Training set

El *training set* es un subconjunto de datos extraído del *data set* completo utilizado para el aprendizaje del sistema. A partir del *training set* el algoritmo de *machine learning* genera un modelo predictivo que se ajuste lo más posible a los datos.

2.2.3. Test set

El *test set* es un subconjunto de datos extraídos del *data set* completo, independiente del *training set.* El *test set* es un conjunto de datos usado únicamente para evaluar el rendimiento de un modelo predictivo específico.

2.2.4. Preparación de los datos de entrenamiento y datos de prueba

La preparación de los datos de prueba y de entrenamiento es un paso crucial para lograr un modelo de predicción eficiente. La preparación de los datos se divide en dos etapas:

2.2.4.1. Preprocesamiento de datos

Consiste en obtener el conjunto de datos de una forma apropiada para el entrenamiento del modelo. Los tres pasos más comunes para el procesamiento son:

- Formateo: consiste en colocar los datos en un formato adecuado como un archivo de texto plano un archivo CSV.
- Limpieza: consiste en remover o restaurar los datos faltantes. En un conjunto de datos pueden existir instancias incompletas y no contiene datos necesarios para resolver el problema de predicción.
- Muestreo: consiste en seleccionar únicamente los datos representativos del problema para construir el modelo de predicción a partir de ellos.

2.2.4.2. Transformación de datos

El conocimiento del dominio del problema y el algoritmo de *machine learning* a utilizar influirán en este paso. Las tres transformaciones más comunes son:

- Escalado: los datos preprocesados pueden contener atributos con una mezcla de escalas para varias cantidades. Muchos métodos de *machine learning* requieren que los atributos de los datos tengan la misma escala. Por ejemplo, entre 0 y 1 para el valor más pequeño y más grande para una característica determinada.

- Descomposición: puede haber características que representan un concepto complejo que puede ser más útil para un método de *machine learning* cuando se divide en las partes constituyentes. Un ejemplo es una fecha que puede tener componentes de día y hora que, a su vez, podrían dividirse aún más. Quizás solo la hora del día es relevante para el problema que se está resolviendo.

- Agregación: puede haber características que pueden agregarse en una sola característica que sería más significativa para el problema que se está tratando de resolver.

2.2.5. Entrenamiento del modelo

El entrenamiento consiste en alimentar un algoritmo de aprendizaje con datos de entrenamiento. El algoritmo encuentra patrones en los datos de entramiento de tal manera que los parámetros de entrada correspondan al objetivo. La salida del proceso de entrenamiento es un modelo que puede utilizarse para hacer predicciones. Este proceso también es llamado aprendizaje.

2.2.6. Hiperparámetros

Los hiperparámetros son parámetros para el uso exclusivo del algoritmo de entrenamiento y no del modelo como tal. Por tanto, sirven para parametrizar el proceso de entrenamiento del modelo y la configuración del mismo. Ejemplo de un hiperparámetro sería el número de capas que se desean en una red neuronal.

2.2.7. Validación cruzada

La validación cruzada es una técnica para evaluar modelos de *machine learning* mediante el entrenamiento de varios modelos sobre subconjuntos de los datos de entrada disponibles y evaluarlos sobre el conjunto de datos de prueba. La validación cruzada se utiliza para detectar si un modelo ha subajustado o sobreajustado los datos de entrenamiento y, por tanto, si realiza predicciones imprecisas sobre nuevos conjuntos de datos.

Se dice que un modelo ha subajustado los datos de entrenamiento cuando no ha sido capaz de captar las relaciones entre los datos de entrada X y los valores esperados Y del conjunto de datos de entrenamiento. Por otro lado, se dice que un modelo ha sobreajustado los datos de entrenamiento cuando realiza predicciones acertadas sobre los datos de entrenamiento, pero no lo hace en los datos de prueba.

La técnica de validación cruzada utilizada en este estudio es conocida como “k-fold”, consiste en dividir el conjunto de datos de entrenamiento en un número k de subconjuntos sobre los cuales se entrenan los modelos para evaluar su eficiencia.

2.2.8. Medidas de desempeño

Existen medidas utilizadas para conocer el desempeño de un modelo de predicción luego de su entrenamiento. Estas medidas sirven para conocer la eficiencia del modelo, es decir, la confiabilidad de sus predicciones.

2.2.8.1. Matriz de confusión

La matriz de confusión muestra los resultados obtenidos por medio de la predicción del modelo y resultados reales. Una matriz de confusión es de tamaño n x n, en donde n es el número de las posibles categorías. La tabla I muestra una matriz de confusión de n=2, la cual describe muy bien un clasificador para categorizar sitios web *phishing.*

- VN es la cantidad de predicciones negativas correctas
- FP es la cantidad de predicciones positivas incorrectas
- FN es la cantidad de predicciones negativas incorrectas
- VP es la cantidad de predicciones positivas correctas

A partir de los valores obtenidos de la matriz de confusión se pueden obtener otras medidas de desempeño útiles.

Tabla I. **Matriz de confusión de un clasificador de dos categorías**

	Predicho negativo	Predicho positivo
Real negativo	VN	FP
Real positivo	FN	VP

Fuente: elaboración propia.

2.2.8.2. Precisión

La precisión de las predicciones hechas por el clasificador se obtiene a partir de la siguiente fórmula:

$$precisión = \frac{VP}{VP + FP}$$

2.2.8.3. Sensibilidad

La sensibilidad es la tasa de instancias positivas que son clasificadas correctamente por el clasificador. Se obtiene a partir de la siguiente fórmula:

$$sensibilidad = \frac{VP}{VP + FN}$$

2.2.8.4. Medida F1

En clasificadores binarios la medida F1 es una medida de exactitud de las predicciones. Considera la precisión y la sensibilidad en su cálculo. La medida F1 es la media armónica entre la precisión y la sensibilidad y se obtiene a partir de la siguiente forma:

$$F1 = \frac{2}{\frac{1}{precisión} + \frac{1}{sensibilidad}}$$

2.2.8.5. Exactitud

La exactitud se define como la relación entre la cantidad de predicciones correctas y la cantidad total de observaciones. La exactitud de un modelo se obtiene de la siguiente forma:

$$exactitud(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(\hat{y_i} = y_i)$$

En donde n es la cantidad de observaciones, $\hat{y}$ es el valor predicho por el modelo y y es el valor real.

2.2.8.6. Pérdida Hamming

La pérdida Hamming se define como la relación entre la cantidad de predicciones incorrectas y la cantidad total de observaciones. La pérdida Hamming de un modelo se obtiene de la siguiente forma:

$$perdida(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(\hat{y_i} \neq y_i)$$

En donde n es la cantidad de observaciones, $\hat{y}$ es el valor predicho por el modelo y y es el valor real.

2.3. Seguridad informática

La seguridad informática es el proceso de prevenir y detectar el uso de sistemas informáticos con intensiones maliciosas. La seguridad informática cubre cuatro áreas principales:

- Confidencialidad: consiste en asegurar que solo los usuarios autorizados pueden acceder a los recursos, datos e información.

- Integridad: consiste en asegurar que solo los usuario autorizados deben ser capaces de modificar datos cuando sea necesario.

- Disponibilidad: consiste en asegurar que los datos deben estar disponibles para los usuarios cuando sea necesario.

- Autenticación: consiste en asegurar que el usuario esté comunicándose realmente con quien piensa que está comunicándose.

2.3.1. Phishing

El *phishing* es una forma de uso de sistemas informáticos con intensiones maliciosas. Se describe como el arte de imitar un sitio web legítimo y acreditado con el fin de obtener información privada de un usuario de Internet, tal como nombres de usuario, contraseñas y números de identificación personal. Se dice que un ataque *phishing* es un ataque hacia la confidencialidad y datos de autenticación del usuario. Un sitio web *phishing* es un sitio web falso generado por personas deshonestas para suplantar sitios web de empresas honestas y legítimas.

Un usuario que sufre un ataque *phishing* puede ser víctima de suplantación de identidad, robo de información financiera tal como números de tarjetas de crédito, números de cuentas bancarias, robo de contraseñas y demás información sensible. El principal objetivo de un atacante *phishing* es obtener un beneficio económico a través del robo de información financiera de la víctima.

Los ataques *phishing* causan riesgos de seguridad a gran escala a la comunidad de Internet y a aquellas empresas en Internet que manejan información sensible. Aún si los usuarios de Internet son conscientes de este tipo de ataques muchos usuarios llegan a ser víctimas debido a la dificultad para el usuario común de identificar un sitio web de este tipo.

2.3.2. Técnicas de *phishing*

El *phishing* se encuentra en constante desarrollo debido a la facilidad que tienen los atacantes de crear réplicas completas de un sitio web haciendo uso del código fuente HTML. Realizando únicamente pequeños cambios en el código fuente del sitio resulta fácil engañar a la víctima dirigiéndola a sitios web fraudulentos.

Existe un número de diferentes técnicas usadas para obtener información de los usuarios. Conforme la tecnología se vuelve más avanzada así las técnicas de los cibercriminales también se vuelven más avanzadas. A continuación, se describen las técnicas utilizadas más comúnmente.

2.3.2.1. Spear phishing

Consiste en el envío de un solo correo electrónico a una persona o un departamento en específico que parece proceder de un destinario confiable. Esta técnica requiere conocer el correo electrónico de la persona o departamento que se desea atacar y también haber definido cuál es el tipo de información que se busca obtener.

2.3.2.2. Email/Spam

Es la técnica más común de realizar un ataque *phishing*, el correo electrónico es enviado a millones de usuario con una solicitud de para brindar datos personales. La mayoría de los mensajes contienen una nota urgente que requiere que el usuario ingrese sus credenciales para actualizar información de la cuenta, cambiar detalles o verificar la cuenta.

2.3.2.3. Inyección de contenido

En esta técnica el atacante cambia una parte del contenido en la página de un sitio web confiable. Esto con el objetivo de despistar al usuario y enviarlo a una página fuera del sitio web de confianza en donde luego se solicita que ingrese información personal.

2.3.2.4. Phishing a través de motores de búsqueda

Esta técnica involucra motores de búsqueda tales como Google, Yahoo!, Bing, entre otros, desde los cuales el usuario es dirigido a sitios que pueden ofrecer productos o servicios a bajo costo. Cuando el usuario intenta comprar el

producto ingresando los detalles de tarjeta de crédito o cualquier otro método de pago, dichos datos son colectados por cibercriminales.

2.3.3. Características de un sitio web *phishing*

No existe un acuerdo actualmente acerca de las características definitivas de sitios web *phishing.* Sin embargo, varias investigaciones previas han probado que existen algunas características importantes para su predicción y detección. Dichos estudios han agrupado estas características en cuatro grupos:

- Características basadas en la barra de dirección web
- Características basadas en anormalidades
- Características basadas en HTML y JavaScript
- Características basadas en el dominio

2.3.3.1. Características basadas en la barra de dirección web

- Uso de la dirección IP: si la dirección IP es usada en lugar de un nombre de dominio en la URL, por ejemplo: 128.95.2.3, el usuario puede estar bastante seguro de que alguien está intentando robarle información personal.
- URL larga para ocultar una parte sospechosa: los atacantes pueden hacer uso de URLs largas para ocultar una parte sospechosa de la barra

de dirección web. Científicamente no existe un largo confiable para distinguir URLs de sitios *phishing* de sitios legítimos. Sin embargo, una longitud mayor de cincuenta y cuatro caracteres puede considerarse sospechosa.

- Uso de servicios para acortar la URL: acortar la URL es un método que permite reducir el largo de la URL y aun así dirigir al sitio web requerido. Esto es posible gracias a servicios en Internet como “TinyURL”. Si una URL está acortada por uno de estos servicios puede considerarse como sospechosa.

- URLs que contienen el símbolo ‘@’: los atacantes utilizan trucos para dar la impresión que la URL es legítima utilizando el símbolo @ en la URL. El navegador puede ignorar todo previo al símbolo @ dado que la dirección real sigue al símbolo @.

- Redireccionamiento usando “//”: la existencia de “//” dentro de la ruta URL significa que el usuario será redirigido a otro sitio web. Dicha práctica no es común en sitios web legítimos y un sitio que lo implemente puede considerarse como sospechoso de *phishing.*

- Agregar un prefijo o sufijo al nombre de dominio separado por ‘-‘: el guion es un símbolo raramente usado en URLs legítimas. Lo atacantes utilizan como recurso agregar sufijos o prefijos a los nombres de dominio para engañar al usuario y hacerlo creer que están tratando con un sitio web legítimo.

- Subdominios y multisubdominios: asumiendo que se tiene la siguiente URL: https://www.usac.edu.gt. Un dominio siempre incluye un dominio de

alto nivel o TLD por sus siglas en inglés (*Top Level Domain*), en este caso corresponde a “gt”. El “edu” corresponde a una abreviatura de “educational” y combinado “.edu.gt” es llamado dominio de segundo nivel o SLD por sus siglas en inglés (*Second Level Domain*), y “usac” es el nombre del dominio. Por tanto, se distingue que la URL legítima contiene dos puntos en la URL ya que se puede ignorar escribir www. Si una URL contiene más de tres puntos se puede considerar como sospechosa de dirigir a un sitio web *phishing* ya que contiene múltiples subdominios.

- HTTPS (*Hyper Text Transfer Protocol with Secure Sockets*): los sitios web legítimos utilizan nombres de dominio seguros cada vez que información sensible debe ser transferida. La existencia del protocolo HTTPS es muy importante para dar la impresión de legitimidad a un sitio web. Sin embargo, no es suficiente ya que se han detectado sitios web *phishing* que cuentan con el protocolo HTTPS mediante el uso de certificados extendidos por emisores poco confiables. Por tanto, debe verificarse el nombre del emisor de dichos certificados para identificar el nivel de confianza del sitio.

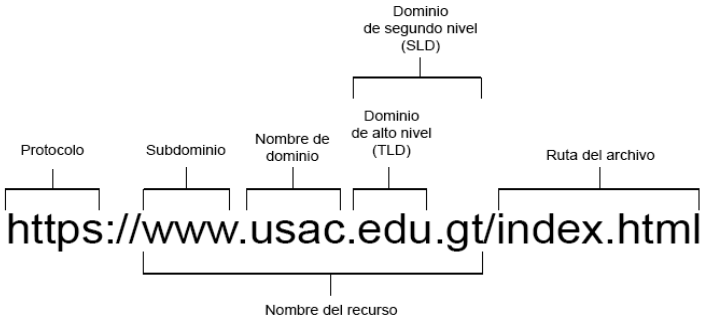
- Favicon: un favicon es un ícono asociado a una página web específica. Muchos navegadores muestran el favicon como un recordatorio visual de la identidad del sitio web en la barra de direcciones. Si el favicon se carga desde un dominio distinto al que se muestra en la barra de direcciones, es probable que el sitio web se considere un intento de *phishing.*

- Uso de puerto no estándar: esta característica es útil para validar si un servicio en particular (por ejemplo, HTTP) está activo o inactivo en un servidor específico. Con el objetivo de controlar las intrusiones, es mucho

mejor simplemente abrir los puertos que se necesita. Varios servidores de seguridad, servidores proxy y traducción de direcciones de red (NAT) bloquearán, de manera predeterminada, todos o la mayoría de los puertos y solo abrirán los seleccionados. Si todos los puertos están abiertos, un atacante puede ejecutar casi cualquier servicio que desee y, como resultado, la información del usuario está amenazada.

- Existencia del token “HTTPS” en el nombre dominio: un atacante puede agregar el token "HTTPS" a la parte del dominio de una URL para engañar a los usuarios. Si este es el caso se considera que el sitio es *phishing.*

Figura 3. **Anatomía de una URL**



Fuente: elaboración propia.

2.3.3.2. Características basadas en anormalidades

- Solicitudes a URLs externas: en sitios web legítimos la mayoría de objetos externos como imágenes están enlazados al mismo nombre de dominio. Si menos del 20 % de objetos un sitio web contiene enlaces a nombres de dominio externos se puede considerar como legítimo; si

entre el 20 % y el 50 % de los enlaces son externos se puede considerar un sitio web sospechoso y cualquier sitio que exceda el 50 % de los enlaces a nombres de dominio externos se puede considerar un sitio web *phishing.*

- URL del “anchor”: un “anchor” es un elemento HTML definido por la etiqueta <a>. Este elemento es la estructura que representa un hipervínculo en un sitio web. Si el nombre de dominio hacia el que dirige el anchor es diferente al nombre de dominio del sitio web el sitio puede clasificarse como *phishing.*

- Enlaces en las etiquetas <meta>, <script> y <link>: se espera que estas etiquetas de un sitio web, si tienen enlaces, estos enlaces dirijan al mismo dominio del sitio. De lo contrario puede considerarse como sospechoso de un sitio *phishing.*

- Dominio de formularios: los dominios de formularios que contienen una cadena vacía o "about: blank" se consideran dudosos porque se debe tomar una acción sobre la información presentada. Además, si el nombre de dominio es diferente del nombre de dominio de la página web, esto revela que la página web es sospechosa porque la información enviada rara vez es manejada por dominios externos.

- URL anormal: si la información obtenida de la base de datos WHOIS no coincide con la identidad a la que se le atribuye el sitio web se considera que es un sitio web *phishing.*

2.3.3.3. Características basadas en HTML y JavaScript

- Cantidad de redireccionamientos: la línea fina que distingue los sitios web *phishing* de los legítimos es la cantidad de veces que se ha redirigido un sitio web. Basado en los datos los sitios web legítimos han son redirigidos una vez como máximo. Por otro lado, los sitios web de *phishing* que contienen esta característica se han redirigido al menos 4 veces.

- Modificación de la barra de estado: un atacante puede usar JavaScript para mostrar a los usuarios una URL falsa en la barra de estado. Para extraer esta función se debe extraer el código fuente de la página web, en particular el evento "onMouseOver", y verificar si realiza algún cambio en la barra de estado.

- Deshabilitación del *clic* derecho: atacantes *phishing* pueden utilizar JavaScript para deshabilitar la función de *clic* derecho de tal manera que los usuarios no pueden acceder al código fuente de la página en busca de elementos sospechosos.

- Uso de ventanas PopUp: en sitios web legítimos es poco común que se le pida al usuario ingresar información sensible o credenciales en una ventana PopUp. Si un sitio web utiliza este mecanismo para solicitar dicha información puede considerarse sospechoso de ser un sitio fraudulento.

- Uso de IFrame: IFrame es una etiqueta HTML utilizada para mostrar una página web adicional en una que se muestra actualmente. Los atacantes

pueden hacer uso de la etiqueta "iframe" y hacerla invisible, es decir, sin bordes de marco. En este sentido, los atacantes utilizan el atributo "frameBorder" que hace que el navegador muestre una delineación visual.

2.3.3.4. Características basadas en el nombre de dominio

- Edad del dominio: la edad de un nombre de dominio puede obtenerse de la base de datos WHOIS. La mayoría de sitios web legítimos tienen una edad igual o mayor a dos años. Aquellos sitios cuyo nombre de dominio sea menor a dos años pueden considerarse sospechosos ya que los sitios fraudulentos no suelen persistir por mucho tiempo en Internet.

- Registros DNS: los registros DNS pueden obtenerse de la base de datos WHOIS. Si los registros DSN de un sitio web están vacíos o el registro del *host* no se encuentra, entonces el sitio web puede clasificarse como sospechoso de ser un sitio fraudulento.

- Tráfico del sitio web: si el nombre de dominio no tiene tráfico o no está reconocido en la base de datos Alexa el sitio web pude clasificarse como *phishing*, de lo contrario si el sitio web está clasificado entre las primeras 100 000 posiciones se puede decir que el sitio es legítimo. Esta característica tiene una importancia significativa al intentar detectar sitios fraudulentos.

- PageRank: PageRank es un valor que va desde "0" hasta "1". PageRank tiene como objetivo medir la importancia de una página web en Internet. Cuanto mayor sea el valor de PageRank, más importante será la página

web. Aproximadamente el 95 % de las páginas web de *phishing* no tienen PageRank. Además, el 5 % restante de las páginas web de *phishing* puede alcanzar un valor de PageRank de hasta "0,2".

- Índice de Google: esta función examina si un sitio web está en el índice de Google o no. Cuando Google indexa un sitio, este se muestra en los resultados de búsqueda. Por lo general las páginas web de *phishing* son simplemente accesibles por un período corto y, como resultado, es posible que muchas páginas web de *phishing* no se encuentren en el índice de Google.

- Número de enlaces apuntando al sitio: el número de enlaces que apuntan a la página web indica su nivel de legitimidad. El 98 % de los sitios *phishing* no tienen enlaces que los señalen. Por otro lado, los sitios web legítimos tienen al menos 2 enlaces externos que los apuntan.

- Característica basada en reportes estadísticos: algunas entidades en Internet, tales como PhishTank y StopBadware, formulan numerosos informes estadísticos sobre sitios web de *phishing* en cada período de tiempo dado. Algunos son mensuales y otros trimestrales. Si el sitio web posee características detectadas por estas entidades puede considerarse como un sitio *phishing.*

3. METODOLOGÍA

3.1. Herramienta Scikit-Learn

Scikit-learn es la herramienta de *machine learning* utilizada para la implementación de los modelos de predicción. Scikit-Learn es un módulo escrito en el lenguaje de programación Python. Expone una amplia variedad de algoritmos de *machine learning,* tanto para aprendizaje supervisado como no supervisado, usando una interfaz consistente y permitiendo una fácil comparación de métodos para una aplicación en específico.

Todos los objetos dentro de Scikit-Learn comparten una API común básica y uniforme que consiste de tres interfaces complementarias: una interfaz estimadora para construir y ajustar modelos, una interfaz predictora para realizar predicciones y una interfaz transformadora para realizar conversiones en los datos.

3.2. Hardware

Para realizar este estudio se han entrenado los modelos de predicción bajo las mismas capacidades de procesamiento (CPU) y memoria.

3.2.1. CPU

Para el entrenamiento de los modelos de predicción se ha utilizado una computadora con un procesador Intel Core i7-7700HQ con reloj de 2.80GHz.

3.2.2. RAM

La computadora en la cual fueron entrenados los modelos de predicción cuenta con una memoria de acceso aleatorio o RAM de 16 GB.

3.3. Descripción del *data set*

El *data set* utilizado para este estudio consta de 11 055 instancias, es decir una muestra de 11 055 sitios web. Cada instancia consta de 30 columnas de las cuales las primeras 29 columnas son características de sitios web que han demostrado ser efectivas en la predicción de sitios web *phishing*. La última columna de la instancia es la etiqueta asociada a sitio web. Cada una de las columnas puede tomar los valores descritos en la tabla II:

Tabla II. **Descripción de los valores en el *data set***

Valor	Descripción
-1	*Phishing*
0	Sospechoso
1	Legítimo

Fuente: elaboración propia.

Estos valores son resultado de la aplicación de las reglas inteligentes aplicadas mediante un proceso de minería de datos. Cada una de las características y la regla inteligente aplicada a cada una se describen en la tabla III.

Tabla III. **Características y reglas inteligentes del *data set***

Característica	Regla inteligente
Uso de dirección IP	$SI \begin{cases} Usa\ dirección\ IP\ en\ lugar\ de\ dominio \rightarrow Phishing \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
URL larga para ocultar una parte sospechosa	$SI \begin{cases} Longitud\ de\ la\ URL < 54 \rightarrow Legítimo \\ Longitud\ de\ la\ URL \geq 54\ y \leq 75 \rightarrow Sopechoso \\ De\ lo\ contrario \rightarrow Phishing \end{cases}$
Uso de servicios para acortar URL	$SI \begin{cases} URL\ Acortada \rightarrow Phishing \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
URL contiene símbolo "@"	$SI \begin{cases} URL\ contiene\ @ \rightarrow Phishing \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
Redireccionamiento usando "//"	$SI \begin{cases} La\ posición\ de\ la\ última\ occurrencia\ de\ //\ en\ la\ URL > 7 \rightarrow \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
Añade prefijo o sufijo al nombre dominio separado por "-"	$SI \begin{cases} Nombre\ de\ dominio\ incluye\ (-) \rightarrow Phishing \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
Subdominio y multi subdominios	$SI \begin{cases} Puntos\ en\ el\ nombre\ de\ dominio = 1 \rightarrow Legítimo \\ Puntos\ en\ el\ nombre\ de\ dominio = 2 \rightarrow Sospechoso \\ Puntos\ en\ el\ nombre\ de\ dominio = 3 \rightarrow Phishing \end{cases}$
Uso de protocolo HTTPS	$SI \begin{cases} Utiliza\ HTTPS\ de\ un\ emisor\ confiable \rightarrow Legítimo \\ Utiliza\ HTTPS\ de\ un\ emisor\ no\ confiable \rightarrow Sospechoso \\ De\ lo\ contrario \rightarrow Phishing \end{cases}$
Fecha de expiración del dominio	$SI \begin{cases} El\ dominio\ expira\ en \leq 1\ año \rightarrow Phishing \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
Favicon	$SI \begin{cases} Favicon\ es\ descargado\ de\ un\ dominio\ externo \rightarrow Phishing \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
Uso de puerto no estándar	$SI \begin{cases} Número\ de\ puerto\ se\ encuentra\ en\ el\ estado\ preferido \rightarrow Legítimo \\ De\ lo\ contrario \rightarrow Phishing \end{cases}$
Existencia del *token* "HTTPS" en el nombre dominio	$SI \begin{cases} Usa\ el\ tóken\ HTTPS\ en\ el\ nombre\ de\ dominio \rightarrow Phishin \\ De\ lo\ contrario \rightarrow Legítimo \end{cases}$
Solicitudes a URL externas	$SI \begin{cases} \%\ de\ solicitudes < 22\% \rightarrow Legítimo \\ \%\ de\ solicitudes \geq 22\%\ y \leq 61\% \rightarrow Sospechoso \\ De\ lo\ contrario \rightarrow Phishing \end{cases}$
URL de anchor	$SI \begin{cases} \%\ de\ URLs\ de\ anchor < 31\% \rightarrow Legítimo \\ \%\ de\ URL\ de\ anchor \geq 31\%\ y \leq 67\% \rightarrow Sospechoso \\ De\ lo\ contrario \rightarrow Phishing \end{cases}$
Enlaces en las etiquetas <meta>, <script> y <link>	$SI \begin{cases} \%\ de\ etiquetas < 17\% \rightarrow Legítimo \\ \%\ de\ etiquetas \geq 17\%\ y \leq 81\% \rightarrow Sospechoso \\ De\ lo\ contrario \rightarrow Phishing \end{cases}$

Continúa Tabla III.

Dominio de formulario	$\boldsymbol{SI}\begin{cases}\boldsymbol{Dominio\ es\ about{:}\ blank\ o\ est\acute{a}\ vac\acute{\imath}o\ \rightarrow\ Phshing}\\ \boldsymbol{Se\ refiere\ a\ un\ dominio\ diferente\ \rightarrow\ Sospechoso}\\ \boldsymbol{De\ lo\ contrario\ \rightarrow\ Leg\acute{\imath}timo}\end{cases}$
URL anormal	$SI\begin{cases}URL\ no\ incluye\ el\ nombre\ del\ host\ \rightarrow Phishing\\ De\ lo\ contrario\ \rightarrow Legítimo\end{cases}$
Cantidad de redireccionamientos	$SI\begin{cases}redireccionamientos\ \leq 1\ \rightarrow legítimo\\ redireccionamientos\ \geq 2\ y\ <4\ \rightarrow Sospechoso\\ De\ lo\ contraio\ \rightarrow Phishing\end{cases}$
Modificación de la barra de estado	$SI\begin{cases}El\ evento\ onMouseOver\ modifica\ la\ barra\ de\ estado\ \rightarrow Phishing\\ De\ lo\ contrario\ \rightarrow Legítimo\end{cases}$
***Clic* derecho deshabilitado**	$SI\begin{cases}Click\ derecho\ deshabilitado\ \rightarrow Phishing\\ De\ lo\ contrario \rightarrow Legítimo\end{cases}$
Uso de ventana Pop-up	$Si\begin{cases}La\ ventana\ popup\ contiene\ campos\ de\ texto\ \rightarrow Phishing\\ De\ lo\ contraio\ \rightarrow Legítimo\end{cases}$
Uso de IFrame	$SI\begin{cases}Usa\ iframe\ \rightarrow Phishing\\ De\ lo\ contrario\ \rightarrow Legítimo\end{cases}$
Edad del nombre de dominio	$SI\begin{cases}Edad\ del\ nombre\ de\ dominio\ \geq 6\ meses\ \rightarrow Legítimo\\ De\ lo\ contraio\ \rightarrow Phishing\end{cases}$
Registros DSN	$SI\begin{cases}No\ existen\ registros\ DNS\ para\ el\ dominio\ \rightarrow Phishing\\ De\ lo\ contrario\ \rightarrow Phishing\end{cases}$
Tráfico del sitio	$SI\begin{cases}Ranking\ del\ sitio < 100\ 000\ \rightarrow Legítimo\\ Ranking\ del\ sitio\ > 100\ 00\ \rightarrow Sospechoso\\ De\ lo\ contrario\ \rightarrow Phishing\end{cases}$
PageRank	$SI\begin{cases}PageRank < 0.2\ \rightarrow Phishing\\ De\ lo\ contrario\ \rightarrow Legítimo\end{cases}$
Índice de Google	$SI\begin{cases}Sitio\ indexado\ por\ Google\ \rightarrow Legítimo\\ De\ lo\ contrario\ \rightarrow Phishing\end{cases}$
Número de enlaces apuntando al sitio	$SI\begin{cases}Enlaces\ apuntando\ al\ sitio = 0\ \rightarrow Phishing\\ Enlaces\ apuntando\ al\ stio\ >0\ y\ \leq 2\ \rightarrow Sospechoso\\ De\ lo\ contrario\ \rightarrow Legítimo\end{cases}$
Característica basada en reportes estadísticos	$SI\begin{cases}Si\ el\ host\ o\ nombre\ de\ dominio\ está\ reportado\ \rightarrow Phishing\\ De\ lo\ contrario\ \rightarrow Legítimo\end{cases}$

Fuente: elaboración propia.

3.4. Preparación de los datos

Los datos del *data set* deben ser preparados para que cumplan con el formato y características requeridos por los algoritmos de aprendizaje. Para llevar a cabo esta tarea fue necesario el apoyo de ciertas librerías de software.

3.4.1. Librerías

Las librerías utilizadas principalmente para la preparación de los datos fueron Pandas y algunas clases de Scikit-Learn.

3.4.1.1. Pandas

Pandas es una librería para el lenguaje de programación Python que proporciona estructuras de datos de alto rendimiento y fáciles de usar y herramientas de análisis de datos. En el código Python desarrollado para este estudio se importó la librería Pandas bajo el alias “pd” para uso posterior.

3.4.1.2. OneHotEncoder

“OneHotEncoder” es una clase del módulo “*preprocessing*” de la librería Scikit-Learn para el lenguaje de programación Python. Una instancia de la clase OneHotEncoder codifica características de un *data set* de tipo entero como arreglo numérico “one-hot”. La codificación “one-hot” consiste en un grupo de bits en que las combinaciones permitidas son únicamente aquellas con un solo valor 1 y todos los demás son 0.

Se utilizó una instancia para codificar cada una de las categorías en cada una de las características del *data set.* Esta codificación será de utilidad para alimentar a los algoritmos de *machine learning* de manera eficiente.

La representación de los valores de las características de un atributo que contiene las tres categorías en el *data set* se ejemplifican en la tabla IV:

Tabla IV. **Ejemplo de codificación One-Hot para un atributo con las tres categorías**

Categoría	Valor en el data set	Representación en one-hot
Sito *phishing*	-1	[0 0 1]
Sospechoso	0	[0 1 0]
Legítimo	1	[1 0 0]

Fuente: elaboración propia.

3.4.1.3. Función train_test_split

"train_test_split" es una función del módulo "model_selection" de la librería Scikit-Learn. Fue utilizada para dividir el *data set* en el set de entrenamiento y set de prueba.

3.4.2. Carga de *data set*

La función "read_csv" del módulo Pandas lee un archivo CSV y lo convierte en una estructura de datos tabular de dos dimensiones y columnas etiquetadas llamada DataFrame.Con dicha función se cargó el archivo "phishing-sites.csv" que contiene el *data set* utilizado en el estudio y se obtuvo el DataFrame en la variable "data_set".

Figura 4. **Código Python para carga de *data set***

```
data_set = pd.read_csv('phishing-sites.csv')
```

Fuente: elaboración propia.

3.4.3. Set de entrenamiento y set prueba

El set de entrenamiento y el set de prueba se obtuvieron utilizando la función “train_test_split” explicada anteriormente. Dicha función recibe como parámetros la variable “data_set”, “test_size=0.2”, que indica que la proporción de datos en set de prueba es del 20% del total del data set; “random_state=42” fue utilizado para obtener instancias aleatorias de todos los datos del *data set* para los sets de entrenamiento y de prueba, el valor 42 indica que cada vez que se ejecute el método se obtendrán las mismas instancias en cada uno de los sets.

Figura 5. **Código Python para obtener sets de entrenamiento y de prueba**

```
entrenamiento, prueba = train_test_split(data_set, test_size=0.2, random_state=42)
```

Fuente: elaboración propia.

3.4.4. Características y etiquetas

Tanto las características como las etiquetas de los datos de entrenamiento y de prueba deben ser convertidas a arreglos que los algoritmos de entrenamiento de los modelos son capaces interpretar.

Figura 6. **Código Python para separar las columnas de características de la columna de resultados**

```
etiqueta = "Result"
X_entrenamiento = entrenamiento.drop([etiqueta], axis=1)
y_entrenamiento = entrenamiento[[etiqueta]]

X_prueba = prueba.drop([etiqueta], axis=1)
y_prueba = prueba[[etiqueta]]
```

Fuente: elaboración propia.

Tanto para los datos de entrenamiento como para los datos de prueba se separan las columnas con las características de la columna de resultados. A partir de lo cual se obtienen “X_entrenamiento”, “y_entrenamiento”, “X_prueba” y “y_prueba”, con las características de entrenamiento, las etiquetas de entrenamiento, las características de prueba y las etiquetas de prueba respectivamente.

3.4.5. OneHot Encoding

Tanto las características de entrenamiento como las de prueba se codifican utilizando una instancia de OneHotEncoder con el parámetro “categories=auto”, que indica que identifique las categorías automáticamente a partir de los datos proporcionados.

Figura 7. **Código Python para codificar los datos de entrenamiento y prueba con OneHot Encoding**

```
cat_encoder = OneHotEncoder(categories='auto')
X_entrenamiento = cat_encoder.fit_transform(X_entrenamiento)
X_prueba = cat_encoder.fit_transform(X_prueba)
```

Fuente: elaboración propia.

3.4.6. Formateo de *data sets*

Tanto los datos de entrenamiento como los datos de prueba deben ser convertidos a arreglos.

Figura 8. **Código Python para convertir a arreglos los datos de entrenamiento y prueba**

```
X_entrenamiento = X_entrenamiento.toarray()
y_entrenamiento = y_entrenamiento.values.ravel()

X_prueba = X_prueba.toarray()
y_prueba = y_prueba.values.ravel()
```

Fuente: elaboración propia.

3.5. Modelos predictivos

A continuación, se presenta el proceso de entrenamiento de cada uno de los modelos predictivos utilizando el respectivo algoritmo de aprendizaje.

3.5.1. Clasificador C4.5

Para el clasificador C4.5 se definió un conjunto de hiperparámetros para el entrenamiento del modelo, luego se buscaron los valores más apropiados para cado uno de ellos y así obtener el modelo de predicción óptimo.

3.5.1.1. Hiperparámetros

- *Criterion:* es la función que utilizará el algoritmo para dividir los nodos del árbol. En este caso de utiliza "*entropy"* para que utilice entropía de la información como criterio y sea un árbol C4.5.

- *Max Depth:* es la máxima profundidad del árbol.

- *Splitter:* es la estrategia que utilizará el algoritmo para dividir los nodos del árbol.

- *Min Samples Split:* es la cantidad mínima de instancias de datos de entrenamiento para dividir un nodo del árbol.

- *Min Samples Leaf:* es la cantidad mínima de instancias de datos de entrenamiento para llegar a un nodo hoja.

- *Presort:* define si se desea preordenar los datos de entrenamiento para acelerar la búsqueda de las divisiones de nodos en el árbol.

Figura 9. **Código Python para definir los hiperparámetros del clasificador C4.5**

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
from sklearn import tree

clasificador = tree.DecisionTreeClassifier()

param_distributions = {
    "criterion": ["entropy"], # Eso hace del árbol un C4.5
    "max_depth": [None, 1, 10, 100, 1000],
    "splitter": ["best", "random"],
    "min_samples_split": [2, 3, 4, 5],
    "min_samples_leaf": [1, 2, 3, 4],
    "presort": [True, False]
}

n_iter = 320
random_search = RandomizedSearchCV(clasificador,
                                   param_distributions,
                                   n_iter,
                                   cv=5,
                                   iid=False)
```

Fuente: elaboración propia.

3.5.1.2. Modelo óptimo

Para encontrar el modelo óptimo se utilizó la clase RandomizeSearchCV que entrena modelos utilizando diferentes valores para cada uno de los hiperparámetros. En total se entrenaron 320 modelos diferentes y se obtuvo como óptimo el modelo resultante con estos valores de los hiperparámetros:

- *Criterion: "entropy"*
- *Max Depth: None*
- *Splitter: "random"*

- *Min Samples Split: 2*

- *Min Samples Leaf: 1*

- *Presort: True*

3.5.2. Clasificador Naive Bayes

Para el clasificador Naive Bayes se definió un conjunto de hiperparámetros para el entrenamiento del modelo, luego se buscaron los valores más apropiados para cado uno de ellos y así obtener el modelo de predicción óptimo.

3.5.2.1. Hiperparámetros

- Priors: permite darle prioridad a una de las clases del conjunto de entrenamiento, ya que ambas clases, "*phishing*" y "legítimo", tienen la misma prioridad que el entrenamiento define como "None".

- Var Smoothing: parte de la variación más grande de todas las características que se agregan a las variaciones para la estabilidad del cálculo.

Figura 10. **Código Python para definir los hiperparámetros del clasificador Naive Bayes**

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
from sklearn.naive_bayes import GaussianNB
import numpy

clasificador = GaussianNB()

param_distributions = {
    "priors": [None],
    "var_smoothing": numpy.arange(0.01, 3, 0.01)
}

n_iter = 299
random_search = RandomizedSearchCV(clasificador,
                                   param_distributions,
                                   n_iter,
                                   cv=5,
                                   iid=False)
```

Fuente: elaboración propia.

3.5.2.2. Modelo óptimo

Para encontrar el modelo óptimo se utilizó la clase RandomizeSearchCV que entrena modelos utilizando diferentes valores de “var_smoothing” hasta encontrar el óptimo para el conjunto de datos. La variable “var_smoothing” tomará todos los valores entre 0.1 hasta 3, incrementado cada vez por 0.01. En total se entrenaron 299 modelos diferentes y se obtuvo como óptimo el modelo resultante con estos valores en los hiperparámetros:

- *Priors: None*
- *Var_smoothing: 0.64*

3.5.3. Red neuronal

Para la red neuronal se definió un conjunto de hiperparámetros para el entrenamiento del modelo, luego se buscaron los valores más apropiados para cado uno de ellos y así obtener el modelo de predicción óptimo.

3.5.3.1. Hiperparámetros

- *Hidden layer sizes:* es la cantidad de neuronas presente en la capa oculta de la red neuronal.

- *Activation:* es la función de activación para la capa oculta de la red neuronal.

- *Solver:* es la función que se utiliza para la optimización de la matriz de pesos de la red neuronal.

Figura 11. **Código Python para definir los hiperparámetros de la red neuronal**

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint
from sklearn.neural_network import MLPClassifier

clasificador = MLPClassifier()

param_distributions = {
    "hidden_layer_sizes": [(50, ), (100, ), (150, )],
    "activation": ["relu", "tanh", "logistic", "identity"],
    "solver": ["lbfgs", "sgd", "adam"]
}

n_iter = 36
random_search = RandomizedSearchCV(clasificador,
                                   param_distributions,
                                   n_iter,
                                   cv=5,
                                   iid=False)
```

Fuente: elaboración propia.

3.5.3.2. Modelo óptimo

Para encontrar el modelo óptimo se utilizó la clase RandomizeSearchCV que entrena modelos utilizando diferentes valores para cada uno de los hiperparámetros. En total se entrenaron 36 modelos diferentes y se obtuvo como óptimo el modelo resultante con estos valores en los hiperparámetros:

- *Hidden layer sises: 150*
- *Activation: "tanh"*
- *Solver:* "Adam"

4. RESULTADOS

A continuación, se presentan los resultados de las medidas de rendimiento para cada uno de los modelos predictivos con datos de entrenamiento y datos de prueba.

4.1. Rendimiento en datos de entrenamiento

A continuación, se muestran los resultados obtenidos con datos de entrenamiento para cada uno de los modelos predictivos.

4.1.1. Precisión

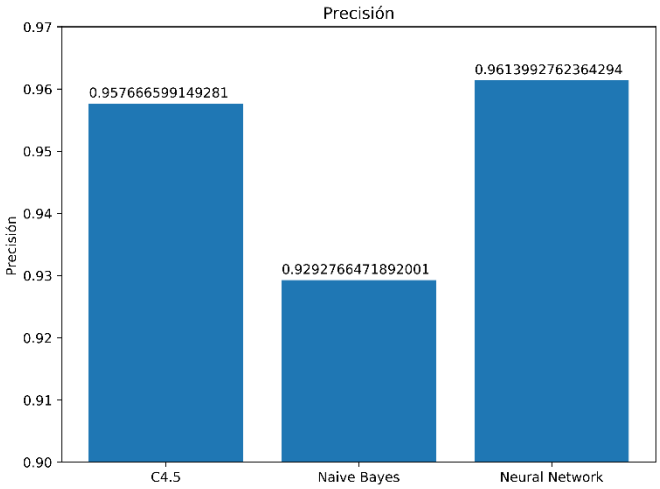
Las redes neuronales obtuvieron la mayor tasa de precisión, 96,14 %. Es decir que, del total de sitios clasificados como *phishing* por el modelo, el 96,14 % de ellos fueron clasificados correctamente como *phishing* (verdaderos positivos) y el 3,86 % fueron clasificados incorrectamente como *phishing* (falsos positivos).

Tabla V. **Precisión de modelos de predicción en datos de entrenamiento**

C4.5	95,77 %
Naive Bayes	92,93 %
Red neuronal	96,14 %

Fuente: elaboración propia.

Figura 12. **Precisión de modelos de predicción en datos de entrenamiento**



Fuente: elaboración propia.

4.1.2. Sensibilidad

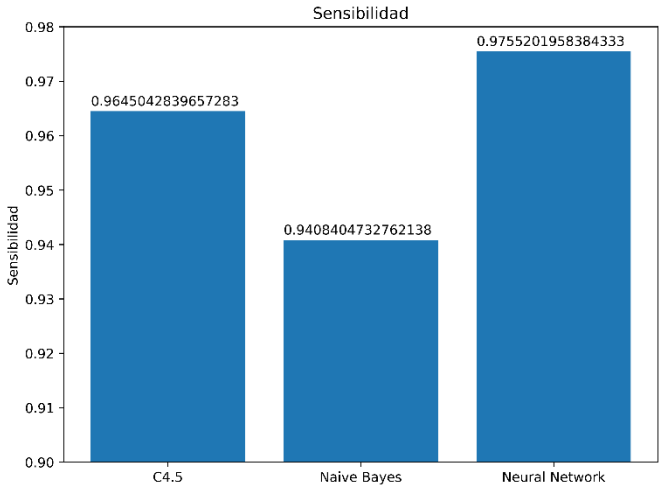
Las redes neuronales obtuvieron la mayor sensibilidad, 97,55 %. Es decir que, del total de sitios etiquetados como *phishing*, el modelo predictivo clasificó el 97,55 % de ellos correctamente como *phishing* (verdaderos positivos) y el 2,45 % fueron clasificados incorrectamente como sitios legítimos (falsos negativos).

Tabla VI. **Sensibilidad de modelos de predicción en datos de entrenamiento**

C4.5	96,45 %
Naive Bayes	94,08 %
Red neuronal	97,55 %

Fuente: elaboración propia.

Figura 13. **Sensibilidad de modelos de predicción en datos de entrenamiento**



Fuente: elaboración propia.

4.1.3. Medida F1

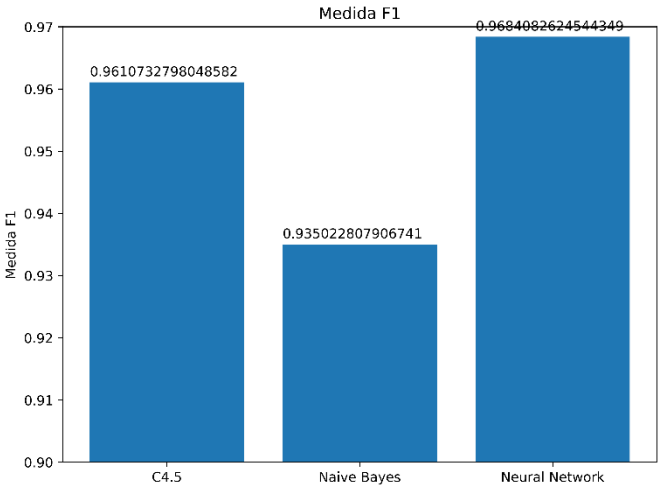
Las redes neuronales obtuvieron el mayor puntaje en medida F1, 96,84 %, lo cual indica que, en promedio, es el que tiene las tasas de precisión y sensibilidad más altas en comparación al clasificador Naive Bayes y el árbol de decisión C4.5.

Tabla VII. **Medida F1 de modelos de predicción en datos de entrenamiento**

C4.5	96,11 %
Naive Bayes	93,50 %
Red Neuronal	96,84 %

Fuente: elaboración propia.

Figura 14. **Medida F1 de modelos de predicción en datos de entrenamiento**



Fuente: elaboración propia.

4.1.4. Exactitud

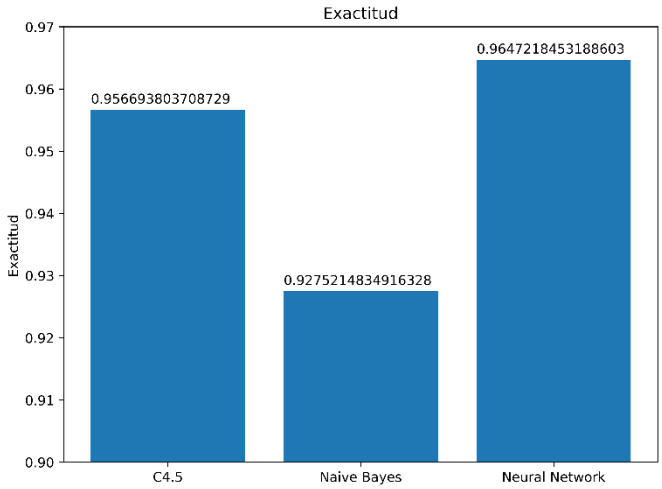
Las redes neuronales obtuvieron la mayor exactitud, 96,47 % Es decir que, sobre la cantidad total de observaciones, el modelo fue capaz de clasificar correctamente el 94,47 % de ellas.

Tabla VIII. **Exactitud de modelos de predicción en datos de entrenamiento**

C4.5	95,67 %
Naive Bayes	92,75 %
Red neuronal	96,47 %

Fuente: elaboración propia.

Figura 15. **Exactitud de modelos de predicción en datos de entrenamiento**



Fuente: elaboración propia.

4.1.5. Pérdida Hamming

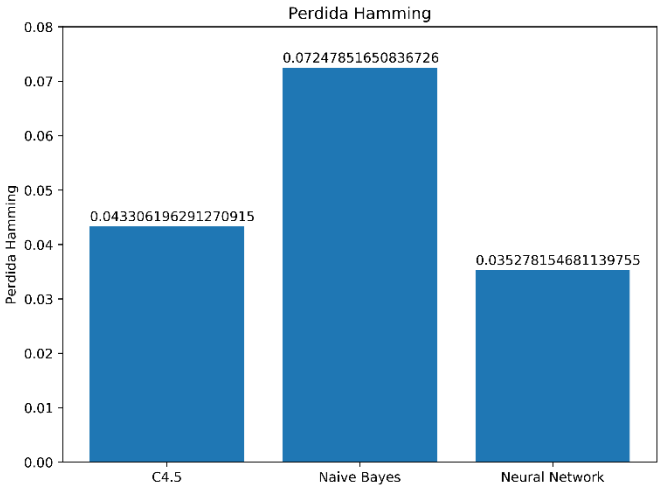
Las redes neuronales obtuvieron la menor cantidad de pérdida Hamming, 3,53 %. Es decir que, sobre la cantidad total de observaciones, el modelo clasificó incorrectamente el 3,53 % de ellas.

Tabla IX. **Pérdida Hamming de modelos de predicción en datos de entrenamiento**

C4.5	4,33 %
Naive Bayes	7,25 %
Red Neuronal	3,53 %

Fuente: elaboración propia.

Figura 16. **Pérdida Hamming de modelos de predicción en datos de entrenamiento**



Fuente: elaboración propia.

4.2. Rendimiento en datos de prueba

A continuación, se muestran los resultados obtenidos con datos de prueba para cada uno de los modelos predictivos.

4.2.1. Precisión

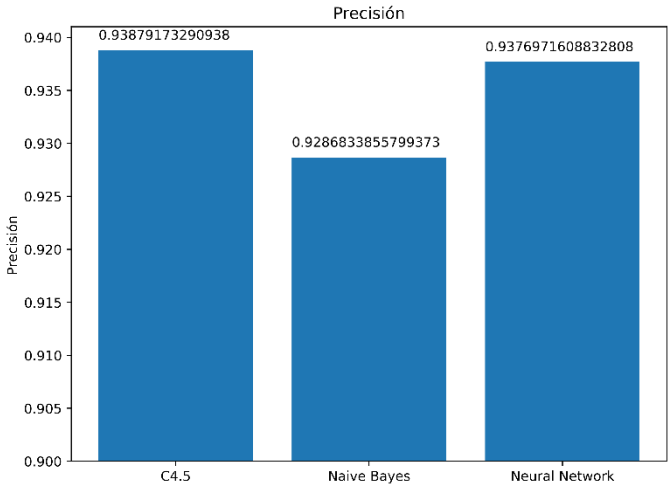
El árbol de decisión obtuvo la mayor tasa de precisión, 93,88 %. Es decir que, del total de sitios clasificados como *phishing* por el modelo, el 93,88 % de ellos fueron clasificados correctamente como sitios *phishing* (verdaderos positivos) y el 6,12 % fueron clasificados incorrectamente como sitios *phishing* (falsos positivos).

Tabla X. **Precisión de modelos de predicción en datos de prueba**

C4.5	93,88 %
Naive Bayes	92,87 %
Red Neuronal	93,77 %

Fuente: elaboración propia.

Figura 17. **Precisión de modelos de predicción en datos de prueba**



Fuente: elaboración propia.

4.2.2. Sensibilidad

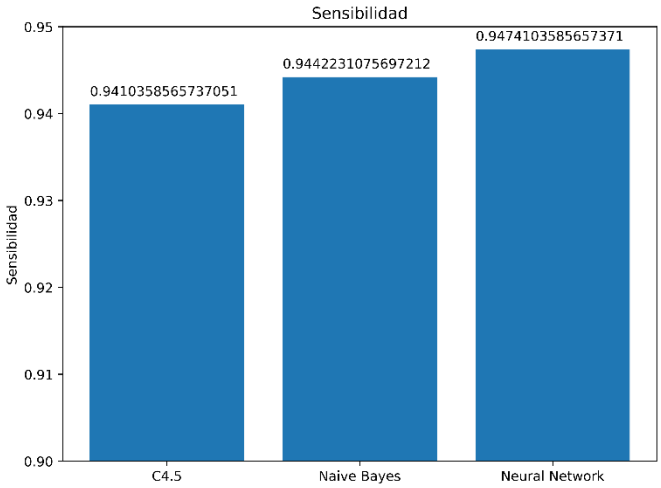
Las redes neuronales obtuvieron la mayor sensibilidad, 94,74 %. Es decir que, de la cantidad de sitios etiquetados como *phishing*, las redes neuronales clasificaron el 94,74 % de ellos correctamente como *phishing* (verdaderos positivos) y el 5,26 % fueron clasificados incorrectamente como sitios legítimos (falsos negativos).

Tabla XI. **Sensibilidad de modelos de predicción en datos de prueba**

C4.5	94,10 %
Naive Bayes	94,42 %
Red Neuronal	94,74 %

Fuente: elaboración propia.

Figura 18. **Sensibilidad de modelos de predicción en datos de prueba**



Fuente: elaboración propia.

4.2.3. Medida F1

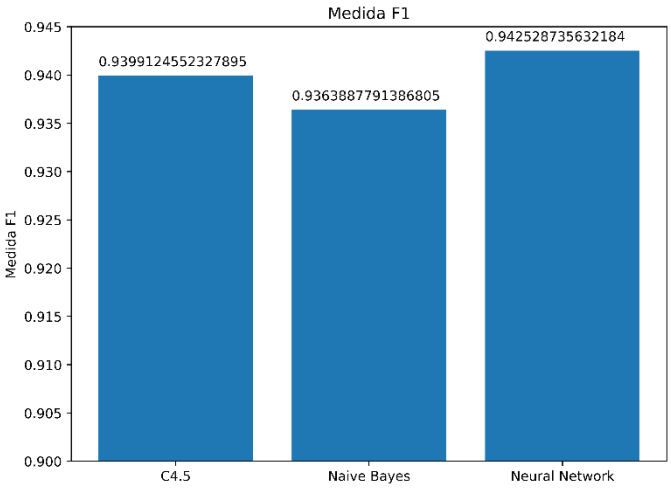
Las redes neuronales obtuvieron el mayor puntaje en medida F1, 94,25 %, lo cual indica que es el modelo que, en promedio, tiene las tasas de precisión y sensibilidad más altas en comparación al clasificador Naive Bayes y el árbol de decisión C4.5.

Tabla XII. **Medida F1 de modelos de predicción en datos de prueba**

C4.5	93,99 %
Naive Bayes	93,64 %
Red Neuronal	94,25 %

Fuente: elaboración propia.

Figura 19. **Medida F1 de modelos de predicción en datos de prueba**



Fuente: elaboración propia.

4.2.4. Exactitud

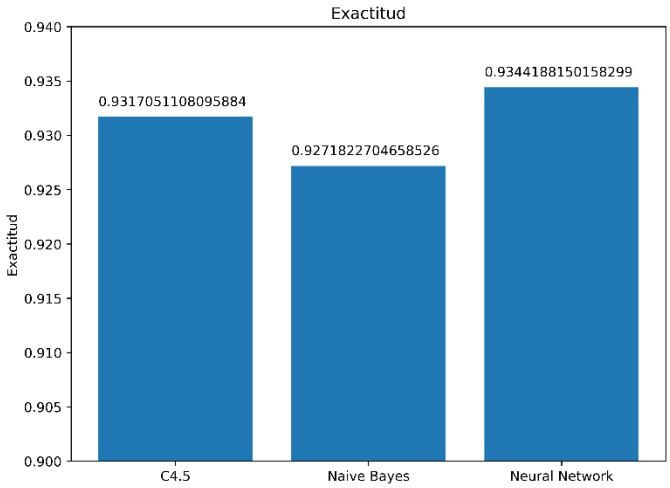
Las redes neuronales obtuvieron la mayor exactitud, 93,44 %. Es decir que, sobre la cantidad total de observaciones, el modelo fue capaz de clasificar correctamente el 93,44 % de ellas.

Tabla XIII. **Exactitud de modelos de predicción en datos de prueba**

C4.5	93,17 %
Naive Bayes	92,72 %
Red Neuronal	93,44 %

Fuente: elaboración propia.

Figura 20. **Exactitud de modelos de predicción en datos de prueba**



Fuente: elaboración propia.

4.2.5. Pérdida Hamming

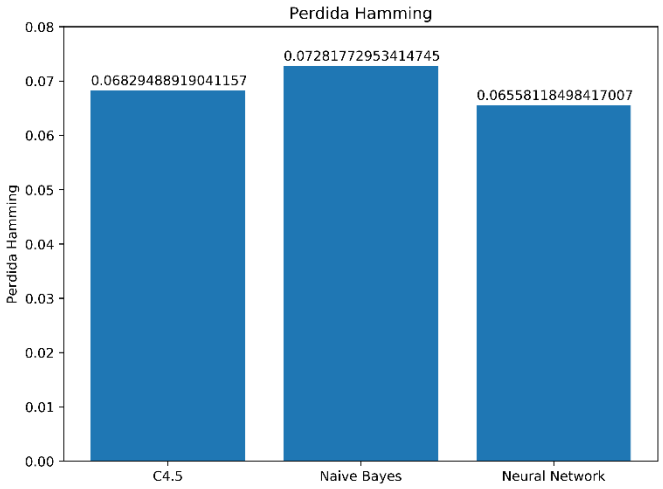
Las redes neuronales obtuvieron la menor cantidad de pérdida Hamming, 6,56 %. Es decir que, sobre la cantidad total de observaciones, el modelo clasificó erróneamente el 6,56 % de ellas.

Tabla XIV. **Pérdida Hamming de modelos de predicción en datos de prueba**

C4.5	6,83 %
Naive Bayes	7,28 %
Red Neuronal	6,56 %

Fuente: elaboración propia.

Figura 21. **Pérdida Hamming de modelos de predicción en datos de prueba**



Fuente: elaboración propia.

4.3. Resumen de resultados con datos de entrenamiento

A continuación, se presenta un cuadro resumen con las medidas de eficiencia predictiva de cada uno de los modelos predictivos sobre los datos de entrenamiento. Los resultados en negrilla indican cuál es el mejor de los tres modelos predictivos para cada una de las medidas.

Tabla XV. **Resumen de las medidas de eficiencia sobre los datos de entrenamiento**

	Precisión	Sensibilidad	Medida F1	Exactitud	Pérdida Hamming
C4.5	95,77 %	96,45 %	96,11 %	95,67 %	4,33 %
Naive Bayes	92,93 %	94,08 %	93,50 %	92,75 %	7,25 %
Redes Neuronales	**96,14 %**	**97,55 %**	**96,84 %**	**96,47 %**	**3,53 %**

Fuente: elaboración propia.

4.4. Resumen de resultados con datos de prueba

A continuación, se presenta un cuadro resumen con las medidas de eficiencia predictiva de cada uno de los modelos predictivos sobre los datos de prueba. Los resultados en negrilla indican cuál es el mejor de los tres modelos predictivos para cada una de las medidas.

Tabla XVI. **Resumen de las medidas de eficiencia sobre los datos de prueba**

	Precisión	Sensibilidad	Medida F1	Exactitud	Pérdida Hamming
C4.5	93,88 %	94,10 %	93,99 %	93,17 %	6,83 %
Naive Bayes	92,87 %	94,42 %	93,64 %	92,72 %	7,28 %
Redes Neuronales	**93,77 %**	**94,74 %**	**94,25 %**	**93,44 %**	**6,56 %**

Fuente: elaboración propia.

CONCLUSIONES

1. Los sitios web *phishing* poseen un conjunto de características que brinda información útil para su detección. Gracias a estudios previos, fue posible determinar veintinueve características utilizadas para el entrenamiento de modelos predictivos de sitios *phishing.*

2. La red neuronal obtuvo los mejores resultados en todas las medidas de eficiencia predictiva sobre los datos de entrenamiento.

3. En los datos de prueba la red neuronal obtuvo los mejores resultados en cuatro de las cinco medidas de eficiencia, siendo superados por el árbol de decisión C4.5 en la precisión. Sin embargo, dado que para la detección de sitios *phishing* la sensibilidad del modelo es más importante que la precisión, se concluye que la red neuronal es el mejor modelo predictivo.

4. Tras el análisis comparativo de las medidas de eficiencia predictiva se puede concluir que las redes neuronales son el modelo más preciso para la predicción de sitios web *phishing*, haciendo uso de las características discriminativas.

RECOMENDACIONES

1. Considerar el uso de una red neuronal sobre otros modelos predictivos para la implementación de una herramienta de detección de sitios web *phishing.*

2. Mantener versiones actualizadas del modelo predictivo mediante el reentrenamiento constante con nuevos datos obtenidos por los procedimientos de minería de datos.

3. Mantener una investigación constante sobre las nuevas técnicas utilizadas por los atacantes *phishing* para así identificar nuevas características discriminativas.

4. Ampliar este estudio comparando la efectividad predictiva de otros algoritmos de *machine learning* como Random Forest o Support Vector Machines.

5. Ampliar este estudio comparando otras librerías de *machine learning* tales como Keras, TensorFlow o PyTorch, ya que cada librería cuenta con implementaciones diferentes de los algoritmos que, potencialmente, podrían generar modelos predictivos más precisos.

BIBLIOGRAFÍA

1. ABURROUS, Maher y HOSSAIN, Muhammad. *Predicting phishing websites using classification mining techniques with experimental case studies*. [en línea]. ResearchGate. <https://www.researchgate.net/publication/224151983_Predicting_Phishing_Websites_Using_Classification_Mining_Techniques_with_Experimental_Case_Studies>. [Consulta: mayo, 2019]

2. BASNET, Ram; MUKKAMALA, Srinivas y SUNG, Andrew*. Detection of phishing attacks: a machine learning approach.* [en línea]. Springer. <https://link.springer.com/chapter/10.1007/978-3-540-77465-5_19>. [Consulta: mayo, 2019]

3. DHAMIJA, Rachna y TYGAR, Doug. *The battle against phishing: dynamic security skins*. [en línea]. Berkeley. < https://people.eecs.berkeley.edu/~tygar/papers/Phishing/Battle_against_phishing.pdf>. [Consulta: junio, 2019]

4. GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* 2a ed. O'Reilly Media. Estados Unidos: 2019. 856p.

5. GOODFELLOW, Ian; BENGIO, Yoshua y COURVILLE, Aaron. *Deep learning (adaptive computation and machine learning series).* 1a ed. The MIT Press. Estados Unidos: 2016. 801p.

6. RAMI, Mohammad. *Predicting phishing websites based on self-structuring neural network. Neural computing and applications.* [en línea]. ResearchGate. < https://www.researchgate.net/publication/259384371_Predicting_Phishing_Websites_based_on_Self-Structuring_Neural_Network>. [Consulta: julio, 2019]

7. R. M., Mohammad; F., Thabtah y L., McCluskey. *An assessment of features related to phishing websites using an automated technique.* [en línea]. ResearchGate. < https://www.researchgate.net/publication/261081735_An_assessment_of_features_related_to_phishing_websites_using_an_automated_technique>. [Consulta: julio, 2019]

8. R.M., Mohammad; F., Thabtah y L., McCluskey. *Intelligent rule-based phishing websites classification.* [en línea]. ResearchGate. < https://www.researchgate.net/publication/261636543_Intelligent_Rule_based_Phishing_Websites_Classification>. [Consulta: agosto, 2019]

9. SHALEV-SHWARTZ, Shai y BEN-DAVID, Shai. *Understanding machine learning: from theory to algorithms.* 1a ed. Cambridge University Press. Inglaterra: 2016. 449p.