UNIVERSIDAD
NACIONAL
DE COLOMBIA

Implementación de modelo computacional para la detección de Ingeniería Social basado en Aprendizaje de Máquina y Procesamiento de Lenguaje Natural

Juan Camilo López Solano

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2022

Implementación de modelo computacional para la detección de Ingeniería Social basado en Aprendizaje de Máquina y Procesamiento de Lenguaje Natural

Juan Camilo López Solano

Trabajo de investigación presentada(o) como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director (a):

Ph.D. Jorge Eliecer Camargo Mendoza

Línea de Investigación:

Sistemas inteligentes

Grupo de Investigación:

UNsecure Lab

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá, Colombia

2022

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

Juan Camilo López Solano

Fecha 18/04/2022

Resumen

Implementación de modelo computacional para la detección de Ingeniería Social basado en Aprendizaje de Máquina y Procesamiento de Lenguaje Natural

La seguridad informática o ciberseguridad se encarga de la protección de datos y servicios ante individuos no autorizados y protege las características de la información como la integridad, la confidencialidad y la disponibilidad. Existen múltiples amenazas y ataques que ponen en riesgo la seguridad informática como el ransomware, el malware o programas malignos, los ataques de denegación de servicios, las fallas de inyección, la ingeniería social, entre otros. En muchas ocasiones la parte más vulnerable de los sistemas son los usuarios, por este motivo los ciberdelincuentes usan la ingeniería social para adquirir información de forma ilícita de los usuarios. La ingeniería social consiste en la manipulación de los individuos mediante el engaño para que divulguen información privada o confidencial. Este tipo de ciberataque es muy difícil de detectar ya que puede ser ejecutado por cualquier individuo en cualquier momento y explota aspectos psicológicos de los humanos para engañarlos.

En el presente trabajo se presenta la implementación de un modelo computacional basado en técnicas de Procesamiento de Lenguaje Natural para extraer características en textos y alimentar tres algoritmos de Aprendizaje de Máquina (r*edes neuronales, máquinas de vector de soporte* y *bosques aleatorios*) para detectar posibles ataques de ingeniería social en textos. Los tres algoritmos fueron entrenados y evaluados, mostrando resultados que superan el 80% de exactitud en la detección de ataques de ingeniería social.

Palabras clave: Cybersecurity, Social Engineering, Natural Language Processing, Machine Learning

Abstract

Implementation of computational model for Social Engineering detection based on Machine Learning and Natural Language Processing

Computer security or cybersecurity is responsible for the protection of data and services against unauthorized people and protects information characteristics such as integrity, confidentiality, and availability. There are multiple threats and attacks that put computer security at risk such as ransomware, malware, denial of services attacks, injection failures, social engineering, among others. In many cases, the most vulnerable part of systems are users, for this reason cybercriminals use social engineering to illegally acquire information from users. Social engineering consists of the manipulation of people through deception to make them disclose private or confidential information. This type of cyber-attack is very difficult to detect since it can be executed by any individual at any time and exploits psychological aspects of humans to deceive them.

This paper presents the implementation of a computational model based on Natural Language Processing techniques to extract characteristics in texts and used to train three Machine Learning algorithms (*Neural Network, Support Vector Machine* and *Random Forest*) to detect possible social engineering attacks in texts. The three algorithms were trained and tested showing an accuracy over 80% in the task of detecting social engineering attacks.

Keywords: Cybersecurity, Social Engineering, Natural Language Processing, Machine Learning

Este Trabajo Final de maestría fue calificado en abril de 2022 por el siguiente evaluador:

Luis Fernando Niño Vásquez PhD.
Profesor Facultad de Ingeniería
Universidad Nacional de Colombia

Contenido

Lista de figuras

Lista de tablas

Introducción

El internet se ha convertido en el mayor medio de comunicaciones del mundo. Diario se hace uso servicios como el correo electrónico, las redes sociales, mensajería instantánea, entre otros, para comunicarnos y compartir información, lo que ha reducido la interacción cara a cara. El rápido desarrollo de la tecnología ha permitido que estos servicios sean sumamente accesibles para todas las personas, siendo el único requisito tener acceso a internet para poder acceder a ellos. Sin embargo, en los últimos años estos medios de comunicación y transferencia de datos han sido usados para actividades ilícitas, creciendo así el número de ciberataques considerablemente (Infoblox, 2020). Por ende, se hace necesario un estudio ampliado de técnicas y métodos para mejorar la seguridad informática, prevenir los ciberataques y las amenazas a la información.

La ciberseguridad o seguridad informática es el área encargada de proteger datos y servicios de personas no autorizadas. Por consiguiente, es necesario realizar mejoras continuamente, y de esta manera evitar el acceso a información sensible de usuarios y organizaciones (Bezuidenhout et al., 2010). La seguridad informática es también la responsable de proteger las características de la información, tales como la confidencialidad, integridad y disponibilidad de esta (Mokhor et al., 2017).

Por otra parte, la ingeniería social es un tipo de vulneración a dicha ciberseguridad, definida por Hadnagy (2010) como el arte o la ciencia de manipulación del comportamiento propio de las personas o individuos; o en palabras de Dan y Gupta (2019) se entiende como manipulación o control de individuos por medio del engaño y la influencia sobre dicha persona, usada con el objetivo de divulgar información privada y/o confidencial. Este tipo de ataque es bastante común, dado que no son necesarios conocimientos técnicos. El atacante se vale de aspectos psicológicos y emocionales para acceder a la información de los individuos, quienes comúnmente son usuarios de los sistemas y suelen ser la parte más vulnerable de estos (Stajano y Wilson, 2011). Junto con el crecimiento de diferentes

herramientas tecnológicas ha crecido simultáneamente el número de ciberataques. Un ejemplo del aumento de estas herramientas son las redes sociales. Hoy en día las personas se comunican cada vez más por medio de redes sociales, ya que estas facilitan la comunicación instantánea entre individuos, siendo el único requisito una conexión a internet.

Además, otra ventaja de las redes sociales y la rápida evolución de estas, es que permiten un alcance máximo de audiencias por parte de compañías, empresas y organizaciones, quienes, de no ser por las redes sociales y el compromiso de los individuos con las mismas, no podrían alcanzar estas altas cifras (Del Pozo, 2019). Retomando el aumento de ciberataques, según el reporte de datos en inteligencia sobre amenazas o *Cyberthreat Intelligence Report* de Infoblox (2020), el mayor número de ciberataques registrados están relacionados con violación o filtración de datos personales (*Data Breach), Phishing*, *Smishing*, *Pharming*, estafas y extorsión, involucrando la mayoría de estos, ingeniería social.

Aunque es común relacionar los ataques de ingeniería social con transferencias de dinero al atacante, muchos de estos son realizados con el objetivo de acceder a datos de compañías o de individuos, como información personal o declaraciones de impuestos de empleados, según informa Infoblox. Esta práctica ha sido recurrente en años recientes representando un gran peligro para los usuarios de los sistemas, ya que tiene un gran potencial de daño, como se discute en la décima conferencia internacional de computación semántica del IEEE, en el año 2016 (Sawa et al., 2016). Por otra parte, un reporte de la empresa de telecomunicaciones Verizon indica, que el 29% de las vulneraciones a la seguridad involucran ingeniería social vía email, llamada telefónica, SMS, reuniones presenciales y sitios web. Además, la misma compañía publica un reporte de una prueba realizada, en la cual enviaron 150.000 emails a varios individuos. Los resultados indican que en la primera hora un 50% de los participantes abrieron el email y entraron a links maliciosos que se encontraban en el contenido del mail, lo que se conoce como *Phishing* (Bhardwaj et al., 2014).

En lo que respecta a la motivación para efectuar ataques de ingeniería social, según una gráfica de un artículo del *International Journal of Advanced Computer Research* (Nabie-Schmick, 2016) hay diferentes factores que motivan a los atacantes: acceso a información

prioritaria, ganancias económicas, ventaja competitiva, venganza, simple diversión y 'otras', siendo sus porcentajes 30%, 23%, 21%, 10%, 11% y 5% respectivamente. Lo anterior indica que son diversas las causas, no principalmente un beneficio económico, y, por ende, cualquier persona que posea motivación y medios puede efectuar un ataque usando la ingeniería social. Esa es la razón por la cual varias compañías establecen, generalmente, políticas de seguridad informática, en las cuales ha sido un tema recurrente la educación sobre ingeniería social como método preventivo. Sin embargo, estos entrenamientos suelen ser olvidados con frecuencia (Bezuidenhout et al., 2010). Según Sawa y compañía (2016) la eficacia de estos entrenamientos a la hora de detectar un posible ataque de ingeniería social depende de la metacognición, definiéndola como la conciencia del usuario de su estado mental durante un ataque.

Por último, a pesar de diferentes intentos como los modelos SEADM (Bezuidenhout et. al., 2010) y SEADMv2 (Mouton et. al., 2016), estos modelos no son del todo fiables debido a que necesitan de la interacción del usuario y recaen en la metacognición de este. Por lo tanto, es necesario un sistema que detecte los ataques de ingeniería social de forma automática, que pueda ser aplicado a un gran rango de tipos de ataques, y a su vez requiera la interacción mínima de los usuarios (Bhakta y Harris, 2015).

Debido a lo anterior, la presente investigación realiza una revisión de literatura para ayudar futuras investigaciones, y desarrolla un modelo computacional para detectar ataques de ingeniería social de forma automática, analizando únicamente texto, basado en técnicas de Procesamiento de Lenguaje Natural y algoritmos de clasificación de Aprendizaje de Máquina.

Otras contribuciones de este trabajo fueron la contrucción de un dataset de textos de ingeniería social en español el cual contiene 1118 textos clasificados con la etiqueta de ataque o no ataque de ingeniería social[1]. Además, de la publicación de la investigación:

López, J., & Camargo, J., (Para ser presentada en marzo 2022). ***Social Engineering Detection Using Natural Language Processing and Machine Learning****.* En la

[1] https://github.com/juclopezso/social-engineering-detection

conferencia *“The 5th International Conference on Information and Computer Technologies (ICICT), 2022”.*

Este trabajo presenta un método de detección automático de ataques de ingeniería social teniendo en cuenta únicamente textos de diálogos como entrada, detectando los ataques mediante el uso de procesamiento de lenguaje natural, para la extracción de características del texto, y aprendizaje de máquina para la detección automática estos ataques.

1. Estado del arte

En este capítulo se describe el estado del arte sobre la detección de ataques de ingeniería social y se detallan los conceptos necesarios para el desarrollo de la investigación.

1.1. Ciberseguridad

La ciberseguridad se encuentra en muchas dimensiones y contextos, debido a que abarca cualquier tema relacionado a la seguridad y ciberespacios. Esta se aplica en múltiples disciplinas como en ciencias de la computación, ingeniería, educación, salud, políticas, sociología y psicología. La ciberseguridad es definida, dependiendo del contexto, por múltiples autores. Craigen y compañía (2014) hacen una revisión de múltiples definiciones previas, para finalmente definir la ciberseguridad como la organización y recolección de recursos, procesos y estructuras usadas para proteger el ciberespacio de amenazas que afecten la propiedad y los derechos de la información.

Según Kaspersky (2022) la ciberseguridad corresponde a “la práctica de defender las computadoras, los servidores, los dispositivos móviles, los sistemas electrónicos, las redes y los datos de ataques maliciosos.” También definen los diferentes tipos de ciberseguridad, como la seguridad de red, de las aplicaciones, de la información y operativa. La seguridad de red consiste básicamente en mantener las redes protegidas de intrusos o *malware*. La seguridad de las aplicaciones se basa en mantener el software y las aplicaciones libres de amenazas de cualquier tipo. La seguridad de la información corresponde a la protección de la integridad y privacidad de datos en tránsito o almacenamiento. Y por último, la seguridad operativa corresponde a los procesos y decisiones del manejo de datos.
Kaspersky menciona diferentes tipos de ciberamenazas: los delitos cibernéticos, ejecutados por grupos de individuos o personas aisladas, con el objetivo de obtener

beneficios financieros o sabotaje; los ciberataques, que corresponden a la recopilación de información con objetivos relacionados a la política y, finalmente, el ciberterrorismo, que corresponde a la debilitación de sistemas electrónicos, en miras de causar terror o pánico.

Teniendo en cuenta las amenazas descritas anteriormente, se enumeran a continuación los métodos más usados para amenazar la seguridad informática:

1.1.1 Malware

Una traducción literal de *malware* sería "programa maligno". El malware se refiere a cualquier forma de software malicioso. Este es una de las mayores amenazas a la seguridad informática debido al amplio alcance de estos ataques. Su objetivo es infiltrarse dentro de un sistema de forma ilícita para robar información, dañar, espiar un sistema, u otras acciones ilegales. Dentro del malware se encuentran distintitos tipos de virus como los troyanos, gusanos, *ransomware, rootkits, keyloggers,* entre otros (Srivalli y Prasanna, 2019). En el reporte anual de ciberamenzas de Sonic Wall (2021), se reportaron 5.600 millones de ataques de *malware* a individuos y organizaciones en el año 2020, un 43% menos que en el año 2019. Además, en el mismo reporte se informa que los ataques de tipo *ransomware* aumentaron en el año 2020 un 62% respecto al año 2019, con un total de 304.6 millones de ataques reportados.

1.1.2 Ataques de denegación de servicios

Los ataques de denegación de servicios o *DoS* (*Denial of Service*) es un tipo de ciberataque que consiste en evitar que los usuarios legítimos de un servicio o sistema puedan acceder a este. Además del DoS, existe otro tipo de ataque de denegación de servicio llamado *DDoS* por sus siglas en inglés (*Distributed Denial of Service*). Su objetivo es el mismo del ataque *DoS,* sin embargo, el ataque se realiza de forma distribuida a un objetivo. Existen varios métodos por los cuales se realizan los ataques de denegación de servicios, como *HTTP-DoS, Ping Flood Attack* o *SYN Flood Attack* (Khorshed et al., 2014). En el mes de septiembre del año 2020, varias compañías de los Estados Unidos fueron amenazadas con ataques de denegación de servicios por parte de grupos de *hackers.* Estos grupos exigían a sus víctimas pagos de 10 a 20 *Bitcoins* para

que no fueran atacadas. Los atacantes afirmaban que realizarían el ataque enviando 2 Terabytes por pegundo (TBps) a los sitios web de las compañías si estas no pagaban el dinero exigido (Gatlan, 2020).

1.1.3 Ataques de inyección de código malicioso

Los ataques de inyección consisten en la inyección de código malicioso en cualquier tipo de base de datos, con el objetivo de robar información o tomar el control de esta. Este ataque puede ser realizado a bases de datos de tipo relacional y no relacional, también mediante el protocolo *Lightweight Directory Access Protocol* (LDAP). Este es uno de los ataques más usados en la actualidad, según el reporte anual de OWASP (2021), este tipo de ataque el tercero más perpetrado en el año 2021.

1.1.4 Ataques de ingeniería social

La ingeniería social es un tipo de ciberataque que puede ser usado para extraer información de una víctima de forma ilícita y/o manipular su comportamiento. El ciberataque más común que usa la ingeniería social es el *phishing.* Este ciberataque consiste en enviar mensajes fraudulentos a las víctimas, los cuales contienen algún tipo de enlace maligno. Este enlace redirige a la víctima a un sitio web malicioso con la finalidad de robar información de la víctima, como número de tarjetas de crédito, número de identificación, ente otros.

1.2 Ingeniería social

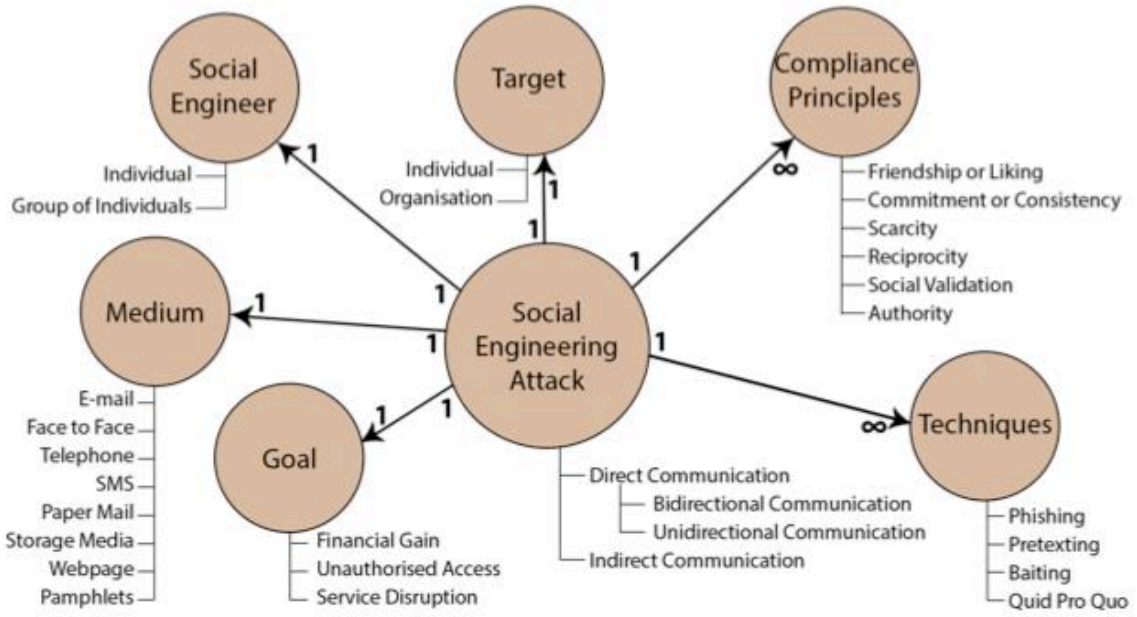
Como se menciona anteriormente en la introducción, la ingeniería social es uno de los métodos más efectivos para poder acceder a información personal de los usuarios de los sistemas, o incluso sacar provecho económico de los mismos. Sin embargo, al ser la ingeniería social uno de los métodos más comunes y exitosos de vulneración a la seguridad, gracias a que no se necesitan conocimientos extensos del área de la informática, es un fenómeno que ha sido estudiado ampliamente.

Existen muchos tipos de ingeniería social y pueden ser usados en múltiples situaciones con un alcance limitado únicamente por el atacante. A gran escala, la ingeniería social se

divide en dos categorías: ingeniería social basada en interacción humana e ingeniería social basada en interacción tecnológica (Ivaturi y Janczewski, 2011).

Mouton y compañía (2014) proponen un modelo ontológico de los ataques de ingeniería social, definiendo a grandes rasgos estos ataques como se muestra en la figura 1-1.

Figura 1-1: Modelo ontológico de ataques de ingeniería social



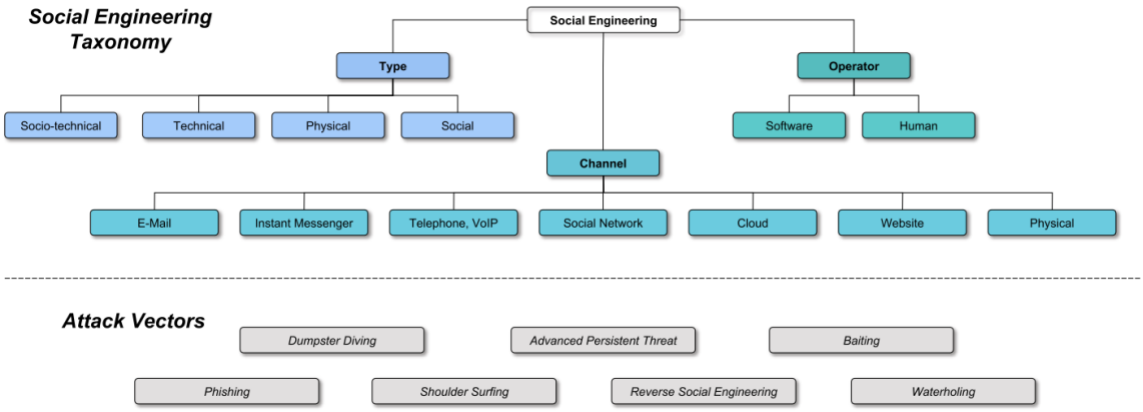
Fuente: (Mouton et al., 2014)

En el modelo se pueden ver seis componentes principales de los ataques de ingeniería social: la meta, que puede ser beneficios económicos, obtener acceso no autorizado a un servicio o sistema y la interrupción de servicios; el medio, que a grandes rasgos puede ser tecnológico o físico (personal); la ingeniería social, la cual puede ser a sólo una persona o un grupo de personas; el objetivo o víctima, estos pueden ser personas u organizaciones; los principios de cumplimiento, que se refiere a los principios psicológicos que son explotados o aprovechados por los ingenieros sociales (véase numeral 1.2.2) y finalmente, las técnicas, que son los métodos por los cuales los ingenieros sociales realizan un ataque (véase numeral 1.2.3). El modelo también muestra tres tipos de comunicación:

- **Comunicación bidireccional (directa):** Este tipo de comunicación ocurre cuando dos o más individuos son parte de la conversación. Por ejemplo, una conversación cara a cara entre personas o en un ambiente virtual como un chat.
- **Comunicación unidireccional (directa):** Este ocurre cuando la comunicación tiene un único sentido, hay sólo un emisor. Por ejemplo, un email, el cual no se puede responder. Los ataques de *phishing* usan este tipo de comunicación.
- **Comunicación indirecta:** Ocurre cuando no hay comunicación directa entre las partes, por ejemplo, los ataques de *baiting*, donde el medio de comunicación es un dispositivo de almacenamiento.

Los autores Krombholz y compañia (2015) proponen una taxonomía muy detallada de los ataques de ingeniería social. Ellos los dividen en cuatro componentes: canales, operadores, tipos y vectores de ataques.

Figura 1-2: Taxonomía de los ataques de ingeniería social



Fuente: (Krombholz et al., 2015)

A continuación se detalla el modelo propuesto por Krombholz y su grupo investigativo:

- **Canales:** Son servicios, formas o herramientas que permiten la comunicación entre los individuos; como el email, mensajes instantáneos, teléfono, redes sociales, servicios en la nube, sitios web y físicos
- **Operadores:** Son los encargados de realizar el o los ataques de ingeniería social, en los que se encuentran humanos, en caso de que el ataque se realice de forma

directa y personal a la víctima. Y el software, que se refiere a las herramientas que permiten hacer ataques de ingeniería social de forma automática y a múltiples víctimas a la vez.

- **Vectores de ataques:** Son los medios mediante el cual un atacante puede explotar alguna vulnerabilidad o amenaza, para otros autores estos vectores de ataque mencionados son las técnicas para realizar ingeniería social. Son gran cantidad de técnicas, mediante las cuales se puede realizar ingeniería social. Los autores proponen los siguientes: Phishing, Dumpster diving, Shoulder surfing, Amenazas Persistentes Avanzadas (APT), Ingeniería social inversa, Cebo o carnada y Waterholing (véase numeral 1.2.3).

La ingeniería social se divide en distintos tipos dependiendo de diversos factores, como la forma de acercamiento a la víctima. Son cuatro tipos los que se describen en la literatura (Krombholz et al., 2015) :

- **Acercamientos físicos:** El atacante intenta establecer contacto físico con el objetivo. Este acercamiento puede ser con la intención de obtener información privada y personal del individuo. Otro ejemplo de este acercamiento es el *Dumpster Diving*, que consiste en hurgar en la basura del objetivo buscando información valiosa o confidencial de este.
- **Acercamientos sociales**: Es uno de los tipos más exitosos de ingeniería social. En esta práctica, el atacante se vale de aspectos psicológicos de la persona, así como de principios de persuasión (véase numeral 1.2.2). Este método es conocido, ya que el víctimario intenta crear una relación con el usuario, con el fin de explotarla más adelante.
- **Ingeniería social inversa**: En este tipo de ataque se intenta hacer creer a la víctima que el atacante hace parte o es del área de soporte de alguna entidad reconocida, con el objetivo de que el usuario haga el acercamiento al atacante (por ejemplo, debido a un nuevo puesto laboral, buscando ayuda, algún servicio, entre otros). Durante este contacto, el víctimario intentará acceder a cualquier tipo de información personal o datos confidenciales de la víctima.
- **Acercamientos técnicos:** Usualmente se realizan por internet. El atacante hace una búsqueda de información personal de los usuarios vía internet, ya que las

personas suelen divulgar con gran facilidad sus datos personales en plataformas de libre acceso, como lo son las redes sociales. Además, existen herramientas de código libre, que permiten minar información de individuos u organizaciones automáticamente, como *Maltego.*

- **Acercamientos socio técnicos**: Este tipo de ataque suele ser bastante exitoso, ya que usa todos los tipos de acercamiento descritos. Un ejemplo es, un dispositivo de almacenamiento infectado dejado para ser encontrado por la posible víctima. El víctimario se aprovecha de la curiosidad de los individuos para infectar con malware los dispositivos a los que se conecte el dispositivo.

Por otro lado, distintos autores proponen varias fases por las que pasan los ataques de ingeniería social.

1.2.1 Fases de los ataques de ingeniería social

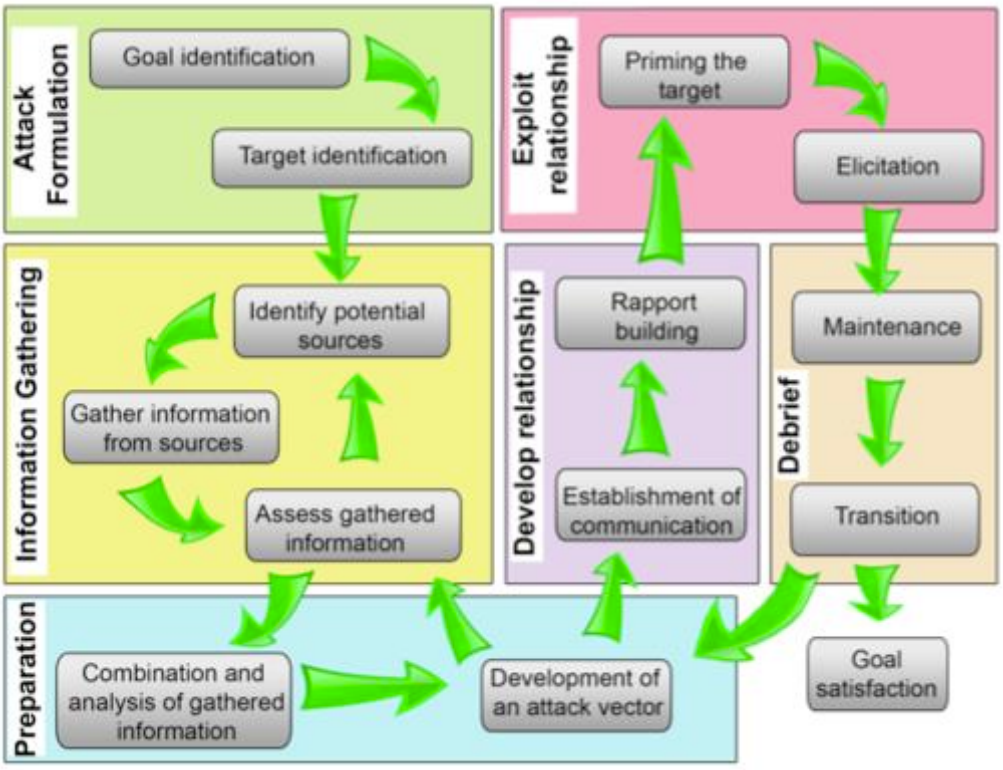
Los autores Bhagyavati (2007), Ivaturi y Janczewski (2011) proponen cuatro grandes fases de un ataque de ingeniería social:

- **(1) Fase de preparación:** En esta fase se define a cual organización o indivíduo se realizará el ataque y se recolecta información de este. La recolección de información de la víctima puede ser mediante internet, *dumpster diving*, u otros métodos. Si es una organización, se puede recolectar información yendo a la locación física de sus oficinas, y analizando características o costumbres de los empleados, como su cargo, horario laboral, relaciones, u otros datos valiosos para explotarlos en las siguientes fases.
- **(2) Fase pre-ataque:** En esta fase se define el ataque que se va a realizar basado en la información obtenida y se realiza la estrategia del ataque.
- **(3) Fase de ataque:** Se perpetra el ataque planeado previamente a la o las víctimas establecidas. El plan se debe revisar antes de efectuar el ataque, debido a la situación en la que se realice. Se debe tener múltiples estrategias dependiendo del ataque que se quiera perpetuar.
- **(4) Fase pos-ataque:** El ingeniero social trata de controlar la situación después de realizar el ataque, borrando evidencias del ataque e integrando la información

obtenida. Después de esto, el atacante puede efectuar otro tipo de ciberataque para afectar la organización u individuo atacado.

Mouton y compañía (2014) proponen un modelo más detallado, describiendo seis fases (figura 1-3) del ciclo en las cuales se realiza un ataque de ingeniería social. Estas fases son una extensión de las propuestas por Mitnick (2003).

Figura 1-3: Fases de ataques de ingeniería social



Fuente: (Mouton et al., 2014)

- **(1) Formulación del ataque:** Esta es la primera fase de un ataque de ingeniería social según Mouton y los demás coautores. Se debe definir que es lo que se quiere lograr con este ataque y a quién se atacará, si a un individuo o una organización.
- **(2) Recolección de información:** Esta fase es muy importante, ya que la relación que se cree con el objetivo depende la información que se recolecte de este. Se

debe recolectar toda la información posible de la víctima y que sea útil para el ataque. Esta información puede proceder de fuentes públicas, como sitios de internet, redes sociales, foros, blogs personales, etcétera. La información privada puede conseguirse mediante técnicas como el *dumpster diving.* Después de recolectar la información, esta debe ser revisada para saber si es relevante o no. Si no hay suficiente información de la víctima se debe reiniciar este paso hasta que haya suficiente información.

- **(3) Preparación:** En esta fase se debe preparar el plan antes de ejecutar el ataque. Se debe estudiar el escenario en que se realizará el ataque y desarrollar un vector de ataque que contenga una meta, un objetivo y un atacante. Además, el vector de ataque debe tener en cuenta el medio, los principios psicológicos y técnicas para realizar el ataque.
- **(4) Desarrollo de una relación:** Este paso es bastante relevante, debido a que entre mejor sea la relación con la víctima, más probabilidades hay de que el ataque sea exitoso. Este paso se divide en dos partes, la primera es buscar un medio para comunicarse con la víctima, y el segundo es la creación de la relación. Este paso puede ser el más largo del proceso y pueden usarse varias técnicas de ingeniería social para ejecutarlo.
- **(5) Explotar la relación:** Para ejecutar este paso con éxito, primero se debe preparar a la víctima, haciéndola llegar al estado emocional deseado. Por ejemplo, contando una historia triste para que su estado emocional cambie. Una vez la víctima esté en el estado emocional deseado, se extrae la información deseada de la víctima. La información puede ser una contraseña o cualquier información que sea valiosa para el atacante.
- **(6) Interrogación o informar:** Después de explotar la relación es importante regresar a la víctima al estado emocional inicial, previo al ataque. Finalmente, se define si la información obtenida es suficiente. Si no lo es, se regresa al paso de recolección de información.

Las fases de los ataques de ingeniería social hacen que este tipo de ataque se pueda hacer de forma sistemática y repetible en diferentes escenarios. Como se ha expuesto previamente, la base de la ingeniería social es la manipulación del comportamiento de las

personas, aprovechándose de los aspectos psicológicos de las personas mediante los principios de persuasión.

1.2.2 Principios de persuasión y aspectos psicológicos

La ingeniería social explota aspectos psicológicos de los humanos, por esto, es necesario conocer los aspectos psicológicos y principios de persuasión que involucran estos ataques. Gragg (2003) propone siete aspectos psicológicos que son explotados mediante la ingeniería social:

- **Emociones fuertes:** Las emociones fuertes incluyen emociones como sorpresa, ira, anticipación, miedo, excitación y pánico. Estas emociones son generadas por el atacante en su víctima y hacen que esta piense de forma menos clara o razonable. También sirven de distracción para el atacante, puesto que hacen que la víctima pierda su habilidad de evaluación de la situación.
- **Sobrecarga:** Cuando una persona tiene que lidiar con mucha información, se produce una sobrecarga, lo cual produce que la persona sólo absorba la información en vez de evaluarla. La sobrecarga también hace que la persona tenga mucha información que procesar en poco tiempo y reduzca su habilidad toma de decisión.
- **Reciprocidad:** Este principio consiste en dar a las personas algo a cambio del mismo valor. Esto generalmente ocurre cuando lo que se ha recibido es de mayor valor de lo que se ha dado. Este principio es usado en la *ingeniería social inversa* (véase numeral 1.2.3), cuando el atacante ayuda al víctimario y espera recibir una compensación a cambio.
- **Relaciones falsas:** Consiste en crear una relación con un individuo, con el propósito de explotar esta relación. Para construir de forma rápida esta relación, se puede hacer creer a la víctima que es muy parecida al atacante, tocando temas de interés de la víctima durante una conversación. También se puede establecer una relación usando el principio de reciprocidad. Una vez se crea la relación, es posible explotarla usando cualquier vector de ataque de ingeniería social.

- **Difusión de responsabilidad y deber moral:** Este ocurre cuando se hace sentir que la víctima no es la única responsable de sus actos. Por ejemplo, en un ambiente laboral, el atacante hace sentir a la víctima que una decisión suya puede hacer que el atacante pierda su empleo o tenga consecuencias laborales graves. De esta manera, el atacante hace sentir culpable a la víctima y se aprovecha de la situación.
- **Autoridad:** Las personas tienden a obedecer la autoridad. Por ejemplo, un atacante puede hacerse pasar por el director de una compañía y enviar emails o llamar a algún empleado, solicitando información confidencial. Si el empleado no está preparado podría caer ante el engaño del atacante.
- **Integridad y consistencia:** Los individuos usualmente asumen los compromisos o responsabilidades adquiridas en el trabajo, aunque no parezcan ser muy coherentes, esto hace que un trabajador efectué cualquier tarea que su jefe le indique. Además, las personas también tienden a confiar en la información que otras personas les dan, aunque esta información sea falsa.

Estos aspectos psicológicos son aprovechados por lo ingenieros sociales para realizar ataques a sus víctimas. Estos son usados a su vez junto a los principios de persuasión. Stajano y Wilson (2011) proponen siete principios por los cuales las personas son estafadas, a su vez teniendo en cuenta los principios de influencia de Cialdini (1993):

- **Principio de distracción:** Las distracciones son fundamentales en muchos escenarios de fraude, así como en los actos de magia. Las distracciones consisten en mantener la atención de la persona en lo que los estafadores quieren, mientras se aprovechan de la situación. Un ejemplo de este principio puede verse en el ataque de *Cartas Nigerianas* donde las personas son distraídas del fraude prometiéndoles altas sumas de dinero.
- **Principio de cumplimiento social:** El cumplimiento social se refiere a la autoridad, las personas tienden a no cuestionar la autoridad. Este principio puede ser explotado cuando el atacante personifica alguna autoridad, por ejemplo, cuando un atacante se hace pasar por una entidad bancaria y envía email que contiene *phishing* a la víctima. La víctima no cuestiona el email ya que proviene de su banco de confianza y cae en este ataque.
- **Principio de manada:** Este principio se basa en que las personas tienden a bajar la guardia cuando todos alrededor parecen tener el mismo riesgo. En el comercio

online, como la plataforma *E-bay,* muchos perfiles de vendedores tienen buena reputación gracias a cómplices. Esto hace creer a las personas que son vendedores confiables y al comprar algún artículo a estos vendedores, resultan estafadas. Otro ejemplo se presenta en la política, cuando se crean identidades falsas para apoyar a cierto candidato en internet. A esta situación se le conoce como “astrosurfing”.

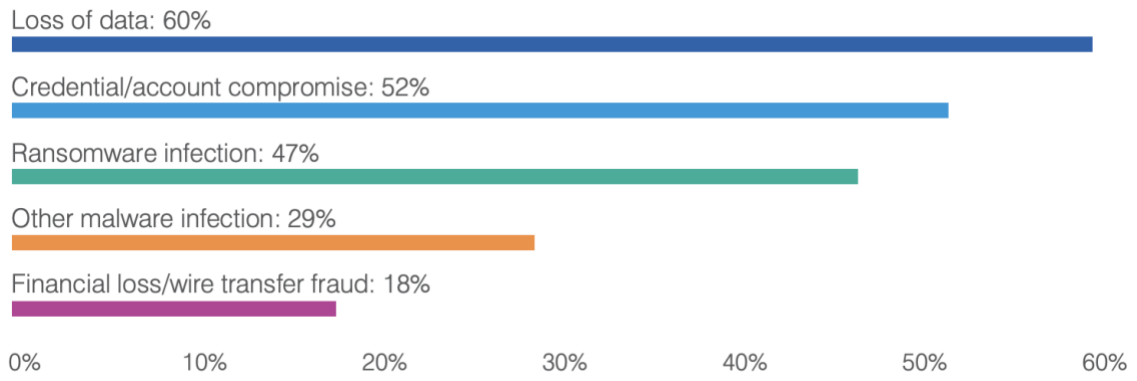
- **Principio de deshonestidad:** Este principio consiste en aprovechar algún acto ilícito de la víctima para extorsionarle. Este principio se ve reflejado, por ejemplo, en un ambiente laboral, cuando un trabajador infecta de *malware* algún equipo de la organización mientras realizaba algún acto ilícito u obsceno en el ambiente de trabajo. Cuando la organización indague sobre la procedencia de este virus, el trabajador no cooperaría con el área de seguridad, debido al estigma social que conllevaría si dice la verdad.
- **Principio de amabilidad:** Las personas tienden a ser buenas y ayudar al prójimo. Los estafadores se aprovechan de este principio en muchas situaciones, en muchos casos, los atacantes crean situaciones donde generan lástima a la víctima para estafarla. Por ejemplo, un atacante puede enviar emails falsos a sus víctimas, creando situaciones en donde ocurre un desastre natural, con el objetivo de pedir dinero para las víctimas de este desastre.
- **Principio de codicia:** Los estafadores pueden aprovecharse de las personas si conocen sus necesidades y deseos. Este principio es explotado por los estafadores, aprovechando las necesidades de la gente: económicas, laborales, sexuales, etcétera. La mejor defensa contra este principio es pensar que, si algo parece muy bueno para ser real, seguramente no lo es.
- **Principio de tiempo:** Cuando las personas están bajo presión cambian la forma de tomar decisiones, esto puede ser aprovechado por un atacante. Cuando no se tiene suficiente tiempo para tomar una decisión las personas tienden a tomar la decisión que mejor satisfaga la situación actual, aún cuando esta decisión pueda no ser la más óptima.

1.2.3 Técnicas de ataques de ingeniería social

En el contexto de la ciberseguridad, los vectores de ataque son el medio por el cual un atacante puede obtener acceso de forma no autorizada a un sistema, servicio o red. Los vectores de ataque son desarrollados durante la fase de preparación de un ataque de ingeniería social. Estos vectores de ataque deben tener en cuenta qué técnica de ingeniería social va a ser usada para ejecutar el ataque. Las técnicas son los métodos mediante los cuales un atacante puede engañar a su víctima o extraer información de esta. Las técnicas más usadas en ingeniería social son:

- **Phishing:** El *phishing* es un tipo de ciberataque, distribuido mediante un canal de comunicación electrónico, donde el atacante busca que la víctima actúe de una manera beneficiosa para él (Khonji et al., 2013). Estos pueden ser transmitidos mediante los servicios como el Email, HTTP, SMS, entre otros. Existen múltiples motivos por los cuales se realizan ataques de este tipo, los más destacados son: beneficios financieros, ocultación de identidad y la fama (Khonji et al., 2013). Según el reporte anual de Footprint (2021) sobre ataques de *phishing* del 2020, el 74% de organizaciones de los Estados Unidos sufrió un ataque exitoso de *phishing*, lo que corresponde un 14% de aumento respecto al año anterior. Los principales efectos de los ataques tipo phishing estudiados en el reporte fueron: perdida de datos, datos personales o credenciales comprometidas, infección por *malware*, infección por *ransomware*, fraude bancario o perdida de dinero. La figura 1-4 muestra los porcentajes de los efectos mencionados en organizaciones que sufrieron este tipo de ataques.

Figura 1-4: Efectos de ataques de phishing en organizaciones



Fuente: (Footprint, 2021)

- **Spear Phishing:** El *Spear Phishing* es una versión mejorada o sofisticada del *Phishing*, ya que su finalidad es la misma, pero este ataque se enfoca en víctimas especificas. Al ser estos ataques dirigidos a individuos u organizaciones específicas, son más difíciles de detectar que el phishing común, dado que usan técnicas más avanzadas para captar la atención de las víctimas.

- **Whaling o Whaterholing:** El *whaling* es el mismo ataque de *Spear Phishing*, pero se enfoca a individuos con posiciones sociales altas como ejecutivos de compañías, políticos, celebridades, entre otros.

- **Smishing:** El termino se deriva de la combinación de *SMS* (*Short Message Service*) y *Phishing*. El método consiste en adquirir información de la víctima de la misma forma que el phishing, pero mediante el uso de mensajes de texto (*SMS*). Este método es cada vez más usado por ingenieros sociales, gracias al incremento en el uso de teléfonos celulares. Además, puede ser combinado con otros vectores de ataque de ingeniería social como el *Catfishing*.

- **Vishing:** El *Vishing* es un método muy similar al *Phishing*, sin embargo, en este ataque se usa la voz como medio de comunicación. Este ataque puede ser ejecutado mediante llamadas telefónicas o VoIP (Ivaturi y Janczewski, 2011). Un

ejemplo de este ataque es cuando una máquina hace una llamada a la víctima haciéndose pasar por una empresa reconocida y pregunta al usuario datos personales o valiosos y así el víctimario extrae información sensible de la víctima.

- **Dumpster diving:** Una traduccíon de *Dumpster Diving* sería "bucear en contenedores de basura". Este método consiste en la búsqueda de información valiosa en cualquier tipo de contenedor de basura (Long, 2011). Este método puede ser muy eficaz para la obtención de información de la víctima, puesto que esta puede desechar mucha información que ya no considera valiosa. Por ejemplo, de una factura de compra de una tienda se podrían extraer datos de la víctima como su nombre, correo electrónico, número de identificación, artículos que compra o en los que está interesado, etcétera. El escenario más riesgoso de este método es el empresarial, muchas empresas desechan documentos importantes sobre clientes, estados de cuentas o empleados, sin una política estricta de seguridad.

- **Shoulder surfing:** Este método consiste en simplemente mirar (sobre el hombro) qué esta haciendo la víctima (Long, 2011). El método es muy simple pero bastante efectivo ya que el atacante podría ver algún dato importante de la víctima como una contraseña, número de tarjeta de crédito, etcétera. Este método puede ser aplicado en situaciones donde haya algún tipo de información sensible a la vista, por ejemplo, en un cajero automático. Con un simple vistazo el atacante puede memorizar la contraseña de la cuenta de la víctima y con algún otro método de ingeniería social o robo, podría hurtarle su tarjeta del banco y así robar el dinero de su cuenta. Otro ejemplo de este método es cuando el atacante ojea algún documento o nota de su víctima que esté a la vista y contenga información confidencial.

- **Ingeniería social inversa:** Este tipo de ataque también es conocido como *Quid pro quo* que se refiere a dar una cosa por otra. En este ataque, el víctimario crea una situación donde la víctima requiera su ayuda, el víctimario ayuda a la víctima y aprovecha la situación para extraer información privilegiada de la víctima (Krombholz et al., 2015). Una situación donde se puede ver cómo funciona este ataque es, por ejemplo, un eventual daño de la red de una organización, durante el

cual el atacante se hace pasar por el técnico de la red. Posteriormente el atacante ayudaría de forma maliciosa a los usuarios de la organización mientras les extrae información.

- **Cebo o carnada (Baiting):** El *Baiting* consiste en dejar un dispositivo de almacenamiento como una memoria USB infectado con *Malware* en una locación concurrida, donde una posible víctima sea capaz de encontrarlo. Cuando la víctima encuentre y posteriormente inserte este dispositivo a su computador, este se infectará con algún tipo de *Malware*.

- **Catfishing:** El *Catfishing* consiste en personificar a otra persona en redes sociales o medios digitales para engañar y aprovecharse de ellas (Simmons y Lee, 2020). Este método podría considerarse un tipo de *Phishing*, ya que busca que la víctima actué de forma en que beneficie al víctimario. Este método es muy efectivo, es difícil de detectar y generalmente es usado con otros ataques de ingeniería social. Según Malwarefox (2021), la red social Facebook en el año 2021 tenía aproximadamente 2.100 millones de cuentas en su plataforma, de las cuales se estima que aproximadamente el 10% son falsas.

- **Tailgating:** El *Tailgating* es el método mediante el cual el atacante entra a una zona restringida, simplemente siguiendo a alguna persona que tenga acceso a esta (Ivaturi y Janczewski, 2011). Un ejemplo de este método es cuando un atacante se hace pasar por empleado de alguna empresa u organización y espera que un empleado real pase por la puerta asegurada, el empleado amablemente le sostiene la puerta abierta para que el atacante pase sin causar sospechas.

- **Estafas y cartas nigerianas:** Las estafas, generalmente, buscan perjudicar a una víctima de forma económica. Por ejemplo, ofreciendo productos que ofrecen beneficios irreales, o premios falsos a las víctimas. Las cartas nigerianas suelen ser un ataque de *Spam* que busca mover grandes sumas de dinero a nivel internacional. Este ataque es realizado principalmente mediante el email y consiste en que el atacante crea una situación donde supuestamente consigue dinero de forma legal como una herencia, y cuenta a la víctima que no puede usar este dinero

por alguna excusa, que suele ser política. Entonces, el atacante pide a la víctima alguna parte del valor de ese dinero adquirido a cambio de todo el supuesto dinero del atacante. Este es un tipo de estafa muy común y se ha usado durante años para engañar a las personas.

1.3 Métodos de detección de ingeniería social

Debido a la importancia de la detección de ataques de ingeniería social, existen múltiples investigaciones que abordan este problema. Las principales investigaciones relacionadas a la detección y prevención de ingeniería social se enfocan en desarrollar métodos o sistemas que se pueden categorizar en dos grupos: los métodos no automáticos, que ayudan a la prevención de los ataques, presentan las fases de los ataques y desarrollan sistemas o modelos para su detección de forma manual, y los métodos automáticos, que usan herramientas tecnológicas y no necesitan la interacción del usuario para detectar este tipo de ciberataques.

1.3.1 Métodos no automáticos

Los métodos de detección de ingeniería social denominados no automáticos abarcan sistemas y métodos como la enseñanza de ciberseguridad, ingeniería social, y temas relacionados, hasta modelos ontológicos que ayudan a la detección de estos ataques. El común denominador de estos métodos es la necesidad de interacción del usuario para poder detectar los ataques de ingeniería social.

Denning y compañía (2012) diseñan y construyen un juego de mesa para ayudar en la enseñanza de temas relacionados a la ciberseguridad, enfocado a estudiantes de escuelas en niveles de primaria y secundaria. Deciden crear un juego porque gracias a estos es más interactiva la enseñanza, se permite explorar ideas y además de ser entretenidos, ayudan a retener la atención de las personas. Durante la investigación, el juego fue enviado por los investigadores a 150 educadores. Después, recibieron retroalimentación de 22 de ellos sobre su experiencia usando el juego con sus estudiantes. Las respuestas de 14 educadores fueron: 11 de 14 educadores reportaron que sus estudiantes se interesaron

sobre el tema de ciberseguridad, 10 de 14 dijeron que usarían el juego de nuevo en sus clases y 13 de 14 recomendarían el juego a más personas.

Pero no todos los métodos de enseñanza son efectivos contra los ataques de ingeniería social, Junger, et al. (2017) realizan un experimento que consistió en dos intervenciones para ayudar a las personas a protegerse contra ataques de ingeniería social mediante señales y avisos sobre la protección de datos personales. El experimento fue realizado en un centro comercial de una ciudad en los Países Bajos. Las personas llenaron formularios donde se les preguntaba su email, número de cuenta de banco, y detalles sobre sus compras online. La tasa de divulgación de datos personales fue relativamente alta, según los siguientes resultados: 79.1% de los individuos dieron su email, 43.5% dieron datos de su banco, y 89.8% de los sujetos dieron datos de los tipos de productos que compran online. Los investigadores reportan indicios de un efecto adverso en las advertencias mostradas a los individuos. Una de las conclusiones de la investigación es que las personas tienden a confiar en las demás personas, lo que las hace vulnerables a ataques de ingeniería social.

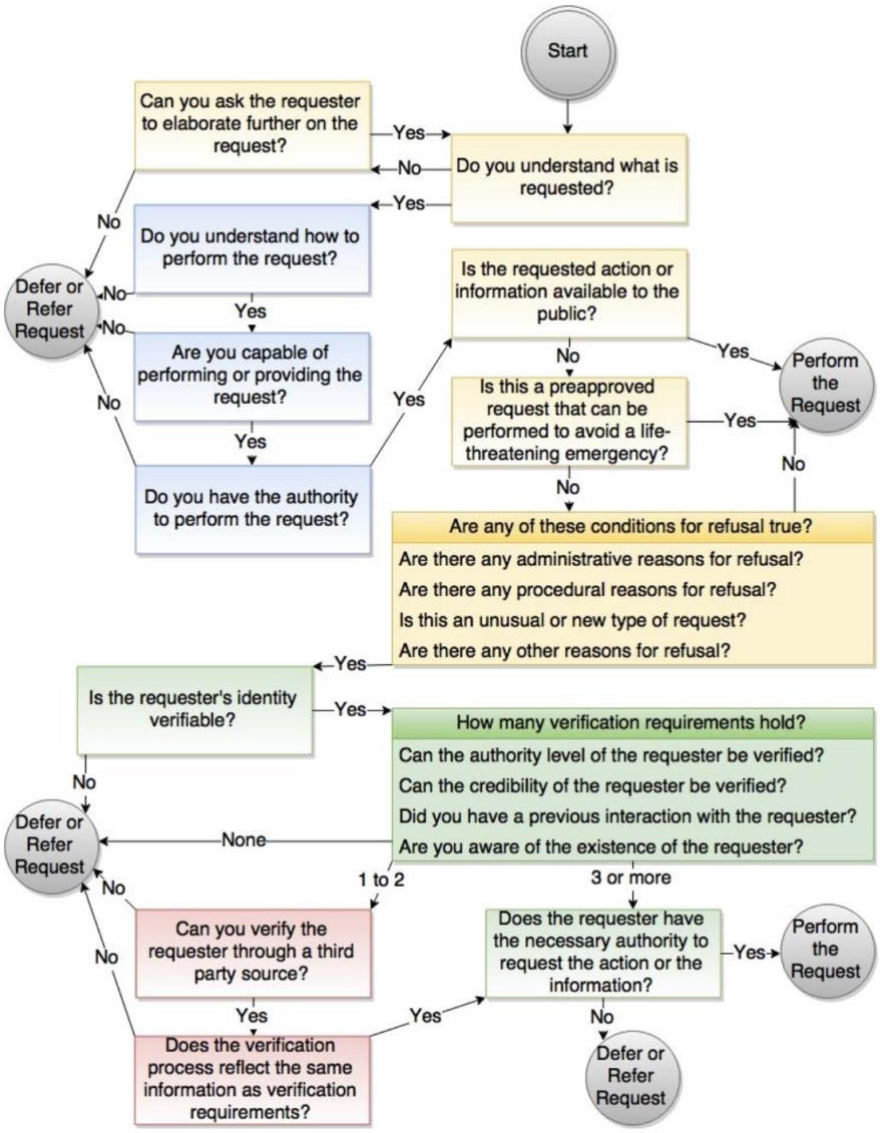
Sólo la enseñanza ha predominado en la prevención de ataques de ingeniería social. Sin embargo, esta puede ser olvidada rápidamente, volviéndola ineficaz contra estos ataques (Bezuidenhout, et al. 2016). Por esta razón se han desarrollado múltiples modelos que ayudan a los individuos a detectar posibles ataques de ingeniería social.

Bezuidenhout y su grupo investigativo (2016) proponen un modelo de detección de ingeniería social llamado “SEADM”, basado en un ambiente de *call centers*. Este modelo se enfoca en dos perspectivas principales: la perspectiva de la psicología y la perspectiva de las ciencias de la computación. En la perspectiva de la psicología mencionan emociones que hacen vulnerable a un individuo ante la ingeniería social como el afecto, la sobrecarga, la reciprocidad, entre otros. Afirman que este es un modelo rápido y efectivo para determinar si una persona en una llamada telefónica está tratando de manipular al individuo receptor de la llamada para que divulgue información privilegiada. El modelo es basado en árboles de decisión con múltiples nodos donde cada nodo es una pregunta al individuo que recibe la llamada. El individuo debe determinar la respuesta de esa pregunta y avanzar al siguiente nodo hasta llegar a una decisión. Las posibles decisiones finales

son: elevar la solicitud a un supervisor, y proveer acceso a la solicitud. El modelo es evaluado en escenarios simulados de posibles ataques de ingeniería social.

En la investigación desarrollada por Mouton et al. (2016) se hace una revisión al modelo SEADM, enfocándolo a los tres tipos de comunicación: bidireccional, unidireccional e indirecta. Este modelo es llamado “SEADMv2”. El modelo es basado en árboles de decisión y hace el proceso de toma de decisión más amigable al usuario. El modelo tiene cuatro tipos de estados: estados de la solicitud, estados del solicitante, estados del receptor, y estados de externos. Los estados de la solicitud se indican en color amarillo en la figura 1-5 y tratan la información de la solicitud. Los estados del receptor se indican en azul en la figura 1-5, estos manejan las acciones del individuo que recibe la solicitud y si este puede realizarla. Los estados del solicitante, indicados en verde en la figura 1-5, manejan al solicitante y si la información de este puede ser verificada. Los estados de externos, indicados en rojo en la figura 1-5, involucran una entidad externa al modelo y verifican si el solicitante puede ser verificado con un medio externo. El modelo se evalúa en tres escenarios propuestos por los investigadores, explicando a detalle cada una de las decisiones que debe tomar el individuo al tratar de identificar si la conversación es un ataque de ingeniería social o no. El modelo no es evaluado en conversaciones reales, sólo en escenarios simulados.

Figura 1-5: Modelo de detección de ingeniería social SEADMv2



Fuente: (Mouton et al., 2016)

El modelo SEADMv2 es implementado por Mouton y su grupo (2017) en una aplicación móvil para el sistema operativo *Android,* y una aplicación web. La aplicación fue creada usando la metodología desarrollo rápido de aplicaciones (RAD). El sistema fue diseñado con una arquitectura de tres capas: capa de interfaz, capada de servicio y capa de back-end. La capa de interfaz consiste en la aplicación móvil Android y la aplicación web. La capa de servicio tiene la lógica de negocio del sistema y es donde implementa el modelo

SEADMv2. Esta es la capa más importante del sistema, ya que en ella recae toda la implementación del modelo. La capa de back-end se encarga de almacenar los datos analíticos de los usuarios de la aplicación web y móvil. Se hizo un experimento para probar la eficiencia de la aplicación para detectar posibles ataques de ingeniería social. El experimento consistió en evaluar 20 individuos en diez escenarios diferentes, ocho de estos escenarios eran maliciosos, involucrando ataques de ingería social, y los dos restantes eran escenarios inofensivos. Primero, evaluaron las respuestas de los individuos en los escenarios sin usar la aplicación, luego evaluaron las respuestas de los mismo usando la aplicación móvil creada. Los resultados del experimento muestran que la aplicación móvil implementando el modelo SEADMv2 reduce el número de errores de los individuos a la hora de detectar ataques de ingeniería social, respecto al no uso de modelo.

1.3.2 Métodos automáticos

Los modelos de detección de ingeniería social dependen de la interacción del usuario para evaluar si hay riesgo de un posible ataque de ingeniería social y recaen en la metacognición para definir su propio estado emocional al usar estos modelos (Sawa, et al., 2016). Debido a esto, se han desarrollado sistemas y métodos, que no requieren de la interacción del usuario para detectar ataques de ingeniería social. La mayoría de estos sistemas utilizan algoritmos de aprendizaje de máquina, procesamiento de leguaje natural o combinan ambos.

Sawa y compañía (2016) proponen un sistema de detección de ingeniería social, el cual usa técnicas de procesamiento de lenguaje natural para detectar preguntas y comandos para extraer los temas de una conversación. Estos temas son comparados con una lista negra y se determina si la pregunta o el comando son maliciosos. Este sistema usa únicamente el texto de la conversación. El sistema analiza cada oración en un árbol sintáctico, después detecta si hay alguna pregunta o algún comando. Los temas potenciales del texto se identifican encontrando pares de tipo verbo/sustantivo que relacionan los verbos con su objeto directo. Finalmente, cada tema es comparado con una lista negra de temas definida manualmente. Un ejemplo del proceso sería así:

- **Texto de entrada**: Reiniciar el router
- **Acción**: reiniciar
- **Recurso**: router

Los investigadores usaron el software *Stanford Parser* para analizar cada oración. La lista negra usada contiene pares de temas genéricos como (“enviar”, “dinero”) o (“dar”, “identificación”). El sistema fue evaluado usando dos corpus, el primero son diálogos transcritos de ataques de ingeniería social; el segundo son diálogos de la Suprema Corte de Estados Unidos, los cuales no contienen ataques de ingeniería social. Los resultados del sistema al evaluarlo con el primer corpus que contiene 74 oraciones arrojan una precisión del 100% y 60% de exhaustividad, pero no reportan datos de exactitud. Para el segundo corpus, que contiene 10.000 oraciones, arroja una precisión del 100%, y no se reportan más métricas. El sistema resulta ser completamente preciso a la hora de detectar ataques de ingeniería social, pero al no reportar más métricas, es difícil saber que tan fiable es el sistema.

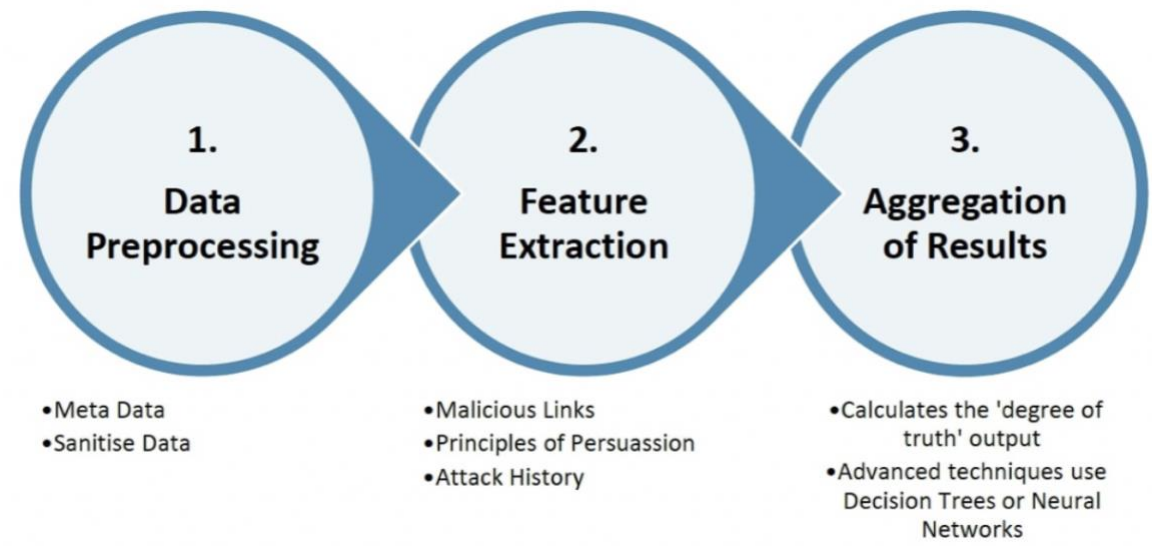
Lansley et al. (2019) desarrollaron un sistema de detección de ingeniería social basado en el ambiente de chats en línea. Primero, se preprocesa el texto de los diálogos y se extraen características para entrenar algoritmos de clasificación. El pipeline del sistema es creado en el lenguaje de programación *Python* y usan las librerías *SymSpelly* para revisar la ortografía de las palabras del texto; y *Scikit-Learn* para implementar los algoritmos de clasificación. Las características extraídas de los datos fueron: puntaje de links, puntaje de ortografía y puntaje de intención. Donde:

- **Puntaje de enlaces:** Se usa una interfaz de programación de aplicaciones (API) para determinar si alguna URL encontrada en el texto es *phishing* o no. La API retorna un valor de reputación y este es normalizado.
- **Puntaje de ortografía**: Usa la librería *SymSpelly* para determinar si las palabras del texto están con ortografía correcta. Se hace el conteo de estos errores de ortografía por cada oración y se normaliza el valor.
- **Puntaje de intención**: Se define manualmente una lista negra de palabras y se busca ocurrencias de estas en el texto ingresado.

Los algoritmos de clasificación usados fueron: árbol de decisión *(Decision Tree),* bosques aleatorios *(Random Forest)* y redes neuronales convolucionales *(CNN)*. El sistema es evaluado con dos datasets, uno real con 147 textos, uno sintético con 1200 textos. Los resultados de exactitud para el dataset real por cada algoritmo de clasificación fueron: árbol de decisión: 0.681, bosques aleatorios: 0.683, y CNN: 0.687. Por otra parte, los valores de exactitud para el dataset sintético fueron: árbol de decisión: 0.918, bosques aleatorios: 0.894, y CNN: 0.917.

Lansley, Polatidis y Stelios (2019) proponen un método para detectar ingeniería social de forma automática basado en los principios de persuasión, usando procesamiento de lenguaje natural y *redes neuronales,* el cual llaman "SEADer". Este método consiste en tres fases: (1) preprocesamiento de datos, (2) extracción de características y (3) agregación de resultados, como se ve en la figura 1-6.

Figura 1-6: Fases sistema de detección de ingeniería social SEADer



Fuente: (Lansley, Polatidis y Stelios, 2019)

En la primera fase es usada la librería *CoreNLP* de Standford para procesar y limpiar el texto de los diálogos. En la segunda fase se extraen características de los textos basándose en los principios de persuasión y ataques de *phishing.* Las características extraídas son tres: puntaje de intención, puntaje de ortografía y puntaje de enlaces. El método usa los algoritmos de clasificación de árbol de decisión, bosque aleatorio y

perceptrón multicapa (*Multi-Layer Perceptron*). El método fue evaluado usando dos datasets, uno real y uno sintético. Los resultados de exactitud para el dataset real fueron: árbol de decisión: 0.681, bosques aleatorios: 0.683, y perceptrón multicapa: 0.691. Por otra parte, los valores de exactitud arrojados para el dataset sintético fueron: árbol de decisión: 0.918, bosques aleatorios: 0.917, y perceptrón multicapa: 0.925.

Lansley, Mouton, Kapetanakis y Polatidis (2020) proponen un nuevo método para la detección de ingeniería social basado en el SEADer (Lansley, 2019). Este método usa las mismas fases del SEADer, y añade un paso final que consiste en la implementación de un método de ensamble basado en votación suave entre los algoritmos de clasificación de árbol de decisión, bosque aleatorio y *Gaussian Naive Bayes.* Este método de ensamble es evaluado usando el dataset sintético del SEADer junto al algoritmo de clasificación de Perceptron multicapa. El valor de exactitud para el algoritmo de perceptrón multicapa es de 0.922; el valor de exactitud para el método de ensamble es de 0.924; siendo este valor 0.025 mayor al resultado del perceptrón multicapa.

Por otro lado, existen diversas investigaciones de detección automática de *phishing,* el cual, como se ha descrito antes, es un método de ataque que usa la ingeniería social. Los autores Sahingoz, et al. (2019) proponen un sistema de detección de *phishing* basado en URLs. Ellos crean un dataset propio de varias fuentes con 73.575 URLs, donde 36.400 son URLs legítimas y 37.175 son URLs *phishing.* El sistema consiste en varias fases donde se analizan y extraen características de las URLs usando técnicas de procesamiento de lenguaje natural, vectores de palabras e híbridos, para finalmente entrenar siete algoritmos de clasificación. Los algoritmos usados fueron: *bayesiano ingenuo* (*Naive Bayes), árboles aleatorios, K-vecinos más cercanos (KNN), Adaboost, K-estrella (K-star), optimización mínima secuencial (SMO)* y *árboles de decisión.* Los resultados de la evaluación del método muestran distintos resultados donde sobresalen los algoritmos de clasificación de árboles de decisión junto a características de *NLP*, con una exactitud de 97.02%; el algoritmo de bosque aleatorio con características de *NLP, con* una exactitud de 97.86%; y el algoritmo bayesiano ingenuo con características hibridas, con una exactitud de 95.86%. Estos fueron los algoritmos con mejores resultados de exactitud de los siete usados en la investigación.

El *smishing,* como se mencionó anteriormente (véase numeral 1.2.3) es un tipo de *phishing* pero usa los mensajes de texto (SMS) para robar información de una víctima. Balim y Gunal (2019) desarrollan un modelo automático para la detección de este tipo de ataques. El modelo consiste en tres fases: (1) preprocesamiento, (2) extracción de características y (3) clasificación. El dataset usado para el modelo consiste en 5.574 mensajes de texto, los cuales contienen ataques de ingeniería social. En la fase de preprocesamiento utilizan técnicas de procesamiento de lenguaje natural, como tokenización, eliminación de signos de puntuación, conversión del texto a minúsculas y lematización. En la fase de extracción de características extraen las siguientes características del texto: frecuencia de palabras, extracción de enlaces, extracción de emails, extracción de números telefónicos, conteo de caracteres, extracción de símbolos de dinero, e idioma del texto. En la fase de clasificación se entrenan los modelos *máquinas de vector de soporte* (*SVM*), *regresión logística* (*LR*) y bosques aleatorios. Los resultados de precisión para cada algoritmo de clasificación fueron 0.880, 0.880 y 0.890 para *SVM, LR* y bosques aleatorios respectivamente. El valor de puntaje F1 para cada algoritmo fue 0.927, 0.927 y 0.924 para *SVM, LR y* bosques aleatorios respectivamente.

En la tabla 1-1 se muestran las características principales de los métodos, sistemas y modelos de detección automática de ingeniería social mencionados previamente.

Tabla 1-1: Características principales métodos de detección automáticos de ingeniería social

Título del artículo	Características principales	Conjunto de datos	Métricas de desempeño
Detection of Social Engineering Attacks Through Natural Language Processing of Conversations. (Sawa, et al., 2016)	- Detección de ingeniería social - Uso de procesamiento de lenguaje natural - Extracción de temas de conversaciones - Comparación de temas extraídos con lista negra definida previamente	- Un primer corpus con 74 oraciones que contienen ataques de ingeniería social - Un segundo corpus con 10.000 oraciones sin ataques de ingeniería social	- Para el primer corpus: precisión de 100% y exhaustividad de 60% - Para el segundo corpus: precisión de 100%
Seen the villains: Detecting Social Engineering	- Detección de ingeniería social	- Un primer dataset con datos reales	- Para el primer dataset una

Attacks using Case-based Reasoning and Deep Learning. **(**Lansley et al., 2019)	- Uso de procesamiento de lenguaje natural y aprendizaje de máquina - Uso de tres características: puntaje de enlaces, puntaje de ortografía y puntaje de intención - Uso de tres algoritmos de clasificación: árbol de decisión, bosques aleatorios y redes neuronales convolucionales	que contiene 147 entradas - Un segundo dataset con datos sintéticos que contiene 1.200 entradas	exactitud máxima de 68.7% - Para el segundo dataset una exactitud máxima de 91.7%
Seader: A social engineering attack detection method based on natural language processing and artificial neural networks. (Lansley, Polatidis y Stelios, 2019**)**	- Detección de ingeniería social - Uso de procesamiento de lenguaje natural y aprendizaje de máquina - Uso de tres características: puntaje de enlaces, puntaje de ortografía y puntaje de intención - Uso de tes algoritmos de clasificación: árbol de decisión, bosques aleatorios y perceptrón multicapa	- Un primer dataset con datos reales que contiene 147 entradas - Un segundo dataset con datos semisintéticos que contiene 747 entradas	- Para el primer dataset una exactitud máxima de 69.1% - Para el segundo dataset una exactitud máxima de 92.5%
SEADer++: social engineering attack detection in online environments using machine learning. **(**Lansley et al., 2020)	- Detección de ingeniería social - Uso de procesamiento de lenguaje natural y aprendizaje de máquina - Uso de tes algoritmos de clasificación y métodos de ensamble: árbol de decisión, bosques aleatorios, perceptrón multicapa y gaussian naive bayes.	- Un primer dataset con datos reales que contiene 147 entradas - Un segundo dataset con datos semisintéticos que contiene 747 entradas	- Para el primer dataset una exactitud máxima de 69.1% - Para el segundo dataset una exactitud máxima de 92.6% y un puntaje F1 máximo de 0.769
Machine learning based phishing detection from URLs. Expert Systems with Applications. (Sahingoz, et al., 2019)	- Detección de phishing - Uso de procesamiento de lenguaje natural y aprendizaje de máquina - Uso de los algoritmos de clasificación: bayesiano ingenuo (Naive Bayes), árboles aleatorios, K-	- Dataset que consta de 73.575 URLs	- Exactitud máxima de 97.98%, precisión de 97%, recall de 99% y puntaje F1 de 0.980

	vecinos más cercanos (KNN), Adaboost, K-estrella (K-star), optimización mínima secuencial (SMO) y árboles de decisión		
Automatic Detection of Smishing Attacks by Machine Learning Methods. (Balim y Gunal, 2019)	- Detección de smishing - Uso de procesamiento de lenguaje natural y aprendizaje de máquina - Las características usadas fueron: frecuencia de palabras, extracción de enlaces, extracción de emails, extracción de números telefónicos, conteo de caracteres, extracción de símbolos de dinero, e idioma del texto - Uso de los algoritmos de clasificación: máquinas de vector de soporte (SVM), regresión logística y bosques aleatorios	- Dataset con 5.574 mensajes de texto	- Resultado de exactitud de 88% aproximadamente, puntaje F1 alrededor del 92%, y exhaustividad alrededor de 98%

Fuente: (Sawa, et al., 2016), **(**Lansley et al., 2019), (Lansley, Polatidis y Stelios, 2019), **(**Lansley et al., 2020), (Sahingoz, et al., 2019), **(**Balim y Gunal, 2019).

2. Diseño del modelo

El objetivo de la investigación es implementar un modelo que detecte ataques de ingeniería social de forma automática, sin requerir la interacción de un usuario y usando únicamente textos de diálogos, valiéndose de algoritmos de clasificación de aprendizaje de máquina y técnicas de procesamiento de lenguaje natural.

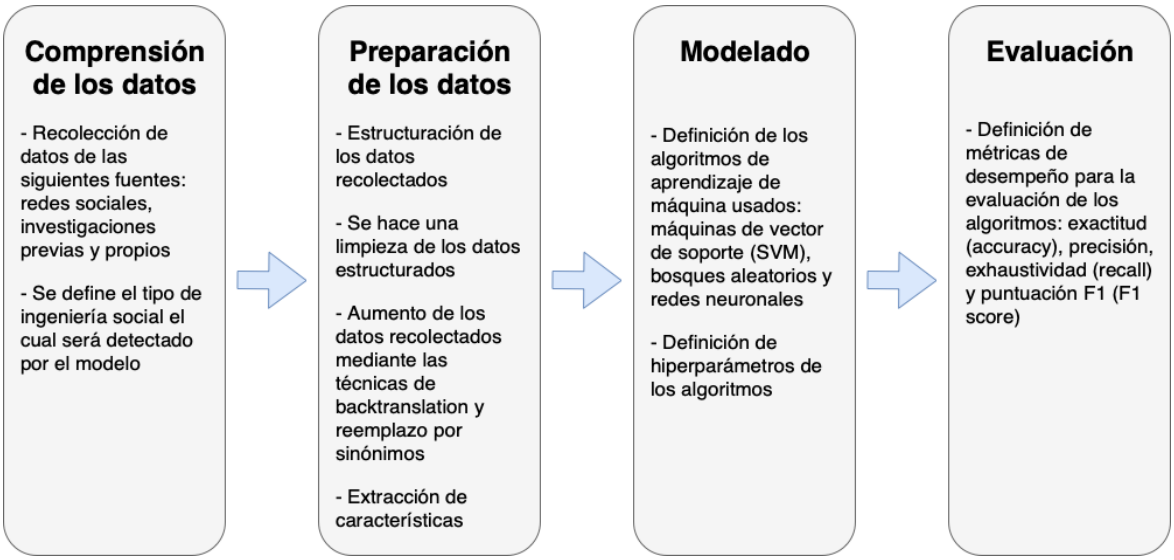
El enfoque de la investigación es cuantitativo (Hernández-Sampieri y Torres, 2018), ya que recolecta datos de ataques de ingeniería social para crear un modelo computacional usando aprendizaje de máquina y procesamiento de lenguaje natural, delimitando el estudio sólo a los ataques de ingeniería social, dando como resultado datos cuantitativos.

Como ya existen antecedentes de sistemas de detección de ingeniería social, el tipo de estudio de la investigación es descriptivo (Hernández-Sampieri y Torres, 2018). Y el tipo de diseño es no experimental (Gregar, 1994) ya que los datos recolectados de ataques de ingeniería social ya sucedieron, no se interviene en estos, ni en los individuos, ni en las situaciones en las cuales ocurren estos ataques.

Basado en la metodología *Cross Industry Standard Process for Data Mining* (CRISP-DM) (Wirth y Hipp, 2000) se desarrolló la investigación en cuatro fases: fase de comprensión de datos, fase de preparación de los datos, fase de modelado y fase de evaluación.

En la figura 2-1 se presenta un panorama general de las fases del modelo propuesto.

Figura 2-1: Fases modelo de detección de ingeniería social



Fuente: Elaboración propia

2.1 Comprensión de los datos

En esta fase se recolectaron y analizaron los datos necesarios para entender la naturaleza de estos y determinar su consistencia y calidad. Además, con base en los datos recolectados, se define qué tipos de ataques de ingeniería social podrán ser detectados por el modelo.

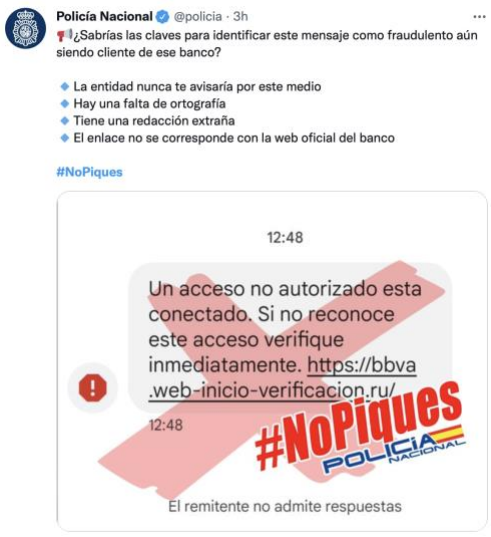
Para la investigación es importante poder implementar un modelo que fuera útil en el contexto en que esta fue desarrollada, por lo cual, los datos de ataques de ingeniería social recolectados fueron textos de diálogos únicamente en el idioma español. Como no existe una base de datos pública de estos ataques, fue necesario recolectar estos datos de distintas fuentes.

La primera fuente de datos corresponde a las redes sociales Facebook y Reddit. De estas redes sociales se extrajeron los datos de forma manual, buscando reportes de usuarios que sufrieron algún ataque de ingeniería social. Además, fueron añadidos textos de forma manual y extraídos de fuentes propias como emails y mensajes de texto. En total se recolectaron 43 textos.

La segunda fuente de datos fue la red social Twitter. Haciendo una exploración manual de esta red social, se encontró que muchos usuarios reportaban ataques de ingeniería social sufridos mediante imágenes o capturas de pantalla. También, múltiples organizaciones de ciberseguridad y comisarias de policías locales divulgaban ataques que les fueron reportados para advertir a las personas sobre estos ciberataques. Por lo cual, se extrajeron los datos de la red social Twitter siguiendo los siguientes pasos:

- **(1) Búsqueda de tweets:** Para extraer los datos de forma programática se utilizó la librería *Tweepy*, la cual es una librería del lenguaje de programación Python que facilita el uso de la API de Twitter (Tweepy, 2021). Los datos extraídos fueron tweets que contenían imágenes, en los cuales se reportaba un ataque de ingeniería social. Un ejemplo de estos datos se ilustra en la Figura 2-2.

Figura 2-2: Reporte ataque ingeniería social en Twitter



Fuente: Twitter Policia Nacional de España, Twitter, 2021, https://twitter.com/policia

La búsqueda realizada en Twitter se basó en los tweets relacionados a los hashtags: “#nopiques”, “#phishing”, “#smishing”, “#desconfia”, “#malware”, “#ciberdelincuentes”, “#ciberseguridad”, “#fraude”, “#scam” y “#ransomware”. Por cada hashtag o etiqueta se recolectaron 2000 tweets. En total se recolectaron 20.000 tweets.

- **(2) Extracción de imágenes de los tweets:** Cómo los tweets contenían imágenes de los reportes, sólo se recolectaron los tweets que tuvieran contenido multimedia. Por lo que se recolectaron en total 4790 imágenes de los 20.000 tweets recolectados. De las imágenes recolectadas, 3555 eran imágenes repetidas. Se eliminaron estas imágenes y finalmente se obtuvieron 1235 imágenes.
- **(3) Análisis y transcripción de imágenes:** Las imágenes obtenidas en el paso anterior fueron analizadas manualmente con el objetivo de transcribir únicamente las que tuvieran algún ataque de ingeniería social. Cómo muchas de estas imágenes borraban o tachaban información de los textos como enlaces de internet o URLs, números telefónicos y emails, se transcribieron como entidades, así:
  - **(Link) o (Enlace):** Cualquier URL ilegible encontrado en el texto
  - **(Teléfono):** Cualquier número telefónico ilegible en el texto
  - **(Email):** Cualquier email ilegible en el texto

Como ejemplo, se muestra como sería el texto transcrito para la figura 2-3: “CaixaBank: Estimado cliente, hemos detectado un intento de acceso sospechoso en su cuenta, por su seguridad debe verificar sus datos: (Link)”

Figura 2-3: Reporte ataque ingeniería social en Twitter para trascripción

Fuente: Twitter Policia Nacional de España, Twitter, 2021, https://twitter.com/policia

Además, los textos de las imágenes se transcribieron de forma exacta, sin ignorar ningún carácter ni ignorando errores de ortografía. Finalmente, los textos recolectados que contenían algún ataque de ingeniería social en la primera y segunda fuente de datos fueron 89.

La segunda fuente de datos también se usó para recolectar textos que no fueran ataques de ingeniería social. Para esto se usaron de igual manera la red social Twitter y la librería Tweepy. Se buscaron los tweets relacionados a los hashtags y parámetros de consulta: “#colombia”, “#deportes”, “#cine”, “eltiempo”, “premio”, “gana”, “apuesta”, “redes”, “chat”, “banco” y “pop”. Estos tweets no contenían imágenes por lo que no fue necesario hacer transcripción de estos. Los textos recolectados que no contenían ataques de ingeniería social de la segunda fuente de datos fueron 327.

La tercera fuente de datos corresponde a los datos de la investigación llamada “Seader: A social engineering attack detection method based on natural language processing and artificial neural networks” de Lansley et al. (2019). Esta fuente de datos consiste en 149 textos clasificados como ataques y no ataques de ingeniería social, en un archivo en formato Comma Separated Values (CSV). Como estos textos se encontraban en idioma inglés, fueron traducidos programáticamente al español usando la librería *Googletrans* del

lenguaje de programación Python. Esta librería hace uso de la *API* del traductor de Google para traducir los textos ingresados al idioma deseado (Googletrans, 2022).

Finalmente, se crearon tres sub datasets para dividir los datos y aumentarlos de forma más sencilla en la siguiente fase. El "sub dataset 1" contiene datos de fuentes de las redes sociales Facebook, Twitter y Reddit, y los datos de fuente propia. El "sub dataset 2" contiene datos únicamente de la red social Twitter y no contiene ataques de ingeniería social. Y el "sub dataset 3" es el dataset original de la investigación "Seader: A social engineering attack detection method based on natural language processing and artificial neural networks" (Lansley et al. 2019). Los datasets recolectados en esta fase se distribuyen como se muestra en la Tabla 2-1.

Tabla 2-1: Distribución textos iniciales de ataques de ingeniería social

Dataset	Fuente	Clasificación	Cantidad
Sub dataset 1	Facebook, Reddit, Twitter y propias	Ataque	109
		No ataque	105
Sub dataset 2	Twitter	Ataque	0
		No ataque	327
Sub dataset 3	"Seader: A social engineering attack…"	Ataque	78
		No ataque	71
Total	**Todas**	Ataque	187
		No ataque	503

Fuente: (Lansley et al. 2019). Elaboración propia

Al analizar los datos obtenidos se evidencia que en todos los ataques hay al menos un interlocutor, sin importar si es un ataque de ingeniería social o no. Por esta razón, se determina que los tipos de ataques de ingeniería social que se abarcaron en la investigación son los ataques de ingeniería social que usan los tipos de comunicación unidireccional y bidireccional.

2.2 Preparación de los datos

En esta fase se realizan las actividades de selección, limpieza, estructuración y formateo de los datos. Se seleccionaron los textos de ataques de ingeniería social que fueron usados para la construcción del modelo, y se realizó una limpieza de estos. Posteriormente, se generaron nuevos datos a partir de los textos previos, usando técnicas de aumento de datos. Finalmente, se formatearon o transformaron los textos en valores numéricos usando técnicas de procesamiento de lenguaje natural para posteriormente alimentar los algoritmos de clasificación.

2.2.1 Selección de datos y limpieza de datos

Los seleccionados para el desarrollo de la investigación son únicamente textos de diálogos, por lo que únicamente se necesitaron los textos de los datos recolectados de todas las fuentes de datos, y etiquetar cada uno de estos. Cada sub dataset creado se encuentra almacenado en archivos con formato CSV. Estos archivos consisten en dos columnas, la primera es el texto del dialogo recolectado. La segunda es la etiqueta de este ataque. Si el texto es o contiene un ataque de ingeniería social, la etiqueta tiene el valor “true”, si no, la etiqueta tiene el valor “false”.

Más adelante, se limpiaron los textos recolectados de dos formas. De forma manual, se anonimizaron datos de los textos como nombres de personas y nombres de empresas, marcas u organizaciones. De forma programática, usando expresiones regulares mediante la librería *Re* de Python, se eliminaron del texto todos los emojis y los símbolos de tweets como “#” y “@”. También, se eliminaron caracteres especiales encontrados como: “…”, “´”, “"” y saltos de línea.

2.2.2 Aumento de datos

El aumento de datos consiste en el uso de algoritmos y técnicas para construir datos a partir de un dataset disponible (Shorten et al., 2021). Como los textos recolectados que

contenían ataques de ingeniería social fueron relativamente pocos (187), fue necesario aumentar estos datos mediante técnicas de aumento de datos, usando procesamiento de lenguaje natural. Dos técnicas de aumento de datos fueron usadas: *Backtranslation* y *Reemplazo de palabras* o también llamado *Reemplazo por sinónimos* (*Synonym Replacement*).

- **Backtranslation:** Esta es una de las técnicas más populares para aumentar textos, esta técnica consiste en traducir el texto a un idioma diferente al original, después, volver a traducir el texto traducido al idioma original (Feng et at., 2021). Mediante esta técnica se aumentaron los textos recolectados del "sub dataset 1". Estos textos fueron traducidos de forma programática usando la librería Googletrans de lenguaje de programación Python. Los textos fueron traducidos al idioma inglés y traducidos de nuevo al idioma español. Finalmente, se obtuvieron 109 textos más que contenían ataques de ingeniería social y 105 textos que no contenían estos ataques.
- **Reemplazo de palabras:** Esta técnica consiste en reemplazar palabras en el texto por otras palabras, generalmente reemplazadas por sinónimos (Coulombe, 2018). Está técnica se usó para aumentar los datos del "sub dataset 1", mediante la librería *Spacy*, usando el modelo preentrenado "es_core_news_lg". *Spacy* es una librería enfocada al procesamiento de lenguaje natural en Python, la cual soporta más de 60 idiomas y permite hacer múltiples tareas de procesamiento de lenguaje natural, como reconocimiento de entidades, tokenización, lematización, clasificación de texto, entre otras (Spacy, 2022). El modelo "es_core_news_lg" contiene una tabla de vectores de palabras con 500.000 llaves, 500.000 vectores únicos de dimensión 300, construido con corpus de noticias en español. Para hacer el reemplazo de palabras se tokenizaron las palabras del texto, se ignoraron los tokens que tuvieran una longitud de caracteres menor a tres y que fueran signos de puntuación, y se seleccionaron dos palabras del texto para ser reemplazadas por sus sinónimos. Se buscaron diez sinónimos de cada palabra usando la librería Spacy con el modelo mencionado, usando el método *Vectors.most_similar,* el cual retorna las *n* entradas más similares usando la similitud por coseno dado un vector. La similitud de coseno es una medida de similitud que se calcula entre dos vectores usando el valor del coseno del ángulo entre estos dos vectores en un espacio. Cuando dos vectores

tienen la misma orientación su similitud de coseno es 1, cuando los vectores son perpendiculares entre sí, tienen una similitud de coseno de 0, y cuando dos vectores tienen una dirección opuesta, su similitud de coseno es menor a 0. La ecuación de la similitud de coseno se ilustra en la ecuación 2.1. Una vez encontrados los *n* vectores más similares, se escoge aleatoriamente uno de estos vectores, se busca su *hash* en la tabla de vectores y se obtiene la palabra representada por ese vector. Finalmente, la palabra inicial se reemplaza por su sinónimo en el texto y se crea una copia de este. Al finalizar este proceso, se obtuvieron 109 textos que contenían ataques de ingeniería social y 105 textos que no contenían estos ataques.

$$\cos(\beta) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \, \|\vec{w}\|} \tag{2.1}$$

Donde $\vec{v}$ y $\vec{w}$ son vectores diferentes en un espacio.

Al final del proceso de aumento de datos, los datos fueron consolidados y se distribuyen como se ilustra en la tabla 2-2.

Tabla 2-2: Distribución de dataset final ataques de ingeniería social

Clasificación	Cantidad	Proporción
Ataque	405	36.2%
No ataque	713	63.8%
Total	1118	100%

Fuente: Elaboración propia

2.2.3 Selección de características

Como no es posible alimentar los algoritmos de clasificación con las cadenas de caracteres de los textos, es necesario representar los textos como un vector de características numéricas. Para esto es necesario definir los siguientes conceptos:

- **Conteo:** Es el recuento de una característica o dato relacionado a información personal o de contacto (como emails o teléfonos) en el texto.
- **Tokenización:** La tokenización consiste en dividir el texto en varias partes que pueden ser oraciones o palabras, llamadas tokens.
- **Lematización:** Es una técnica que consiste en cortar una palabra para eliminar las terminaciones inflexibles de esta y obtener su forma canónica o lema. Las palabras flexibles son palabras que pueden tener morfemas flexivos, estas pueden variar en tiempo, género, número, etcétera.
- **Stop-words:** Estas son palabras muy comunes en cualquier idioma, pero en el contexto de aprendizaje de máquina no son muy relevantes. Algunas de estas palabras son 'de', 'la', 'que', 'el', 'en', 'estará', 'estaremos', 'tengas', 'tengamos', entre otras.
- **Expresiones regulares:** En el contexto del área de la computación, las expresiones regulares son patrones utilizados para encontrar combinaciones de cadenas de caracteres.
- **Reconocimiento de palabras nombradas (NER):** Esta técnica consiste en clasificar y etiquetar partes de un texto en categorías relevantes como personas, organizaciones, fechas, etcétera.

Las características seleccionadas para representar el texto a valores numéricos fueron propuestas basadas en los principios de persuasión (véase numeral 1.2.2) y en el estado del arte de los métodos automáticos de detección de ingeniería social mencionados previamente (véase numeral 1.3.2). En total se extrajeron siete características de los textos, las cuales fueron:

- **Conteo de emails:** En los intentos de ataques de ingeniería social, los atacantes suelen enviar información para contactarlos, prometiendo algún tipo de premio o remuneración económica. Los atacantes aprovechan el principio de distracción asegurando poder entregar sumas de dinero a las víctimas, con el fin de distraerlas del fraude. Esta característica es usada en la investigación de Balim y Gunal (2019). Este conteo se realizó mediante expresiones regulares.

- **Reporte de URLs:** Esta característica es usada en las investigaciones de Lansley et al. (2019), Lansley, Polatidis y Stelios (2019). Este ítem consiste en la revisión de las URLs encontradas en el texto mediante una herramienta externa. Las URLs son extraídas usando expresiones regulares. La API de Virus Total es usada para analizar las URLs encontradas en los textos, la cual retorna reportes de distintas bases de datos de antivirus. Cada base de datos reporta los valores 0 o 1, donde 0 es quiere decir que la URL no es maliciosa en esa base de datos, y 1 significa que la URL si es maliciosa en esa base de datos. Los valores de cada base de datos son sumados para cada URL.
- **Conteo de enlaces y URLs:** Como no todas las URLs maliciosas han sido reportadas o están en las bases de datos de antivirus y en muchos casos de ataques de ingeniería social son usadas URLs para engañar a las personas, como en los ataques de phishing, cartas nigerianas, smishing, entre otros, se propone esta característica. Por otra parte, muchas de las URLs extraídas de los textos transcritos fueron anonimizadas en las fuentes de datos, por lo tanto, se transcribieron como la entidad “(Link)”.
- **Conteo de números telefónicos:** Esta característica consiste en la búsqueda de números telefónicos en el texto. La característica es deteminante, ya que muchos ataques de ingeniería social incluyen números telefónicos para poder llamar al atacante y este puede aprovechar este acercamiento para realizar el ataque de forma más efectiva. Este conteo se realizó mediante expresiones regulares.
- **Revisión ortográfica:** La revisión ortográfica es una característica usada en las investigaciones de Lansley et al. (2019), Lansley, Polatidis y Stelios (2019), y Lansley, et al. (2020). Esta característica es usada debido a la gran cantidad de errores ortográficos en muchos de los intentos de suplantación a empresas, atípicos en las muestras de texto originales de dichas empresas. En esta revisión ortográfica no se tienen en cuenta palabras que no tienen ortografía definida en el idioma, mediante el reconocimiento de palabras nombradas (NER), como nombres de personas, nombres de productos, marcas, obras de arte y locaciones. Para revisar la ortografía del texto se le realizó una tokenización al texto, posteriormente, cada token que no fuera *stop-word* se comparó con una lista de palabras ortográficamente correctas. Si el token no se encontraba en esta lista, se considera un error ortográfico.

- **Ocurrencias en lista negra de palabras:** Con la búsqueda de ocurrencias en la lista negra de palabras es posible detectar una intención maliciosa en un texto. Esta característica es usada en las investigaciones de Sawa et al. (2016) y Lansley et al. (2019). La lista negra de palabras contiene palabras extraídas de la investigación de Lansley et al. (2019) traducidas al español, las cuales son palabras generales de políticas de seguridad. La lista negra también contiene otras palabras relacionadas a temas de ciberseguridad definidas de forma propia. Para obtener las ocurrencias de las palabras de la lista negra en el texto se realizó la tokenización del texto, después la lematización de las palabras de la lista negra y del texto, y se compararon para contar la ocurrencia.
- **Ocurrencias en verbos modales y advertencias:** Esta característica es usada para determinar si el atacante intenta persuadir a la víctima usando los principios de persuasión de tiempo o de cumplimiento social. La lista contiene verbos modales en español y advertencias como “urgente”, “advertencia”, entre otras. Esta característica es usada de forma similar en las investigaciones de Lansley et al. (2019), Lansley, Polatidis y Stelios (2019), y Lansley, et al. (2020). Para obtener las ocurrencias de las palabras de la lista verbos modales y advertencias en el texto se realizó la tokenización del texto, después la lematización de las palabras de la lista y del texto y se compararon para contar la ocurrencia.

2.3 Modelado

Como el objetivo de la investigación es la implementación de un método automático de detección de ingeniería social, se hace uso de algoritmos supervisados de clasificación de aprendizaje de máquina, teniendo en cuenta los datos recolectados y seleccionados. Los algoritmos seleccionados (basándose en la literatura) fueron: bosques aleatorios (Random Forest), máquinas de vector de soporte (SVM) y redes neuronales (NN).

2.3.1 Bosques aleatorios

A grandes rasgos el algoritmo de bosques aleatorios consiste en la agrupación de varios árboles de clasificación, seleccionando de forma aleatoria los datos para construir cada

árbol. Los árboles realizan predicciones con los datos de entrada para posteriormente ser ponderadas, calculando la clase más votada y realizar la predicción de la clase final.

Los árboles de decisión son métodos de aprendizaje supervisado. Este se compone de una raíz, que es la parte inicial del árbol; nodos, que representan las características para toma de decisiones; ramas, que representan la decisión en función de una condición determinada; y hojas, que son el valor final de clasificación. Los árboles de decisión clasifican los datos mediante una medida de impureza. Entonces, para un nodo $\boldsymbol{m}$, $\boldsymbol{N_m}$ es el número de instancias de entrenamiento que alcanza el nodo $\boldsymbol{m}$. $\boldsymbol{N_m^i}$ de $\boldsymbol{N_m}$ pertenece a las clases $\boldsymbol{C_i}$. Dado que una instancia alcanza el nodo m, la estimación de la probabilidad de la clase $\boldsymbol{C_i}$ es (ecuación 2.2) (Merino y Chacón, 2017):

$$\hat{P}(C_i \backslash x, m) \equiv p_m^i = \frac{N_m^i}{N_m} \tag{2.2}$$

Una función para medir la impureza es la entropía (ecuación 2.3):

$$I_m = \sum_{i=1}^{K} p_m^i \, log \, p_m^i \tag{2.3}$$

Para evitar que un árbol de decisión presente sobreajuste, se le realiza una poda, lo cual consiste en quitar ramas y nodos terminales hasta lograr el tamaño requerido. Otros métodos que mejoran el rendimiento de los árboles de decisión son el *Bagging* y *Boosting.* El *Bagging* consiste en dar un peso igual a los modelos.Este es un método de aprendizaje estadístico cuyo propósito es la reducción de la varianza en el modelo. Y el *Boosting* se desarrolla de forma contraria al *Bagging,* el cual da distintos pesos a los modelos para dar más relevancia a los que resalten más (Merino y Chacón, 2017).

El algoritmo de bosques aleatorios esta formado por múltiples arboles de decisión individuales entrenados con muestras aleatorias extraídas de los datos de entrenamiento mediante *Bagging,* donde cada árbol se entrena de forma ligeramente distinta y promedia el resultado de estos. Este algoritmo somete una muestra de los árboles de decisión a una serie de test binarios en cada nodo, hasta llegar a una hoja. El algoritmo, en la etapa de entrenamiento, intenta

optimizar los parámetros de las funciones de test binarios mediante la ecuación 2.4 (Merino y Chacón, 2017):

$$\theta_k^* = argmax_{\theta\ j\epsilon t_j} I_j \quad (2.4)$$

Donde para cada **k-ésimo** árbol se genera un vector aleatorio $\boldsymbol{\theta_k}$ independiente de los últimos vectores aleatorios generados. Además, se usa la siguiente función de ganancia (ecuación 2.5):

$$I_j = H(j) - \sum_{i\epsilon 1,2} \frac{|S_j^i|}{|s_j|} H(S_j^i) \quad (2.5)$$

Donde $\boldsymbol{S}$ representa el conjunto por dividir en un nodo.

2.3.1 Redes neuronales

Las redes neuronales, también conocidas como redes neuronales artificiales (ANNs), son un modelo inspirado en las redes neuronales biológicas del cerebro humano. Estas son capaces de aprender de la experiencia, abstraer características de un conjunto de datos y generalizar basadas en datos previos (Olabe, X. B., 1998).

Las redes neuronales son usadas en múltiples tareas dentro de las cuales se encuentran la conversión de texto a voz, el procesamiento de lenguaje natural, la comprensión de imágenes, el reconocimiento de caracteres, reconocimiento de patrones en imágenes, modelos económicos y financieros, entre otros.

Una red neuronal se compone de multiples capas, con una o varias neuronas, en donde se encuentran la capa de entrada, las capas ocultas y la capa de salida. Cada neurona clasifica los datos recibidos usando una función de activación, además, contiene una regla propagación y una función de salida. La neurona recibe ciertos valores de entrada multiplicados por un peso proveniente de otra neurona, el cual es la importancia de dicha

variable en el aprendizaje de al red neuronal. También, cada neurona contiene un sesgo, el cual puede ser ajustado para entrenar de forma más precisa la red neuronal (Bueno, F., 2019). Existen muchas funciones de activación que pueden ser usadas en la red neuronal, a continuación se definen las tres más usadas (ecuaciones 2.6, 2.7 y 2.8):

- **Función escalonada:**

  $$f(\mathrm{x}) = \begin{cases} 0 \; si \; x \; \leq U \\ 1 \; si \; x \; > U \end{cases} \quad (2.6)$$

  Donde $\boldsymbol{U}$ es el umbral deseado.

- **Función tangente hiperbólica:**

  $$f(\mathrm{x}) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.7)$$

- **Función sigmoide:**

  $$f(\mathrm{x}) = \frac{1}{1 + e^{x}} \quad (2.8)$$

La regla de propagación de las neuronas determina el potencial de la interacción entre una neurona y sus neuronales vecinas (ecuación 2.9). Este valor suele estar entre los valores -1 y 1.

$$z_i^L = \sum_j^N (w_{ij}^L * a_i^{L-1} + b_j^L) \quad (2.9)$$

Donde $\boldsymbol{L}$ es la capa donde se encuentra la neurona, $\boldsymbol{w_{ij}^L}$ son los pesos correspondientes de la capa $\boldsymbol{L}$ en la neuronas $\boldsymbol{i, j,}$ y $\boldsymbol{b_j^L}$ es el sesgo asociada a la neurona $\boldsymbol{j}$.

La función de salida proporciona el valor de la salida de la neurona, usando la función de activación definida (ecuación 2.10). Esta es el resultado de aplicar la función de activación a los valores recibidos por la capa anterior.

$$a_i^L = \sigma(z_i^L) \quad (2.10)$$

Donde σ representa la función de activación.

2.3.2 Máquinas de vector de soporte

Las máquinas de vectores de soporte o *support vector machines* en inglés (SVM), son un algoritmo que puede ser usado para realizar regresión y clasificación. Este algoritmo, cuando está enfocado en realizar clasificación, busca encontrar la mejor separación entre dos clases, definiendo un hiperplano que las separa. Este clasificador es una extensión del *soft margin classifier* o *support vector classifier* (SVC), el cual busca un margen de clasificación lo más ancho posible, donde la mayoría de las observaciones estén en el lado correcto del margen y permitiendo que algunas de estas observaciones estén del lado incorrecto del margen. La identificación del hiperplano de un clasificador de vector de soporte es un problema de optimización convexa. Este proceso incluye un hiperparámetro *C*. Dicho valor controla la dureza de las violaciones del margen y del hiperplano que se toleran en el procedimiento de ajuste (Amat, 2017). El valor óptimo de *C* se identifica mediante la validación cruzada.

Lo que diferencia los clasificadores SVC de SVM es que SVM usa el truco del *kernel*. Este consiste en expandir las dimensiones del espacio original para encontrar un hiperplano para separar las clases, cuando no exista una manera de separarlas en su espacio original. Un *kernel* es una función que retorna el resultado del producto punto entre dos vectores en un nuevo espacio dimensional (Amat, 2017). Los kernels más usados son:

- **Kernel lineal**: Este es el kernel más básico de todos, muchos de los problemas pueden ser clasificados usando este kernel. Su representación se muestra en la ecuación 2.11.

$$K\ (X, X'\ ) = (X \cdot X') \tag{2.11}$$

Donde $\boldsymbol{X}$ es un vector de valores de los datos de entrada

- **Kernel polinómico**: Este es una representación más general del kernel lineal. Este es menos usado y eficiente respecto a otros kernels. Su ecuación se muestra en la ecuación 2.12.

$$K(X, X') = (X \cdot X' + c)^d \quad (2.12)$$

Donde $\boldsymbol{X}$ es un vector de valores de los datos de entrada, ***d*** denota el grado del polinomio y ***c*** controla la penalización en el error de entrenamiento.

- **Kernel radial (RBF):** Este es uno de los kernels más usados en las máquinas de vector de soporte cuando los datos no son lineales. La ecuación 2.13 representa la formula de este kernel.

$$K(X, X') = exp(-\gamma \|X - X'\|)^2 \quad (2.13)$$

Donde $\boldsymbol{X}$ es un vector de valores de los datos de entrada y $\boldsymbol{\gamma}$ es una constante que toma valores entre cero y uno. Cuando el valor es muy pequeño el método comporta como un kernel lineal, si su valor aumenta, también lo hace la flexibilidad del método.

2.4 Evaluación

En esta fase se determinan las métricas usadas para evaluar el desempeño de los algoritmos de aprendizaje de máquina definidos previamente.

Teniendo en cuenta que el modelo computacional clasificará las entradas como ataques de ingeniería social o no ataque de ingeniería social (clasificación binaria), se hizo uso de las métricas comunes de evaluación de algoritmos de clasificación de aprendizaje de máquina.

Es necesario definir los siguientes términos para definir las métricas propuestas:

- **Verdaderos postivos (TP):** Son los casos donde el clasificador predijo que la entrada es un ataque de ingeniería social y el valor real de la entrada es un ataque de ingeniería social.
- **Falsos positivos (FP):** Son los casos donde el clasificador predijo que la entrada es un ataque de ingeniería social pero el valor real de la entrada no es un ataque de ingeniería social.
- **Verdaderos negativos (TN):** Son los casos donde el clasificador predijo que la entrada no es un ataque de ingeniería social y el valor real de la entrada no es un ataque de ingeniería social.
- **Falsos negativos (FN):** Son los casos donde el clasificador predijo que la entrada no es un ataque de ingeniería social pero el valor real de la entrada es un ataque de ingeniería social.

Una vez definidos los términos anteriores, las métricas propuestas fueron:

2.4.1 Exactitud (Accuracy)

La exactitud muestra la relación entre las predicciones correctas y el número total de predicciones. La métrica se define de la siguiente forma (ecuación 2.14):

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

Donde $\boldsymbol{TP}$ son casos los positivos verdaderos, $\boldsymbol{TN}$ son los casos negativos verdaderos, $\boldsymbol{FP}$ son los casos falsos positivos y $\boldsymbol{FN}$ son los casos falsos negativos.

2.4.2 Precisión

La precisión representa el número de predicciones correctas sobre el número total de predicciones correctas. La métrica se define con la siguiente ecuación (ecuación 2.15):

$$Precisión = \frac{TP}{TP + FP} \tag{2.15}$$

Donde $\boldsymbol{TP}$ son casos los positivos verdaderos y FP son los casos falsos positivos.

2.4.3 Exhaustividad (Recall)

La exhaustividad es el número de elementos identificados correctamente como positivos del total de positivos verdaderos. Se define de la siguiente forma (ecuación 2.16):

$$Exhaustividad = \frac{TP}{TP + FN} \tag{2.16}$$

Donde $\boldsymbol{TP}$ son casos los positivos verdaderos y $\boldsymbol{FN}$ son los casos falsos negativos.

2.4.4 Puntuación F1 (F1 Score)

La métrica de puntuación F1 es la media ponderada de la precisión y la exhaustividad, es usada para revisar el balance entre la exhaustividad y la precisión. Se define de con la siguiente ecuación (ecuación 2.17):

$$Puntuación\ F1 = 2 \times \frac{Precisión \times Exhaustividad}{Precisión + Exhaustividad} \tag{2.17}$$

3 Implementación del modelo

En este capítulo se presenta la implementación del modelo de detección de ingeniería social descrito en el capítulo anterior (véase capítulo 2). Presentando los recursos de máquina donde se realizó el experimento, el lenguaje de programación usado (numeral 3.1), y la implementación y evaluación del modelo propuesto (numeral 3.2).

3.1 Recursos de máquina y lenguaje de programación

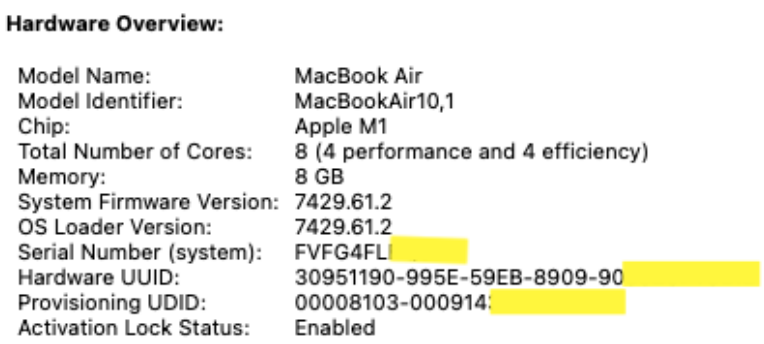
El modelo de detección de ingeniería social fue implementado y evaluado en una máquina con el sistema operativo macOS 12 – Monterey (Macintosh operating system). Este sistema operativo fue creado por la compañía Apple y es diseñado específicamente para su línea de computadores Macintosh. La versión 12 Monterey fue lanzada por la compañía en el mes de octubre del año 2021.

El modelo de la máquina es “MacBook Air (M1, 2020)”, y cuenta con las siguientes características:

- **Procesador:** Apple M1
- **Núcleos:** 8
- **Memoria física:** 8GB
- **Tarjeta de video:** Integrada
- **Disco duro:** 250GB

Se ilustra a detalle la configuración de hardware de la máquina en la figura 3-1 y la configuración de software en la figura 3-2.

Figura 3-1: Especificaciones de hardware de máquina

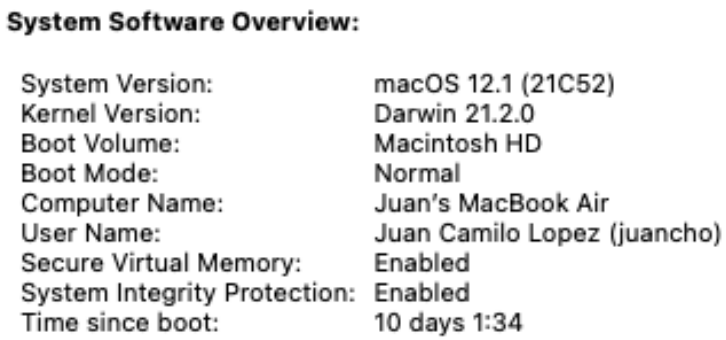


Hardware Overview:

Model Name:	MacBook Air
Model Identifier:	MacBookAir10,1
Chip:	Apple M1
Total Number of Cores:	8 (4 performance and 4 efficiency)
Memory:	8 GB
System Firmware Version:	7429.61.2
OS Loader Version:	7429.61.2
Serial Number (system):	FVFG4FL
Hardware UUID:	30951190-995E-59EB-8909-90
Provisioning UDID:	00008103-000914
Activation Lock Status:	Enabled

Fuente: macOS 12

Figura 3-2: Especificaciones de software de máquina



System Software Overview:

System Version:	macOS 12.1 (21C52)
Kernel Version:	Darwin 21.2.0
Boot Volume:	Macintosh HD
Boot Mode:	Normal
Computer Name:	Juan's MacBook Air
User Name:	Juan Camilo Lopez (juancho)
Secure Virtual Memory:	Enabled
System Integrity Protection:	Enabled
Time since boot:	10 days 1:34

Fuente: macOS 12

El lenguaje de programación usado para el desarrollo del experimento es el lenguaje Python en la versión 3.9. Python es un lenguaje de programación interpretado, multiparadigma y dinámico. Este fue creado en la década de los noventas por Guido Van Rossum en los Países Bajos. El lenguaje posee una licencia de código abierto, compatible con la licencia *GNU General Public License* (GPL) (Python, 2022). Según la compañía TIBOE (2022), en enero del año 2022, Python es el lenguaje de programación más popular, por encima de otros lenguajes como C, C++, Java, C#, Javascript, entre otros.

Python provee miles de librerías y módulos gracias a su popularidad y facilidad de uso. A continuación, se listan las librerías y módulos de Python usados para la implementación y evaluación del modelo:

- **os**: El módulo os permite usar funcionalidades del sistema operativo. Es usado para leer y escribir archivos, manipular rutas, crear archivos temporales, entre otros (Python, 2022).
- **time**: Este módulo permite usar funciones relacionadas con el tiempo. Es usada para obtener el tiempo de la máquina en distintos formatos, establecer cronómetros, entre otros (Python, 2022).
- **re**: Este módulo permite el uso de expresiones regulares para búsqueda de coincidencias en texto (Python, 2022).
- **Matplotlib**: Esta es una librería la cual permite crear visualizaciones estáticas, dinámicas y animadas en Python (Matplotlib, 2022).
- **Requests**: Requests es una librería que permite hacer petición usando el protocolo HTTP de forma sencilla y rápida (The Python Package Index (Pypi), 2022).
- **Pandas:** Pandas es una librería que permite crear estructuras de datos flexibles y rápidas para trabajar con datos relacionales de forma fácil e intuitiva (Pypi, 2022).
- **Numpy:** Es una librería que provee objetos multidimensionales y permite la manipulación matemática de estos, ente otras funciones (Numpy, 2022).
- **PySpellchecker:** Es un módulo el cual permite revisar la ortografía de una palabra, buscándola en una lista de palabras definida. Soporta los idiomas inglés, español, alemán, francés, y portugués (Pypi, 2022).
- **Googletrans:** Como se menciona previamente, esta librería permite traducir cualquier texto deseado, hace uso de la API de Google Translate para realizar las traducciones (Googletrans, 2022).
- **NLTK:** Es una librería la cual permite trabajar de forma sencilla con datos del leguaje humano. Permite realizar procesamientos sobre textos como tokenización, lematización, entre otros.
- **Spacy:** Como se menciona en capítulos anteriores, Spacy es una librería enfocada al procesamiento de lenguaje natural en Python, soporta más de 60 idiomas y permite hacer múltiples tareas de procesamiento de lenguaje natural, como reconocimiento de entidades, tokenización, clasificación de texto, entre otras (Spacy, 2022).

- **Scikit-Learn:** Es un módulo de Python el cual contiene múltiples herramientas para poder hacer tareas de clasificación, regresión, clustering, reducción de dimensional dad, entre otras, de forma simple y eficiente (Scikit-Learn, 2022).

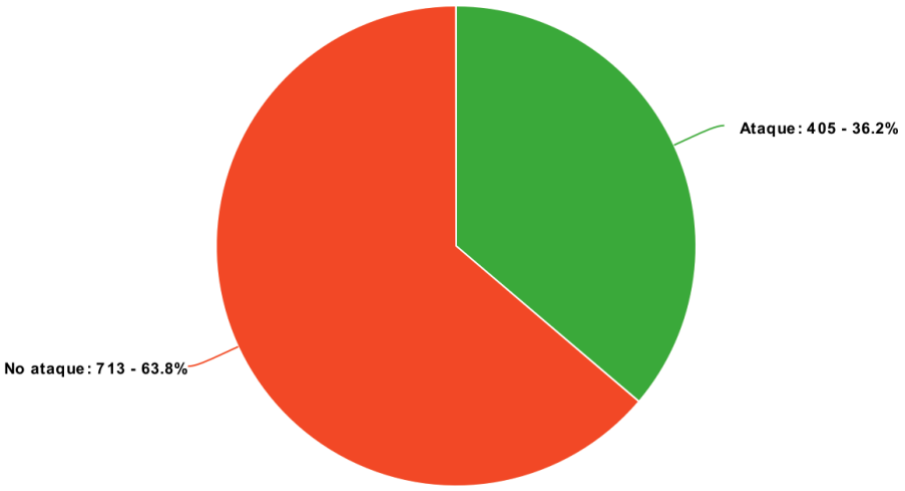
3.2 Implementación y evaluación del modelo

En esta sección se muestra el proceso de implementación y evaluación del modelo propuesto para la detección automática de ataques de ingeniería social.

3.2.1 Resultados de la comprensión y preparación de datos

En la fase de comprensión de datos y hasta la sub-fase de aumento de datos (véase del numeral 2.1 hasta el numeral 2.2.2) se obtuvo un dataset que consiste en textos de diálogos donde puede existir o no un ataque de ingeniería social. El dataset final construido en estas fases consta de 1118 textos, donde 713 de estos no contienen ataques de ingeniería social y los 405 restantes si contienen ataques de ingeniería social, como se ilustra en la figura 3-3.

Figura 3-3: Proporción de datos dataset final



Fuente: Elaboración propia

Se puede ver que la distribución de los datos no es completamente balanceada, pero teniendo en cuenta que el 36.2% de los datos contienen ataques de ingeniería social, no se puede considerar que el dataset es muy desbalanceado.

3.2.2 Resultados de la selección de características

Se implementó en el lenguaje de programación Python cada una de las características seleccionadas sobre los textos del dataset. Para cada característica se describe el proceso y los módulos y librerías usadas:

- **Conteo de emails:** Para extraer los emails del texto se usó el módulo *re* y su función *findall*, de esta forma se guardan una lista de emails en una variable y usando la función *len* se obtiene el tamaño de esta lista. La expresión regular usada fue "`[\w.+-]+@[\w-]+\.[\w.-]+`".
- **Reporte de URLs:** Para extraer las URLs del texto se usó el módulo *re* y su función *findall*, de esta forma se guardan una lista URLs en una variable, la expresión regular usada fue "`(?:https?://)?(?:(?i:[a-z]+\.)+)[^\s,]+`". Después, cada URL de la lista es analizada usando la API de VirusTotal. Para el uso de la API se debe hacer una petición de tipo POST enviando la URL a analizar, la API retorna un identificador para posteriormente hacer una petición tipo GET para obtener el reporte del análisis de la URL. Para hacer las peticiones se usó la librería *requests*, también se usó el módulo *time*, para pausar la ejecución del análisis 30 segundos, ya que la versión gratuita de la API permite hacer máximo dos análisis de URL por minuto. Finalmente, los valores de los reportes son sumados y retornados para todas las URLs encontradas en un texto.
- **Conteo de enlaces y URLs:** Para el conteo de enlaces y URLs, se usa la lista de URLs extraída de la característica anterior y se obtiene la cantidad de estas usando la función *len*. Posteriormente, se buscan los enlaces que se etiquetaron inicialmente con las entidades "(enlace)" y "(link)" usando el módulo *re* y la función *findall*, el cual retorna una lista de ocurrencias y es usada la función *len* para contar el tamaño de la lista. La expresión regular usada es "`(?i)\(link\)|(i?)\(enlace\)`". Finalmente, se suman los dos valores obtenidos y se retorna.

- **Conteo de números telefónicos:** Para extraer los números telefónicos del texto se usó el módulo *re* y su función *findall*, de esta forma se guardan una lista de números telefónicos en una variable y usando la función *len* se obtiene el tamaño de esta lista. La expresión regular usada fue "`\d{10}|\d{7}|(i?)\(telefono\)|(i?)\(teléfono\)`". Con esta expresión se extraen todos los números que tengan 7 o 10 dígitos, y las entidades "(telefono)" y"(teléfono)".
- **Revisión ortográfica:** Para la revisión ortográfica primero se tokeniza el texto usando la clase TweetTokenizer del módulo NLTK, la cual retorna una lista de tokens. Después, se eliminan los tokens que sean stop-words, estos stop-words se importan del módulo NLTK. A continuación, se instancia el módulo PySpellchecker, y se define una lista blanca de palabras de forma manual que contiene nombres de personas, empresas y locaciones, para que no sean analizadas por el módulo. A esta lista blanca de palabras también se añaden de forma programática las palabras en el texto que correspondan a las entidades: 'PERSON', 'PER', 'GPE', 'LOC', 'PRODUCT', 'WORK_OF_ART', usando la librería Spacy y su reconocimiento de palabras nombradas (NER). Finalmente, la función *unknown* retorna una lista de las palabras con errores de ortografía encontradas en el texto y luego se usa la función *len* para retornar el tamaño de esa lista.
- **Ocurrencias en lista negra de palabras:** Para extraer la ocurrencia de palabras en el texto respecto a la lista negra de palabras, se lematizaron las palabras de la lista negra de palabras y los tokens del texto, usando la clase SnowballStemmer de la librería NLTK. Una vez lematizadas, se compararon. Si eran iguales, se suma uno a un contador para luego retornar su valor. Las palabras usadas en esta lista negra de palabras fueron: "admin, administrador, dirección, direcciones, jefe, computador, confidencial, conectar, credenciales, data, datos, base de datos, detalles, dispositivo, email, trabajador, archivo, firewall, info, información, instalar, ip, login, ingresar, máquina, nombre, número, passcode, password, contraseña, frase, servidor, software, especificaciones, sistema, tecnología, tech, usuario, username, db, ataque, acceso, seguridad, cuenta, verificación, autenticación, desactivar, activar y jefe".
- **Ocurrencias en verbos modales y advertencias:** Para extraer la ocurrencia de palabras en el texto respecto a la lista de verbos modales y advertencias, se

lematizaron las palabras de la lista y los tokens del texto, usando la clase SnowballStemmer de la librería NLTK. Una vez lematizadas, se compararon y si eran iguales, se sumaba uno a un contador para luego retornar su valor. Las palabras usadas en esta lista de verbos modales y advertencias fueron: “necesito, debe, debería, tiene, alerta, peligro, urgente, ayuda, deprisa, contar, decir”

Después del proceso de extracción de características cada texto es representado en un vector con siete valores numéricos y la clasificación correspondiente (ataque o no ataque). El vector de características tiene las siguientes etiquetas: (1) Reporte de URLs, (2) Conteo de enlaces y URLs, (3) Conteo de emails, (4) Conteo de números telefónicos, (5) Revisión ortográfica, (6) Ocurrencias en lista negra de palabras, (7) Ocurrencias en verbos modales y advertencias, y (8) Ataque o no ataque. Un ejemplo de este vector se ilustra en la figura 3-3. Donde, el primer elemento de cada fila corresponde al texto analizado y el segundo elemento es el vector de características correspondiente a ese texto.

Figura 3-3: Ejemplo de vector de características de los textos

"Bankbank :Lamentamos informarle que su cuenta ha sido desactivada. Por su seguridad le rogamos que complete la siguiente verificacion: https://bit.ly/3tmana" (2, 1, 0, 0, 1, 5, 0, 1)
"Una persona que estuvo en contacto con usted tuvo un resultado positivo en una prueba de COVID-19 o ha presentado síntomas de esta enfermedad y se recomienda que se auto aislé/se haga la prueba. Más información en (Link)" (0, 1, 0, 0, 1, 0, 0, 1)
"Este es John Candy de Apple.Soy un representante de ventas, ¿te gustaría comprar algún software?Genial, solo voy a necesitar su contraseña.adiós gracias." (0, 1, 0, 0, 2, 2, 0, 1)
"MEXICO ocupa el deshonroso primer lugar en defunciones de personal médico en América, lo que refleja la irresponsabilidad gubernamental." (0, 0, 0, 0, 0, 0, 0, 0)

Fuente: Elaboración propia

Finalmente, se extrajeron las características de todos los textos del dataset, dando como resultado una matriz de dimensiones (1118, 8).

3.2.3 Resultados del modelado y evaluación

Los algoritmos de clasificación seleccionados fueron máquinas de vector de soporte (SVM), bosques aleatorios (RF) y redes neuronales (NN). Para usar estos algoritmos en el lenguaje de programación Python se usó el módulo Scikit-Learn, ya que este tiene implementados múltiples algoritmos de clasificación entre los que se encuentran los mencionados. Además, el módulo permite la evaluación de los algoritmos de forma eficiente y simple y contiene implementadas las métricas definidas para la evaluación de los algoritmos (véase numeral 2.4.1). Adicionalmente, se realiza una medición de la importancia de las características a cada uno de los algoritmos para determinar cual de las características influye más en la detección de ingeniería social en cada algoritmo.

Cada uno de los algoritmos de clasificación fue implementado usando los modelos implementados por Scikit-Learn así: Para el algoritmo SVM se usó el modelo svm.SVC, para el algoritmo de bosques aleatorios se usó el modelo RandomForestClassifier, y para el algoritmo de redes neuronales se usó el modelo MLPClassifier.

Inicialmente se exploraron los mejores hiperparámetros de los modelos de clasificación mediante el módulo GridSearch de Scikit-Learn. Para esto es necesario definir los hiperparámetros más usados de cada uno de los modelos:

- **Bosques aleatorios:**
  - **criterion:** Este parámetro define la función que mide la calidad de las divisiones de las ramas de los árboles, pueden tomar los valores de *gini*, *entropy*.
  - **n_estimators:** Este parámetro corresponde a el número de árboles que va a tener el bosque aleatorio. Normalmente cuantos más árboles haya, el

modelo es mejor, pero a partir de cierto punto se pierde rendimiento. El valor por defecto del parámetro es 100.

- **n_jobs:** Es el número de núcleos que pueden ser usados para entrenar los árboles. Cada árbol es independiente al resto, así que entrenar un bosque aleatorio es una tarea muy paralelizable. Por defecto sólo utiliza 1 núcleo de la CPU. No obstante, para mejorar el rendimiento del modelo, se pueden usar todos los núcleos disponibles en la máquina.
- **max_features:** Este parámetro es usado para garantizar que los datos de entrenamiento de los árboles en el bosque aleatorio sean diferentes. Este parámetro es útil cuando hay atributos relacionados entre sí.

- **SVM:**
  - **kernel:** Especifica el tipo de kernel que será usado por el modelo. Este parámetro puede ser *linear*, *poly*, *rbf*, *sigmoid* y *precomputed*. El valor por defecto es *linear*.
  - **C:** Es el parámetro de regularización. Este parámetro es inversamente proporcional a la fuerza de regularización.
  - **degree:** Este parámetro es el grado del polinomio de la función cuando el kernel es *poly*.
  - **gamma:** Este parámetro es el coeficiente del kernel cuando este sea *poly*, *rbf* o *sigmoid*.

- **Redes neuronales:**
  - **hidden_layer_sizes:** Este parámetro representa la **c**antidad de capas ocultas que tendrá el modelo.
  - **activation:** Define la función de activación para las capas ocultas del modelo. Puede tomar los valores de *identity*, *logistic*, *tanh* y *relu*.
  - **solver:** Este parámetro especifica el algoritmo para la optimización de los pesos en los nodos del modelo. Puede tomar los valores de *lbfgs*, *sgd* y *adam*.
  - **learning_rate:** Los valores que toma este parámetro son *constant*, *invscaling* y *adaptive*. Este controla cuánto debe cambiar el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo.

En la exploración de hiperparámetros se asignó un 70% del dataset para el entrenamiento y 30% del dataset para evaluación. Se usó validación cruzada con cinco pliegues. Los resultados de esta exploración se muestran en la tabla 3-1. Los hiperparámetros que no se muestran en la tabla toman el valor por defecto asignado por la librería Scikit-Learn.

Tabla 3-1: Resultado exploración de hipérparametros algoritmos de clasificación

Algoritmo	Máxima exactitud	Mejores parámetros
Bosques aleatorios	0.8478	▪ criterion: gini ▪ n_estimators: 100
SVM	0.8452	▪ C: 50 ▪ gamma: auto ▪ kernel: rbf
Redes neuronales	0.8529	▪ activation: tanh ▪ hidden_layer_sizes: (100, 100, 100) ▪ learning_rate: constant ▪ solver: adam

Fuente: Elaboración propia

Una vez encontrados los mejores hiperparámetros para cada algoritmo, se evaluaron de nuevo, esta vez usando validación cruzada con diez pliegues y la misma división del dataset: 70% y 30% para entrenamiento y evaluación respectivamente. Además, se evaluaron cuatro veces y se calculó el promedio de los resultados de las métricas para todas las evaluaciones y su desviación estándar. Las métricas de evaluación obtenidas para cada algoritmo se muestran en la tabla 3-2, tabla 3-3 y tabla 3-4, para los modelos de bosques aleatorios, SVM y redes neuronales respectivamente.

Tabla 3-2: Resultados de evaluación métricas de algoritmo bosques aleatorios

Exactitud	Precisión	Exhaustividad	Puntuación F1
0.8204 ± 0.0032	0.8027 ± 0.0050	0.7408 ± 0.0032	0.7506 ± 0.0019

Elaboración propia

Tabla 3-3: Resultados de evaluación métricas de algoritmo SVM

Exactitud	Precisión	Exhaustividad	Puntuación F1
0.8408 ± 0	0.8071 ± 0	0.8196 ± 0	0.8010 ± 0

Elaboración propia

Tabla 3-4: Resultados de evaluación métricas de algoritmo redes neuronales

Exactitud	Precisión	Exhaustividad	Puntuación F1
0.8280 ± 0.0099	0.8043 ± 0.0129	0.7772 ± 0.0166	0.7719 ± 0.0147

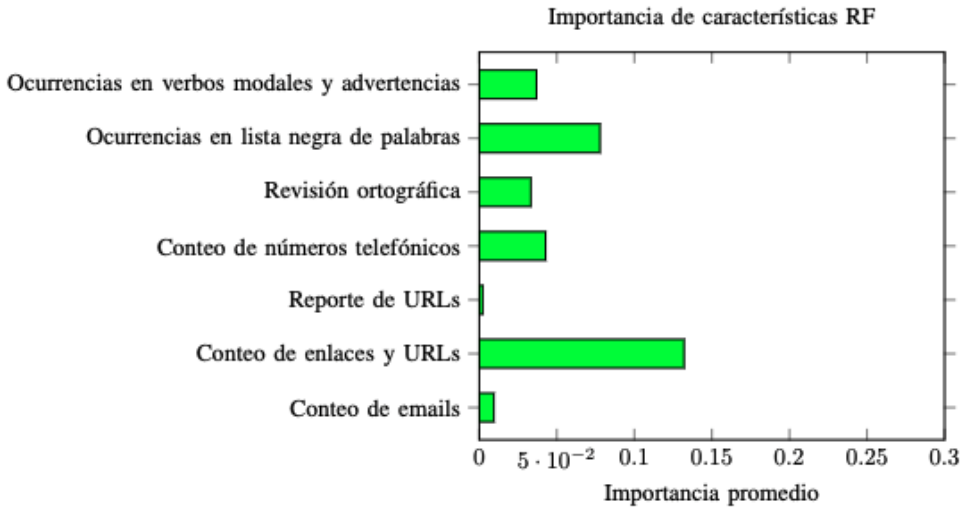
Elaboración propia

Finalmente, se hace una revisión de las características más importantes para cada algoritmo de clasificación, esto se realizó mediante la técnica de Scikit-Learn llamada *Permutation Feature Importance*. Esta es una técnica de inspección de modelo de clasificación que permite encontrar las características más importantes de un modelo, puede ser usada en cualquier modelo, y la técnica consiste en romper la relación entre la función y el objetivo, donde la caída en el puntaje del modelo indica cuánto depende el modelo de la función (Scikit-Learn, 2022).

Para el algoritmo de bosques aleatorios (figura 3-4) todas las características son usadas por el algoritmo. Se puede ver que la característica más importante es el conteo de enlaces y URLs, seguido de la característica de ocurrencias en lista negra de palabras, y el conteo de números telefónicos. Esto puede deberse a que en varios de los textos que son ataques de ingeniería social se presentan URLs maliciosas, además, combinando esta

característica con la lista negra de palabras hace que un texto que contiene estas dos características probablemente sea un ataque de ingeniería social.

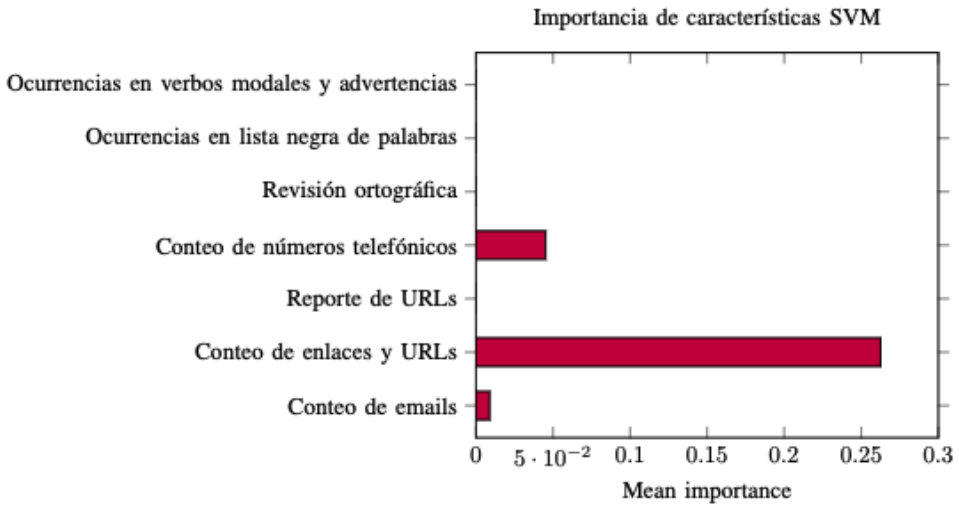
Figura 3-4: Importancia de características de algoritmo bosques aleatorios



Fuente: Elaboración propia

Para el algoritmo SVM, se puede ver (figura 3-5) que sólo usa tres características de las siete seleccionadas. La característica más importante para el algoritmo es el conteo de enlaces y URLs, seguido por el conteo de números telefónicos y el conteo de emails. Esta selección de características del modelo lo hace muy propenso a inferir que si se encuentra una URL en un texto lo clasifique como un ataque de ingeniería social. Aunque sus métricas de evaluación tengan buenos resultados, puede que al evaluarse con un nuevo dataset su puntaje baje considerablemente.

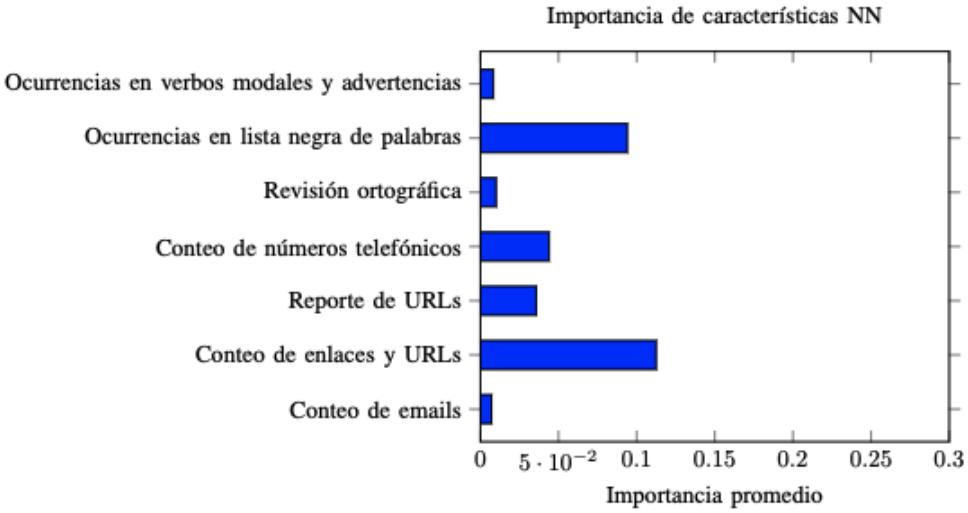
Figura 3-5: Importancia de características de algoritmo SVM



Fuente: Elaboración propia

El algoritmo de redes neuronales hace uso de todas las características seleccionadas (figura 3-6), donde la característica más importante es el conteo de palabras, seguido por la ocurrencias en lista negra de palabras y el conteo de números telefónicos. Estas características son muy similares a las del algoritmo de bosques aleatorios.

Figura 3-6: Importancia de características de algoritmo redes neuronales



Fuente: Elaboración propia

4 Discusión de resultados

Teniendo en cuenta los resultados obtenidos de las métricas de medición de los algoritmos de clasificación, se calcula el promedio de cada métrica para cada algoritmo como se muestra en la tabla 4-1.

Tabla 4-1: Promedio de resultados evaluación métricas de algoritmos de clasificación

Algoritmo	Promedio de exactitud	Promedio de precisión	Promedio de exhaustividad	Promedio de puntuación F1
Bosques aleatorios	0.8204	0.8027	0.7408	0.7506
SVM	**0.8408**	**0.8071**	**0.8196**	**0.8010**
Redes neuronales	0.8280	0.8043	0.7772	0.7719

Elaboración propia

Los resultados de la tabla muestran que el algoritmo de clasificación con mayor exactitud es el SVM, lo cual indica que este algoritmo clasifica correctamente los textos entre ataque o no ataque de ingeniería social aproximadamente un 84% de las veces. El algoritmo con mejor precisión también es el SVM, esto indica que el algoritmo es relativamente preciso detectando ataques de ingeniería social. El único algoritmo que pasa del 0.8 el valor de exhaustividad es el SVM, con un 0.8196, lo que indica que este algoritmo es sensible al detectar ataques de ingeniería social. Finalmente, el algoritmo con mayor puntuación F1 fue el SVM, con 0.8010, esto indica que existe buena relación entre la precisión y la

exhaustividad del algoritmo. Los algoritmos de bosques aleatorios y redes neuronales también hacen un buen trabajo en la clasificación de ataques de ingeniería social, pero sus métricas son un poco inferiores a las del algoritmo SVM.

Los tres clasificadores evaluados tienen buenos resultados en sus métricas de rendimiento, todos obtienen una exactitud promedio por encima del 80%, lo que los hace buenos algoritmos de clasificación para la detección de ingeniería social teniendo en cuenta el dataset creado y las características extraídas.

Considerando los demás sistemas y métodos de detección automática de ingeniería social (véase numeral 1.3.2), como el presentado por Sawa et al. (2016) que consiste en la detección de ingeniería social mediante el uso de técnicas de procesamiento de lenguaje natural, donde al evaluar el sistema obtienen resultados de precisión del 100% y una exhaustividad del 60% siendo estas las únicas métricas reportadas. Se puede afirmar que el sistema es muy preciso cuando detecta ataques de ingeniería social, pero es poco sensible para clasificar los ataques positivos de ingeniería social. Se aprecia que este sistema es más preciso que el desarrollado en este proyecto, pero es menos sensible a los ataques positivos de ingeniería social, además, como no registran métricas de exactitud, no es posible comparar su rendimiento.

Como se menciona previamente (véase numeral 1.3.2), Lansley et al. (2019) desarrollan un sistema de detección de ingeniería social usando técnicas de procesamiento de lenguaje natural y algoritmos de clasificación. Estos algoritmos son evaluados en un dataset de datos reales y otro dataset con datos sintéticos. Los resultados de las métricas de exactitud de los algoritmos propuestos, evaluados con el dataset real, fueron: árbol de decisión: 68.1%, bosques aleatorios: 68.3%, y CNN: 68.7%. Los resultados de este sistema son buenos, no obstante, comparados con el sistema descrito y desarrollado en este documento, son inferiores en lo que a exactitud respecta. Esto puede ser al tamaño del dataset real, y a las características extraídas del texto. Lansley et al. (2019) usan un dataset de 147 textos, y tres características: puntaje de links, puntaje de ortografía y puntaje de intención.

Lansley, Polatidis y Stelios (2019) proponen un modelo basado en aprendizaje de máquina y procesamiento de lenguaje natural, el cual consiste en tres fases: preprocesamiento de datos, extracción de características y agregación de los resultados. El modelo es implementado en Python y además, se hace uso de las mismas características que el sistema propuesto por Lansley et al. (2019) y evalúan el modelo usando los mismos datasets de la investigación mencionada. Para el dataset real, los resultados de las métricas de exactitud de los algoritmos de clasificación de la investigación fueron: árbol de decisión: 68.1%, bosques aleatorios: 68.3%, y perceptrón multicapa: 69.1%. Estos resultados de exactitud son altos, pero al no reportar más métricas es difícil saber el rendimiento completo del sistema. Comparando los puntajes con el modelo desarrollado como resultado de esta investigación, se aprecia que en términos de exactitud el sistema de Lansley, Polatidis y Stelios (2019) es un poco menos exacto a la hora de detectar ataques de ingeniería social que el modelo presentado.

Lansley et al. (2020) proponen un nuevo método para la detección de ingeniería social basado en el SEADer (Lansley, 2019). Este método usa las mismas fases del SEADer, y añade un paso final que consiste en la implementación de un método de ensamble basado en votación suave. Este método de ensamble es evaluado usando el dataset sintético del SEADer junto al algoritmo de clasificación de Perceptron multicapa, el valor de exactitud para el algoritmo de perceptrón multicapa es de 92.2%, el valor de exactitud para el método de ensamble es de 92.4%. Es difícil comparar este modelo con otros ya que los modelos son evaluados teniendo en cuenta el dataset sintético creado en la investigación, no el dataset real.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

- Para entrenar y evaluar el modelo propuesto se tuvo que construir un dataset de textos de ataques de ingeniería social en español, buscando en distintas fuentes de datos como redes sociales, datos usados en investigaciones previas y datos propios. Además, fue necesario aumentar los datos para mejorar el rendimiento del modelo.
- El modelo propuesto en esta investigación de detección automática de ataques de ingeniería social hace uso de algoritmos de clasificación de aprendizaje de máquina para clasificar las entradas como ataques o no ataques de ingeniería social, y usa técnicas de procesamiento de lenguaje natural para extraer características de los textos ingresados.
- Las características usadas en el modelo fueron propuestas teniendo en cuenta los principios de persuasión y los usados en la literatura. Estas fueron: conteo de emails, conteo de enlaces y URLs, reporte de URLs, conteo de números telefónicos, revisión ortográfica, ocurrencias en lista negra de palabras, ocurrencias en verbos modales y advertencias.
- El modelo fue implementado haciendo uso de herramientas y librerías de código libre y gratuitas, lo que permite que sean implementadas y usadas por cualquier usuario sin costo alguno.
- Los tres algoritmos evaluados en el modelo arrojan buenos resultados en la detección de ataques de ingeniería social, donde su exactitud es mayor a 80% al

igual que su precisión. Donde, el algoritmo de máquina de vector de soporte sobresale en las métricas de exhaustividad y puntaje F1 (0.81 y 0.80 respectivamente).

- La característica más importante para los tres algoritmos de clasificación fue el conteo de enlaces y URLs. Los algoritmos de bosques aleatorios y redes neuronales usaron todas las demás características para el proceso de clasificación, mientras que el algoritmo de máquina de vector de soporte sólo usó las características de conteo de enlaces y URLs, conteo de números telefónicos y conteo de emails. Esto puede afectar considerablemente el rendimiento del algoritmo si se evalúa con un nuevo dataset.

5.2 Trabajo futuro

El reto más grande de la investigación fue la construcción del dataset de ataques de ingeniería social en español. Para mejorar este proceso, se recomienda hacer alianzas con instituciones de ciberseguridad que tengan bases de datos de estos ataques y que permitan el uso de estos para construir un dataset mucho más robusto.

Para el proceso de selección de características, se pueden revisar investigaciones sobre detección de tipos de ciberataques que usen ingeniería social, como el phishing, smishing, spam, cartas nigerianas, etcétera. Esto para tener características aún más significativas para los algoritmos de clasificación en la tarea de detectar ataques de ingeniería social.

Respecto al proceso de clasificación, se pueden explorar más algoritmos de clasificación a parte de los propuestos en esta investigación. También se recomienda hacer uso de métodos de ensamble entre los algoritmos para obtener mejores resultados a la hora de detectar ataques de ingeniería social. Además, se puede explorar el uso de aprendizaje profundo.

6 Bibliografía

Amat, J. (Abril 2017). Máquinas de Vector Soporte (Support Vector Machines, SVMs) https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

Balim, C., & Gunal, E. S. (Noviembre 2019). Automatic Detection of Smishing Attacks by Machine Learning Methods. In 2019 1st International Informatics and Software Engineering Conference (UBMYK) (pp. 1-3). IEEE.

Bezuidenhout, M., Mouton, F., & Venter, H. S. (2010). Social engineering attack detection model: SEADM. Proceedings of the 2010 Information Security for South Africa Conference, ISSA 2010.

Bhakta, R., & Harris, I. G. (2015). Semantic analysis of dialogs to detect social engineering attacks. Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015).

Bhardwaj, T., Sharma, T. K., & Pandit, M. R. (2014). Social engineering prevention by detecting malicious URLs using artificial bee colony algorithm. 355-363.

Bueno, F. (2019). Redes neuronales: entrenamiento y comportamiento.

Cialdini, Robert. (1993). Influence: Science and Practice.

Coulombe, C. (2018). Text data augmentation made simple by leveraging nlp cloud apis. arXiv preprint arXiv:1812.04718.

Craigen, D., Diakun-Thibault, N., & Purse, R. (2014). Defining cybersecurity. Technology Innovation Management Review, 4(10).

Dan, A., & Gupta, S. (2019). Social engineering attack detection and data protection model (SEADDPM). In Advances in Intelligent Systems and Computing (Vol. 811, pp. 15-24). https://doi.org/10.1007/978-981- 13-1544-2

Del Pozo, I. (2018). Social engineering: Application of psychology to information security. 2018 6th International Conference on Future Internet of Things and Cloud Workshops

Denning, T., Lerner, A., Shostack, A., & Kohno, T. (2013, November). Control-Alt-Hack: the design and evaluation of a card game for computer security awareness and education. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 915-928).

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). A survey of data augmentation approaches for nlp. arXiv preprint arXiv:2105.03075.

Footprint (2021). 2021 State of the Phish. An In-Depth Look at User Awareness, Vulnerability and Resilience. https://www.proofpoint.com/sites/default/files/threat-reports/pfpt-us-tr-state-of-the-phish-2021.pdf

Gatlan, S. (3 de septiembre de 2020). FBI: Thousands of orgs targeted by RDoS extortion campaign. *BleepingComputer*. https://www.bleepingcomputer.com/news/security/fbi-thousands-of-orgs-targeted-by-rdos-extortion-campaign/

Googletrans (11 de enero 2022). Googletrans 3.0.0 documentation. https://py-googletrans.readthedocs.io/en/latest/

Gragg, D. (2003). A multi-level defense against social engineering. SANS Reading Room, 13, 15.

Gregar, J. (1994). Research Design (Qualitative, Quantitative and Mixed Methods Approaches). Book published by SAGE Publications, 228.

Hadnagy, C. (2010). Social Engineering: The Art of Human Hacking.

Hernández-Sampieri, R., & Torres, C. P. M. (2018). Metodología de la investigación (Vol. 4). México^ eD. F DF: McGraw-Hill Interamericana.

Infoblox. (2020). Cyberthreat Intelligence Report. The Infloblo Q3 2020.

Ivaturi, K., & Janczewski, L. (Junio 2011). A taxonomy for social engineering attacks. In International Conference on Information Resources Management (pp. 1-12). Centre for Information Technology, Organizations, and People.

Janczewski, L., & Colarik, A. (Eds.). (2007). Cyber warfare and cyber terrorism. IGI Global.

Junger, M., Montoya, L., & Overink, F. J. (2017). Priming and warnings are not effective to prevent social engineering attacks. Computers in human behavior, 66, 75-87.

Kaspersky. (2022). ¿Qué es la ciberseguridad?. Recuperado el 02 de enero de 2022 de https://latam.kaspersky.com/resource-center/definitions/what-is-cyber-security

Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing detection: a literature survey. IEEE Communications Surveys & Tutorials, 15(4), 2091-2121.

Khorshed, M. T., Ali, A. S., & Wasimi, S. A. (2014). Combating Cyber Attacks in Cloud Systems Using Machine Learning. In Security, Privacy and Trust in Cloud Systems (pp. 407-431). Springer, Berlin, Heidelberg.

Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and applications*, *22*, 113-122.

Lansley, M., Mouton, F., Kapetanakis, S., & Polatidis, N. (2020). SEADer++: social engineering attack detection in online environments using machine learning. Journal of Information and Telecommunication, 4(3), 346-362.

Lansley, M., Polatidis, N., & Kapetanakis, S. (Septiembre 2019). Seader: A social engineering attack detection method based on natural language processing and artificial neural networks. In International Conference on Computational Collective Intelligence (pp. 686-696). Springer, Cham.

Lansley, M., Polatidis, N., Kapetanakis, S., Amin, K., Samakovitis, G., & Petridis, M. (2019). Seen the villains: Detecting Social Engineering Attacks using Case-based Reasoning and Deep Learning. In ICCBR Workshops (pp. 39-48).

Long, J. (2011). No tech hacking: A guide to social engineering, dumpster diving, and shoulder surfing. Syngress.

López, J., & Camargo, J., (Para ser presentada en Marzo 2022). Social Engineering Detection Using Natural Language Processing and Machine Learning.The 5th International Conference on Information and Computer Technologies (ICICT), 2022.

Malwarefox. (2021). How to Spot Fake Facebook Profile. https://www.malwarefox.com/spot-fake-facebook-profile/

Matplotlib. (14 de enero 2022). Matplotlib: Visualization with Python. https://matplotlib.org

Merino, R. F. M., & Chacón, C. I. Ñ. (2017). Bosques aleatorios como extensión de los árboles de clasificación con los programas R y Python. Interfases, (10), 165-189.

Mitnick, K. D., & Simon, W. L. (2003). The art of deception: Controlling the human element of security. John Wiley & Sons.

Mokhor, V. V, Tsurkan, O. V, Tsurkan, V. V, & Herasymov, R. P. (2017). Information security assessment of computer systems by socio-engineering approach. CEUR Workshop Proceedings, 2067, 92-98.

Mouton, F., Leenen, L., & Venter, H. S. (2016). Social Engineering Attack Detection Model: SEADMv2. Proceedings - 2015 International Conference on Cyberworlds, CW 2015, 216-223.

Mouton, F., Malan, M. M., Leenen, L., & Venter, H. S. (Agosto 2014). Social engineering attack framework. In 2014 Information Security for South Africa (pp. 1-9). IEEE.

Mouton, F., Teixeira, M., & Meyer, T. (Agosto 2017). Benchmarking a mobile implementation of the social engineering prevention training tool. In 2017 Information Security for South Africa (ISSA) (pp. 106-116). IEEE.

NLTK. (14 de enero 2022). Documentation - Natural Language Toolkit. https://www.nltk.org

Numpy. (14 de enero 2022). Numpy documentation. https://numpy.org/doc/stable/

Olabe, X. B. (1998). Redes neuronales artificiales y sus aplicaciones. Publicaciones de la Escuela de Ingenieros.

OWASP. (2021). OWASP Top 10 - 2021. https://owasp.org/Top10/

Python. (14 de enero 2022). History and License. https://docs.python.org/3/license.html

Python. (14 de enero 2022). os — Interfaces misceláneas del sistema operativo. https://docs.python.org/es/3.9/library/os.html?highlight=#module-os

Python. (14 de enero 2022). re — Operaciones con expresiones regulares. https://docs.python.org/es/3.9/library/re.html

Python. (14 de enero 2022). time — Tiempo de acceso y conversiones. https://docs.python.org/es/3.9/library/time.html?highlight=time#module-time

Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications, 117, 345-357.

Sawa, Y., Bhakta, R., Harris, I. G., & Hadnagy, C. (2016). Detection of Social Engineering Attacks Through Natural Language Processing of Conversations. Proceedings - 2016 IEEE 10th International Conference on Semantic Computing, ICSC 2016, 262□265. https://doi.org/10.1109/ICSC.2016.95

Scikit-Learn (14 de enero 2022). Inicio - scikit-learn - Machine Learning in Python. https://scikit-learn.org/dev/index.html

Scikit-Learn (15 de enero 2022). 4.2. Permutation feature importance. https://scikit-learn.org/stable/modules/permutation_importance.html

Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Text data augmentation for deep learning. Journal of big Data, 8(1), 1-34.

Simmons, M., & Lee, J. S. (Julio 2020). Catfishing: A Look into Online Dating and Impersonation. In International Conference on Human-Computer Interaction (pp. 349-358). Springer, Cham.

SonicWall. (2021). SonicWall 2021 Cyber Threat Report.

Spacy. (11 de enero 2022). Respositorio de código en Github de Spacy. https://github.com/explosion/spaCy

Srivalli, & Prasanna, L. (2019). Cyber attacks. International Journal of Engineering and Advanced Technology, 8(6 Special Issue 3), 1934-1936. https://doi.org/10.35940/ijeat.F1372.0986S319

Stajano, F., & Wilson, P. (2011). Understanding scam victims: Seven principles for systems security. Communications of the ACM, 54(3), 70-75. https://doi.org/10.1145/1897852.1897872

Stajano, F., & Wilson, P. (2011). Understanding scam victims: seven principles for systems security. Communications of the ACM, 54(3), 70-75.

The Python Package Index. (14 de enero 2022). Powerful data structures for data analysis, time series, and statistics - Pandas. https://pypi.org/project/pandas/

The Python Package Index. (14 de enero 2022). Pure python spell checker based on work by Peter Norvig - Pyspellchecker. https://pypi.org/project/pyspellchecker/

The Python Package Index. (14 de enero 2022). Python HTTP for Humans - Requests. https://pypi.org/project/requests/

TIBOE. (14 de enero 2022). TIOBE Index for January 2022. https://www.tiobe.com/tiobe-index/

Tweepy. (08 de enero 2022). Tweepy. https://www.tweepy.org

Wirth, R., & Hipp, J. (Abril 2000). CRISP-DM: Towards a standard process model for data mining. In Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining (Vol. 1, pp. 29-39). London, UK: Springer-Verlag.