Sección de Matemáticas
Universidad de La Laguna

Nuria Reyes Dorta

Técnicas analíticas para detectar problemas de ciberseguridad

Analytical techniques to detect cybersecurity problems

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Julio de 2023

Dirigido por
Pino Caballero Gil
Carlos Rosa Remedios

Pino Caballero Gil
Departamento Ingeniería
Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Carlos Rosa Remedios
Departamento Ingeniería
Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Agradezco a las personas que me han apoyado tanto en los buenos como en los malos momentos. También agradezco a mis tutores por todo lo que me han enseñado y por las oportunidades que me han ofrecido.

Nuria Reyes Dorta
La Laguna, 10 de julio de 2023

Resumen · Abstract

Resumen

En este trabajo se analiza la aplicación práctica de varios algoritmos de Machine Learning, y en particular de Deep Learning, para el análisis de un conjunto de datos público que permite clasificar URL fraudulentas con el objetivo de prevenir diferentes tipos de ciberataques. Se presta especial atención a la necesaria fase de preparación de datos, así como a la construcción de cada una de las capas ocultas de las redes neuronales utilizadas en los modelos de Deep Learning. La comparación realizada de los algoritmos utilizados para el análisis del conjunto de datos y la evaluación de los modelos con diferentes conjuntos de datos ofrece resultados interesantes de los que aquí se extraen conclusiones prometedoras. Además, en esta memoria también se proporciona una primera aproximación a la aplicación de Quantum Machine Learning para el mismo objetivo, de la que se obtienen algunos resultados alentadores.

En este documento se incluye un breve estado de arte de varios problemas actuales de ciberseguridad y algunas medidas que se utilizan para resolverlos, relacionadas con Machine Learning. A continuación se comentan los conjuntos de datos más utilizados en ciberseguridad. Además, se introducen los conceptos matemáticos de Machine Learning y de computación cuántica necesarios para la comprensión de este trabajo.

Palabras clave: *Ciberseguridad– Dataset – Curva ROC– Matriz de Confusión– Machine Learning– Regresión Logística– Árbol de Decisión– Support Vector Machine- Red Neuronal- Programación Cuántica.*

Abstract

This work analyzes the practical application of several Machine Learning algorithms, and in particular Deep Learning, for the analysis of a public dataset that allows the classification of fraudulent URLs with the aim of preventing different types of cyberattacks. Special attention is paid here to the necessary data preparation phase, as well as to the construction of each of the hidden layers of the neural networks, used in the Deep Learning models. The comparison made of the algorithms used for the analysis of the dataset and the evaluation of the models with different datasets offers interesting results from which promising conclusions are drawn. In addition, this report also provides a first approximation to the application of Quantum Machine Learning, from which some encouraging results are obtained. This document includes a brief state of the art of several current cybersecurity problems and some measures used to solve them, related to Machine Learning. Next, the most widely used datasets in cybersecurity are discussed. In addition, the mathematical concepts of Machine Learning and quantum computing necessary for the understanding of this work are introduced.

Keywords: *Cybersecurity– Dataset – ROC curve– Confusion Matrix–Malicious URL– Machine Learning– Logistic Regression– Decision Tree– Support Vector Machine– Neural Network – Quantum Computing.*

Contenido

1

Preliminares

Hoy en día, con los grandes y rápidos avances tecnológicos, todas las empresas y organismos requieren cada vez más de ordenadores para llevar a cabo su actividad y almacenar todo tipo de información. Una de las consecuencias negativas de este cambio es que dichas entidades son ahora más sensibles y vulnerables ya que pueden ser objeto de ciberataques cuyo objetivo principal puede ser el robo de datos, el bloqueo de actividad, etc. Por ello, es absolutamente imprescindible que todas las entidades con actividad informática apliquen medidas que permitan detectar y evitar los ciberataques más frecuentes, antes de su materialización [1].

1.1. Tipos de ciberatacantes

Un ciberataque se define como un conjunto de acciones ofensivas contra sistemas de información con el objetivo de dañar, robar datos, alterar información, o espiar o destruir organizaciones o personas.

Los atacantes se pueden clasificar según el método que utilizan para llevar a cabo sus ataques. A continuación, se definen algunos de los tipos de ciberatacantes más relevantes [2]:

- *Hacker*: término general que se utiliza para referirse a quien posee conocimientos profundos sobre ciberseguridad y en particular sobre metodologías de acceso a sistemas informáticos.
- *Hacker* ético o de sombrero blanco, o *white hat hacker*: hacker especializado en comprobar posibles intrusiones o vulnerabilidades en sistemas informáticos, para garantizar la ciberseguridad de los sistemas que analiza.
- *Hacker* de sombrero gris, o *grey hat hacker*: hacker que se dedica a realizar acciones sin autorización, para descubrir y revelar vulnerabilidades en sistemas, pero no siempre siguen los procedimientos legales adecuados.
- *Hacker* malicioso o *black hat hacker*: es un hacker que actúa con intenciones maliciosas, con el objetivo de obtener beneficios personales ilegales, realizando

ciberataques para robar información, causar daños o llevar a cabo toda una variedad de actividades dañinas, como el robo de información, el sabotaje de sistemas, el acceso no autorizado o el fraude.
- *Cracker*: tipo de hacker malicioso que se dedica a romper la seguridad de los sistemas informáticos con el objetivo de dañar los sistemas de información que ataca, robar información confidencial o realizar actos ilegales.
- *Script kiddy*: Es alguien con habilidades técnicas básicas o nulas que utiliza herramientas y programas desarrollados por otros para llevar a cabo ataques. Suele realizar sus acciones solo por diversión, sin un objetivo claro más allá de causar molestias o interrupciones.
- *Insider*: suele ser una persona que ha sido despedida, que conoce el entorno y las vulnerabilidades de seguridad y ataca por venganza.
- *Phreaker*: persona que se dedica a explorar y manipular sistemas de telecomunicaciones, especialmente sistemas telefónicos, para obtener acceso no autorizado o realizar actividades ilegales.
- Espía cibernético: agente de inteligencia, gubernamental o corporativo, que realiza actividades de espionaje en el ámbito digital, para acceder a información clasificada o confidencial de otro país, organización o individuo.
- *Hacktivista*: persona o grupo que realiza ataques informáticos con motivaciones políticas o sociales, con el objetivo de hacer una declaración o causar un impacto en la opinión pública mediante el acceso no autorizado a sistemas o la difusión de información comprometedora.

Para evitar que los ciberatacantes puedan robar, dañar, etc, se hace necesario utilizar un conjunto de medidas tecnológicas, procedimientos y políticas diseñadas para prevenir esos posibles ataques, daños, o accesos ilegales a ordenadores, redes, programas o datos. El término ciberseguridad se refiere a la práctica de proteger los sistemas informáticos, redes, dispositivos y datos digitales contra ataques, daños, acceso no autorizado y otras amenazas cibernéticas. Es decir, el objetivo de la ciberseguridad es proteger los sistemas frente a todos los tipos de ciberatacantes, con el fin de que no puedan acceder a los sistemas.

1.2. Estadísticas de ciberataques recientes

Los problemas de ciberseguridad suelen suponer un impacto económico muy importante para las empresas y organismos atacados y, especialmente, en su reputación. El 43 % de las víctimas de ciberataques han sido pequeñas empresas según un informe sobre violación de datos de 2019. Este dato demuestra que el cibercrimen busca sacar provecho económico de la escasa seguridad que tienen este tipo de empresas, debido a que casi no tienen recursos para poder defenderse, convirtiéndolas así en las más vulnerables [3].

Para tener una idea de la importancia de estos delitos, se mencionan a continuación algunos datos estadísticos publicados por el Ministerio del Interior de España [4]. En 2021 se llevaron a cabo en España 305.477 ciberdelitos, de los cuáles: 267.011 fueron fraudes informáticos, 17.319 fueron amenazas y coacciones, 10.476 falsificaciones informáticas y 5.342 accesos e interceptación ilícita, 1.628 delitos sexuales, 2.138 interferencias en los datos y en el sistema, y 137 contra la propiedad industrial/intelectual. Las cinco comunidades con mayor índice de ciberdelicuencia fueron: Madrid con 53.039, Cataluña con 49.755, Andalucía con 42.493, Comunidad Valenciana con 29.508 y Galicia con 22.010. Por último, en la Figura 1.1 se muestra la evolución del número de ciberdelitos cometidos en estos últimos años.

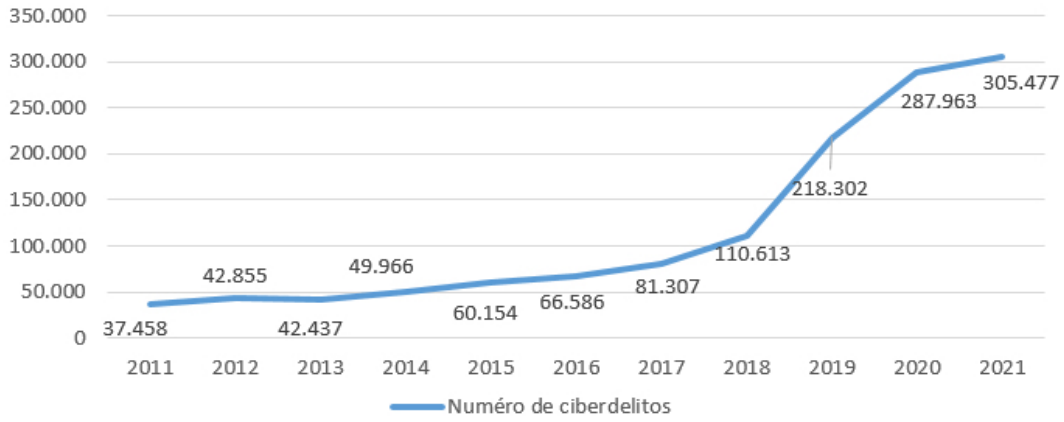

Figura 1.1. Evolución del número de ciberdelitos en España entre 2011 y 2021

Se puede apreciar que desde que empezó la pandemia vinculada a la Covid-19 han aumentado significativamente los ciberataques, debido a la necesidad de hacer todo telemáticamente con ordenadores. Posteriormente, dicho crecimiento no se ha compensado, lo que evidencia la importancia de trabajos como éste, en los que se analicen soluciones de ciberseguridad.

1.3. Tipos de ciberataques analizados

Cuando una empresa queda expuesta a un ciberataque, normalmente tiene pérdidas económicas que pueden llegar a ser muy importantes, e incluso suponer su bancarrota. Además, puede que el ciberataque implique una filtración de datos que tenga implicaciones sobre terceros. Todo esto se relaciona con lo que se conoce como "riesgo de ciberseguridad".

A continuación se definen algunos de los tipos de ciberataques más relacionados con el estudio realizado.

1.3.1. *Malware*

El *malware* es un conjunto de herramientas informáticas diseñadas con el objetivo de ser utilizadas par desarrollar actividades maliciosas en un sistema informático. Se distinguen diferentes tipos de *malware*:

- Virus: es un tipo de programa informático malicioso que se propaga e infecta otros archivos o programas. En particular, al ejecutarse, suele adjuntarse al código de otra aplicación instalada en el ordenador afectado y cuando se ejecuta dicha aplicación, el código del virus también lo hace, infectando así otros archivos o ejecutando código malicioso.
- Gusano (Worm): malware similar a un virus, pero con la capacidad de replicarse y propagarse sin infectar otros archivos o programas pues se propaga a través de redes y sistemas, explotando sus vulnerabilidades de seguridad.
- Troyano: malware que se presenta como un programa legítimo, pero en realidad contiene un código malicioso oculto. Una de sus características es que no se autorreplica sino que normalmente necesita de ingeniería social para su propagación. Su principal característica es que se suele camuflar como una herramienta útil para abrir una vía de comunicación que el atacante utiliza para controlar el sistema atacado. Se considera uno de los ataques más peligrosos ya que a menudo están diseñados para robar información financiera.
- *Ransomware* o secuestro de datos: es un tipo de malware que cifra los archivos o bloquea el acceso a un sistema y exige un rescate para restaurar el acceso o descifrar los archivos.
- *Spyware* o programa espía: es un tipo de malware diseñado para recopilar información sobre la actividad de la víctima sin su consentimiento. Dicha información normalmente es enviada a otro ordenador de manera remota sin consentimiento de la víctima.
- *Adware*: malware que muestra anuncios no deseados o intrusivos en un dispositivo infectado. A menudo se instala sin el consentimiento del usuario y puede ralentizar el sistema y afectar la experiencia de navegación.
- *Rootkit*: tipo de malware diseñado para ocultar su presencia y proporcionar acceso privilegiado al atacante en un sistema comprometido. Puede modificar o reemplazar componentes del sistema operativo para evitar su detección y facilitar el control total del sistema.
- *Botnet*: red de dispositivos infectados por malware controlados por un atacante de forma remota. Los dispositivos infectados se utilizan para llevar a cabo actividades maliciosas, como ataques de denegación de servicio distribuido o DDoS (Distributed Denial of Service) o envío masivo de correo no deseado (*spam*).

1.3.2. *Phishing*

El *phishing* es un tipo de software malicioso que se emplea, por lo general, para robar datos de usuarios. Se basa en engañar a la víctima para que abra un mensaje de texto, SMS o correo electrónico mediante un enlace malicioso. Se trata de uno de los ataques más peligrosos, ya que es una técnica sencilla y muy fácil de utilizar. Algunos tipos de ataque de este tipo son:

- *Phishing* por correo electrónico: es el tipo más común de phishing. Los atacantes envían correos electrónicos falsos que se hacen pasar por entidades legítimas, como bancos, empresas o servicios en línea, con el objetivo de engañar a los destinatarios para que revelen información personal, como contraseñas, números de tarjetas de crédito o datos de inicio de sesión.
- *Smishing* o *phishing* por SMS: se lleva a cabo a través de un mensaje *SMS*, donde se solicita a la víctima que verifique sus datos en un enlace que lo dirige a una web falsa.
- *Vishing* o *phishing* por llamada telefónica: la víctima recibe una llamada o un correo pidiendo que contacte con un número gratuito que solicita los datos personales y bancarios.
- *Pharming*: tipo de *phishing* en el que los atacantes redirigen el tráfico de Internet de los usuarios a sitios web falsos sin su conocimiento. Se necesita que los conocimientos del atacante sean mayor debido a que tiene que atravesar los sistemas de seguridad de una web. Este tipo de ataque se utiliza para obtener acceso a los servidores
- *Whaling*: este ataque se centra en un perfil de alto directivo. El objetivo es robar información vital, ya que los que ocupan altos cargos suelen tener acceso ilimitado a información confidencial. El delincuente manipula a la víctima para que le haga transferencias de un alto valor.
- *Spear phishing*: ataque informático que tiene como objetivo una persona o empleado de una compañía. Para este tipo de ataque se necesita un estudio previo en el que los criminales recopilan información de la víctima para ganarse su confianza.

1.3.3. Ataque *web*

Los ataques *web* se utilizan para engañar a los usuarios que utilizan las redes o *webs* y los sistemas. Algunos de los tipos más relevantes de ataques *web* son los siguientes:

- Inyección de código *SQL*: los atacantes insertan código malicioso, como comandos SQL, en entradas de datos de aplicaciones o páginas web que presentan errores en sus diseños. Los ciberdelincuentes son capaces de engañar a la aplicación para que haga uso de los comandos que deseen y acceder a la base de datos que quieran.

- *XSS* o *Cross-Site Scripting*: los atacantes insertan código malicioso en páginas web o campos de entrada para que se ejecute en el navegador de los usuarios. Concretamente, utiliza recursos web para ejecutar secuencias de comandos en el navegador de la víctima.
- *CSRF* o *Cross-Site Request Forgery*: los atacantes engañan a los usuarios para que realicen acciones no deseadas en una aplicación web sin su conocimiento o consentimiento. Esto se logra mediante la explotación de la confianza de la aplicación en la autenticidad de las solicitudes enviadas por los usuarios.
- Denegación de Servicio (*DoS*, Denial of Service) es uno de los ataques web más comunes. Tiene como objetivo inhabilitar el uso del sistema atacado, con el fin de bloquear el servicio para el que está destinado. Hay varias formas de ataques *DoS* [5], tales como:
  1. Ataque *ICMP Flood*: este ataque permite agotar el ancho de banda de la víctima y consiste en enviar una gran cantidad de información usando paquetes *ICMP Echo Request*.
  2. *Ping of the Dead*: es similar al anterior pero este utiliza paquetes de 65536 bytes, haciendo que el sistema operativo no sepa cómo manejarlo y a su vez se bloquee por intentarlo.
  3. Ataque *Fraggle Dos*: consiste en enviar mucho tráfico *UDP* a una dirección IP de *broadcast*.
- Ataque distribuido de denegación de servicio o *DDoS*: es un ataque de red que consiste en colapsar a una víctima desde múltiples equipos de origen.
- *Remote to User (R2L)*: es un ataque de red en el que el atacante entrega una serie de paquetes a otra máquina o servidor a través de una red donde el atacante no tiene autorización de usuario local.
- Ataque *User to Root (U2R)*: es un ataque a la red donde el atacante intenta acceder a los recursos de la red como un usuario normal, y después de varios intentos, obtiene acceso completo.
- Ataque *Probe*: es un tipo de ataque en el que el intruso busca equipos de red donde hay fallos en el diseño o puertos abiertos, luego los usa para obtener acceso no autorizado a información personal.

1.4. Medidas de ciberseguridad

Antes de describir diferentes tácticas para afrontar los ciberataques mencionados, es conveniente explicar la diferencia entre *software* libre y *software* propietario [6]. Por una parte, el *software* libre es aquel cuyo código fuente está disponible de forma que se puede ser estudiado, modificarse y utilizarse con cualquier fin. Por otra parte, el *software* propietario es aquel cuyo código fuente no está disponible para los usuarios, impidiendo que puedan estudiarlo y modificarlo. Algunas de las ventajas del *software* libre son las siguientes:

- Fácil acceso y bajo coste de adquisición.
- No es necesario adquirir nuevas licencias. No obstante, existe *software* libre cuyas actualizaciones si requieren la obtención de licencias.
- Mayor capacidad y flexibilidad para acceder a programas y adaptarlos.
- La disponibilidad del código fuente permite auditarlo y verificarlo, lo que habilita un espacio para evaluar la seguridad y fiabilidad de mayor intensidad que en el caso de *software* propietario.

El *software* libre permite identificar las posibles vulnerabilidades que puede contener, pero a cambio, sabemos que el ciberatacante también puede acceder al código fuente, lo que le permite encontrar fallos en la seguridad que puede aprovechar más fácilmente. De hecho, se puede apreciar que el sistema operativo con menor número de ataques es el sistema libre y abierto, *Linux* [7]. La elección del *software* es uno de los pasos importante que se debe decidir al usar un sistema informático. Una vez tomada esa decisión, es necesario desarrollar un plan de seguridad que se adapte a dicho *software*.

A continuación, se establece una clasificación de las medidas más comunes en ciberseguridad para proteger los sistemas frente a los ciberataques más comunes. En primer lugar, se indican algunas de las medidas más tradicionales.

1.4.1. Medidas de ciberseguridad tradicionales

Algunos de los mecanismos más usados para garantizar la seguridad en empresas [2] son los siguientes:

- Limitar el acceso a algunos programas o ficheros mediante cifrado o claves.
- Otorgar los privilegios mínimos y necesarios a todos los usuarios del sistema, evitando así dar más permisos de los necesarios.
- Aplicar una gestión de la explotación del software.
- Controlar la información que se recibe y que sale del sistema.

Las tres características de la información que se deben proteger en todas las empresas u organizaciones son: confidencialidad, integridad y disponibilidad [2].

- Confidencialidad: característica según la cual solo los usuarios autorizados pueden leer la información. La información confidencial siempre debe ser cifrada para que aquellas personas que no tengan autorización no puedan acceder a ella.
- Integridad: característica según la cual está garantizado que la información no ha sido modificada. Para garantizar la integridad es conveniente llevar a cabo algunos procedimientos como:
  1. Monitorizar el tráfico de la red para detectar las posibles intrusiones.
  2. Reuniones para evaluar y modificar las políticas de seguridad.
  3. Implementar procesos de copias de seguridad y recuperación.

- Disponibilidad: característica según la cual solo los usuarios autorizados pueden acceder a la información. Además, asegura que una organización tenga acceso oportuno y confiable a los activos y sistemas de información. Para garantizarla es conveniente implementar políticas de control como:
  1. Acuerdos de nivel de servicio o *SLA, Service Level Agreement.*
  2. Disponibilidad de sistemas a través de redundancia y alta disponibilidad.
- Autenticidad: característica según la cual está garantizado que el usuario que está asociado a un documento o quiere acceder a un sistema es quien dice ser.

Uno de los denominadores comunes de varias de las características mencionadas es la necesidad de autorización para acceder a la información. Esto implica definir un control de accesos para controlar quién accede a la información. Posiblemente ésta es una de las medidas de seguridad más importantes que se pueden establecer. Además, el tema del control de acceso es bastante complejo por lo que a continuación solo se indican unas pocas nociones básicas [8].

A continuación se describen brevemente las dos técnicas de ciberseguridad más esenciales: control de accesos y cifrado.

Control de accesos

El medio de autenticación para control de accesos más usado es el basado en contraseñas, que se utilizan para acceder a servicios o para proteger el contenido de los ficheros almacenados en los ordenadores [6]. Para evitar una mala gestión de contraseñas, es recomendable seguir las siguientes pautas:

- No compartir nunca las contraseñas con otras personas.
- Utilizar combinaciones de al menos ocho caracteres en los cuales se mezclen números, letras y símbolos.
- No utilizar la misma contraseña en diferentes servicios, ya que si un atacante obtiene la contraseña de un servicio, automáticamente podría utilizar esa contraseña en otros servicios que utilicemos.
- Para la pregunta de seguridad con respuesta que se utiliza cuando olvidamos nuestra contraseña y queremos recuperarla o cambiarla es recomendable registrar respuestas tan complejas como la contraseña.
- Utilizar un gestor de contraseñas para que solo se necesite recordar una contraseña robusta que dé entrada a la aplicación. El resto de contraseñas se almacenan cifradas y solo se pueden recuperar con la contraseña inicial.
- Implementar un ciclo de vida de contraseñas robustas, hay que cambiar las contraseñas de forma regular, especialmente aquellas vinculadas a servicios críticos en los que se gestione y haga uso de dato sensibles.

Cuando se realizan bajas o altas en la organización, hay que pedir siempre la cancelación de credenciales o de cambio de contraseña. La parte más débil

del control de acceso es la administración de contraseñas.Todos los accesos, al igual que otros eventos del sistema, se registran en archivos *log*. Entre los errores comunes que se puede encontrar en la administración de contraseñas son:

1. Comprobación defectuosa de la identidad.
2. Falta de borrado y desactivación de usuarios y activos no utilizados.
3. Reinicio de contraseña sin comprobar la identidad del peticionario.
4. Compartición de contraseñas.
5. Cesión de sesiones.

Existen varios modelos de autorización para control de accesos, como *Bell-LaPadula* y el de *Clark-Wilson*. Por un lado, el modelo de seguridad Bell-Lapadula consiste en dividir el permiso de acceso de los usuarios a la información en función de etiquetas de seguridad. Por otro lado, el modelo Clark-Wilson es un modelo orientado a la integridad de los datos.

Cifrado

El cifrado es la técnica de ciberseguridad más popular. Cifrar es una operación reversible por la que convertimos un mensaje en un conjunto de datos cuasi aleatorios, el mensaje cifrado. Lo importante es que el descifrado tenga una dificultad computacional cuando no se tiene la clave. Entre las ciberamenazas que se pueden mitigar usando cifrado destacan:

- Modificación y generación malintencionadas de datos.
- Compromiso de medios de autenticación.
- Compromiso de claves.
- Análisis de tráfico.
- Infracción de derechos de autor.
- Denegación de la autoría propia.
- Denegación de la autoría ajena.
- Código malicioso.
- Errores en los mensajes.
- Errores en el almacenamiento.

En consecuencia, algunos de los usos de las técnicas de cifrado son para:

- Mantener secreta la información y los mensajes secretos.
- Identificar al autor de una información o de un mensaje.
- Identificar a las partes de un acceso o comunicación.
- Probar el conocimiento de algo, sin revelarlo.
- Comprobar la consistencia de unos datos.
- Probar la existencia e identificar inequívocamente un mensaje.
- Probar cuando se ha enviado o recibido un mensaje.
- Ocultar el hecho de que se transmiten mensajes cifrados.

1.4.2. Medidas de ciberseguridad basadas en *Machine Learning*

Machine Learning o aprendizaje automático es una rama de la Inteligencia Artificial (IA) que permite que las máquinas aprendan sin ser expresamente programadas para ello. Es decir, es capaz de convertir una muestra de datos en un programa informático capaz de extraer datos de nuevos conjuntos de datos para los que no ha sido entrenado previamente [10].

El aprendizaje automático se basa en reconocer un conjunto de patrones que permitan extraer conclusiones acerca de un fenómeno observado. Por tanto, el elemento principal que requiere el aprendizaje es la experiencia después de una etapa de experimentación sobre un fenómeno para realizar posteriormente un aprendizaje sobre el mismo.

Matemáticamente, se pueden definir varios conceptos formales de Machine Learning como se muestra a continuación.

Se denota por O al conjunto de todas las posibles observaciones del fenómeno concreto observado y $S = \{o_1, \cdots, o_n\}$ al conjunto finito de observaciones que se han tomado. Este espacio de observaciones suele descomponerse como producto de dos espacios de dimensión menor, $O = X \times Y$. Se espera encontrar una relación entre X y Y. Por defecto, O es un espacio con estructura medible. Debido a eso, se introduce algunas notas sobre Espacios de Medida.

Sea $A \subseteq P(O)$ una colección de subconjuntos de O. Se dice que A es una θ-álgebra en O si cumple las siguientes propiedades:

- $O \in A$
- $O \backslash \mathrm{B}, \mathrm{B} \in A$
- $\forall \quad \{A_n\}_{n \in \mathbb{N} \subseteq A} \Longrightarrow \cup_n A_n \in A$

Sea A una θ-álgebra en O. Se llama (O, A) al espacio medible, si existe una aplicación:

$$\begin{gathered} \mu : A \longrightarrow [0, \infty] \\ B \longrightarrow \mu(B) \end{gathered}$$

Se dice que es una medida cuando:

- $\mu(\varnothing) = 0$
- $\{A_n\}_n \subseteq A, A_n \cap A_m = \varnothing, n \neq m \quad \mu(\cup_n A_n) = \sum_n \mu(A_n)$

(O, A, μ) se conoce como espacio de medida.

De la definición anterior se deduce la siguiente Proposición.

Proposición 1 *Sea (O, A, μ) un espacio de medida. $\forall C, D \in A$.*

i $\mu(D) = \mu(D \cap C) + \mu(D \cap C^c)$
ii *Si* $C \subseteq D \Longrightarrow \mu(D) = \mu(C) + \mu(D \setminus C)$ *y* $\mu(C) \leq \mu(D)$
iii *Si* $C \subseteq D$ *y* $\mu(C) = \infty \Longrightarrow \mu(D) = \infty$
iv $\mu(C \cup D) \subseteq \mu(C) + \mu(D)$
v $\mu(C \cup D) + \mu(C \cap D) = \mu(C) + \mu(D)$

Demostración. i $D = (D \cap C) \cup (D \cap C^c) \Longrightarrow \mu(D) = \mu(D \cap C) + \mu(D \cap C^c)$
ii $D = C \cup (D \setminus C) \Longrightarrow \mu(D) = \mu(C) + \mu(D \setminus C) \Longrightarrow \mu(C) \leq \mu(D)$
iii Por ii se tiene que $\mu(D) = \mu(C) + \mu(D \setminus C)$ Como $\mu(C) = \infty \Longrightarrow \mu(D) = \infty$
iv $C \cup D = C \cup (D \setminus C) \Longrightarrow \mu(C \cup D) = \mu(C) + \mu(D \setminus C)$ Usando que $(D \setminus C) \subseteq D$ $\mu(C \cup D) = \mu(C) + \mu(D \setminus C) \leq \mu(C) + \mu(D)$
v $C \cup D = C \cup (D \setminus (C \cap D)) \Longrightarrow \mu(C \cup D) = \mu(C) + \mu(D \setminus (C \cap D)) \Longrightarrow \mu(C \cup D) + \mu(C \cap D) = \mu(C) + \mu(D \setminus (C \cap D)) + \mu(C \cap D)$ Tenemos que, $\mu(D) = \mu(C) + \mu(D \setminus C)$, tomando $C = C \cap D$, tenemos que $\mu(C) + \mu(D \setminus (C \cap D)) + \mu(C \cap D) = \mu(C) + \mu(D)$

Un conjunto medible A se dice nulo, o de medida nula, si $p(A) = 0$. Se dice que la medida es completa si todo subconjunto de un conjunto medible de medida nula es también medible (y, en consecuencia, de medida nula).

En la mayoría de los casos se puede suponer que el espacio completo tiene medida finita, debido a que se trabaja con un trozo finito o compacto de él, o bien porque a pesar de ser infinito presenta algunas características de concentración haciendo que algunas de sus partes relativamente pequeñas concentren todo el interés mientras que el resto tiene medida 0. Por lo tanto, se considerará que no solo hay una medida en O, sino que además es una medida de probabilidad, es decir, $p(O) = 1$. Si no fuera así se consideraría está otra medida $p'(C) = \frac{1}{p(O)} p(C)$.

Por tanto, se puede tomar p como una probabilidad que indica cómo de probable es que se den ciertas observaciones en el mundo. En el caso de que p fuese capaz de medir observaciones aisladas, se diría que p indica la probabilidad de que cada observación sea posible. Pero la mayoría de las veces O es no numerable, por lo tanto, las observaciones aisladas tienen medida 0. En la mayoría de los casos se considera una función asociada a la medida, se llama función de densidad que nos indica el peso de cada punto del espacio O respecto a la medida.

Matemáticamente, el problema del aprendizaje automático se traduce en ser capaz de dar una reconstrucción formal de la medida p.

En general, se puede clasificar el *Machine Learning* en tres categorías [12]:

- Aprendizaje supervisado: enfocado en determinar las probabilidades de nuevos eventos en función de eventos observados anteriormente.
- Aprendizaje no supervisado: permite encontrar patrones no etiquetados.
- Aprendizaje reforzado: la máquina guía su propio aprendizaje a través de recompensas y castigos.

En *Machine Learning* se pueden distinguir las siguientes 7 etapas [10]:

1. Obtención de datos, para cualquier algoritmo de *Machine Learning* se necesita poseer un gran números de datos para entrenar el modelo.

2. Preprocesamiento de datos, muchas veces los datos con los que se trabajan son categóricos, por lo que es necesario realizar un preprocesamiento y transformar esos datos en numéricos.
3. Extracción de características, se identifican los elementos que deben extraerse y someterse a análisis.
4. Selección de características, se identifica que atributos son necesarios para entrenar los modelos.
5. Entrenamiento, se entrena el modelo en base de un algoritmo de *Machine Learning*.
6. *Testing* de modelo, una vez entrenado el modelo es necesario validarlo con unos datos que se han separado del conjunto de datos original.
7. Análisis de los resultados, se buscan los errores a corregir y ajustar el modelo.

En el capítulo 3 se explica el funcionamiento de algunos de los algoritmos de *Machine Learning* más útiles en ciberseguridad [10]. Por ejemplo, se puede utilizar *Machine Learning* en los siguientes campos de ciberseguridad :

- Detección de intrusiones en la red.
- Detección de anomalías.
- Identificación de familias de *malware*.
- Detección de *Phishing*.
- Detección de *Spam*.
- Evaluación de riesgo.
- Detección de *Deep Fake*.

A continuación se detallan tres ejemplos concretos en los que se utiliza *Machine Learning*.

Detección de *Phishing*

La detección de *phishing* puede plantearne como un problema de aprendizaje supervisado [13]. En concreto es un problema de clasificación. Se tiene que generar un modelo a partir de un conjunto de datos que permita predecir si una página web es *phishing* o no.

Un ejemplo de un trabajo dedicado a la detección de *phishing* es el de CANTINA+ (continuación de otro trabajo previo denominado CANTINA). Se utilizaron dos conjuntos de datos, uno de 8,118 URLs y otro de 14,883 URLs. Entre los algoritmos de *Machine Learning* que se han utilizado, se encuentran *Support Vector Machines*, Regresión Lógica, Redes Bayesianas, Árboles de decisión, *Random Forest* y *Adaboost*. Se utilizaron dos evaluaciones, una aleatoria y otra basada en el tiempo. El mejor resultado se obtiene con el algoritmo de Redes Bayesianas.

Detección de *Deep Fake*

Mediante técnicas de *Deep Learning* se pueden generar o modificar vídeos usando redes neuronales. Los *Deep Fakes* funcionan mediante modelos de redes neuronales generativas antagónicas, GANs por sus siglas en inglés. Los algoritmos aprenden a crear imágenes de personas tras haber analizado una base de datos de imágenes de ejemplo [14].

Intel presentó un detector de *deep fake* en tiempo real (*Fake Catcher*), el cual analiza el "flujo sanguíneo" en píxeles de vídeo. Actualmente tiene una precisión del 96 %. El detector utiliza *hardware* y *software* de Intel.

La mayoría de los detectores que utilizan el aprendizaje profundo analizan los datos sin procesar. Es decir, intentan encontrar signos de falta de autenticidad e identificar que está mal en el vídeo. Sin embargo, *FakeCatcher* busca pistas auténticas en vídeos reales, viendo que nos hace humanos. Las señales de flujo sanguíneo comentadas se recogen en toda la cara. Por lo tanto, utilizando *Deep Learning* es posible detectar si un vídeo es real o falso [15].

Proyecto *OPOSSUM*

El proyecto *OPOSSUM* utiliza *Machine Learning* para analizar el comportamiento de un usuario cuando se autentifica y cómo usa un usuario una aplicación determinada, debido a que toda la comunicación entre el usuario y la aplicación pasa forzosamente por *OPOSSUM* [16]. El proyecto *OPOSSUM* aumenta el contexto de la petición, optimizando los datos sobre los cuales se harán las predicciones, para ello se emplea fuentes externas como *Shodan*, *Spyse* o *Alienvault*. Todos estos datos aportan mucha información que será procesada por modelos de *Machine Learning* para detectar anomalías en el comportamiento del usuario. Debido a que los comportamientos de los usuarios evolucionan con el tiempo y el propio usuario puede haber sido ascendido dentro de la organización dándole más acceso a la información. Es por ello que en *OPPOSSUM* se investiga e implementan modelos de *Machine Learning* basados en técnicas como el *Adaptive Learning*.

2

Análisis de *datasets* disponibles

Las medidas de seguridad que utilizan *Machine Learning* requieren siempre un conjunto de datos para poder utilizar los algoritmos de aprendizaje automático. Por este motivo, en este capítulo se analizan algunos de los conjuntos de datos más utilizados en ciberseguridad.

Una base de datos es un conjunto de informaciones sobre un tema determinado, exhaustivo, no-redundante y estructurado [17]. El adjetivo exhaustivo se debe a que la base de datos contiene el 100 % de datos sobre un tema. No-redundante significa que no hay ningún dato almacenado dos veces. Al estar la información almacenada una única vez, está correcta o incorrecta pero no se puede contradecirse. Estructurado hace referencia a almacenar los datos de tal manera que su tratamiento sea eficaz.

Es importante entender los datos útiles para ciberseguridad y sus características. Los datos sin procesar obtenidos desde fuentes cibernéticas relevantes se pueden usar para evaluar varios patrones de comportamiento malicioso con el fin de poder desarrollar una seguridad que ayude a alcanzar nuestro objetivo.

A continuación se nombran algunos conjuntos de datos y se da una breve descripción de cada uno.

2.1. *DARPA*

El conjunto más reciente de *DARPA* es un conjunto de datos de detección de intrusos en escenarios específicos de 2000 [18]. Contiene datos de escenarios de asaltos de *LLDOS* 1.0, *LLDOS* 2.0.2 y *Windows NT*. Es un conjunto de datos que consta de comunicaciones entre las IP de origen y las *IP* de destino [19]. Además, contiene diferentes ataques entre direcciones *IP*. Cubre cuatro categorías principales de ataques que son: denegación de servicios (*DoS*), de usuarios a remoto (*U2R*), remoto a local (*R2L*) y de sondeo.

2.2. *KDD'99 Cup*

KDD'99 Cup es uno de los conjuntos de datos que más se han utilizado para evaluar los métodos de detección de anomalías [20], donde se puede clasificar los ataques en cuatro tipos que son: denegación de servicio (*DoS*), remoto a local (*R2L*), usuario a remoto (*U2R*) y sondeo, que es vigilancia y otros sondeos, como por ejemplo, exploración de puertos. Además, se puede utilizar para evaluar modelos de ataques basados en *Machine Learning* [21]. Se desarrolló en la universidad de California [18]. Proviene del conjunto de datos *DARPA* 1998 , tiene 4.900.000 ataques repetidos, 22 tipos de ataques y 41 características fijas.

2.3. *NSL-KDD*

NSL-KDD es una versión modificada del conjunto de datos anterior que elimina registros innecesarios [18]. Como resultado, un modelo de seguridad basado en este conjunto de datos no favorecerá entradas más frecuentes [22]. También se utiliza para los tipos de ataques *DoS*, *R2L*, *U2R*. Este conjunto de datos tiene una carencia debido a la falta de conjuntos de datos públicos para *IDS* basados en red. Aun así, se puede aplicar para ayudar a los investigadores a comparar diferentes métodos de detección de intrusos.

2.4. *CICIDS*-2017

CICIDS-2017 fue generado en 2017 y contiene asaltos del mundo real que ocurrieron durante el año de su publicación [23]. La finalidad de este conjunto de datos es examinar el tráfico de la red utilizando marcas de tiempo, direcciones *IP* de origen y destino, puertos de origen y destinos, protocolos y ataques [18]. Comprende 86 características completos con estructura de red y tráfico, datos etiquetados, tráfico de red registrado y protocolos de ataques frecuentes que se distribuyen proporcionalmente [23], Además, contiene los ataques comunes benignos y que más se han actualizado, que se asemejan a los verdaderos datos del mundo real (*PCAP*).

2.5. *CSE-CICIDS*-2018

CSE-CICIDS-2018 es el resultado de un proyecto de colaboración entre el Centro de Seguridad de las Comunicaciones (*CSE*) y el Instituto Canadiense de Ciberseguridad [24] [18]. Es una versión mejorada del anterior conteniendo menos datos duplicados, elimina datos cuestionables y se puede exportar en formato *CSV*, lo cual lo hace idóneo para usar sin preprocesamiento [21]. Incluye siete

escenarios de ataque diferentes que son: fuerza bruta, *Heartbleed*, *Botnet*, *DoS*, *DDos*, ataques web e infiltración de la red desde dentro. Incluye una descripción detallada de las intrusiones junto con modelos de distribución abstractos para aplicaciones, protocolos o entidades de red de nivel inferior.

2.6. *Kyoto* 2006+

Kyoto 2006+ fue construido sobre tres años de datos de tráfico real, desde noviembre de 2006 hasta agosto de 2009 [25]. Consta de 14 características estadísticas derivadas del conjunto de datos *KDD Cup*'99. Además, costa de 10 características adicionales que se pueden usar para el análisis y evaluación de la red *IDS*. Las primeras 14 características se extrajeron del conjunto de datos *KDD Cup*'99, el cuál es muy popular. Las características que se extrajeron son significativas y esenciales de los datos de tráfico sin procesar obtenidos por los sistemas *honeypot* que se implementan en la Universidad de *Kyoto*. El *honeypot* es una trampa que se le ponen a los *hackers* para obtener información del ataque informático y de donde procede.

2.7. *ISCX*2012

El conjunto de datos *ISCX*2012 contiene 19 características, de las cuales el 19,11 % representa ataques *DDos* [20]. El Instituto Canadiense de Ciberseguridad desarrolló *ISCX*'12 [26], para su creación se imitó la actividad del usuario, posteriormente se diseñaron y ejecutaron escenarios de ataques para expresar casos reales de comportamiento malicioso.

2.8. *UNSW-NB*15

El conjunto de datos *UNSW-NB*15 fue desarrollado en la Universidad de Nuevas Gales de Sur en 2015 [20]. Contiene 49 características y nueve formas distintas de ataques, en los cuales se incluyen ataques *DoS*. Este conjunto se puede utilizar para probar sistemas de detección de anomalías basados en aprendizaje automático en aplicaciones cibernéticas.

2.9. *Bot-loT*

Bot-loT es un conjunto de datos para el análisis forense de red en Internet de las cosas que incluye tráfico de red de *loT* legítimo y simulado, así como diversos ataques. Por razones forenses, *Bot-IoT* se puede usar para evaluar la confiabilidad usando varias estad,ísticas y enfoques de aprendizaje automático [20].

2.10. Comparativa

El primer artículo analizado para realizar una comparativa entre los diferentes *datasets* mencionados es [27], que utiliza los siguientes 4 *datasets* para la detección de anomalías.

- *UNSW-NB15*
- *NSL-KDD*
- *CIC-IDS*-2017
- *Kyoto*

Para analizar esos *datasets* se usaron los siguientes cuatros métodos:

- *Local Outlier Factor* (LOF): es un algoritmo que identifica valores atípicos midiendo la desviación local de un punto de datos dado con respecto a sus vecinos. Donde un valor atípico es un punto de los datos que es diferente o está lejos del resto de los puntos de datos.
- *One-Class Support Vector Machine* (OC-SVM): solo usa una clase para estimar un modelo y detecta nuevos datos diferentes de esa clase como valores atípicos.
- *Isolation Forest* (IF): se crea un conjunto de árboles que aíslan las anomalías en lugar de ajustar instancias normales, que es un enfoque diferente para valores atípicos.
- *Robust Covariance* (RC): implementa un determinante de covarianza mínimo que es un algoritmo muy robusto para estimar la matriz de covarianza en datos multivariados.

Al observar la tabla 3 del artículo [27], se pueden clasificar los conjuntos de datos mencionados según el método que se utilice con la medida *F1-score*. Teniendo en cuenta esto, el mejor conjunto para el primer método es *UNSW-NB*15, para el segundo método es el *NSL-KDD* y para el tercer y cuarto método es *CIC-IDS*-2017.

En segundo lugar, leyendo el artículo [20], se encuentra una tabla en la que se puede observar una comparativa de los siguientes *datasets*:

- *NSL-KDD*
- *KDD cup* 99
- *Darpa*

Se emplean los siguiente algoritmos de *Machine Learning*:

- *Support Vector Machine* (SVM)
- Árbol de decisión
- *Deep Belief Network* (DBN)
- *Artificial Neural Network* (ANN)
- *Naïve Bayes* (NB)

Al observar la tabla IV del artículo [20], para clasificar los conjuntos de datos según el algoritmo de *Machine Learning* que se está utilizando se toma la medida *accuracy*. Teniendo en cuenta esto, se concluye que el mejor conjunto para el primer método es *NSL-KDD* 2016, para el segundo método es *KDD* 2018, para el tercer método es *NSL-KDD* 2019, para el cuarto método es *KDD cup* 2019, y para el último es *KDD Cup* 99 2018. Además, si se observa la Fig. 3 del artículo [20] y tomando como medida el *recall* de cada uno de los conjuntos de datos se puede observar que, en general, el conjunto *NSL-KDD* tiene mejores resultados.

Por tanto, se puede concluir que dependiendo del algoritmo que se utilice para analizar el conjunto de datos es mejor uno u otro.

3

Pruebas de concepto

En este capítulo se muestra cómo se han aplicado varios algoritmos de *Machine Learning* para identificar páginas web fraudulentas.

A la dirección única y específica de cada una de las páginas que existen en Internet se le denomina *URL* (*Uniform Resource Locator*). Entre las *URLs* se pueden encontrar versiones fraudulentas, que son links que parecen llevar a páginas legítimas pero en realidad redirigen a páginas falsas en las que los ciberdelicuentes aprovechan para robar información personal como contraseñas, cuentas bancarias, etc.

En la era digital actual, la detección de *URLs* fraudulentas se ha convertido en una preocupación cada vez más importante debido al creciente número de ciberataques que buscan engañar a los usuarios para obtener información confidencial. De hecho, los cibercriminales utilizan técnicas cada vez más sofisticadas para crear esas *URLs* maliciosas que parecen legítimas, lo que dificulta su detección. Sin embargo, con el avance de las técnicas de Inteligencia Artificial y en particular de *Machine Learning*, se pueden desarrollar modelos altamente efectivos para detectar estas *URLs* fraudulentas ya que esos algoritmos permiten analizar grandes cantidades de datos y aprender a reconocer patrones para elaborar predicciones. En la bibliografía se pueden encontrar diferentes trabajos en los que se aplican técnicas de *Machine Learning* con ese objetivo [28, 29]

El objetivo de este capítulo es analizar la aplicación de técnicas de *Machine Learning*, y en particular de *Deep Learning*, para lograr una alta probabilidad de detección temprana de URLs fraudulentas. Además, se introduce una prueba de concepto usando computación cuántica en un conjunto de datos reducidos para analizar su viabilidad en el campo de la ciberseguridad.

Este capítulo se estructura como sigue. La sección 3.1 incluye una breve introducción de los algoritmos de *Machine Learning* y *Deep Learning*. La sección 3.2 presenta los principales detalles del modelo y programa aplicado, cuyos resultados se muestran en la sección 3.3. Una breve evaluación y discusión se muestra en la sección 3.4. Además, en la Sección 3.5 se hace una breve introducción a la computación cuántica y su aplicación en ciberseguridad.

3.1. Algoritmos de *Machine Learning*

En esta sección se detalla el funcionamiento de cada uno de los algoritmos de *Machine Learning* utilizados.

3.1.1. Regresión logística

La regresión logística es una técnica de análisis de datos que utiliza las matemáticas para encontrar las relaciones entre dos factores de datos. Uno de los algoritmos más utilizados para la clasificación en *Machine Learning* se basa en regresión logística. Se utiliza para estimar la probabilidad de que una variable independiente pertenezca a una clase o a otras [30].

Para definir la región de clasificación, se escoge como clase 1 la región para la cual la regresión produce un valor mayor a $0,5$.

$$y = g_\theta(x) + E \tag{3.1}$$

$$g_\theta(x) = \frac{1}{1 + e^{-\theta^t \cdot x}} \tag{3.2}$$

donde

- y es el identificador de la clase.
- x es la variable de entrada.
- E es el error de predicción.
- $g_\theta(x)$ es el modelo matemático.
- θ es el vector de pesos lineales.

El objetivo de la regresión logística es encontrar los valores de θ que minimizan el error entre los valores observados y la estimación del modelo.

En la Fig. 3.1 se puede apreciar un ejemplo de problema de regresión logística el cual es clasificar si un correo es *spam* o no. Por lo que puede observarse, se ve que $\hat{p} \geq 0{,}5$ por lo tanto es un correo *spam*.

Una vez que se ha calculado la probabilidad de que una clase x, se calcula la predicción del modelo de regresión logística mediante la ec. 3.3

$$\hat{y} = \begin{cases} 0 \text{ si } \hat{p} < 0{,}5 \\ 1 \text{ si } \hat{p} \geq 0{,}5 \end{cases} \tag{3.3}$$

donde

$$\hat{p} = g_\theta(x)$$

Una vez visto como se calcula la probabilidad y se hace predicciones de un modelo de regresión logística se observa como se entrena. El objetivo del entrenamiento es determinar el vector de parámetros θ, de forma que el modelo estime probabilidades altas para instancias positivas (y=1) y probabilidades

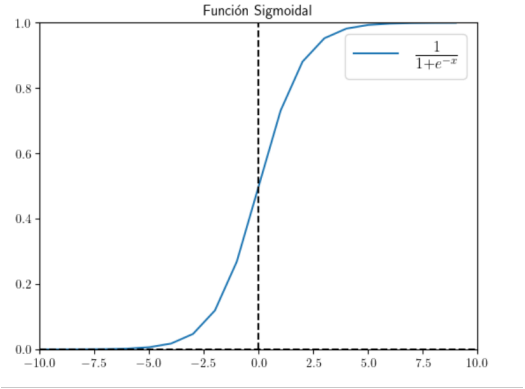

Figura 3.1. Función *Sigmoidal*

bajas para estancias negativas (y=0). En la ec. 3.4 se muestra la función de pérdida de una sola instancia de entrenamiento x.

$$\hat{y} = \begin{cases} -\log(\hat{p}) & \text{si } y = 1 \\ -\log(1-\hat{p}) & \text{si } y = 0 \end{cases} \tag{3.4}$$

Esta función de pérdida tiene sentido porque $-\log(\hat{p})$ se hace más grande cuando $\hat{p}$ se acerca a 0 para una instancia positiva y lo mismo ocurre cuando se acerca a 1 para una instancia negativa.

Por lo tanto, nuestro problema se reduce a su función de costo que se conoce como la entropía cruzada condicional.

$$\min_{\theta} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(-y_i \log(g_\theta(x_i)) - (1-y_i)\log(1-g_\theta(x_i)) \tag{3.5}$$

donde

$$g_\theta(x_i) = \frac{1}{1+e^{-\theta^T x_i}} \tag{3.6}$$

m es el número total de elementos de datos en el conjunto de datos de entrenamiento, y_i es la variable dependiente de la fila i del conjunto de datos de entrenamiento, g_θ es el modelo matemático indicado por la función sigmoidea y x_i es el vector de variables independientes de la fila i del conjunto de datos de entrenamiento.

3.1.2. Árbol de decisión

Un árbol de decisión es un mapa que refleja todos los posibles resultados de una serie de decisiones relacionadas. Entre los algoritmos de *Machine Learning* hay un conjunto basado en árboles de decisiónes que permiten realizar tanto tareas de clasificación como de regresión. Esta herramienta se puede aplicar

a algoritmos complejos para predecir valores de respuestas mediante reglas de decisión.

La función objetivo a maximizar es la ganancia de información en cada división [31]:

$$IG(D_P, f) = I(D_P) - \sum_{j=1}^{m} \frac{N_j}{N_p} \cdot I(D_j) \tag{3.7}$$

donde m son las regiones en las que se divide el nodo principal y además se tiene que:

D_p : el conjunto de datos principal.
f : la característica para realizar la división.
I : la medida de impureza
N_j : el número de muestras en el j-ésimo nodo secundario.
N_p : el número total de muestras en la nodo principal.
D_j el conjunto de datos del j-ésimo nodo secundario.

Como se puede contemplar, la ganancia de información es simplemente la diferencia entre la impureza del nodo principal y la suma de las impurezas del nodo secundario. Cuanto menor es la impureza de los nodos secundarios, mayor es la ganancia de la información. Sin embargo, por simplicidad y para reducir el espacio de búsqueda combinatoria, la mayoría de las bibliotecas implementan árboles de decisiónes binarios, donde cada nodo principal es dividido en dos nodos secundarios, D_L y D_R:

$$IG(D_P, f) = I(D_P) - \frac{N_L}{N_P} \cdot I(D_L) - \frac{N_R}{N_P} \cdot I(D_R) \tag{3.8}$$

Para la medida de impureza se necesitan los siguientes conceptos:

- Entropía

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \cdot \log_2(pi|t) \tag{3.9}$$

- Impureza de *Gini*

$$I_G(t) = \sum_{i=1}^{c} p(i|t) \cdot (1 - p(i|t)) = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{3.10}$$

- Error de clasificación

$$I_E(t) = 1 - \max_i \{p(i|t)\} \tag{3.11}$$

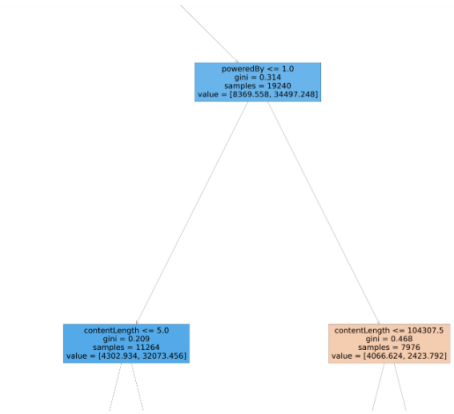

Figura 3.2. Árbol de decisión

donde la $p(i|t)$ es la proporción de las muestras que pertenecen a la clase c para un determinado nodo t
En la Fig. 3.2 se puede visualizar como es un árbol de decisión.

Analizando la Fig. 3.2 cabría preguntarse como el árbol encuentra preguntas como *poweredBy* $\leq$ 1,0 o *contentLength* $\leq$ 5,0.

1. Para todas y cada unas de las características en D_p, $f_j^{(i)}$:
   - Hace una pregunta para separar D_p en D_L y D_R.
   - Calcula las impurezas $I(D_L)$ Y $I(D_R)$.
   - Calcula la ganancia de la información utilizando la ec. 3.9

$$IG(D_P, f_j^{(i)}) = I(D_P) - \frac{N_L}{N_P} \cdot I(D_L) - \frac{N_R}{N_P} \cdot I(D_R) \tag{3.12}$$

2. Sea

$$f_q^{(p)} = \arg \max_{i,j} \{IG(D_P, f_j^{(i)})\} \tag{3.13}$$

3. Entonces, la mejor pregunta de división es

$$f_q^{(k)} \leq f_q^{(p)} \text{para cualquier } k \tag{3.14}$$

El máximo de la ec. 3.14 a menudo ocurre cuando algunas de las impurezas secundarias es cero o muy pequeña.

3.1.3. *Support Vector Machine*

Las máquinas de vector soporte o SVM (*Support Vector Machine*) son un conjunto de algoritmos de aprendizaje supervisado que se utilizan para la clasificación, la regresión y la detección de valores atípicos. *Support Vector Machine* se

fundamenta en el *Maximal Margin Classifier*, que a su vez se basa en el concepto de hiperplano [33].

El objetivo de este algoritmo se basa en encontrar un hiperplano en un espacio N- dimensional que clasifique la variable dependiente, donde N es el número de variables independientes.

Además, el objetivo de optimización de este algoritmo se fundamenta en maximizar el margen. Donde el margen es la distancia del hiperplano de separación(límite de decisión) y las muestras de entrenamiento que están más cerca de ese hiperplano, que se denominan vectores soporte.

Un hiperplano se define como un subespacio plano y afín de dimensiones $p-1$. En un espacio de dimensión dos el hiperplano sería una recta debido a que es un subespacio de una dimensión. Si ahora se plantea para un espacio de tres dimensiones, el hiperplano sería un plano. De forma matemática se puede expresar a través de la ec. 3.15.

$$\alpha_0 + \alpha_1 \cdot x_1 + ... + \alpha_p \cdot x_p \tag{3.15}$$

A continuación, se introduce el concepto de linealmente separable. Se dice que un conjunto de datos es linealmente separable si la distribución de las observaciones es tal que se pueden separar linealmente de forma perfecta en dos clases que denotaremos como ($+1$ y -1). Un hiperplano de separación cumple que:

$$\begin{cases} \alpha_0 + \alpha_1 \cdot x_1 + ... + \alpha_p \cdot x_p > 0 \text{ si } y = 1 \\ \alpha_0 + \alpha_1 \cdot x_1 + ... + \alpha_p \cdot x_p < 0 \text{ si } y_i = -1 \end{cases} \tag{3.16}$$

Esas dos condiciones se pueden simplificar en una sola.

$$y_i \cdot (\alpha_0 + \alpha_1 \cdot x_1 + ... + \alpha_p \cdot x_p) > 0, \text{para } i = 1, ..., n \tag{3.17}$$

donde n es el número total de datos que tenemos y p es el número total de variable dependiente que tenemos.

Observando a la definición de hiperplano para casos perfectamente separables linealmente se obtiene un número infinito de posibles hiperplanos. Por lo tanto, es necesario encontrar el más óptimo. El *maximal margin hyperplane* se define como el hiperplano que consigue que la distancia mínima entre el hiperplano y las observaciones sea lo más grande posible. Para esto se recurre a métodos de optimización que lo podemos encontrar explicados en [32].

El *maximal margin hyperplane* es una forma muy simple y natural de clasificación siempre y cuando exista un hiperplano de separación. En la mayoría de los casos no se pueden separar linealmente de forma perfecta, por lo que no existe el hiperplano de separación ni se puede obtener el *maximal margin hyperplane*.

Para solucionar este problema, se extiende el concepto de *maximal margin hyperplane* para obtener un hiperplano que casi separe las clases, pero dejándole

cometer algunos errores. A estos hiperplanos se les conoce como *Support Vector Classifier* o *Soft Margin*.

Además, cuando se tratan datos que no son linealmente separables en el espacio original no significa que no lo sean en un espacio de mayores dimensiones. Se puede entrar en la página [33] en la que se pueden visualizar imágenes de conjunto de datos que no son linealmente separables en una dimensión pero al aumentar la dimensión se vuelven linealmente separables.

La idea fundamental de los *kernels*(núcleos) que se utilizan para tratar datos inseparables lineales es crear combinaciones no lineales de las características originales para proyectarlas hacia un espacio de dimensiones mayores, mediante una función de mapeo, ϕ, donde pasan a ser separables lineales. Un ejemplo de función de mapeo puede ser, la función que lleva el par (x_1, x_2) al par $(x_1, x_2, x_1^2 + x_2^2)$.

Los núcleos que se emplean son [31]:

- Lineal:
$$K(x^{(i)}, x^{(j)}) = x^{(i)} \cdot x^{(j)} \tag{3.18}$$
- *RBF* (*Radial Basis Function*) o el núcleo Gaussiano :
$$K(x^{(i)}, x^{(j)}) = \exp(-\frac{||x^{(i)} - x^{(j)}||^2}{2 \cdot \theta^2}) \tag{3.19}$$
- Polinomial de grado K :
$$K(x^{(i)}, x^{(j)}) = (x^{(i)} \cdot x^{(j)})^k \tag{3.20}$$
- Polinomial de grado hasta k : para alguna $c > 0$:
$$K(x^{(i)}, x^{(j)}) = (c + x^{(i)} \cdot x^{(j)})^k \tag{3.21}$$
- *Sigmoid*:
$$K(x^{(i)}, x^{(j)}) = \tanh\left(ax^{(i)} \cdot x^{(j)} + b\right) \tag{3.22}$$

En la página [33] se puede contemplar al final de la páginas dos imágenes en las cuales en una se utiliza el núcleo *poly* y en la otra se utiliza el núcleo *rbf*.

La función a maximizar que se utiliza es [31]:

$$\max_{\alpha} [\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} \cdot K(x^{(i)}, x^{(j)})] \tag{3.23}$$

cumpliendo:

$$0 \leq \alpha_i \leq C \qquad \forall i \qquad \sum_{i=1}^{N} \alpha_i y^{(i)} = 0 \tag{3.24}$$

donde $y^{(i)}$ es el valor de la variable dependiente de la fila i, N es el número de datos observado del conjunto de entrenamiento y la C es una variable para controlar la penalización por clasificación errónea. Los valores grandes de C corresponden a grandes penalizaciones por errores. Mientras que al elegir valores más pequeños para C se es menos estrictos con respecto a la clasificación errónea de errores. Por lo tanto, se puede usar el parámetro C para controlar el ancho del margen y , por lo tanto, ajustar la compensación de la varianza del sesgo.

3.1.4. Redes neuronales

La primera neurona que se formalizó fue creada por *Warren McCulloch* (neurocientífico, psiquiatra) y *Walter Pitts*(matemático) [34]. Es el primer modelo matemático que intenta replicar la actividad eléctrica de una neurona biológica. En la Fig 3.3 se puede apreciar la neurona *McCulloch-Pitts*, donde θ es el valor de un potencial que representa ese umbral de activación y $w \in \mathbb{R}^+$.

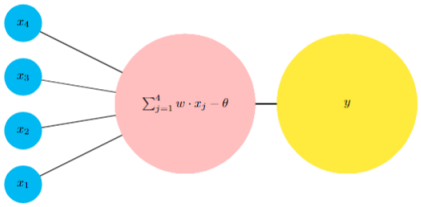

Figura 3.3. Neurona *McCulloch-Pitts*

La función de activación que se utiliza es [34]:

$$s(z) = \begin{cases} 1 \text{ si } z \leq 0 \\ 0 \text{ si } z < 0 \end{cases} \tag{3.25}$$

donde $z = \sum_{i=1}^{4} w \cdot x_i - \theta$. Por lo tanto, $y = s(z)$

Como ya se mencionó, entre las redes neuronales se encuentran como unidades básicas el perceptrón y *Adaline*, así como la neurona logística [35]. En la Fig. 3.1.4 se muestra la estructura de un perceptrón.

θ es el nivel de umbral [34]. Los w_i son los pesos que indican que tan relevantes son las entradas. Con los pesos se le da un mayor grado de flexibilidad al modelo para ajustarse a los problemas.

Una posible cuestión es como se calculan esos w_i. La Regla de Perceptrón se basa en una variante de la Regla *Hebiana* [36]. La Regla *Hebiana* dice que si tiene dos neuronas que trabajan juntas entonces su conexión debe fortalecerse.

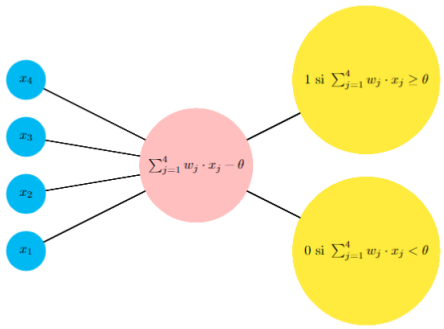

Figura 3.4. Estructura de un perceptrón

De forma matemática se puede definir la regla de actualización de los pesos para el perceptrón como:

$$w_i := w_i + \alpha \cdot (y - \hat{y}) \cdot x_i \tag{3.26}$$

donde α es la taza de aprendizaje, $\hat{y}$ es el valor estimado, y es el verdadero valor de nuestra variable dependiente.

Similar a la regla *Hebiana* pero se incluye una retroalimentación de la salida de neurona.

El *Adaline* y la neurona logística son similares al perceptrón pero cambian en la función de activación. En el *Adaline* se utiliza la función lineal mientras que en el caso de la neurona logística se utiliza la función logística que es la *sigmoidal*.

Otra cuestión que se podrían plantear es como se encuentran los valores de los w_i. Tanto para el *Adaline* como para la Neurona Logística, la función de activación es derivable, por lo que se puede utilizar el gradiente descendiente para encontrar los parámetros del modelo.

$$w_i := w_i - \theta \frac{\mathrm{d}}{\mathrm{d}w} J(w) \tag{3.27}$$

Para el *Adaline* se utiliza la función de costo del error cuadrático medio, por lo tanto, se tendría:

$$J(w) = \frac{1}{N} \sum (y_i - \hat{y}_i)^2 \tag{3.28}$$

Para la neurona logística se usa:

$$J(w) = -y \log\left(g_\theta(x)\right) - (1 - y) \log\left(1 - g_\theta(x)\right) \tag{3.29}$$

donde $g_\theta(x) = \frac{1}{1+e^{-\theta^t x_i}}$ y N es el número de observaciones del conjunto de datos de entrenamiento.

Aparte de estas redes neuronales, existen las denominadas redes *feed-forward*, que se diferencian de las anteriores por tener capas ocultas. En cada

una de las capas ocultas se encuentran varias neuronas con la siguiente estructura. Las neuronas de una misma capa no están conectadas entre ellas y todas comparten la misma función de activación. Por otro lado, cuando hay dos capas consecutivas, sucede que todas las neuronas de una capa se conectan con todas las neuronas de la otra capa, lo que hace que la red sea una red densa. En la Fig. 3.5 se puede apreciar la estructura de la red neuronal *feed-forward*.

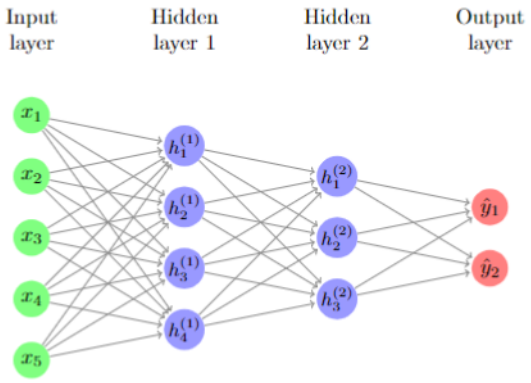

Figura 3.5. Estructura de una red neuronal *feed-forward*

Se obtiene la siguiente nomenclatura:

- vector de entrada: $x = [x_1, ..., x_n]^t$.
- nº de neuronas en la capa p: $n^{(p)}$.
- función de activación de la capa p : $f_p(.)$.
- peso de conexión entre neuronas i de la capa p con la neurona j de la capa p+1: $w_{ji}^{(p)}$.
- vector de conexión entre las salidas de la capa p, y la neurona i de la capa p+1: $w_i^{(p)} = [w_{1i}^{(p)}, ..., w_{n(p-1)i}^{(p)}]^t$

Para calcular los w_i, se utiliza el *backpropagation* para minimizar la función de costo ajustando los pesos y sesgos de la red. El nivel de ajuste está determinado por los gradientes de la función de costo con respecto a esos parámetros. En estas citas se explica brevemente [37] y [38].

3.1.5. Matriz de confusión

Para evaluar estos modelos se utiliza la matriz de confusión, que es una de las métricas más utilizadas para evaluar modelos de clasificación. En la Fig.

3.8 se puede observar un ejemplo de matriz de confusión. De esta estructura se puede extraer la siguiente información [39], [40]:

$$\text{Precisión} = \frac{\text{VP}}{\text{VP+FP}} \tag{3.30}$$

$$\text{Sensibilidad} = \frac{\text{VP}}{\text{VP+FN}} \tag{3.31}$$

$$\text{Especificidad} = \frac{\text{VN}}{\text{VN+FP}} \tag{3.32}$$

$$\text{Exactitud } (\textit{Accuracy}) = \frac{\text{VP+ VN}}{\text{VP+FN+VN+FP}} \tag{3.33}$$

donde

VP son Verdaderos Positivos
FN son Falsos Negativos
FP son Falsos Positivos
VN son Verdaderos Negativos

La Precisión o *Precision* es el porcentaje de los casos bien clasificados dentro de una clase.

La Sensibilidad o *Recall* es el porcentaje de los casos positivos bien clasificados, es decir, es la capacidad de un algoritmo para predecir un resultado positivo cuando el resultado real es positivo.

La Especificidad o *Specifity* es el porcentaje de los casos negativos bien clasificados.

La Exactitud o *Accuracy* es el cociente de los datos bien clasificados entre la suma de todos los datos. Es decir, es el porcentaje de las predicciones correctas frente al total.

3.1.6. La curva *ROC*

Se puede definir la curva *ROC*, tal y como se define en [30]. La curva *ROC* (*Receiver Operating Characteristic*), es una herramienta utilizada en los clasificadores binarios. Es muy similar a la curva precisión/sensibilidad, pero la curva *ROC*, en lugar de trazar la precisión frente a la sensibilidad, lo que hace es trazar la tasa de Verdaderos Positivos frente a la tasa de Falsos Positivos *FPR* (*False Positive Rate*). La *FPR* es la proporción de instancias negativas que se clasifican de manera incorrecta como positivas. La *TNR* (Tasa Negativa Real) es la proporción de las instancias negativas clasificadas correctamente como negativas. La *TNR* también se llama Especificidad. Por lo tanto, la curva *ROC* traza la sensibilidad frente a 1-especificidad. Además, una forma de comparar clasificadores es medir el área debajo de la curva o *AUC* (*Area Under the Curve*). Decimos que un clasificador es perfecto cuando tiene un área debajo de la curva *ROC*

igual a 1. Más adelante se podrá visualizar la tabla V donde se muestra el área debajo de la Curva *ROC* de cada uno de los algoritmos de *Machine Learning* utilizados. En la Fig. 3.6 se puede visualizar un ejemplo de curva *ROC*.

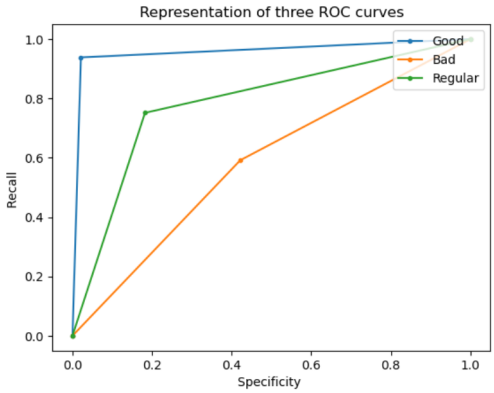

Figura 3.6. Curva *ROC*

3.2. Propuesta e implementación

Para la detección de *URLs* fraudulentas se utilizó el *dataset* obtenido de la página https://machinelearning.inginf.units.it/data-and-tools/hidden-fraudulent-urls-dataset. Contiene la siguiente información:

- *url* es la *URL* actual.
- *compromissionType* es la variable que indica si el sitio web está comprometido por phising, defacement, o es normal.
- *isHiddenFraudulent* es una variable dependiente que indica si la URL es fraudulenta o no.
- *contentLength* es una variable que toma valores enteros y se ha obtenido al enviar una solicitud *HTTP HEAD* a la *URL*. Además indica el tamaño del cuerpo del mensaje, en bytes, enviado al destinatario.
- *serverType* es una cadena que indica el servidor, como por ejemplo *Apache*, *Microsoft IIS*.
- *poweredBy* es una cadena que indica la plataforma de aplicaciones que subyace al servidor web.
- *contentType* contiene información del *charset* que es del tipo de codificación.
- *lastModified* es una variable que indica cuándo fue su última fecha de modificación.

Tanto la variable *poweredBy* como *serverType* se han preprocesado para mantener el nombre del marco y el número de la versión principal y secundaria.

	url	compromissionType	isHiddenFraudulent	contentLength	serverType	poweredBy	contentType	lastModified
0	http://www.sinduscongoias.com.br/index.html	defacement	False	2474	Apache/2.2	NaN	text/html	Sat, 05 Jan 2013 19:36:29 GMT
1	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:30:53 GMT
2	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:30:58 GMT
3	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:31:01 GMT
4	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:31:05 GMT

Figura 3.7. Cinco primeras filas del *dataset* utilizado

En la Fig. 3.7, se pueden observar las 5 primeras filas del *dataset* utilizado.

Se han desarrollado varios planteamientos para el tratamiento de los datos y en todos se ha eliminado la columna de *lastModified* dado que son datos no relevantes para el estudio.

Antes de implementar algún modelo, se decidió sustituir el valor NaN por 0 en la columna *PoweredBy* por el mismo motivo. De idéntica manera, después se eliminaron las filas en las que faltaba algún dato en alguna columna. Haciendo eso se obtuvo un conjunto de datos de 181.916 filas y 6 columnas, contando con la variable dependiente. Entre ellos en total había 8.618 URLs fraudulentas. Por tanto, se trata de un conjunto no balanceado de datos, teniendo que usar la parametrización siguiente en los modelos usados: *class_weight="balanced"*. Además, para el entrenamiento se utilizaron 145.532 datos aleatorios de nuestro conjunto de datos. En total, en nuestro conjunto de entrenamiento había 6.965 *Urls* fraudulentas.

Lo primero que se hizo fue comprobar la bondad de los datos sin las URLs. Utilizando el algoritmo de árbol de decisión de *Machine Learning* se obtuvo un 87.77 % de exactitud. En la Fig. 3.8 se puede contemplar cuál es la matriz de confusión resultante.

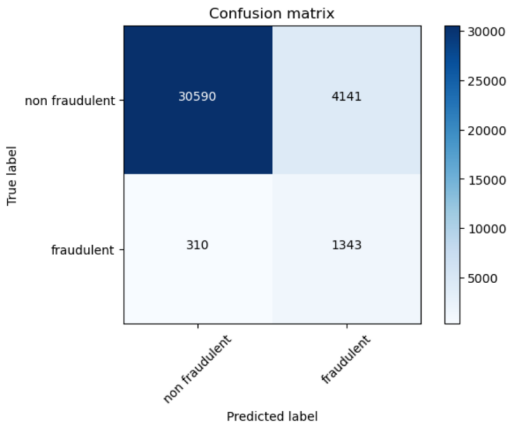

Figura 3.8. Matriz de confusión resultante

En la Fig. 3.8 se puede apreciar que del conjunto de entrenamiento solo se identificaron bien 1.653 *URLs* fraudulentas entre 1.659, es decir, que solo se identificó bien el 81.25 % . Por otra parte, de las *URLs* no fraudulentas se identificó bien el 88.08 %. Esto lleva a pensar que si se analizan las *URLs* tal vez se podría conseguir más precisión ya que ahora mismo 310 *URLs* fraudulentas no se están clasificando bien.

A continuación se procedió a analizar únicamente las URLs y crear un árbol de decisión eliminando las variables *serverType*, *compromissionType*, *poweredBy* y *contentType*. Se definieron nuevas variables, como la longitud de la *URL*, contar cuántas veces aparece la '/' en el *path* de *URL* y más variables parecidas. En la Fig. 3.10 se pueden visualizar todas las definiciones nuevas que se crearon junto con las variables que teníamos al principio excepto *lastModified*. Al hacer esto, el conjunto de datos generado tiene el mismo número de filas y 15 columnas contando con la variable dependiente.

Al añadir las nuevas variables, se devolvió una exactitud de 93.18 %. En la Fig. 3.9 se muestra la nueva matriz de confusión.

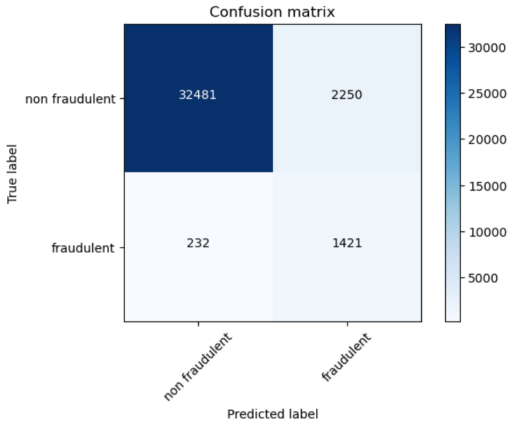

Figura 3.9. Nueva matriz de confusión

En este caso, se puede observar que del conjunto de entrenamiento identificó más datos correctamente que el anterior. De las que son fraudulentas, identificó bien el 85.96 % mientras que de las *URLs* no fraudulentas se identificó bien el 93.52 % . Con este programa se obtuvo que 232 *URLs* fraudulentas no se clasifican bien, lo que lleva a plantear si uniendo ambos programas se podría obtener más especificidad. Además está la pregunta de qué pasaría si se aplican otros algoritmos.

Una vez realizado esto, se pasó a la preparación de los datos. En este apartado, se decidió dejar las definiciones creadas para extraer la información de las *URLs* y dejar las variables independientes *serverType*, *compromissionType*, *poweredBy*, *contentType* y *contentLenght*.

Además, se utilizó la normalización de los datos en todos los algoritmos empleados de *Machine Learning* y *Deep Learning* excepto en el árbol de decisión. Los algoritmos que se emplearon para analizar finalmente el conjunto de datos fueron:

- Regresión logística
- Árbol de decisión
- *Support Vector Machine*
- Redes neuronales

Se eligió el modelo de Árbol de Decisión [41] pues clasifica muy bien este problema. Además, se eligieron las herramientas de Regresión Logística y *Support Vector Machine* dadas las precisiones obtenidas en trabajos como [42]. Otro trabajo en el que se utiliza la Regresión Logística es en [43]. Otro ejemplo de estudio en el que se utilizan estos algoritmos y además se emplea redes neuronales es el trabajo [44].

Para el árbol de decisión y las redes neuronales se muestra en la Fig. 3.10 la tabla de correlación de las variables independientes junto con nuestra variable dependiente que es la variable *isHiddenFraudulent*.

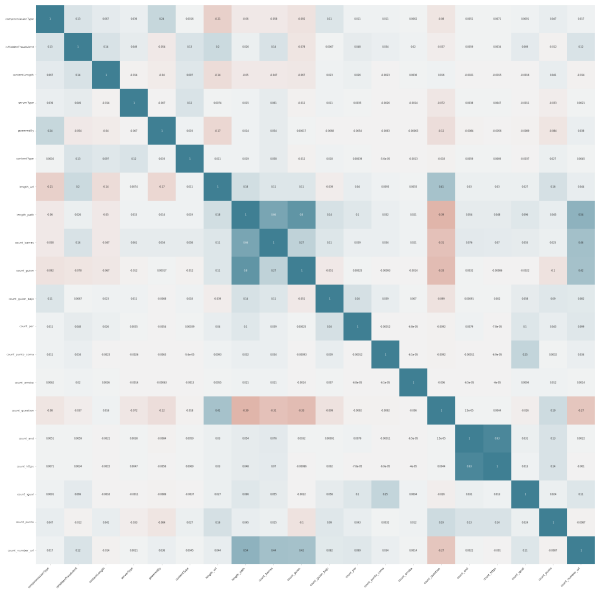

Figura 3.10. Correlación del conjunto de datos

Para los demás algoritmos de *Machine Learning* se ha empleado el siguiente conjunto de datos que se puede observar en la Fig. 3.11, donde se muestra la correlación entre las variables independientes junto con nuestra variable dependiente que es la misma que en el caso anterior.

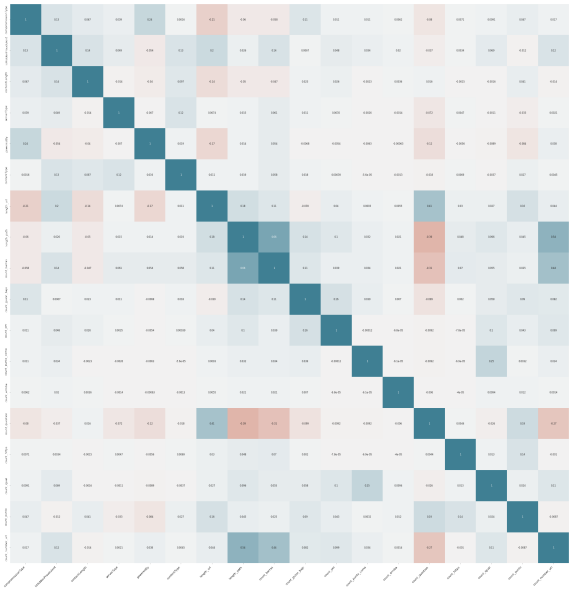

Figura 3.11. Correlación del conjunto de datos para los demás algoritmos

3.3. Resultados

En esta sección se comparan los resultados obtenidos de todos los métodos empleados.

En el algoritmo de *Support Vector Machine* se han empleado diferentes *kernels* o núcleos (*sigmoid*,*Radial Basis Function* (*RBF*) y *poly*. Se elegió el núcleo *RBF* usado en el trabajo [45].

En las Tablas 3.1 y 3.2 se muestra la comparativa de los métodos empleados, excepto las redes neuronales.

Tabla 3.1. Comparación de los resultados obtenidos que no son fraudulentos

Algoritmo *Machine Learning*	*Precision* (%)	*Recall* (%)	*F1-score* (%)
Regresión Logística	99	82	89
SVC (*Sigmoid*)	97	58	72
SVC (*Poly*)	100	96	98
SVC (*rbf*)	100	98	99
Arbol de decisión	100	99	100

En el caso de las redes neuronales, se optó por construir tres diferentes.

En la primera red neuronal se creó una capa oculta en la que se utiliza la función de activación *sigmoid*. Además se utilizó la función de pérdida '*binary_crossentropy*', el optimizador de '*adam*' y la métrica '*binary_accuracy*'. En la Fig. 3.12 se aprecia su matriz de confusión.

Tabla 3.2. Comparación de los resultados obtenidos que si son fraudulentos

Algoritmo *Machine Learning*	*Precision* (%)	*Recall* (%)	*F1-score* (%)
Regresión Logística	16	75	27
SVC (Sigmoid)	6	59	11
SVC (Poly)	55	93	69
SVC (RBF)	68	94	79
Arbol de decisión	89	91	90

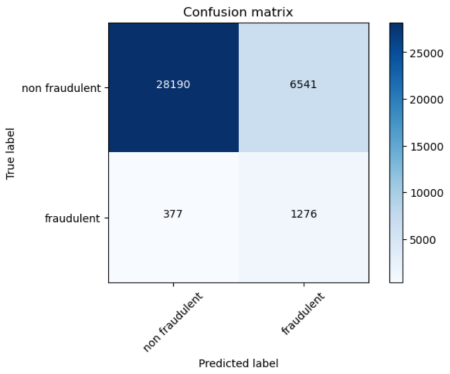

Figura 3.12. Primera red neuronal

La segunda red neuronal se creó con tres capas ocultas, y en todas se utilizó la función de activación *sigmoid*. También se utilizó la misma función de pérdida y la misma métrica que en la primera red neuronal, el optimizador que se utilizó fue '*rmsprop*'. En la Fig. 3.13 se puede contemplar la matriz de confusión de esta red neuronal.

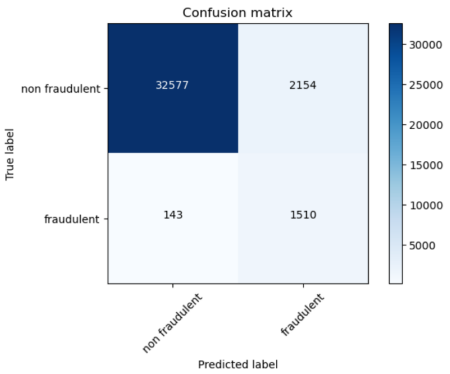

Figura 3.13. Segunda red neuronal

En la tercera y última red neuronal se crearon tres capas ocultas. En las dos primeras capas se utilizó la función de activación 'relu', mientras que en la última

capa se utilizó la función de activación *sigmoid*. También se utilizó la misma función de pérdida, optimizador y métrica que en la primera red neuronal. En la Fig. 3.14 se puede visualizar la matriz de confusión de esta última red neuronal.

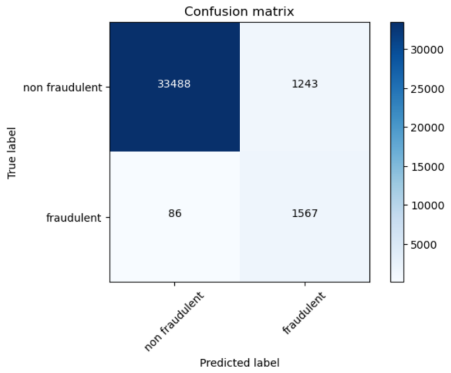

Figura 3.14. Tercera red neuronal

En la Tabla 3.3 se encuentra el resumen de las matrices de confusión de cada una de las redes neuronales.

Tabla 3.3. Comparativa de las tres redes neuronales (son Fraudulentos)

Algoritmo *Machine Learning*	*Precision* (%)	*Recall* (%)	*Accuracy* (%)
Primera red neuronal	16	77	80,99
Segunda red neuronal	41	91	93,69
Tercera red neuronal	56	95	96,35

Una vez hechas las comparaciones se decidió replantear si los modelos usados están sobreajustados (*overfitting*) o subajustados (*underfitting*). Para ello, se decidió analizar, en el caso de las redes neuronales, el *binary_accuracy* y el *val_binary_accuracy*, en la Fig 3.15 se puede observar la representación de ambas variables. Se dice que un modelo está subajustado cuando se obtiene una precisión baja y, además, la precisión de validación también es baja. En el caso de sobreajustado se obtiene un precisión alta en el entrenamiento pero en la validación se tiene una precisión baja.

Por lo que se puede observar en la Fig. 3.15, vemos que la representación de la variable *val_binary_accuracy* y la variable *binary_accuracy* están casi al mismo nivel. Como hay interacciones donde se produce picos drástico, para evitarlo implementaremos una medida para intentar que no se produzcan tanto. La medida que se va a implementar es *Dropout*, que consiste en "apagar" para cada una de las capas un porcentaje de las neuronas para que no sean entrenadas, evitando así que se modifique su coeficiente. En nuestro caso en la primera

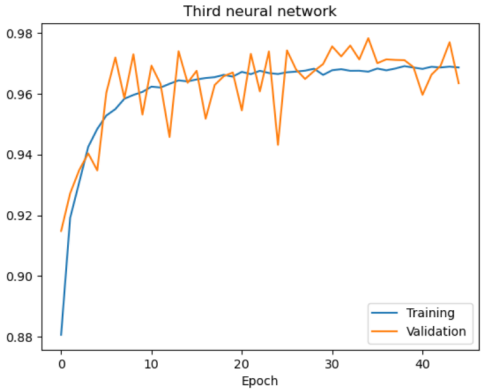

Figura 3.15. Tercera red Neuronal

capa "apagaremos" el 20 %. En la Fig. 3.16 se puede observar el resultado que obtenemos.

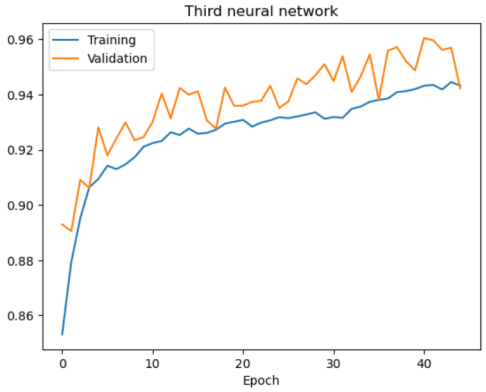

Figura 3.16. Tercera red Neuronal con *Dropout*

Como se puede contemplar ya ese pico no se produce tanto.

Otras medidas que se podrían aplicar a modelos que están sobreajustados son el *Early stopping*, que consiste en detener el entrenamiento en el momento que se observa un incremento en el valor de error de validación. También se podría simplificar el modelo, es decir, disminuir el número de capas y/o el números de neuronas en cada capa. Otro método que se podría aplicar es el que viene explicado en el siguiente artículo [46].

En el caso que estuviera subajustado, se podrían crear más capas de neuronas y/o neuronas por capas, es decir hacer el modelo más complejo.

A continuación, se puede observar en la Tabla 3.4 el área debajo de la curva *ROC*.

Tabla 3.4. Área debajo de la curva *ROC*

Algoritmo *Machine Learning*	Área (%)
Regresión Logística	78,46
SVC (*Sigmoid*)	58,50
SVC (*Poly*)	94,49
SVC (*RBF*)	95,86
Arbol de decisión	94,98
Primera red Neuronal	79,18
Secunda red Neuronal	92,57
Tercera red Neuronal	95,61

Por lo que se puede apreciar, si lo clasificamos por el área en primer lugar estaría *SVC* (*RBF*), después estaría la Tercera Red Neuronal y por último sería el Árbol de decisión.

Al estar utilizando un conjunto público podríamos comparar nuestros resultados con los de otros investigadores. Como es el caso del siguiente artículo [42]. En él se puede examinar que los autores decidieron analizarlo de una manera más compleja teniendo en cuenta la información de la página. Por ejemplo, una de sus variables dentro del *dataset* es *TCP_conversation_exhange*. Esta variable cuenta la cantidad de paquetes que hay entre el *honeypot* y el sitio web por el protocolo *TCP*. Nosotros, al hacer el procesamiento de los datos, nos centramos más en la información que nos proporcionaba la propia *Url*. Además, se aprecia que los otros autores los compara por la exactitud mientras que nosotros los hacemos por el *Recall*. Para comparar nuestros datos, se utilizará su segunda tabla ya que nuestro planteamiento a la hora de abordar los datos ha sido distinto. Como solo coincidimos en dos métodos compararemos esos dos métodos. En la Tabla 3.5 se pueden encontrar las exactitudes obtenidas por ambos programas.

Tabla 3.5. Comparación de la exactitud de ambos programas

Algoritmo *Machine Learning*	Exactitud (1) (%)	Exactitud (2) (%)
Support Vector Machine	97,70	97,41
Regresión Logística	81,48	90,58

La Exactitud (1) es la exactitud obtenida en nuestro programa y la Exactitud (2) es la exactitud obtenida por los investigadores. Podemos apreciar que la exactitud que ellos obtienen en la Regresión Logística es mejor que la nuestra. Esto se debe probablemente a su procesamiento de datos.

3.4. Evaluación y discusión

Para completar la evaluación se probaron los modelos ya entrenados con otro conjunto de datos diferente. Este nuevo conjunto de datos fue obtenido de la dirección https://github.com/ESDAUNG/PhishDataset/blob/main/data_bal%20-%2020000.xlsx.

Este nuevo conjunto solo tiene las *URLs* y el *compromissionType* ya que se sabe que son *URLs* de *phising*.

Por tanto, se decidió borrar el resto de columnas del conjunto de datos que eran *serverType*, *contentLength*, etc. Entrenando de nuevo cuatro de los modelos con el conjunto de datos original se obtuvo la Tabla 3.6:

Tabla 3.6. Predicción del conjunto original (son Fraudulentos)

Algoritmo *Machine Learning*	*Precision* (%)	*Recall* (%)	*Accuracy* (%)
Árbol de Decisión	50	91	95, 43
Tercera red neuronal	38	95	92, 85
Support Vector Machine (poly)	30	94	90, 01
Support Vector Machine (RBF)	37	95	92, 48

A continuación, en la Tabla 3.7 se muestran los resultados obtenidos de la predicción del nuevo conjunto con algunos de los modelos mencionados en la Tabla 3.2 y en la Tabla 3.4, siendo entrenados con el conjunto original de datos y eliminando las columnas mencionadas anteriormente.

En la Tabla 3.7 se visualizan los resultados obtenidos cuando los modelos ya entrenados intentan predecir las *URLs* fraudulentas del nuevo conjunto:

Tabla 3.7. Predicción del nuevo conjunto (son Fraudulentos)

Algoritmo *Machine Learning*	*Precision* (%)	*Recall* (%)	*Accuracy* (%)
Árbol de decisión	46	75	42,95
Tercera red neuronal	46	85	43,29
Support Vector Machine (poly)	47	75	44,81
Support Vector Machine (RBF)	49	87	47,33

En la Tabla 3.7 se puede observar que el programa, al intentar predecir un nuevo conjunto, lo máximo que es capaz de identificar de *URLs* fraudulentas es el 87 % . Probablemente ese porcentaje se debe a que el programa ha encontrado nuevas *urls* que son fraudulentas pero que no sabía que lo eran porque en su base de datos no encontró alguno semejante y, por tanto, lo identifica como no fraudulento.

3.5. Aplicación de computación cuántica

Una vez hecho descrito el análisis realizado del conjunto de datos con los algoritmos de *Machine Learning* con computación tradicional, se introduce a continuación una línea de trabajo enfocada a analizar algoritmos de *Machine Learning* vinculados a la computación cuántica.

Por ese motivo, en primer lugar, se hace una breve introducción sobre computación cuántica.

3.5.1. Cúbit

En la computación clásica se utilizan los *bit* [47]. Un *bit* $b \in \{0, 1\}$ es la unidad mínima de información que se emplean en los computadores clásicos. Mientras tanto, en los computadores cuánticos la unidad mínima de información se denomina cúbit. Al igual que el *bit*, el cúbit puede estar en dos estados básicos que denotaremos como$| 0\rangle, | 1\rangle$. Además, se tiene la propiedad del principio de superposición que permite a un cúbit no estar restringido a sus dos estados básicos, sino que puede estar en una combinación lineal de ambos. Se define el estado de un cúbit ψ sobre la base estándar $\{| 0\rangle, | 1\rangle\}$ como:

$$| \psi\rangle = \alpha \mid 0\rangle + \beta \mid 1\rangle \tag{3.34}$$

donde $|\alpha|^2 + |\beta|^2 = 1$, además $\alpha, \beta \in \mathbb{C}$ y se denominan α, β amplitudes de estado.

3.5.2. Esfera de *Bloch*

Al fijarse en la definición anterior, se aprecia que se puede representar un cúbit en un espacio de tres dimensiones. Para llegar a esta representación, lo primero que se hace es expresar en forma polar ambas amplitudes: $\alpha = r_0 \cdot e^{i\phi_0}, \beta = r_1 \cdot e^{i\phi_1}$.

A partir de ellas se puede representar nuestro cúbit como:

$$| \psi\rangle = r_0 \cdot e^{i\phi_0} \mid 0\rangle + r_1 \cdot e^{i\phi_1} \mid 1\rangle \tag{3.35}$$

Extrayendo $e^{i\phi_0}$ se obtiene: $| \psi\rangle = e^{i\phi_0} \cdot (r_0 \mid 0\rangle + r_1 \cdot e^{i(\phi_1 - \phi_0)} \mid 1\rangle)$ donde ϕ_0 se denomina fase global y $(\phi_1 - \phi_0)$ se denomina fase relativa.

Proposición 1 *La fase global de un cúbit no altera la probabilidad de medir sus estados base.*

Demostración. Sea ψ un cúbit con $\alpha, \beta \in \mathbb{C}$ de manera que $| \psi\rangle = \alpha \mid 0\rangle + \beta \mid 1\rangle$. Aplicando una fase global $c = e^{i\phi}$, se tiene que:

$$| c\psi\rangle = c\alpha \mid 0\rangle + c\beta \mid 1\rangle$$

. Como $|c| = |e^{i\phi}| = 1$, para todo ϕ se obtiene lo siguiente:

$$| 0\rangle = |c\alpha|^2 = |c|^2|\alpha|^2 = |\alpha|^2$$

$$| 1\rangle = |c\beta|^2 = |c|^2|\beta|^2 = |\beta|^2$$

■

Por otro lado, usando que $|\alpha|^2 + |\beta|^2$=1 y la definición de α y β. Se logra:

$$|\alpha|^2 + |\beta|^2 = |r_0 \cdot e^{i\phi_0}|^2 + |r_1 \cdot e^{i\phi_1}|^2 = |r_0|^2|e^{i\phi_0}|^2 + |r_1|^2|e^{i\phi_1}|^2 = 1$$

Dado que r_0 y r_1 son números reales, y además se tiene que $r_0^2 + r_1^2 = 1$. Se puede encontrar un ángulo $\frac{\theta}{2}$ de forma única tal que: $r_0 = \cos(\frac{\theta}{2})$, $r_1 = \text{sen}(\frac{\theta}{2})$. Reescribiendo un cúbit se consigue:

$$| \psi\rangle = \cos(\frac{\theta}{2}) | 0\rangle + e^{i\phi} \text{sen}(\frac{\theta}{2}) | 1\rangle$$

Sus valores cumplen que $0 \leq \phi < 2\pi$ y $0 \leq \frac{\theta}{2} < \frac{\pi}{2}$. De esta manera ya se puede obtener una representación de los cúbit como puntos en la superficie de una espera en un espacio $\mathbb{R}^3$. El ángulo *theta* corresponde a la latitud y el ángulo ϕ a la longitud.

3.5.3. Puertas cuánticas

Primero se introducen algunos conceptos necesarios.

Definición 1 *Sean dos espacios vectoriales V y W, la función $T : V \longmapsto W$, es una transformación lineal si cumple las siguientes propiedades:*

1. *para todo $v \in V$ y para todo $w \in W$, $T(v + w) = T(v) + T(w)$.*
2. *para todo $v \in V$ y para todo $c \in \mathbb{C}$, $T(cv) = cT(w)$.*

Definición 2 *Sea $M \in \mathbb{C}^{n\times n}$ diremos que M es una matriz unitaria si cumple que $MM^\dagger = I$, donde $M^\dagger$ es la matriz traspuesta conjugada.*

Definición 3 *Una puerta cuántica es un operador que actúa sobre cúbits y se representa por una matriz unitaria.*

Ejemplo 1 *Puerta NOT [49]: Es una transformación lineal que va:*

$$X : \mathbb{C}^2 \longmapsto \mathbb{C}^2$$
$$| 0\rangle \longmapsto | 1\rangle$$
$$| 1\rangle \longmapsto | 0\rangle$$

La matriz unitaria asociada a X es:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Lo primero que se va a ver es que la matriz asociada a X es unitaria.

Para ello, primero se calcula su traspuesta.

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Una vez calcula su traspuesta, se calcula su conjugada. En este caso resulta ser ella misma.

Por último, se tiene que comprobar que el producto de X y $X^\dagger$ es la identidad:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Una vez comprobado que efectivamente la matriz asociada a X es unitaria. Se comprueba que efectivamente las imágenes de $\mid 0\rangle, \mid 1\rangle$ son las definidas en la función X. Para ello, se tiene que tener en cuenta que $\mid 0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ y $\mid 1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \mid 1\rangle$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \mid 0\rangle$$

Efectivamente las imágenes de f están bien definidas.

Ejemplo 2 *La puerta Hadamard: Es una transformación lineal que va [49]:*

$$\begin{gathered} H : \mathbb{C}^2 \longmapsto C^2 \\ \mid 0\rangle \longmapsto \tfrac{1}{\sqrt{2}}(\mid 0\rangle + \mid 1\rangle) \\ \mid 1\rangle \longmapsto \tfrac{1}{\sqrt{2}}(\mid 0\rangle - \mid 1\rangle) \end{gathered}$$

La matriz unitaria asociada a H es:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Se deja como ejercicio hacia el lector comprobar que H es una matriz unitaria y las imágenes están bien definidas.

Ejemplo 3 *Puerta CNOT [49]*

$$
\begin{gathered}
C : \mathbb{C}^4 \longmapsto \mathbb{C}^4 \\
|\, a\, b\rangle \longmapsto |\, a\, a \oplus b\rangle \\
|\, 0\, 0\rangle \longmapsto |\, 0\, 0\rangle \\
|\, 0\, 1\rangle \longmapsto |\, 0\, 1\rangle \\
|\, 1\, 0\rangle \longmapsto |\, 1\, 1\rangle \\
|\, 1\, 1\rangle \longmapsto |\, 1\, 0\rangle
\end{gathered}
$$

La matriz unitaria asociada a C es:

$$
\begin{pmatrix}
1\,0\,0\,0 \\
0\,1\,0\,0 \\
0\,0\,0\,1 \\
0\,0\,1\,0
\end{pmatrix}
$$

Se deja como ejercicio hacia el lector comprobar que la matriz asociada a la puerta CNOT es unitaria y comprobar que las imágenes de C están bien definidas. Para poder realizarlo se tiene que tener en cuenta que :

$$
|\, 0\, 0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \; |\, 0\, 1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \; |\, 1\, 0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \; |\, 1\, 1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
$$

3.5.4. Circuitos cuánticos

Dado que en un ordenador clásico no se trabaja con un único *bit*, en los ordenadores cuánticos tampoco. Por ello, se hace imprescindible combinar cúbits para poder conseguir sistemas que permitan trabajar con mayor cantidad de información.

Definición 4 *Sean $A \in \mathbb{C}^n$ y $B \in \mathbb{C}^m$ de la forma*

$$
A = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}
$$

Se define el producto de Kronecker para vectores como:

$$
A \otimes B = \begin{bmatrix} a_1 \cdot B \\ \vdots \\ a_n \cdot B \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ \vdots \\ a_1 b_m \\ a_2 b_1 \\ \vdots \\ a_n b_m \end{bmatrix}
$$

Ejemplo 4 *Se muestra una forma sencilla de combinar dos cúbits. Sean* $\mid \psi_1\rangle = [\alpha_1, \beta_1]^T$ *y* $\mid \psi_2\rangle = [\alpha_2, \beta_2]^T$*. Por lo tanto, el sistema cuántico definido por ambos es:*

$$\mid \psi_1\rangle \otimes \mid \psi_2\rangle = \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_2\\ \beta_2\end{bmatrix} = \begin{bmatrix}\alpha_1\alpha_2\\ \alpha_1\beta_2\\ \beta_1\alpha_2\\ \beta_1\beta_2\end{bmatrix}$$

Definición 5 *Se dice que un sistema cuántico* $\mid \psi\rangle$ *está en un estado entrelazado si no existen* $\mid \psi_1\rangle, \mid \psi_2\rangle$ *de forma que* $\mid \psi\rangle = \mid \psi_1\rangle \otimes \mid \psi_2\rangle$*. Si se puede poner, entonces, diremos que es un estado producto.*

Ejemplo 5 *Demostrar que* $\frac{1}{\sqrt{2}} \mid 0\ 0\rangle + \frac{1}{\sqrt{2}} \mid 1\ 1\rangle$ *es un estado de entrelazamiento.*

Se supone por reducción al absurdo que es un estado producto. Entonces existen $\alpha_0, \alpha_1, \beta_0, \beta_1$*, de manera que:*

$$\begin{gathered}(\alpha_0 \mid 0\rangle + \alpha_1 \mid 1\rangle) \otimes (\beta_0 \mid 0\rangle + \beta_1 \mid 1\rangle) = \\ \alpha_0\beta_0 \mid 0\ 0\rangle + \alpha_0\beta_1 \mid 0\ 1\rangle + \alpha_1\beta_0 \mid 1\ 0\rangle + \alpha_1\beta_1 \mid 1\ 1\rangle = \\ \frac{1}{\sqrt{2}} \mid 0\ 0\rangle + \frac{1}{\sqrt{2}} \mid 1\ 1\rangle\end{gathered}$$

De esa igualdad se obtiene que $\alpha_0\beta_1 = 0 \wedge \alpha_1\beta_0 = 0$*. Por lo tanto, si* $\alpha_0\beta_1 = 0 \Longrightarrow \alpha_0 = 0 \vee \beta_1 = 0$*. Si* $\alpha_0 = 0 \Longrightarrow \alpha_0\beta_0 = 0 \quad \#$*, debido a que* $\alpha_0\beta_0 = \frac{1}{\sqrt{2}}$*. Por lo tanto,* $\beta_1 = 0 \Longrightarrow \alpha_1\beta_1 = 0 \quad \#$*, debido a que,* $\alpha_1\beta_1 = \frac{1}{\sqrt{2}}$*. Se concluye, por tanto que es un estado de entrelazamiento.*

Definición 6 *Sean dos matrices complejas* A *y* B *de la forma:*

$$A = \begin{bmatrix}a_{1,1} & \cdots & a_{1,n}\\ \vdots & \ddots & \vdots\\ a_{m,1} & \cdots & a_{m,n}\end{bmatrix}, \quad B = \begin{bmatrix}b_{1,1} & \cdots & b_{1,t}\\ \vdots & \ddots & \vdots\\ b_{p,1} & \cdots & b_{p,t}\end{bmatrix}.$$

Se define el producto de Kronecker como:

$$A \otimes B = \begin{bmatrix}a_{1,1} \cdot B & \cdots & a_{1,n} \cdot B\\ \vdots & \ddots & \vdots\\ a_{m,1} \cdot B & \cdots & a_{m,n} \cdot B\end{bmatrix} = \begin{bmatrix}a_{1,1}b_{1,1} & \cdots & a_{1,1}b_{1,t} & a_{1,2}b_{1,1} & \cdots & a_{1,n}b_{1,t}\\ \vdots & & \vdots & \vdots & & \vdots\\ a_{1,1}b_{p,1} & \cdots & a_{1,1}b_{p,t} & a_{1,2}b_{p,1} & \cdots & a_{2,n}b_{p,t}\\ a_{2,1}b_{1,1} & \cdots & a_{2,1}b_{1,t} & a_{2,2}b_{1,1} & \cdots & a_{2,n}b_{1,t}\\ \vdots & & \vdots & \vdots & & \vdots\\ a_{m,1}b_{1,1} & \cdots & a_{m,1}b_{p,t} & a_{m,2}b_{p,1} & \cdots & a_{m,n}b_{p,t}\end{bmatrix}$$

Definición 7 *Un registro cuántico es una colección de cúbits* $\psi_1, \cdots, \psi_n$ *que se utilizan para el cálculo.*

Definición 8 *Un circuito cuántico es una sucesión de puertas cuánticas* $P_1, \cdots, P_n$ *que se aplican sobre un registro cuántico.*

En la Fig. 3.17 se puede contemplar un ejemplo de un circuito cuántico.

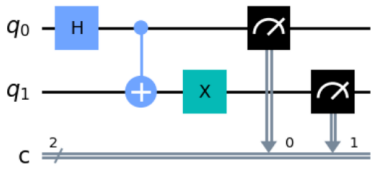

Figura 3.17. Ejemplo de circuito cuántico

3.5.5. Aplicación de *Machine Learning* cuántico

Tras esta breve introducción a la computación cuántica, nos aventuramos a programar con la librería *Qiskit*. En este trabajo se ha decidido utilizar un conjunto de datos clásicos y aplicarle un algoritmo de *Machine Learning* cuántico (*Quantum Machine Learning (QML)*), a través de los siguientes pasos:

- Para la codificación de los datos, que consiste en transferir los datos originales a cúbits mediante un mapeo de características, se ha elegido el algoritmo *ZFeatureMap*.
- Para la aplicación de un circuito cuántico parametrizado o *Ansatz*: circuito cuántico cuya principal característica es que posee un conjunto de pesos ajustables que deben minimizar una función objetivo. El *Ansatz* que se ha usado ha sido *RealAmplitudes*.
- Para la elección del algoritmo de optimización, que es una función equivalente a la de un modelo clásico de *Deep Learning*, se han empleado *COBYLA*(*Constrained Optimization By Linear Approximation optimizer*) y *SLSQP* (*Sequential Least Squares Programming optimizer*).

Adaptación del *dataset*

Debido a que se necesita trabajar con variables numéricas es necesario codificar las variables categóricas. Para ello, se ha empleado la codificación binaria. Consiste en establecer un orden jerárquico entre los valores de la variable.

A continuación, se detallan los campos seleccionados y los procesos realizados sobre el *dataset* original para permitir la aplicación de algoritmos *QML* considerando que todas las características deben ser numéricas.

- *url*: almacena el número total de caracteres en la *URL*.
- *compromissionType*: variable que indica si el sitio web está comprometido por phishing, defacement o es normal. Se transforma asignando un valor numérico a cada tipo (1, 2 o 0 respectivamente).
- *isHiddenFraudulent*: cambiado a 1 para URL fraudulentas o 0 para las benignas.
- *contentLength*: ya es una variable que toma valores enteros.
- *serverType*: valor numérico asignado a cada tipo de servidor.

- *poweredBy*: valor numérico asignado para cada plataforma de aplicación subyacente al servidor web.
- *contentType*: valor numérico asignado para cada tipo de codificación.

Dado el volumen del *dataset* y los tiempos de procesamiento requeridos, se realizó una primera aproximación con un *dataset* reducido (200 observaciones), balanceado (100 *URLs* maliciosas y 100 no maliciosas).

Así, se analizó la diferencia entre trabajar en el modelo *SVM* clásico con el conjunto de datos completo y la versión reducida. Primero se aplicó el modelo *SVM* clásico al conjunto de datos completo, obteniendo:

Tabla 3.8. Predicción del conjunto de datos completos

Precision (%)	*Recall* (%)	*F1-score* (%)
97	100	98
97	34	50

A continuación, se aplicó el modelo *SVM* clásico al conjunto de datos reducido.

Tabla 3.9. Predicción del conjunto de datos reducido

Precision (%)	*Recall* (%)	*F1-score* (%)
90	100	95
100	91	95

Como se dijo anteriormente, se utiliza el *ZFeatureMap* con 2 repeticiones. Se puede observar en la Fig. 3.18

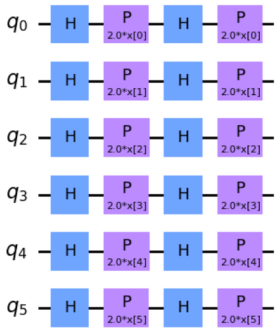

Figura 3.18. Mapeo de características con *ZFeatureMap*

Además para el *Ansazt* se utilizó *RealAmplitudes* con 2 repeticiones. Se puede visualizar en la Fig 3.19

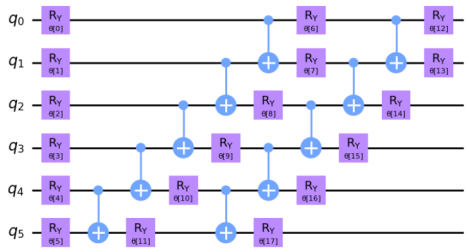

Figura 3.19. Representación *Ansatz RealAmplitudes*

Al aplicar el *VQC* con el optimizador *SLSQP* al conjunto de datos reducido se concluye la tabla 3.10:

Tabla 3.10. Predicción VQC del conjunto de datos reducido optimizador *SLSQP*

Precision (%)	*Recall* (%)	*F1-score* (%)
90	100	95
100	91	95

A continuación, se puede contemplar en la Fig.3.20 la matriz de confusión del modelo de *VQC* con el optimizador *SLSQP*.

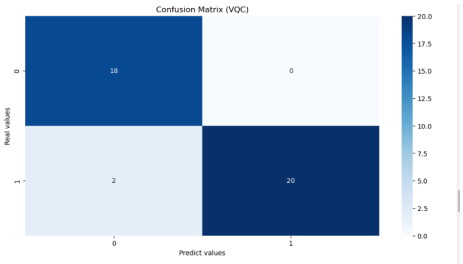

Figura 3.20. Matriz de confusión *VQC* con el optimizador *SLSQP*

Al aplicar el *VQC* con el optimizador *COBYLA* al conjunto de datos reducido se consigue la tabla 3.10:

Por último, se puede ver en la Fig.3.21 la matriz de confusión del modelo de *VQC* con el optimizador *COBYLA*.

Tabla 3.11. Predicción *VQC* del conjunto de datos reducido - optimizador *COBYLA*

Precision (%)	*Recall* (%)	*F1-score* (%)
86	100	92
100	86	93

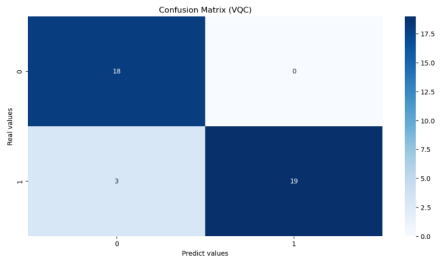

Figura 3.21. Matriz de confusión *VQC* con el optimizador *COBYLA*

4

Conclusiones

En este trabajo se aborda uno de los problemas de ciberseguridad más relevantes [48]. Concretamente, el objetivo ha sido buscar el modelo más óptimo para clasificar *URLs* fraudulentas con objeto de evitar ataques de *phishing*.

Se realizó un análisis con varios *datasets*, concluyéndose que el primero era no balanceado. Por ese motivo se decidió prestar especial atención a los Falsos Positivos y a los Falsos Negativos. Desde el punto de vista de una empresa, lo más recomendable es intentar reducir el número de Falsos Negativos debido a que si se deja pasar una *URL* fraudulenta eso podría implicar grandes pérdidas económicas. Sin embargo, al hacer eso, aumenta el número de Falsos Positivos, además, se puede observar cómo la Precisión disminuye pero aumenta el *Recall*. En el caso de que se quiera intentar conseguir el menor número de Falsos Positivos, sin aumentar mucho el numero de Falsos Negativos, la mejor medida para comparar los modelos sería el *F1-score*.

Se escogieron diferentes modelos y tras analizarlos se concluyó que el mejor para el conjunto de datos usado fue la tercera red neuronal creada. El segundo mejor modelo fue el de *Support Vector Machine* con núcleo RBF. Por último, el tercer mejor modelo fue el de *Support Vector Machine* con núcleo *Poly*.

Con respecto a la parte cuántica, las conclusiones que se han obtenidos es que el optimizador *SLSQP* es un poco mejor que el optimizador *COBYLA* al identificar las *URLs* fraudulentas. Los resultados que se han obtenido son alentadores porque aunque se trabaje con un pequeño conjunto de datos se abre una puerta a otras aplicaciones en el campo de ciberseguridad. En velocidad de entrenamiento, el optimizador *COBYLA* es con diferencia el más rápido ya que no utiliza descenso de gradiente. Por tanto, es una opción a tener en cuenta cuando el factor tiempo es determinante.

Como trabajo futuro relacionado se plantea la mejora de la metodología usada para aplicarla a otros conjuntos de datos relacionados con otros problemas de ciberseguridad y además se plantea mejorar el análisis del *Quantum Machine Learning* haciendo uso de más algoritmos y comparando con diferentes *Ansatz* y mapeos de características.

Bibliografía

[1] Sudar, K.M., Deepalakshmi, P., Nagaraj, P., Muneeswaran, V. (2020). Analysis of cyberattacks and its detection mechanisms. IEEE International Conference on Research in Computational Intelligence and Communication Networks, pp. 12-16.

[2] Postigo Palacios, A. (2020). Seguridad informática. Ediciones Paraninfo.

[3] Cano, J.J. (2020). Retos de seguridad/ciberseguridad en el 2030. Revista Sistemas, (154), 68-79.

[4] Observatorio Español de Delitos Informáticos. http://oedi.es/estadisticas/

[5] Patel, C.M., Borisagar, A.P.V.H. (2012). Survey on taxonomy of ddos attacks with impact and mitigation techniques. International Journal of Engineering Research and Technology, 1(9).

[6] Guardeño, D.A., Martínez, V.G., Encinas, L.H. (2020). Ciberseguridad. Catarata.

[7] Incibe. (2020). Los ordenadores con macOS no son invulnerables, ¿sabes protegerlos?. https://www.incibe.es/empresas/blog/los-ordenadores-macos-no-son-invulnerables-sabes-protegerlos

[8] Aceituno, V. (2004). Seguridad de la Información. Creaciones Copyright.

[9] Morillo, C. (2022). 97 cosas que todo profesional de la seguridad informática debería de saber. Anaya.

[10] González, S. (2021). Por qué el Machine Learning es un gran aliado para la ciberseguridad. https://www.welivesecurity.com/la-es/2021/12/10/por-que-machine-learning-aliado-para-ciberseguridad

[11] Sancho Caparrini, F. (2017). Fundamentos Matemáticos del Machine Learning (I) http://www.cs.us.es/~fsancho/?e=169.

[12] BBVA. (2019) Machine learning: ¿qué es y cómo funciona?. https://www.bbva.com/es/innovacion/machine-learning-que-es-y-como-funciona/

[13] Coronado Huamán, H.H., Han, A., Sanz García, L. (2020). Detección automática de sitios web fraudulentos. Universidad Complutense de Madrid.

[14] Floridi, L. (2021). Artificial intelligence, deepfakes and a future of ectypes. Ethics, Governance, and Policies in Artificial Intelligence, pp. 307-312.
[15] Ciftci, U.A., Demir, I., Yin, L. (2020). Fakecatcher: Detection of synthetic portrait videos using biological signals. IEEE transactions on pattern analysis and machine intelligence.
[16] Passos, L.A., Jodas, D., da Costa, K.A., Júnior, L.A.S., Colombo, D., Papa, J.P. (2022). A review of deep learning-based approaches for deepfake content detection. arXiv preprint arXiv:2202.06095.
[17] Martin, D. (1987). Técnicas avanzadas para las bases de datos. Editorial Omega.
[18] Ridwan, M.A., Radzi, N.A.M., Abdullah, F., Jalil, Y. E. (2021). Applications of machine learning in networking: a survey of current issues and future challenges. IEEE access, 9, pp. 52523-52556.
[19] Thomas, C., Sharma, V., Balakrishnan, N. (2008). Usefulness of DARPA dataset for intrusion detection system evaluation. In Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008, Vol. 6973, pp. 164-171.
[20] Thanh, C.T. (2021). A study of machine learning techniques for cybersecurity. IEEE International Conference on Advanced Computing and Applications, pp. 54-61.
[21] Gondalia, A., Shah, A. (2022). A Survey of Advancement in AnomalyIntrusion Detection System. Research Square.
[22] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. (2009). A Detailed Analysis of the KDD CUP 99 Data Set, IEEE Symposium on Computational Intelligence for Security and Defense Applications.
[23] Canadian Institute for Cybersecurity, Intrusion Detection Evaluation Dataset (CIC-IDS2017). (2017). https://www.kaggle.com/datasets/cicdataset/cicids2017/code
[24] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1, pp. 108-116.
[25] Song, J., Takakura, H., Okabe, Y. (2006). Description of kyoto university benchmark data. http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf
[26] Brunswick, U. (2010). Intrusion detection evaluation dataset (ISCXIDS2012). Canadian Institute for Cybersecurity, 11.
[27] Pérez, D., Alonso, S., Morán, A., Prada, M.A., Fuertes, J.J., Domínguez, M. (2019). Comparison of network intrusion detection performance using feature representation. In International conference on engineering applications of neural networks, pp. 463-475. Springer.

[28] Cubas, J.V., Niño, G.M. (2022). Modelo de machine learning en la detección de sitios web phishing. Revista Ibérica de Sistemas e Tecnologias de Informação, E52, pp. 161-173.
[29] Albán Toapanta, D.F. (2022). Análisis y diseño de un modelo predictivo para detección de phishing basado en url y corpus del correo electrónico. Bachelor's thesis, Quito.
[30] Géron, A. (2020). Aprende machine learning con scikit-learn, keras y tensorflow. España: Anaya.
[31] Bartlett, P.L., Butucea, C., Schmidt-Hieber, J. (2022). Mathematical Foundations of Machine Learning. Oberwolfach Reports, 18(1), 853-894.
[32] Kowalczyk, A. (2017). Support vector machines succinctly. Syncfusion Inc.
[33] León, E.C. (2017). Introducción a las máquinas de vector soporte (SVM) en aprendizaje supervisado. Trabajo de Fin de Grado en Matemáticas, Universidad de Zaragoza.
[34] López Jaimes, D. S. (2022). Data driven initialization for machine learning classification models. Rosario University.
[35] Ramírez J., Chacón Q., Mario, M. (2011). Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década. Revista de Ingeniería Eléctrica, Electrónica y Computación 9, pp. 7-16.
[36] Perceptrón Multicapa http://avellano.fis.usal.es/~lalonso/RNA/introMLP.htm
[37] Kostadinov, S. (2019). Understanding backpropagation algorithm. Towards Data Science. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
[38] McGonagle, J., Shaikouski, G., Williams, C., Hsu, A., Khim, J., Miller, A. (2018). Backpropagation. Brilliant Math Science Wiki, brilliant. org/wiki/backpropagation
[39] Arce, J.I.B. (2019). La matriz de confusión y sus métricas. Health Big Data. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/
[40] Paloma Recuero de los Santos, P. (2021). Cómo interpretar la matriz de confusión: ejemplo práctico. Telefonica. https://telefonicatech.com/blog/como-interpretar-la-matriz-de-confusion-ejemplo-practico
[41] Fernández, C.L., Canaza, J.M.S., Ugarte, D.J.P., Quispe, J.Y.L. (2022). Aplicación de los árboles de decisión en la identificación de sitios web fraudulentos. Innovación y Software, vol. 3(1), pp. 6-16.
[42] Urcuqui, C., Navarro, A., Osorio, J., García, M. (2017). Machine Learning Classifiers to Detect Malicious Websites. Proceedings of the Spring School of Networks SSN, vol. 1950, pp. 14-17.
[43] Chiramdasu, R., Srivastava, G., Bhattacharya, S., Reddy, P.K., Gadekallu, T.R. (2021). Malicious url detection using logistic regression. IEEE International Conference on Omni-Layer Intelligent Systems, pp. 1-6.

[44] Moncada Vargas, A.E. (2021). Comparación de técnicas de machine learning para detección de sitios web de phishing. Tesis, Universidad de Lima.
[45] Li, T., Kou, G., Peng, Y. (2020). Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. Information Systems, vol. 91, p. 101494.
[46] Li, H., Li, J., Guan, X., Liang, B., Lai, Y., Luo, X. (2019). Research on overfitting of deep learning. International conference on computational intelligence and security, pp. 78-81.
[47] Vicente López Oliva " Fundamentos de la computación cuántica".
[48] Águeda, P. (2022). Un hackeo a través del Poder Judicial roba a Hacienda datos de medio millón de contribuyentes. El Diario.
[49] IBM. (2023). Qiskit 'Resumen de Operaciones Cuánticas. https://qiskit.org/documentation/locale/es_UN/tutorials/circuits/3_summary_of_quantum_operations.html

Analytical techniques to detect cybersecurity problems

Nuria Reyes Dorta
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu0101328690@ull.edu.es

Abstract

One of the big cybersecurity problems is detecting Phishing, among them are fraudulent URLs, which are links that seem to send you to a legitimate page but actually redirect you to a page 'false page. For this reason, in this work we propose to use different Machine Learning algorithms, and in particular Deep Learning to classify fraudulent URLs.

1. Introduction

For the detection of fraudulent URLs, the used dataset was obtained from [1]. In Fig. 1 we can watch the dataset.

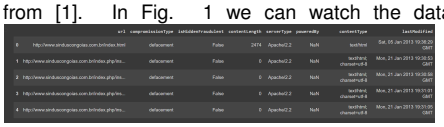

Figure 1: *First five columns of the used dataset*

It contains the following information:

- *url* is the current URL.
- *compromissionType* is the variable that indicates if the website is compromised by phishing, defacement, or normal.
- *isHiddenFraudulent* is a dependent variable indicating whether the URL is fraudulent or not.
- *contentLength* is a variable that takes integer values and was obtained by sending an HTTP HEAD request to the URL. It also indicates the size of the message body, in bytes, sent to the recipient.
- *serverType* is a string indicating the server, such as Apache, Microsoft IIS.
- *poweredBy* is a string that indicates the application platform underlying the web server, that is, it is used to specify with which software the response has been generated by the server.
- *contentType* contains charset information that is of the encoding type.
- *lastModified* is a variable indicating when its last modified date was.

Both the *poweredBy* and *serverType* variables have been preprocessed to hold the framework name and the major and minor version number. Besides, for the implementation, it was decided remove *lastModified* variable. It was created new variable as url length, the number of times they appear '%', '.', etc.

2. Machine Learning

The *Machine Learning* is a artificial intelligence branch which, allows machines to learn without being expressly programmed to do so. That is, it is capable of converting a data sample into a computer program capable of extracting data from new data sets for which it has not been previously trained.
The algorithms that it were used to finally analyze dataset were:

- Logistic Regression
- Decision Tree
- Support Vector Machine
- Neural Networks

3. Results

Next, the results obtained from the Machine Learning algorithms but without the Neural Networks will be shown.

Table 1: *Comparison of the results obtained as non-fraudulent*

Machine Learning algorithm	Precision (%)	Recall (%)	F1-score (%)
Logistic regression	99	82	89
SVC (Sigmoid)	97	58	72
SVC (Poly)	100	96	98
SVC (rbf)	100	98	99
Decision tree	100	99	100

Table 2: *Comparison of the results obtained as fraudulent*

Machine Learning algorithm	Precision (%)	Recall (%)	F1-score (%)
Logistic regression	16	75	27
SVC (Sigmoid)	6	59	11
SVC (Poly)	55	93	69
SVC (RBF)	68	94	79
Decision tree	89	91	90

References

[1] Machine Learning Lab, 'HIDDEN FRAUDULENT URLS DATASET'.

[2] O' Reilly Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow