Universidad de Buenos Aires
Facultad de Ciencias Económicas
Escuela de Estudios de Posgrado

Universidad de Buenos Aires
Facultad de Ciencias Económicas
Escuela de Estudios de Posgrado

MAESTRIA EN CIBERDEFENSA Y CIBERSEGURIDAD

TRABAJO FINAL DE MAESTRÍA

Estudio de aplicaciones de *Machine Learning* en ciberdefensa y ciberseguridad

---

AUTOR: Esp. Cr. Lic. Jorge Alejandro RIOS

DIRECTORA: Dra. Ing. Antonieta Kuz

Junio 2022

Cohorte del Cursante 2018

I. Resumen

El Machine Learning (ML) (Google, 2019) es una rama de la Inteligencia Artificial (IA) cuyo objeto de estudio es la construcción de algoritmos e implementación de software que aprenda de forma autónoma, detectando patrones de comportamiento y relación a partir de grandes volúmenes de datos.

En los últimos años la aplicación de ML se extendió de manera considerable en distintos ámbitos, en el campo de la ciberseguridad se aplica por ejemplo como elemento diferenciador de diferentes soluciones. Numerosas empresas, gobiernos y otros tipos de entes, invierten considerables recursos en la investigación y desarrollo de soluciones que apliquen IA, y en particular ML. Una de las ramas del ML es el Deep Learning (DL) o aprendizaje profundo, el cual utiliza Redes Neuronales Artificiales (RNA) para gestionar su proceso de aprendizaje.

El trabajo final se enfoca en el estudio del Estado del Arte de aplicaciones de ML en ciberdefensa y ciberseguridad, primeramente se realizó una revisión del marco teórico relacionado, para luego analizar la aplicación de ML en soluciones, productos y servicios. Paralelamente se analizó las características particulares de los sistemas software ML, considerando los enfoques ofensivos como defensivos.

Dada la necesidad de los profesionales de la Ciberseguridad de estar actualizados en nuevas herramientas tecnológicas para el desarrollo de su actividad, se plantea el presente trabajo final de maestría como aporte de actualización de aplicaciones de ML en el campo de la ciberseguridad y ciberdefensa.

Palabras Claves: ciberdefensa, ciberseguridad, machine learning, deep learning, aprendizaje automático, inteligencia artificial.

II. Abstract

Machine Learning (ML) (Google, 2019) is a branch of Artificial Intelligence (AI) whose object of study is the construction of algorithms and implementation of software that learns autonomously, detecting behavior patterns and relationships from large volumes of data.

In recent years the application of ML has spread considerably in different areas, in the field of cybersecurity it is applied, for example, as a differentiating element of different solutions. Many companies, governments and other types of entities invest considerable resources in the research and development of solutions that apply AI, and in particular ML. One of the

branches of ML is Deep Learning (DL) or deep learning, which uses Artificial Neural Networks (ANN) to manage its learning process.

The final work focuses on the study of the State of the Art of ML applications in cyber defense and cybersecurity, first a review of the related theoretical framework was carried out, to then analyze the application of ML in solutions, products and services. At the same time, the particular characteristics of ML software systems were analyzed, considering offensive and defensive approaches.

Given the need for Cybersecurity professionals to be updated in new technological tools for the development of their activity, this final master's thesis is proposed as a contribution to update ML applications in the field of cybersecurity and cyberdefense.

Keywords: cyber defense, cybersecurity, machine learning, deep learning, artificial intelligence.

III. Dedicatorias

A Dios que no deja de bendecirme en su infinita misericordia, a mi familia, mi esposa e hija, mi madre, hermanos y amigos que son ejemplos de vida y perseverancia.

IV. Agradecimientos

A la Directora de mi Trabajo Final de Maestría, Dra. Antonieta Kuz, por su guía, acompañamiento y tiempo dedicado.

A mis compañeros de trabajo por compartir sus conocimientos y experiencias profesionales para la realización del trabajo.

A todos los compañeros de la cohorte 2018, a los profesores que fueron y son fuente de inspiración en especial al Dr. Roberto Uzal, al Esp. Ing. Carlos Federico Amaya y la Mg. Adriana Baravalle.

V. Índice General

VI. Índice de Figuras

VII. Índice Tablas

VIII. Abreviaturas y acrónimos.

APT: Amenaza Persistente Avanzada

AWS: Amazon Web Services

AZURE: Conjunto de servicios en la nube de la empresa Microsoft

CSIRT: Computer Security Incident Response Team o Equipo de Respuesta ante Incidencias de Seguridad Informáticas

DARPA: Defense Advanced Research Projects Agency

DDOS: Distributed Denial of Service o denegación de servicio distribuido

DevOps: Combinación de development y operations

DOCKER: Sistema operativo para contenedores

ENISA: European Union Agency for Network and Information Security o Agencia Europea de Seguridad de las Redes y de la Información

FSGM: Fast Gradient Signed Method

GAN: Generative Adversarial Networks

GPU: Graphics processing unit

GCP: Google Cloud Platform

HTTPS: Hypertext Transfer Protocol Secure

IBM: International Business Machines

IDS: Intrusion Detection System o Sistema de detección de intrusiones

INCIBE: Instituto de Nacional Ciberseguridad de España

IoT: Internet of Things o Internet de las Cosas

IP: Internet Protocol o Protocolo de Internet

IPS: Intrusion Prevention System o Sistema de Prevención de Intrusiones

JSMA: Jacobian-based Saliency Map Attack

LAN: Local Area Network o Red de Área Local

L-BFGS Método de Broyden-Fletcher-Goldfarb-Shanno

MAC: Media Access Control

MALWARE: Malicious software.

MOOCs: Massive Open Online Courses o Cursos en línea abiertos y masivos

NGFW: Next Generation Firewall

RDP: Remote Desktop Protocol o Protocolo de Escritorio Remoto

RNA: Redes Neuronales Artificiales

RNP: Red Neuronal Profunda

SDN: Software-defined Networking o Redes Definidas por Software

SOC: Security Operations Center o Centro de Operaciones de Seguridad

SQL: Structured Query Language

SQLi: SQL injection

TI: Tecnologías de la Información

TIC: Tecnologías de la Información y la Comunicación

TPU: Tensor processing unit

SaaS: Software as a Service

URL: Uniform Resource Locator

XSS: Cross-site Scripting

RED TEAMS: Profesionales de seguridad expertos en atacar sistemas y romper defensas

BLUE TEAM: Profesionales de la seguridad que protegen activos críticos de la organización contra cualquier amenaza.

1. Capítulo I. Introducción.

1.1 Descripción del problema.

En un mundo cambiante y digitalizado de continuos avances tecnológicos, la ciberseguridad y ciberdefensa enfrentan nuevos escenarios, uno de ellos es la relación con la IA. Avances en aumento de poder de cómputo, microelectrónica, disponibilidad de contar con grandes cantidades de datos, etc., propiciaron un incremento importante de inversiones para el desarrollo de aplicación de IA en distintos ámbitos. En sectores como industria, agricultura, defensa, seguridad, educación, salud, gobierno, entre otros, se estudia e investiga la aplicación de IA para lograr alguna ventaja competitiva en particular en ML.

Considerando que la IA se aplica en diversos ámbitos, no sería rebuscado considerar que los ciberdelicuentes también la aplican para el diseño de ataques y ciberarmas en un escenario cada vez más complejo.

Ayerbe, A. (2020) señala que detrás de los ciberataques más peligrosos y sofisticados susceptibles de utilizar la IA, pueden encontrarse grupos especializados financiados por Estados, dirigidos hacia infraestructuras críticas de otro país o a generar campañas de desinformación. Analizando a los potenciales atacantes desde su forma de operar, vemos que pueden utilizar IA tanto para explotar vulnerabilidades conocidas como para encontrar otras desconocidas o crearlas.

Desde un enfoque defensivo las técnicas de IA sirven para mejorar la ciberseguridad de sistemas, servicios y productos. En escenarios actuales no basta con sólo defenderse de los ataques, sino que es necesario tener una arquitectura segura y probada con las mismas herramientas que lo harían los atacantes, lo que posibilitaría detectar fallos de seguridad y corregirlos. Aplicaciones defensivas en ciberseguridad la encontramos en áreas como criptología, estenografía, control de acceso, seguridad en redes, entre otras.

Desde un enfoque ofensivo los sistemas de ML son objeto de diferentes tipos de ataques. Los atacantes usan ML para mejorar ataques clásicos o descubrir nuevas variantes. Se pueden distinguir los ataques dirigidos a algoritmos de ML, ataques de sistemas ML contra otros sistemas de ML (Adversarial Machine Learning), ataques mixtos de regresión logística o puertas traseras de ML, entre otros.

Todo este escenario plantea como desafío para los profesionales de la ciberseguridad, el estar preparados para dar respuestas efectivas. Se plantea como problema la dificultad de tener

una actualización continua en aplicaciones de IA en Ciberdefensa y Ciberseguridad utilizando ML.

1.2 Objetivo

El trabajo final tiene como objetivo abordar la problemática relacionada con la Ciberseguridad y Ciberdefensa al aplicar tecnología de ML. Para ello se propone analizar los fundamentos teóricos, el estado del arte, posibilidades y limitaciones, principales aplicaciones tecnológicas implementadas en la faz ofensiva como defensiva.

1.3 Hipótesis

El trabajo final aborda el estado del arte de las principales aplicaciones de ML en el campo de la ciberdefensa y ciberseguridad, tanto de la faz ofensiva como defensiva.

La hipótesis planteada es que la aplicación de IA y en particular ML puede mejorar o poner en riesgo la Ciberseguridad y Ciberdefensa.

Para corroborar la hipótesis se plantean las siguientes preguntas a responder

¿Es necesario fortalecer la formación en IA de profesionales de ciberdefensa y ciberseguridad?

¿El ML puede solucionar problemas reales de ciberdefensa y ciberseguridad?

¿La componente humana marca un factor diferencial?

¿Es útil la aplicación de ML en seguridad defensiva y ofensiva?

1.4 Metodologías y técnicas a utilizar

El tipo de investigación a realizar es exploratoria y descriptiva con análisis cualitativo y proceso deductivo. El tipo de trabajo es de investigación metodológica ya que se plantea realizar una revisión de metodologías y tecnologías de IA en la rama del ML utilizadas para el abordaje y resolución de problemas. Posteriormente a partir de un diagnóstico preliminar se realizó una propuesta de aplicación metodológica con una aplicación tecnológica, a través de análisis de un caso de estudio.

2. Capítulo II - Marco Teórico.

2.1 Generalidades de la Inteligencia Artificial.

La IA contribuye con la Ciberseguridad posibilitando mejorar la resiliencia de los sistemas y productos que utilizan empresas, gobiernos y la sociedad en general. En la actualidad los avances en diferentes campos posibilitan nuevas aplicaciones de la IA, esto propiciado por el mayor conocimiento del funcionamiento del cerebro, avances en la microelectrónica, aumento de la potencia de cómputo, disponibilidad de gran cantidad de datos, la conexión ubicua entre sistemas, entre otros.

Según Pacheco, V.G. (2019) algunos hitos históricos relacionados con aplicaciones de IA que podemos señalar son:

Nacimiento [1952 – 1956]

1950 — Alan Turing crea el *Test de Turing* para determinar si una máquina era realmente inteligente. Para superar el *test*, una máquina tenía que ser capaz de engañar a un humano haciéndole creer que era humana.

1952 — Arthur Samuel escribe el primer programa de computadora capaz de aprender, el software jugaba a las damas y mejoraba su juego partida tras partida.

1956 — Martin Minsky y John McCarthy con la ayuda de Claude Shannon y Nathan Rochester, organizan la conferencia de Darthmouth de 1956, considerada como el evento donde nace el campo de la IA.

1958 — Frank Rosenblatt diseña el Perceptrón, la primera red neuronal artificial (RNA).

Primer Invierno de la IA (AI-Winter) – [1956 – 1980]- Diferentes agencias que financian la investigación en IA cortan los fondos tras numerosos años de altas expectativas y muy pocos avances.

1967 — Se escribe el algoritmo *Nearest Neighbor,* este hito está considerado como el nacimiento al campo del reconocimiento de patrones (pattern recognition) en computadores.

1979 — Estudiantes de la Universidad de Stanford inventan el Stanford *Cart*, un robot móvil capaz de moverse autónomamente por una habitación evitando obstáculos.

La explosión de los 80 [1980 – 1987]- Los años 80 estuvieron marcados por el nacimiento de los sistemas expertos que fueron utilizados en el sector corporativo, esto generó un nuevo interés en la IA.

1981 — Gerald Dejong introduce el concepto *Explanation Based Learning* (EBL), donde un computador analiza datos de entrenamiento y crea reglas generales que le permiten descartar los datos menos importantes.

1985 — Terry Sejnowski inventa *NetTalk* un software que aprende a pronunciar palabras de la misma manera que lo haría un niño.

Segundo AI Winter [1987 – 1993]- La IA a finales de los 80 entró en un nuevo invierno y la reputación del campo no se recuperó hasta inicios del 2000.

1990s — Se potencia el campo de la IA con ML, se gira desde un enfoque orientado al conocimiento hacia uno orientado el dato. Los investigadores crean programas que analizan grandes cantidades de datos y extraen conclusiones de los resultados.

1997 — El ordenador *Deep Blue*, de IBM vence al campeón mundial de ajedrez Gary Kaspárov.

Explosión y adopción comercial [2006-Actualidad]- El aumento de la potencia de cómputo sumado a la disponibilidad de grandes volúmenes de datos volvió a poner el foco en la IA, en especial en el campo del ML. Numerosas empresas incorporan ML en sus procesos, productos y servicios para obtener ventajas competitivas.

2006 — Geoffrey Hinton acuña el término Aprendizaje Profundo (DL), es parte de ML donde se utilizan arquitecturas de Redes Neuronales profundas (RNP) que posibilitan mejorar el aprendizaje.

2011 — El ordenador *Watson* de IBM vence a sus competidores humanos en el concurso Jeopardy, que consiste en contestar preguntas formuladas en lenguaje natural.

2012 — Jeff Dean de Google, con la ayuda de Andrew Ng (Universidad de Stanford), lideran el proyecto *GoogleBrain*, que desarrolla una Red Neuronal Profunda utilizando toda la capacidad de la infraestructura de Google para detectar patrones en vídeos e imágenes.

2012 — Geoffrey Hinton lidera el equipo ganador del concurso de Visión por Computador a Imagenet utilizando una Red Neuronal Profunda (RNP). El equipo venció por un amplio margen de diferencia, dando nacimiento a la actual explosión de ML basado en RNPs.

2012 — El laboratorio de investigación Google X utiliza *GoogleBrain* para analizar autónomamente vídeos de *Youtube* y detectar aquellos que contienen imágenes de gatos.

2014 — Facebook desarrolla *DeepFace*, un sistema ML basado en RNPs que es capaz de reconocer a personas con la misma precisión que un ser humano.

2014 — Google compra *DeepMind*, una *startup* inglesa de DL que recientemente había demostrado las capacidades de las RNP con un algoritmo capaz de jugar a juegos de *Atari* simplemente viendo los píxeles de la pantalla, tal y como lo haría una persona. El sistema ML tenía un algoritmo que con unas horas de entrenamiento era capaz de vencer a humanos expertos en algunos de esos juegos.

2015 — Amazon lanza su propia plataforma de ML.

2015 — Microsoft crea el *Distributed Machine Learning Toolkit*, que permite la distribución eficiente de problemas de machine learning en múltiples computadores.

2015 — Elon Musk y Sam Altman, entre otros, fundan la organización *OpenAI*, dotándola con un presupuesto de 1.000 millones de dólares, cuyo objetivo es asegurar que el desarrollo de la IA tenga un impacto positivo en la humanidad.

2016 – *Google DeepMind* vence en el juego *Go*, que es uno de los juegos de mesa considerado de los más complicados, al jugador profesional Lee Sedol por 5 partidas a 1. Jugadores expertos de *Go* afirman que el sistema ML fue capaz de realizar movimientos creativos que no se habían visto hasta el momento. Ese mismo año usando tecnología de movilidad autónoma, el MIT lanza su primer taxi autónomo en Singapur, a la vez que *Uber* opera con vehículos autónomos en Pittsburg y San Francisco.

2017– El programa Libratus vence al póker en un juego contra cuatro personas. Ese año *DeepMind* logra trasladar lo aprendido en un juego a otro. También ese año la ginoide *Sophia* se convierte en ciudadana saudí.

2018 – *Alpha Zero* aprende por si misma a jugar al ajedrez. También el robot de rescate bípedo Atlas de la Boston Dynamics aprende a caminar por terrenos escarpados, saltar y desplazarse esquivando obstáculos.

En la actualidad estamos en un nuevo verano de la IA, donde sus avances están encontrando aplicabilidad en diferentes áreas, hasta el punto de crear mercados enteros y producir cambios significativos en la estrategia de empresas y estados. En la vida cotidiana usamos tecnología de IA en diferentes situaciones, por ejemplo, cuando empleamos asistentes personales Cortana o *Google Now*, cuando *Amazon* recomienda productos de forma personalizada, cuando en una búsqueda en internet usamos *Google Translator* para traducir páginas enteras, cuando desbloqueamos nuestro teléfono con la huella digital o por reconocimiento facial, entre otros ejemplos.

En referencia a la situación actual de la Inteligencia Artificial, Gallegos y Soto (2014) expresa:

"Actualmente la Inteligencia Artificial es un área de la ciencia de gran interés por ser un área multidisciplinaria donde se realizan sistemas que tratan de hacer tareas y resolver problemas como lo hace un humano, así mismo se trata de simular de manera articula las formas del pensamiento y como trabaja el cerebro para tomar decisiones."

En muchos sectores se está expandiendo la aplicación de la inteligencia IA para obtener alguna ventaja competitiva, en particular en la rama de ML. En la figura 1 se observan diferentes campos de aplicación.

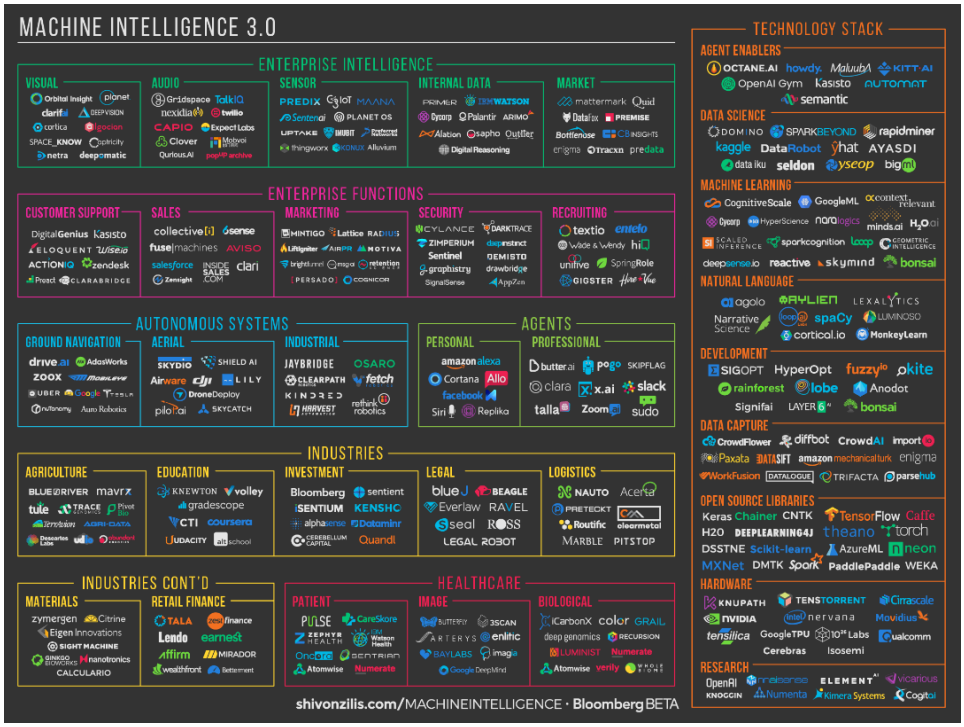

Figura 1: ML 3.0.

Shivon Zilis, S.(0). Inteligencia de la máquina. [Figura]. Recuperado de: http://www.shivonzilis.com/

Los objetivos principales de la IA son: el logro de la capacidad de deducción y razonamiento, procesamiento de lenguaje natural, capacidad de planificación, aprendizaje, percepción y capacidad de manipular cosas. A largo plazo se busca alcanzar la creatividad, la inteligencia social y la inteligencia general. Al presentarse un campo de aplicación cada vez más amplio, existe el temor de que la IA desarrolle capacidades que el ser humano no pueda controlar.

En ese sentido el término "Singularidad" refiere al momento donde un ente artificial sea capaz de resolver un problema de mejor forma que un ser humano. Al respecto el Instituto del futuro de la Universidad de Oxford (Owain, Stuhlmüllery, Cundy, Careyy & Zachary, 2018), en uno de sus estudios proyectivos del año 2018 considera que en un plazo de cuarenta años muchas tareas que realiza el ser humano la resolverá de mejor forma un ente artificial. Luego en ciento veinte años la mayoría de las tareas que realizan los seres humanos la podrán desarrollar dispositivos inteligentes.

Por otro lado la IA es considerada como una tecnología exponencial junto con la informática, biotecnología, nanotecnología, neurociencias y robótica. El término Tecnologías exponenciales (BBVA, 2020) surgió con el avance del poder de cómputo y refiere a las disciplinas que se utilizan para brindar soluciones a grandes problemas que afectan a la humanidad.

Ejemplo de desafios actuales de la ciberseguridad son: detección de intrusiones, protección de la privacidad, defensa proactiva, identificación de comportamientos anómalos, detección de amenazas sofisticadas y cambiantes, entre otros. En ese sentido se están explorando métodos basados en la IA que faciliten el análisis y la toma de decisiones en tiempo real para una rápida detección y reacción ante ciberataques.

En la medida que los atacantes incorporen técnicas de IA a sus actividades, sus actividades serán más difíciles de detectar y tendrán un mayor éxito de causar daño, según un informe de la Agencia de la Unión Europea para la Ciberseguridad ENISA.

Por otro lado el relacionar los campos de la robótica y sistemas inteligentes, dio lugar a la Robótica Inteligente (RI). La misma tiene como objeto de estudio el diseño y producción de robots con un elevado grado de autonomía en sus funciones, pudiendo actuar en entornos cambiantes, y de aplicación a sistemas de armas letales autónomos. Debido a ello su empleo en aplicaciones para la defensa y la seguridad está experimentando un notable incremento, donde Fuerzas Armadas de países avanzados invierten cuantiosos recursos para la obtención de sistemas que incorporan IA y RI para uso militar, por ejemplo, en sistemas de mando y control, inteligencia, sistemas de armas o equipo individual del combatiente.

2.2 Generalidades de Aprendizaje automático o ML

Analizamos algunas de distintos referentes y empresas reconocidas:

Andrew Ng, Profesor de la Universidad de Stanford, define el ML como la ciencia de hacer que las computadoras actúen sin estar explícitamente programadas. (Andrew, 2018).

El equipo de expertos de Nvidia, definen el ML como la práctica de usar algoritmos para analizar datos, aprender de ellos y luego hacer una determinación o predicción sobre algo en el mundo.

Tom Mitchell, profesor del departamento de ML de la Universidad Carnegie Mellon, propone que el campo del ML busca responder a la pregunta ¿cómo podemos construir sistemas informáticos que mejoren automáticamente con la experiencia, y cuáles son las leyes fundamentales que rigen todos los procesos de aprendizaje?

ML permite que los sistemas mejoren automáticamente su desempeño en una tarea observando datos relevantes. El aprendizaje automático fue el principal contribuyente al aumento de la IA en las últimas décadas, desde motores de búsqueda y recomendación de productos hasta sistemas para reconocimiento de voz, detección de fraude, comprensión de imágenes e innumerables otras tareas que alguna vez se basaron en la habilidad y el juicio humano. La automatización de estas tareas ha permitido la ampliación de servicios como el comercio electrónico. (Artificial Intelligence and Life in 2030, 2016, p.52).

Un informe reciente "*The 2021 AI Index: Major Growth Despite the Pandemic*", producido por el grupo de expertos en IA de la Universidad de Stanford del HAI (*Human Centered Artificial Intelligence*), destaca que la inversión privada en IA aumentó significativamente a pesar de que la crisis de COVID impactó negativamente en las economías. Refiere que luego de que Canadá publicó una estrategia nacional de IA en 2017, muchos otros países se embarcaron en proyectos similares. Menciona el éxodo de investigadores académicos al sector corporativo, favoreciendo que las corporaciones dominen herramientas que antes se usaban en entornos de investigación de IA. Según el informe los desafíos éticos de las aplicaciones de IA conforman un foco importante para la comunidad de IA evidenciado por el aumento significativo de artículos que hacen mención a la "ética" como palabra clave.

En ML se utilizan modelos con algoritmos que aprenden a realizar predicciones basadas en grandes cantidades de datos, los algoritmos son entrenados para detectar patrones. Una dificultad que se presenta es la dificultad de obtención de datos para entrenar los modelos.

En el paradigma lógico de programación en su enfoque clásico los expertos humanos ingresan reglas que forman la base de conocimiento y datos son procesados según esas reglas para obtener las respuestas esperadas. Por el contrario, en ML se puede ingresar como entrada los datos además de las respuestas esperadas y las salidas serían las reglas, estas se aplican con posterioridad a nuevos datos para producir respuestas originales. DL o aprendizaje profundo es una

especialización del ML, en este trabajo de forma posterior se analizan en detalles sus características.

A continuación, desarrollamos conceptos principales de ML.

2.2.1 Tipos de Aprendizajes en ML

Según el tipo de aprendizaje los podemos clasificar en tres tipos:

2.2.1.1 Aprendizaje Supervisado.

En los problemas de aprendizaje supervisado se busca inferir una función a partir de datos de entrenamiento etiquetados considerando sus entradas y salidas, para esos datos se conoce cuál es el valor esperado o respuesta correcta. Un ejemplo para entender este tipo de aprendizaje es el caso de un estudiante que rinde un examen, el estudiante marca las respuestas que cree correctas y al finalizar recibe la devolución del profesor con la corrección. La retroalimentación le indica las respuestas acertadas y esto permite al estudiante aprender de sus errores para mejorar su desempeño.

Con el aprendizaje supervisado se crean modelos que combinan entradas y hacen predicciones incluso de datos que no fueron vistos, cuando entrenamos ese modelo le asignamos etiquetas. Se plantea como ejemplo de aplicación de este tipo de aprendizaje es en la determinación si un correo electrónico es *Spam* o no.

Las variables de entrada conocida como atributos describen los datos. Siguiendo el ejemplo del *Spam,* los atributos que se pueden extraer de un correo electrónico serían: texto del mensaje, dirección de envío y recepción, datos de enrutamiento, encabezado, etc. Un proyecto de ML simple podría usar un solo atributo, mientras que otro más complejo podría tener muchos atributos.

Un modelo define la relación entre los atributos y la etiqueta. Continuando con el ejemplo en un modelo de detección de *Spam* se pueden asociar determinados atributos con S (*Spam*). En el modelo se distingue la fase de entrenamiento e inferencia, en la primera se busca la creación o el aprendizaje del modelo y para ello se le muestra ejemplos etiquetados y este aprende gradualmente entre las relaciones entre atributos y la etiqueta. La inferencia significa aplicar el modelo entrenado a ejemplos sin etiqueta, es decir se utiliza el modelo entrenado para obtener predicciones útiles.

Dentro del aprendizaje supervisado se encuentran los problemas de regresión que abarcan los casos cuando los valores a predecir son numéricos. También están los problemas de clasifica-

ción que consideran los casos cuando el valor a predecir es una etiqueta o una cantidad pequeña de valores.

2.2.1.2 Aprendizaje no supervisado.

Permite abordar problemas donde se desconoce el tipo de resultado que se espera o se conoce muy poco sobre la salida, el conjunto de datos no tiene ninguna etiqueta. A diferencia del aprendizaje supervisado no se tiene ningún tipo de retroalimentación basada en resultados de las predicciones realizadas. No hay una variable objetivo y se busca patrones. El método que utilizan los bebes para aprender a hablar es un ejemplo de este tipo de aprendizaje, en principio escuchan hablar a los padres y no entienden nada. Pero a medida que escuchan miles de conversaciones su cerebro comienza a formar un modelo de cómo funciona el lenguaje y reconoce patrones al esperar ciertos sonidos.

Con estos modelos es común realizar agrupamientos en los datos conocido como *Clustering*, esto para encontrar patrones en las entradas o para reducir la cantidad de variables que se manejan en los modelos.

2.2.1.3 Aprendizaje por reforzamiento.

En los problemas de este tipo de aprendizaje el algoritmo aprende observando el mundo que lo rodea. No se indica la acción correcta pero la acción elegida tiene consecuencias, los algoritmos tienen una función de recompensa que indica cuándo lo esté haciendo bien.

Un ejemplo de este tipo de aprendizaje es la forma en la que aprenden los perros. Ante una buena acción percibida por sus dueños reciben una recompensa y ante una mala acción reciben un castigo. Así se asocia a cada acción una recompensa o castigo y esto condiciona su futura toma de decisiones.

2.2.1.4 Sobreajuste.

Es la tendencia a ajustarse a unas características muy específicas de los datos de entrenamiento, que no tienen relación causal con la función objetivo que estamos buscando generalizar. Existen estrategias para minimizar el sobreajuste, una de ellas conocida como “retención” que consiste en dividir nuestro conjunto de datos en uno o varios conjuntos de entrenamientos y uno o varios conjuntos de evaluación. Otra estrategia es la validación cruzada que considera realizar un análisis estadístico para obtener otras medidas del rendimiento estimado, como la media y la varianza para de esta forma entender la variación del rendimiento a traves de los distintos conjuntos de datos.

2.2.2 Algoritmos de ML.

Para el desarrollo del trabajo se revisaron los siguientes algoritmos más usados en ML: regresión lineal, regresión logística, árboles de decisión, Random Forest, SVM o máquina de soporte vectorial, KNN o K vecinos más cercanos, K-means.

Para profundizar la compresión de los algoritmos propuestos se implementaron prototipos utilizando el paquete ANACONDA, la herramienta JUPITER Notebook con el lenguaje Python. Para tratar los Dataset de ejemplos se utilizaron librerías como Scikit-learn, Statsmodels, PyMC.

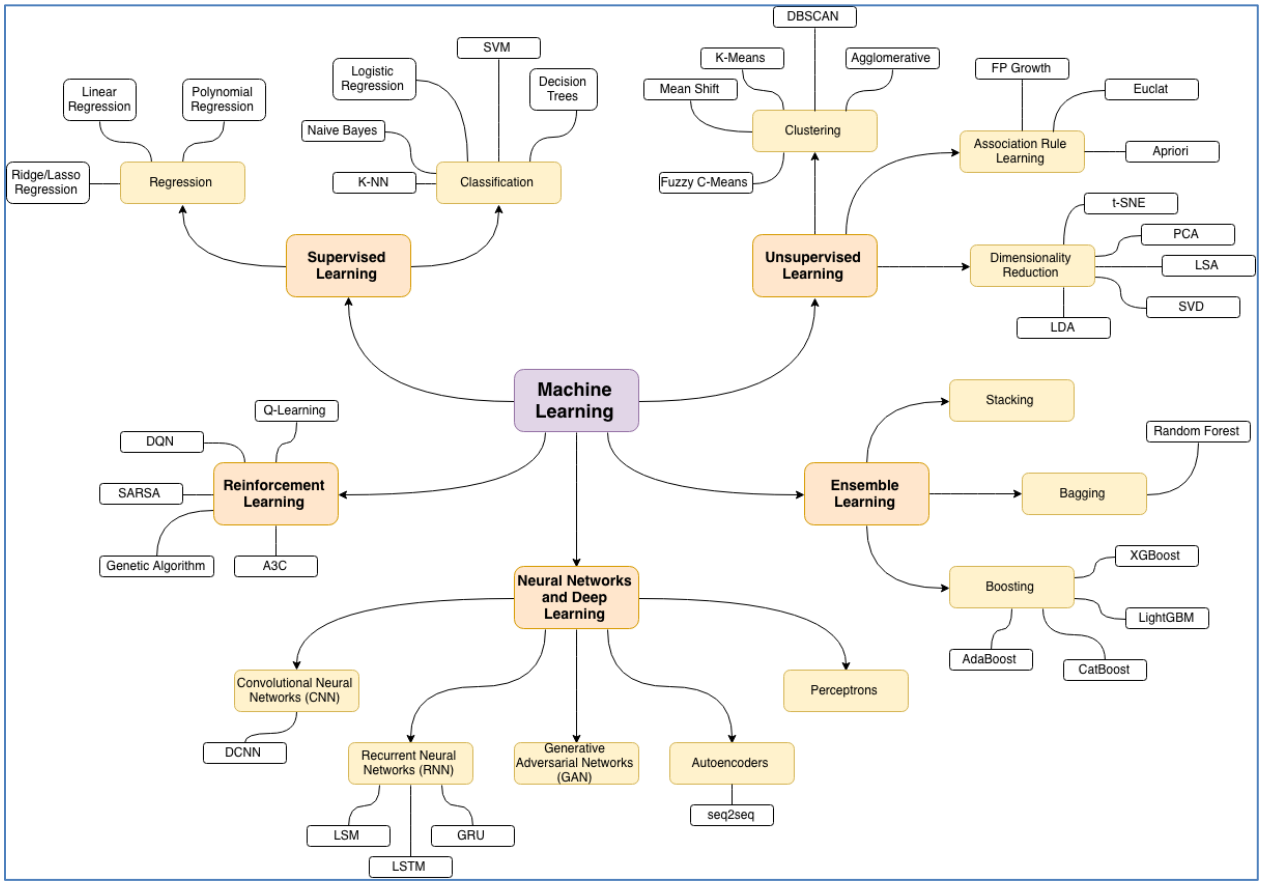

Figura 2: Algoritmos de ML

Fuente: GitHub (2022). Red Neuronal (NN): Mapa de Aprendizaje Automático. [Figura]. Recuperado de: https://github.com/trekhleb/homemade-machine-learning

2.2.2.1 Regresión lineal.

Se utiliza para estimar los valores reales (Por ejemplo, costo de las viviendas, el número de llamadas, ventas totales) basados en variables continuas. Se busca establecer la relación entre las variables independientes y dependientes ajustando una mejor línea recta con respecto a los

puntos dados. Esta línea de mejor ajuste se conoce como línea de regresión y está representada por una ecuación lineal. En las Figura 3,4 y 5, se muestra un ejemplo de implementación de Regresión Lineal en lenguaje *Phyton* utilizando el entorno de desarrollo (Ide) *Jupyter Nootebook*.

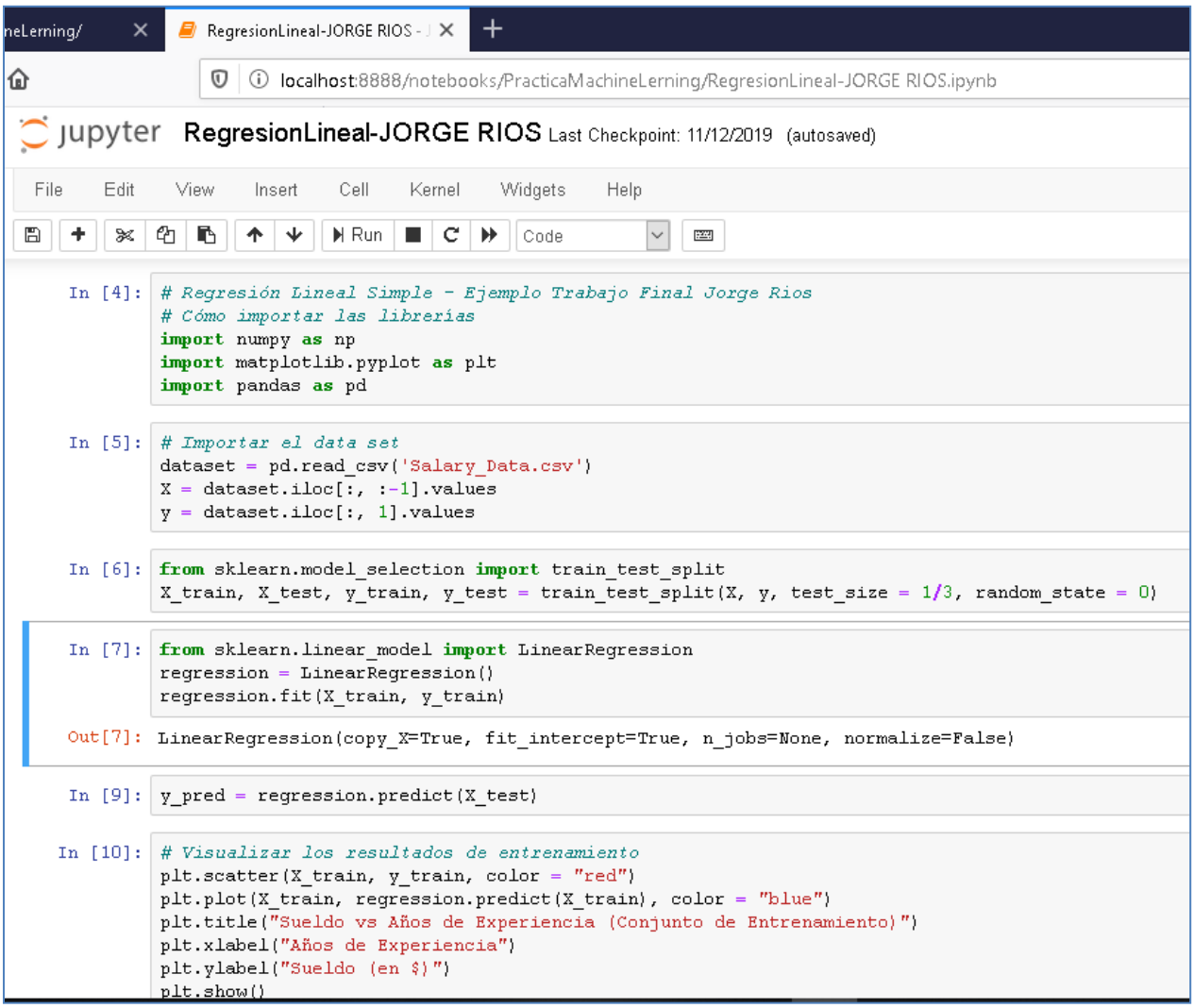

Figura 3: Implementación Regresión Lineal Simple.

Imagen Propia (2022).

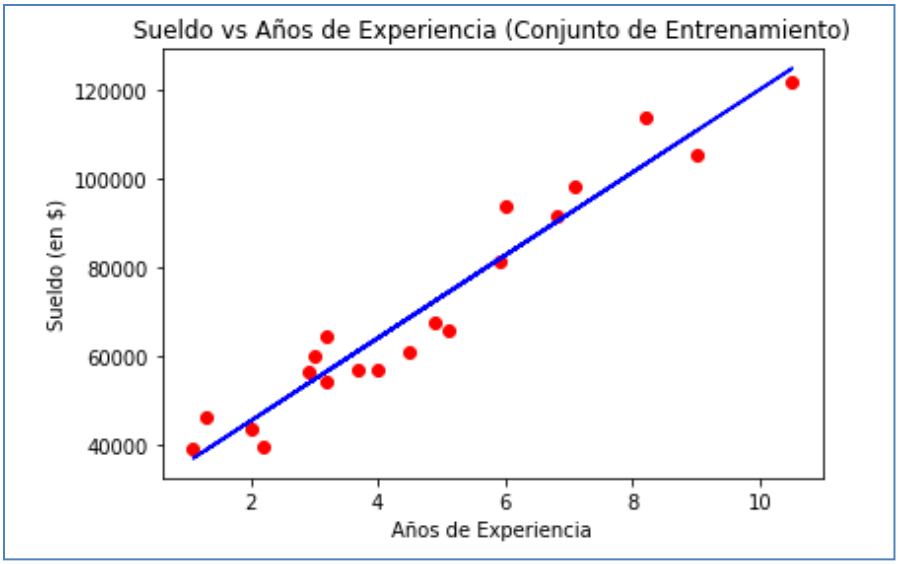

Figura 4: Entrenamiento del algoritmo de Regresión lineal.

Imagen Propia (2022).

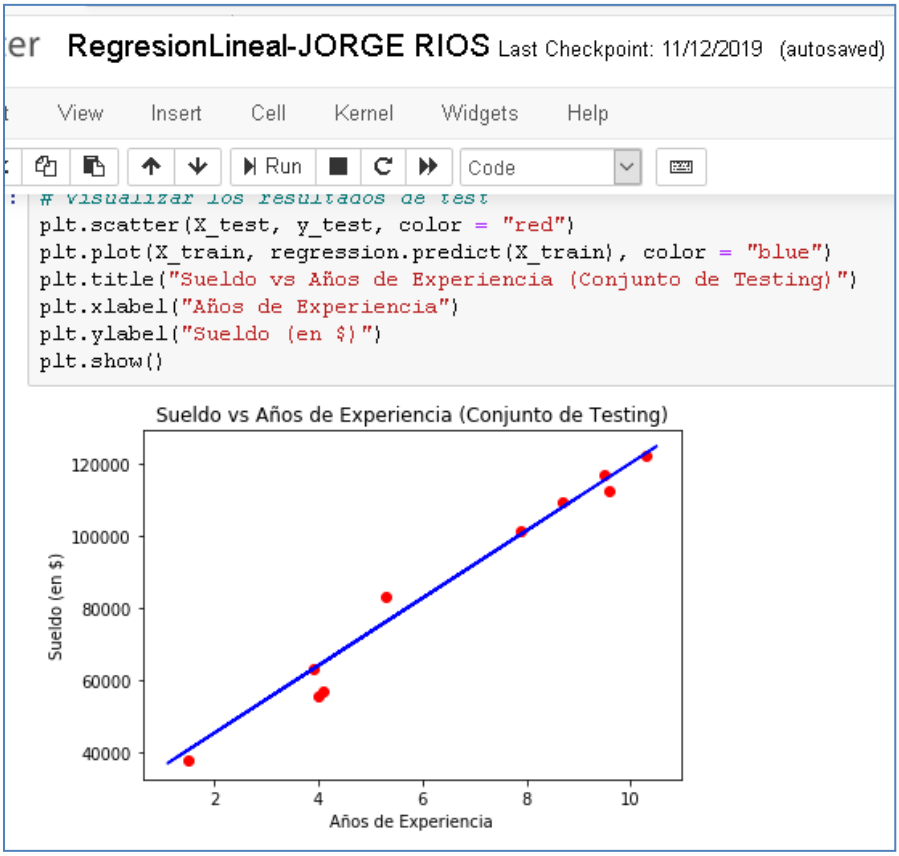

Figura 5: Prueba del Algoritmo Regresión Lineal.

Imagen Propia (2022).

En el ejemplo propuesto se visualiza que el desvío del modelo es pequeño, por lo que sus resultados son bastante confiables.

2.2.2.2 Regresión logística.

Los modelos lineales pueden ser utilizados para clasificaciones; es decir, que primero ajustamos el modelo lineal a la probabilidad de que una cierta clase o categoría ocurra y, a luego, utilizamos una función para crear un umbral en el cual especificamos el resultado de una de estas clases o categorías. La función que utiliza este modelo es la función logística.

2.2.2.3 Árboles de decisión.

Los Arboles de Decisión son diagramas con construcciones lógicas, muy similares a los sistemas de predicción basados en reglas. Sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. Los Arboles de Decisión están compuestos por nodos interiores, nodos terminales y ramas que salen de los nodos interiores. Cada nodo interior en el árbol contiene una prueba de un atributo y cada rama representa un valor distinto del atributo. Siguiendo las ramas desde el nodo raíz hacia abajo, cada ruta finalmente termina en un nodo terminal creando una segmentación de los datos.

2.2.2.4 Random Forest.

Trabaja construyendo una gran cantidad de árboles. Teniendo en cuenta que un sencillo clasificador entrenado podría tener sólo el 60 por ciento de precisión, podemos entrenar muchos clasificadores que sean por lo general acertados y luego podemos utilizar el conocimiento de todos juntos.

2.2.2.5 SVM o máquina de soporte vectorial.

Se propone encontrar un plano que separe los grupos dentro de los datos de la mejor forma posible. La separación significa que la elección del plano maximiza el margen entre los puntos más cercanos en el plano, estos puntos se denominan vectores de soporte.

2.2.2.6 K-means.

Es uno de los algoritmos de agrupamiento más conocidos y una de las técnicas de aprendizaje no supervisado más conocidas. Su funcionamiento busca reducir al mínimo la suma de las distancias cuadradas desde la media dentro del agrupamiento. Para hacer esto establece primero un número previamente especificado de conglomerados, K, y luego va asignando cada observación a la agrupación más cercana de acuerdo a su media.

2.2.2.7 KNN o k vecinos más cercanos.

Método de clasificación no paramétrico que estima el valor de la probabilidad a posteriori de que un elemento x pertenezca a una clase en particular a partir de la información proporcionada por el conjunto de prototipos. La regresión KNN se calcula simplemente tomando el promedio del punto k más cercano al punto que se está probando.

2.3 Generalidades del Aprendizaje Profundo o DL

DL es un campo del ML que se caracteriza por utilizar Redes Neuronales Artificiales (RNA) para conseguir el aprendizaje. El término profundo refiere a la cantidad de capas de representaciones que se utilizan en los modelos, en muchos problemas, DL ofrece mejor rendimiento que soluciones de ML tradicionales. DL automatiza uno de los pasos más complicados del ML que es la "Ingeniería de Atributos", que consiste en que antes de entrenar un modelo primero se deben refinar las entradas para adaptarlas al tipo de transformación, seleccionando los atributos más representativos.

2.3.1 Redes Neuronales Artificiales (RNA).

Son modelos computacionales basados en un gran conjunto de unidades neuronales simples, las cuales imitan el comportamiento de los axones en las neuronas de cerebros humanos.

Su funcionamiento es en capas y mientras más tenga, más profundo es el aprendizaje. El mayor número de capaz implica que el entrenamiento es más costoso requiriendo elevados volúmenes de datos. En la figura 6 se muestra la forma de una neurona simple con sus partes:

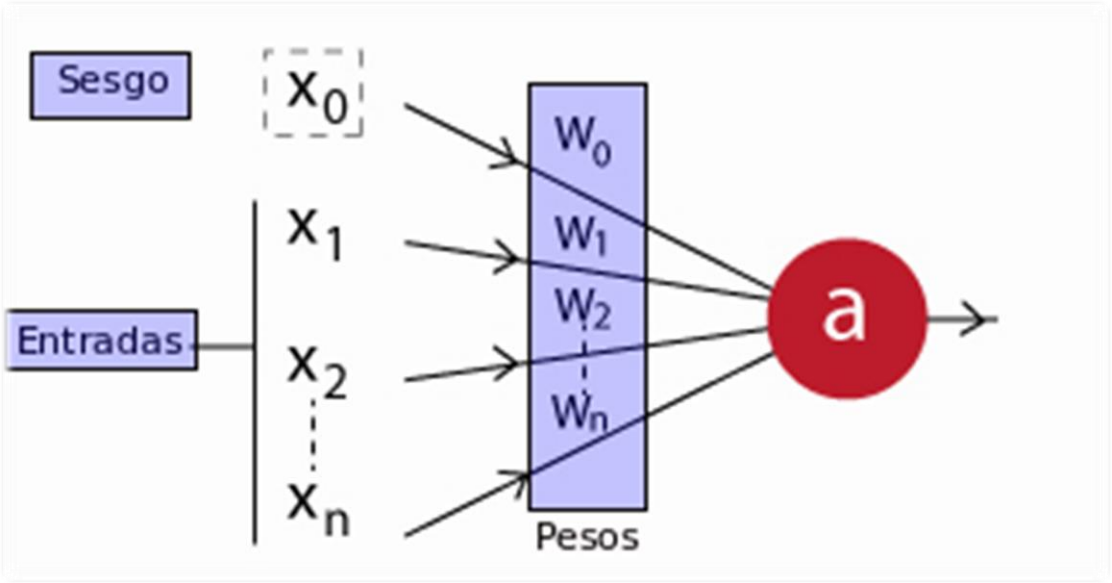

Figura 6: Forma de una neurona artificial simple

Recuperado de: http://iartificial.net

Componentes de la neurona simple:

- $x_1, x_2, \ldots, x_n$: Son los datos de entrada de una neurona, puede ocurrir que estas entradas sean las salidas de otra neurona de la red.
- x_0: Es la unidad de sesgo que consiste en un valor constante que se le suma a la entrada de la función de activación de la neurona. El valor permite cambiar la función de activación hacia la derecha o izquierda, lo que permite a la neurona flexibilidad para aprender. Generalmente tiene el valor 1 y también tiene asociado un peso.
- $w_0, w_1, \ldots, w_n$: Los pesos relativos de cada entrada
- a: Es la salida de la neurona cuya fórmula de cálculo es la siguiente:

$$a = f\left(\sum_{i=0}^{n} w_i \cdot x_i\right)$$

f es la función de activación de la neurona y es la que le otorga flexibilidad a las redes neuronales que le permite estimar complejas relaciones no lineales en los datos. Puede ser una función lineal, una función logística, hiperbólica, entre otras.

En las RNA las neuronas simples están conectadas con muchas otras, los enlaces que se establecen entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adya-

centes. Las RNA aprenden y se forman de manera autónoma y son útiles en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional.

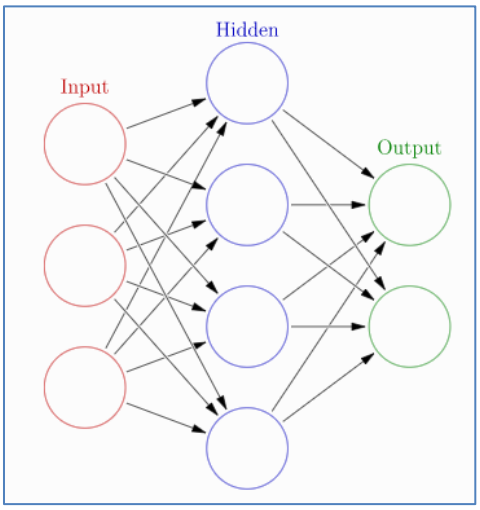

Figura 7: Red Neuronal artificial (RNA)

Fuente: http://developer.ibm.com

El algoritmo de propagación hacia atrás o *backpropagation* permite que la RNA aprender, este funciona mediante la determinación de la pérdida o error (función pérdida) en la salida que luego se propaga hacia atrás en la red ajustando los pesos para minimizar el error resultante de cada neurona. La implementación de *backpropagation* implica cálculos realmente complicados y la transición desde funciones de activación como tanh o sigmoid a funciones como ReLU o SELU simplifica los problemas.

2.3.2 Funcionamiento del DL.

En general las técnicas de ML tratan de realizar la asignación de entradas a salidas objetivo, esto mediante la observación de un gran número de ejemplos de entradas y salidas. En DL se realiza este mapeo de entrada a salidas objetivo por medio de una RNA formada por un gran número de capas organizadas de forma jerárquica. Al comienzo la red aprende algo simple en la capa inicial de la jerarquía y envía esa información a la capa siguiente. La capa siguiente toma esa información, la combina en algo que es un poco más complejo y lo pasa a la siguiente capa. El proceso de aprendizaje por utilizando los datos de ejemplo sigue y cada capa de la jerarquía construye algo más complejo de la entrada que recibió.

Los pesos de cada capa en el primer paso se asignan de forma aleatoria, por ello la salida del modelo no es muy buena teniendo un valor de la función de pérdida alto. Para conseguir un

modelo entrenado se procesan gran cantidad de nuevos casos, esto permiten que los pesos se ajusten de tal forma de reducir el valor de la función de pérdida.

En el cerebro humano las neuronas están conectadas a muchas otras por sinapsis y a medida que se interactúa con el entorno se crean nuevas conexiones o modifican las existentes. Por otro lado, existen y se desarrollan regiones que se especializan en el procesamiento de determinadas entradas como, por ejemplo: visión, audición, lenguaje, entre otras. En la RNA dependiendo del tipo de entradas que contemos se cuenta con diferentes arquitecturas a utilizar.

2.3.3 Arquitecturas de Redes Neuronales.

Algunos de los tipos de arquitecturas de RNA son: Prealimentadas (*Perceptron*, *Perceptron* Multicapa), Recurrentes, Convolucionales, *Generative Adversarial Networks* (muy usadas en seguridad), *Resnet*, *inception*. El campo de los tipos de RNA se va actualizando continuamente lo que aumenta la flexibilidad de los modelos.

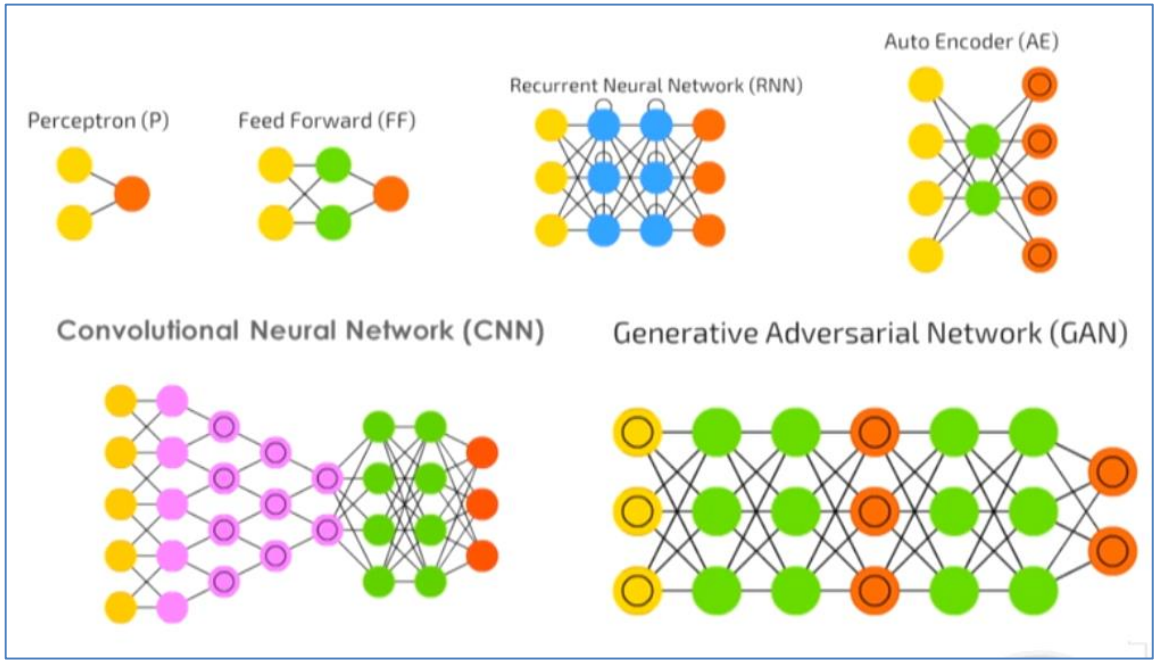

Figura 8: Algunas arquitecturas de RNA

Fuente: Van Veen F. (2016). El zoológico de la red neuronal. [Figura]. Recuperado de: http://asimovinstitute.org/neural-network-z

2.3.3.1 Redes neuronales prealimentadas

Las RNA prealimentadas consisten en un modelo sencillo donde la información se mueve hacia adelante. El perceptron y perceptron multicapa son ejemplo de este tipo de arquitectura se utilizan para problemas de clasificación simple.

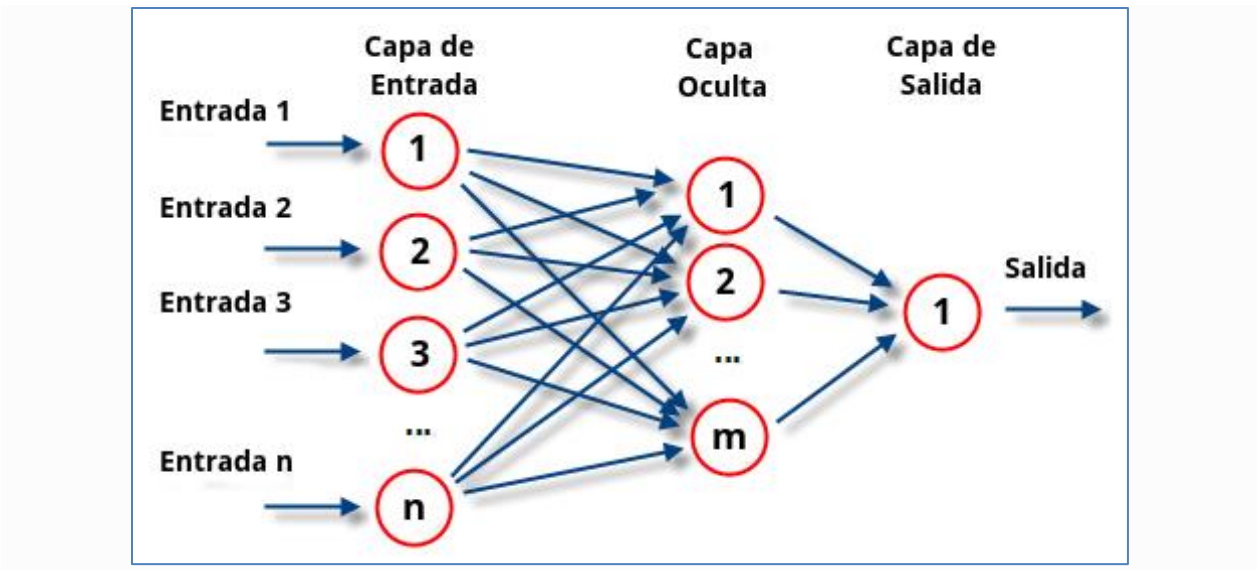

Figura 9: Perceptron Multicapa

Recuperado de: http://agriainnovation.com

2.3.3.2 Redes neuronales convolucionales.

Similares a la *perceptron* multicapa, se componen de neuronas con pesos y sesgos que pueden aprender. Cada neurona recibe entradas, realiza un producto escalar y luego aplica una función de activación, también consta una función de pérdida o costo sobre la última capa totalmente conectada. La diferencia con las RNA convolucionales es que suponen como entradas imágenes, lo que permite codificar ciertas propiedades en la arquitectura para mejorar eficiencia y reducir la cantidad de parámetros en la red. La estructura está formada por tres tipos distintos de capas:

- Capa convolucional.
- Capa de reducción o de *pooling*, que reduce la cantidad de parámetros al quedarse con las características más comunes.
- Capa clasificadora totalmente conectada, la cual proporciona el resultado final de la red.

2.3.3.3 Redes neuronales recurrentes.

Una limitación importante de las RNA prealimentadas es que no cuentan con mecanismos capaces de emular la persistencia del pensamiento, ya que los seres humanos no empiezan el pensamiento desde cero cada segundo. Las RNA recurrentes abordan este problema y cuentan con bucles de retroalimentación que permiten la persistencia de la información.

La RNA recurrente se puede pensar como una red con múltiples copias de ella misma, donde cada copia pasa un mensaje a su sucesor. Implementaciones exitosas con este tipo de RNA se aplicaron a problemas de: reconocimiento de voz, traducción, subtítulos de imágenes, modelado de lenguaje, entre otros.

Las redes de memoria de largo plazo a corto plazo (LSTMs) son un ejemplo de este tipo de red y son capaces de aprender dependencias a largo plazo. Cuentan con estructura como cadena, pero el módulo de repetición tiene una estructura diferente, ya que consta de cuatro capas que interactúan posibilitando tener una memoria a más largo plazo.

2.3.3.4 Redes Neuronales Generativas Adversarias (GAN)

Los modelos generativos están basados en redes neuronales profundas de DL. Las redes neuronales reciben datos de entrada y produce resultados como salida.

Para su funcionamiento se usan dos redes neuronales profundas, donde estas dos redes son adversarias (Discriminador y Generador). Para explicar su funcionamiento se recurre al ejemplo donde se quiere generar fotos de caras de personas. La tarea del Discriminador será decir si una cara es falsa o auténtica, mientras que la tarea del Generador será la de crear las fotos de cara que parezcan auténticas.

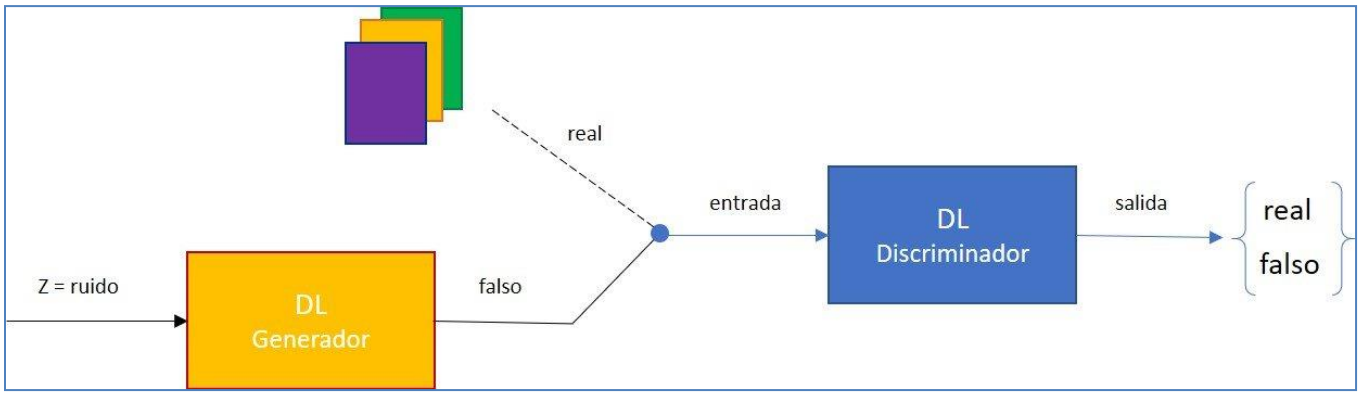

Figura 10: Redes GAN

Recuperado de: http://www.iartificial.net

En el ejemplo de las fotos de cara, para entrenar el Discriminador y que pueda determinar si una foto es real o no, se precisa un conjunto de fotos de personas reales. El Generador crea caras distintas y para ello utiliza un vector aleatorio Z distinto en cada generación. A medida que el generador mejora en la producción de fotos de caras, la tarea discriminadora se vuelve más difícil. El discriminador ayuda al generador a crear cada vez mejores imágenes y tanto el genera-

dor como el discriminador van mejorando simultáneamente hasta el punto que se complica distinguir las imágenes generadas de las reales.

Las redes GAN presentan distintos problemas, como de convergencia, de calidad, de colapso modal entre otros. Para solucionarlos se utilizan distintas técnicas para mejorar su rendimiento que son propuestas por la comunidad científica y están en continua evolución. Aunque nacieron en 2014, a la fecha el crecimiento de investigaciones relacionadas con GAN crecen en forma exponencial.

2.4 Ciclo de vida clásico de ML

Analizando el funcionamiento de los sistemas expertos (SE), podemos apreciar que manipulan conocimiento de expertos humanos acerca de un dominio delimitado y restringido, el cual se incorpora en la base de conocimientos en forma de reglas. El SE cuenta además con una base de datos y mecanismos de inferencia con el cual brinda respuestas de la misma forma que lo haría un experto humano. El enfoque considerado en las SE sigue una visión tradicional, de introducir reglas en la base de conocimiento, y luego con los datos y aplicando un mecanismo de inferencia se obtienen respuestas.

ML en cambio aprende utilizando las respuestas y los datos, aprende sabiendo cual es la respuesta de algo. Este proceso genera un conjunto de reglas, que cuando se empaquetan se denominan modelo de razonamiento o modelo de aprendizaje.

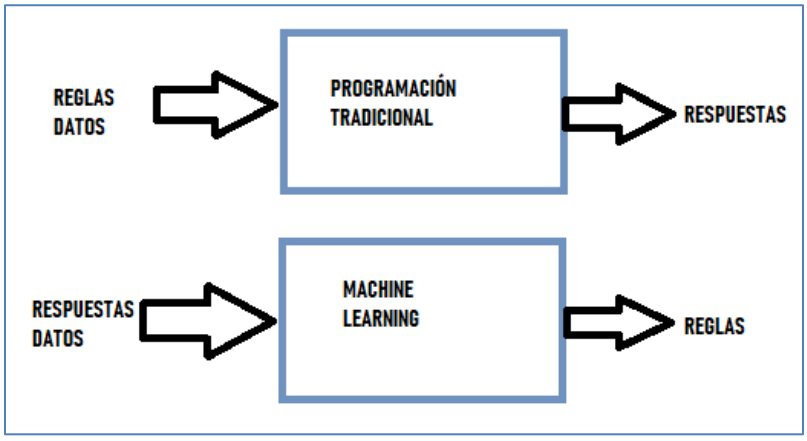

Figura 11: Modelo de aprendizaje ML.

Normalmente el ciclo de vida de los modelos está dividido en dos fases, la de entrenamiento y la de inferencia. La primera fase de entrenamiento es la más costosa y consiste en utilizar un conjunto de datos como instancia de entrenamiento, la Instancia es la estructura básica

para representar la información y está compuesta por una secuencia de atributos que describen cada uno de los ejemplos.

Para definir un modelo tenemos que definir una serie de elementos, siendo una de las primeras actividades la construcción de los datos. En los datos tenemos que hacer una serie de selección de características. En general se dividen los datos en los siguientes conjuntos:

- Conjunto de entrenamiento, que es un conjunto de instancias utilizadas para el proceso de entrenamiento con el objetivo de construir un modelo.
- Conjunto de validación, que consiste en un pequeño conjunto que se configura a partir del conjunto de entrenamiento, sirve para ir validando el modelo durante el periodo de entrenamiento.
- Conjunto de test, es un conjunto de instancias que son utilizadas para comprobar la calidad del modelo final que ha sido generado, sus elementos no son parte del conjunto de entrenamiento ni validación.

Una vez que tenemos definidos nuestros conjuntos podemos seleccionar el algoritmo de ML que se adapte mejor al problema a resolver. Elementos adicionales que pueden identificarse son la función de pérdida (Loss), el algoritmo de optimización, el conjunto de hiperparámetros que permiten modificar el proceso de aprendizaje. Una vez que tenemos todo y ejecutamos una serie de interacciones lo que obtenemos en un modelo que podemos evaluar con nuestro conjunto de test. Entonces podemos considerar el modelo como un conjunto de reglas o patrones, inferidos a partir del conjunto de entrenamiento, con el objetivo de predecir, inferir o definir la agrupación de una instancia.

Adicionalmente una vez que se tiene el modelo, hay que definir un proceso para poder desplegarlo y ponerlo en producción, de manera que otros sistemas o usuarios lo puedan utilizar.

Como normalmente todo el proceso de entrenamiento se repite n veces, se construyen diferentes versiones de ese modelo, esto provoca que tengamos que redesplegar el modelo varias veces incrementando el número de operaciones a realizar, siendo muchas de esas operaciones genéricas.

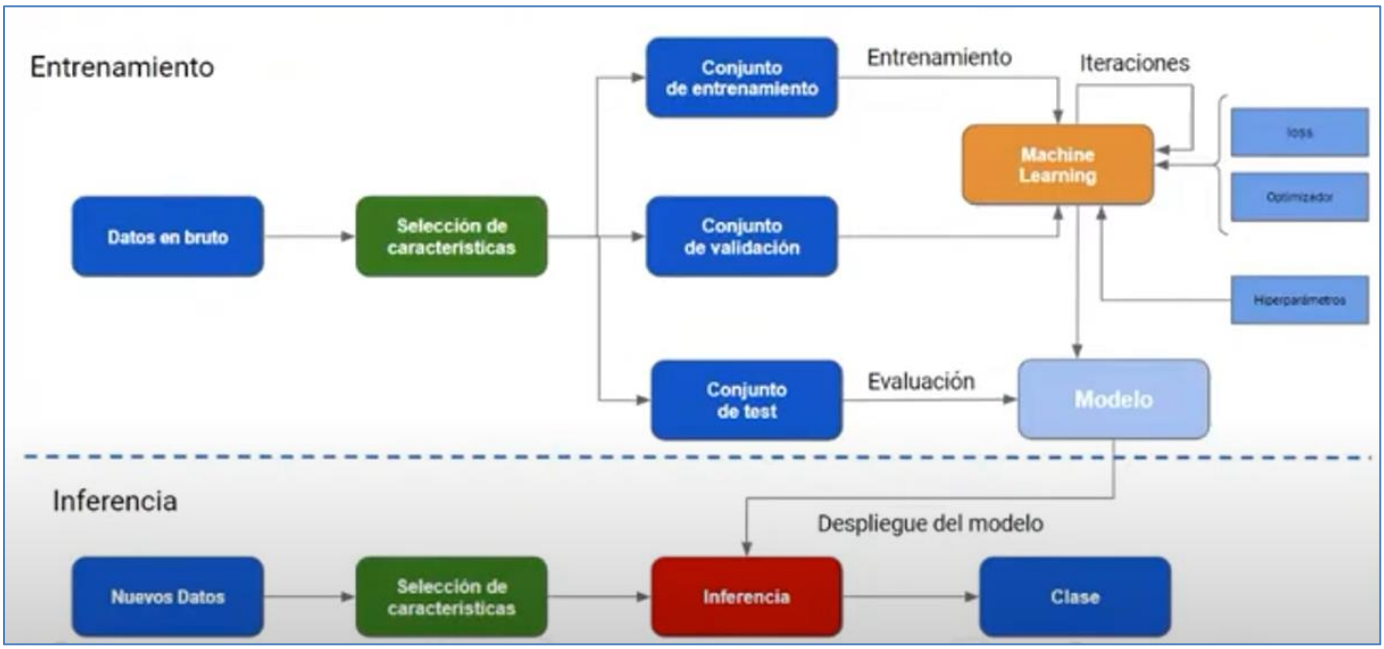

Figura 12: Actividades fase de entrenamiento e inferencia

Martínez M. (). *Paradigma Digital*. [Figura].

Un problema que surge en la construcción de modelos ML es que hay una serie de tareas y elementos adicionales a tener en cuenta. El proceso de construcción de modelos eficientes de ML involucra una serie de actividades a tener en cuenta y no es solo utilizar un algoritmo y un repositorio de datos.

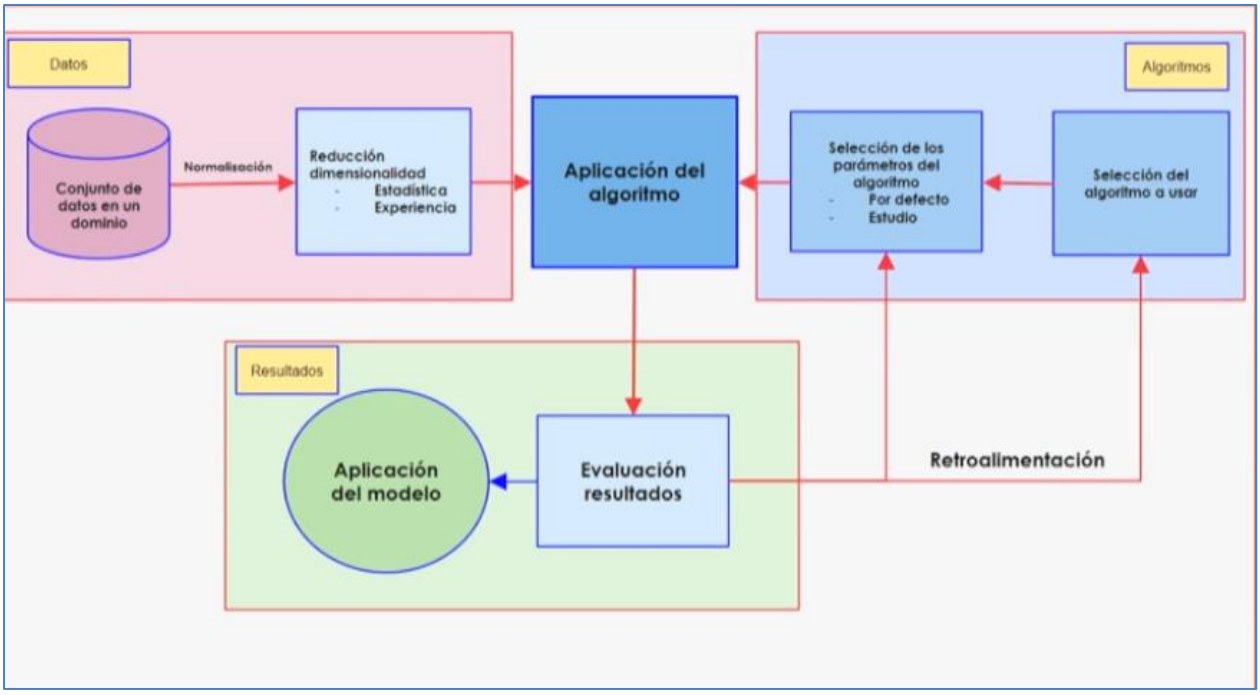

Figura 13: Ciclo de Vida de ML

Muñoz A. (2017). Machine Learning aplicado a ciberseguridad. Limitaciones y seguridad ofensiva. [Figura]. Recuperado de: https://es.slideshare.net/owaspmadrid/machine-learning-aplicado-a-ciberseguridad-limitaciones-y-seguridad-ofensiva

Recolección de datos. Consiste en recolectar los datos de distintas fuentes posibles, por ejemplo, de sitios web, de base de datos, de repositorios de dominios públicos, agregación de distintas fuentes, datos generados por nosotros. Es uno de los pasos que consume mayor cantidad de tiempo y genera complicaciones.

Pre procesamiento de datos. Consiste en adecuar nuestros datos al formato correcto para alimentar el algoritmo de aprendizaje. La limpieza de datos se realiza por ejemplo mediante la eliminación de datos atípicos, datos faltantes.

Exploración de datos. Consiste en realizar un análisis previo para corregir datos faltantes o intentar distinguir algún patrón que facilite la construcción del modelo. Se utilizan medidas estadísticas y visualizaciones en gráficos que permiten por ejemplo detectar y descartar valores atípicos o encontrar las características que tienen más influencia para realizar una predicción. En esta etapa se considera la división de los datos en dos o más conjuntos por ejemplo datos para entrenamiento y datos para validación. También se consideran métricas prefijadas de antemano para validar el modelo.

Entrenamiento del algoritmo. Se proporcionan a los algoritmos de aprendizajes que se procesan en las etapas anteriores, de tal forma que puedan extraer información para luego poder hacer predicciones.

Evaluación del algoritmo. Se evalúa la precisión de las predicciones del algoritmo, si no se llega a un nivel satisfactorio se puede volver al paso anterior para continuar con el entrenamiento del algoritmo modificando algunos parámetros, esto hasta lograr un rendimiento aceptable.

Utilización del modelo. Se pone al modelo a enfrentar el problema real, se puede medir su rendimiento y si es necesario se puede volver a algún paso anterior.

En el capítulo 3 se hace un estudio más profundo de enfoques relacionados con el ciclo de vida, esto para dotar de seguridad a sistemas ML.

2.5 Lenguajes y Herramientas para Desarrollo de ML

Durante el desarrollo del trabajo se exploraron distintos lenguajes, herramientas y entornos de desarrollo (Ide) utilizados en ML. Los lenguajes más utilizados para el trabajo con ML son Python, R, Matlab, Julia.

De los lenguajes relevados se seleccionó Python para profundizar su estudio, Phyton es un lenguaje multiparadigma y multiplataforma. Es un software libre y se destaca por su fácil comprensión y uso, cuenta con numerosas librerías especializadas para ML. A continuación, se reseñan características de recursos relevados:

- Librería *Numpy*, contiene el paquete de computación científica para análisis de datos.
- Ide *Jupyt*er, compuesto por un *shell* interactivo con soporte para múltiples lenguajes. Posee una interfaz web *Notebook*, con soporte para código, texto, expresiones matemáticas, gráficos en línea y multimedia.
- Librería *Matpltlib*, para visualizaciones y gráficos adaptados para publicaciones científicas. Se integra con *Jupyter* lo que ofrece un ambiente ideal para visualizaciones y exploración de datos de manera interactiva.
- Librería Pandas, aporta a Python funciones esenciales para el análisis de datos, estructuras de datos fáciles de usar y de alta performance como el objeto *DataFrame*. Todas sus funciones y estructuras de datos están optimizadas.
- *TensorFlow*, desarrollado por Google, contiene librerías de código libre para computación numérica usando grafos de flujo de datos.

- CNTK, es un conjunto de herramientas desarrolladas por Microsoft de código abierto, entrena algoritmos de DL para aprender como el cerebro humano.
- *Theano,* librería de Python que permite definir, optimizar y evaluar expresiones matemáticas que involucran tensores de manera eficiente.
- Keras, librería de alto nivel, está escrita y mantenida por Francis Chollet, miembro del equipo de *Google Brain*. Permite a los usuarios elegir si los modelos que se construyen serán ejecutados en el grafo simbólico de Theano, *TensorFlow* o CNTK.

3. Capítulo III – Estado del Arte de aplicaciones de ML en Ciberseguridad y Ciberdefensa.

3.1 Panorama actual del ML en ciberseguridad

El ML ha propiciado cambios importantes en diversos sectores, incluyendo el de ciberseguridad y ciberdefensa. Las mejoras en los motores de exploración, en la velocidad de detección y en la capacidad de identificar irregularidades fueron factores que contribuyen a proteger mejor las infraestructuras, en especial frente a las amenazas nuevas y emergentes y las amenazas persistentes avanzadas (APT).

La continua evolución de amenazas cibernéticas proporciona un ámbito propicio para la aplicación de ML como complemento de herramientas existentes, esto por sus capacidades de adaptación y de aprendizaje. Las crecientes aplicaciones del ML en herramientas de ciberseguridad en su faz ofensiva como defensiva permiten agregar características adicionales que potencian las mismas.

ML contribuye a potenciar soluciones tradicionales, pero cuenta con limitaciones que se deben tener en cuenta, una de ellas es la necesidad de disponer de gran cantidad de datos representativos del entorno para entrenamiento y prueba de los sistemas. Esto es un problema ya que la mayoría de los repositorios públicos no tienen buena calidad de datos, y los privados son de difícil acceso. Otra limitación es que requiere un alto poder de cómputo para su entrenamiento, esto se ve atenuado por el avance en hardware específico como las GPU y TPU.

Si bien la utilización de ML tiene sus beneficios, un enfoque más seguro y equilibrado para la ciberseguridad es desplegar una solución de varias capas, donde se aproveche el poder y el potencial de la IA y el ML, pero con el respaldo de otras tecnologías de detección y prevención

A continuación, se describen algunas de las limitaciones a tener en cuenta del ML:

- Conjunto de muestras de entrenamiento: Para entrenar los modelos de ML se requieren grandes cantidades de datos del dominio que se esté abordando, siendo su obtención no siempre posible. Por ejemplo, si los ataques son a una nueva plataforma, lenguaje o aplicación, el obtener el conjunto de muestras para el entrenamiento de los algoritmos puede llevar bastante tiempo.
- Adversario inteligente y adaptable: En ciberseguridad los atacantes no siguen pautas ni aceptan limitaciones, incluso pueden cambiar las reglas de juego completo sin ninguna advertencia.

Un ejemplo de adversario adaptable se relaciona con el uso de la fragmentación por los atacantes dividen el malware en partes y las ocultan en varios archivos separados. Cada uno de ellos es inofensivo en forma individual, recién cuando convergen en una *endpoint* o red comienzan a demostrar un comportamiento malicioso.

Otra referencia la encontramos en la agencia DARPA, entre sus proyectos de investigación avanzada en defensa está el de Máquinas de aprendizaje permanente. El programa *Lifelong Learning Machines* L2M (DARPA, 2021) busca lograr desarrollos que cambien el paradigma en arquitecturas de IA y técnicas de ML. El programa busca desarrollar sistemas que puedan aprender continuamente durante la ejecución y volverse cada vez más expertos en la realización de tareas, sujetos a límites de seguridad y aplicar habilidades y conocimientos previos a situaciones nuevas, sin olvidar el aprendizaje previo.

3.2 Aplicaciones de ML a la Ciberseguridad

Las aplicaciones de ML suponen un gran desafío para profesionales de la ciberseguridad.

ML se aplica en área como la biometría la cuál es amplio uso en seguridad. En ese sentido el proceso de autenticación de los sistemas evolucionó de solo utilizar algo que el usuario sabe (P.ej. contraseñas), a luego incorporar algo que el usuario tiene (P.ej. token, tarjetas, etc), y posteriormente incorporar reconocimientos de aspectos biométricos. Datos biométricos como huellas dactilares, geometría y patrones faciales, retinas e iris ocular, ADN, olores de voz y cuerpo, así como rasgos de comportamiento como la forma de caminar de las personas, se recopilan y utilizan para representar, identificar y verificar la identidad de un individuo.

Un ataque a un sistema biométrico se podría realizar haciendo pasar una voz real cuando no lo es, en ese sentido se pueden identificar avances importantes. Por ejemplo, el propuesto por

la empresa *Descript*, que contando con 30 segundos de una voz humana puede sintetizarla para su posterior uso, haciendo esto que sea un problema diferenciar la voz real de la que no lo es. Otro ejemplo tendiente a engañar a un sistema de reconocimiento que utilice ML es la existencia de posibilidades de hacer máscaras 3D a partir de una foto 2D.

Otra área de aplicación de ML es la detección de amenazas de internet. Ejemplos de aplicaciones son escaneo y monitoreo de sitios web, análisis de sentimientos realizados en redes sociales, detección de *Fake News*, detección de suplantación de perfiles, filtración de datos sensibles, etc.

Áreas en la que se cuenta con amplia experiencia es en tareas de *Antispam* y *Antiphishing*, donde empresas proveedoras de soluciones incorporan técnicas de IA. Grandes operadores de correo como Google, Yahoo, Hotmail, entre otros, tienen dentro de sus sistemas aplicaciones de ML para detectar de forma proactiva el Spam o correo no deseado.

En la Detección de anomalías en redes, ML se aplica en: monitoreo de red, monitoreo de datos, análisis estadístico, análisis de patrones, entre otros ejemplos. Podemos tener un sistema ML que este monitoreando y analizando el tráfico de una red en tiempo real, aprenda cual es el comportamiento normal de una red y basado en ese aprendizaje, cuando detecta algo anómalo empieza a tomar decisiones con respecto a qué hacer, por ejemplo, puede tomar decisiones activas como bloquear el acceso a un protocolo que está siendo vulnerado o pasivas como producir una notificación.

En Criptografía existen aplicaciones de ML que permite descifrar un criptograma sin necesidad de conocer la clave de cifrado, el intercambio de claves privadas (redes neuronales sincronizadas) se puede aplicar para esta tarea.

ML se aplican para la Detección de Malware, resolución automática de Captcha, detección de anomalías de dispositivos IoT, entre otras aplicaciones.

3.3 Herramientas Comerciales que incorporan ML

Soluciones tradicionales de Ciberseguridad utilizan ML para reforzar los esquemas de protección en un panorama de amenazas crecientes, empresas líderes proponen diferentes aplicaciones:

- Fortinet (2021) propone la aplicación de IA para acelerar la prevención de amenazas, detección y respuesta, esto aplicando IA de diferentes tipos y en diferentes ubicaciones para fines complementarios. En *FortiGuard Labs* aplican ML para generar inteli-

gencia global frente a nuevas amenazas. En Web *Application Firewal* y en la plataforma de protección de *endpoint* (EPP), utilizan IA para complementar las técnicas tradicionales.

- Fujitsu (Iria & Alvarez, 2021) desarrolló el sistema IA Deep Tensor, para tomar decisiones frente a ciberataques, validando el riesgo y las acciones a tomar. El sistema se alimenta con gran cantidad de información sobre registros de comportamiento y muestras de ataques reales que se utiliza como entrenamiento para la toma de decisiones.
- Cisco (CISCO, 2021) cuenta con una plataforma avanzada de ML *Cisco AI Network Analytics*, la cual utiliza una colección de algoritmos de ML altamente avanzados para detectar las anomalías más relevantes y comunicarse con conocimiento, junto con recomendaciones de resolución lógica para ayudar a corregirlas. La plataforma se incluye dentro de las capacidades de *Cisco DNA Assurance de Cisco DNA Center*.
- La empresa inglesa DarkTrace (DarkTrace, 2021) entre sus productos cuenta con Darktrace Immune System que es una plataforma de ciberdefensa autónoma Utiliza IA para detectar ciberamenazas sofisticadas, desde amenazas internas y espionaje criminal hasta ramsonware y ataques de estados nación, adaptándose constantemente a entornos cambiantes.
- La empresa Check Point (Point, 2018) dispone entre sus productos de motores basados en Inteligencia Artificial en su plataforma de prevención de amenazas, Campaign *Hunting, Huntress y Context-Aware Detection* (CADET), con el objetivo de mejorar la inteligencia frente a amenazas, buscando ejecutables maliciosos dando acceso y visibilidad a todas las partes de la infraestructura TI: redes, centro de datos, entornos de nube, dispositivos móviles y *endpoints*.
- La empresa Symantec dispone entre otros productos de *Symantec Endpoint Protection* que propone una solución contra amenazas conocidas y desconocidas usando un enfoque de defensas en capas, reduciendo el riesgo de exposición utilizando ML para anticiparse a ataques. El motor de ML avanzado (AML, *Advanced Machine Learning*) de Symantec determina si un archivo es de confianza mediante un proceso de aprendizaje.

- La Empresa Palo Alto (IT digital Security, 2020), dispone de un producto firewall de última generación (NGFW), el mismo incorpora el ML en el núcleo del firewall que ayuda de forma proactiva a detener de forma inteligente las amenazas, proteger dispositivos de IoT y recomendar políticas de seguridad.
- ESET (Kubovic, 2021) tiene incorporado en sus soluciones el motor Augur de ML, el mismo está disponible a través de sus soluciones *ESET Entrerprise Inspector y ESET Dynamic ThreatDefense*. El motor de ESET emula el comportamiento de la muestra y proporciona un conjunto de características y secuencias para su posterior procesamiento. También ejecuta un análisis detallado de ADN que arroja características numéricas de la muestra para posteriormente analizar la información recopilada utilizando tanto los métodos de DL como el procesamiento con modelos múltiples.
- La empresa Karspersky (Karspersky, 2021) dispone del *producto Kaspersky Machine Learning for Anomaly Detection (Kaspersky MLAD)* orientado a la protección de entornos industriales. Los ataques dirigidos son los más peligrosos para las instalaciones industriales porque pueden interrumpir el proceso tecnológico y causar daños irreversibles a los equipos, lo que resulta en importantes pérdidas financieras y de reputación.
- MLAD se propone como un sistema que utiliza RNA para monitorear simultáneamente una amplia gama de datos de telemetría e identificar anomalías en el funcionamiento de los sistemas ciberfísicos presentes en instalaciones industriales modernas. La RNA analiza la telemetría en tiempo real desde varios sensores utilizados en el proceso de producción. Detecta desviaciones menores, como un cambio en la dinámica o correlaciones de las señales, y emite alertas antes de que los valores alcancen sus umbrales e impactan el rendimiento, lo que permite a los operadores de la planta tomar acciones preventivas. Para poder detectar anomalías, la RNA aprende el comportamiento normal de la máquina a partir de datos históricos de telemetría.
- La Empresa MITRE (Citomic, 2020) dispone de una herramienta *MITRE ATTA&CK* que cuenta con útil para identificar tácticas comunes, técnicas y procedimientos, que las APT usa contra plataformas informáticas. Para la visualización se usa ATTA&CK Matrix, una matriz compuesta por un eje de tácticas que representan la motivación de

la técnica empleada y otro eje de técnicas muestran cómo el ciberatacante desarrollaría la acción para lograr su objetivo.

- Según el informe de Gartner Research Top 10 Strategic Technology Trends for 2020, se espera que gran parte de los ciberataques de IA proyectados a 2022, aprovechen el envenenemiento de datos de entrenamiento, robo de modelos o ejemplos adversarios para atacar sistemas de ML.

Por ello en 2020 MITRE junto a otras empresas como Microsoft, IBM, Airbus, presentaron *Adversarial ML Threat Matrix*, que es un marco específicamente diseñado para identificar, responder y remediar ciberataques dirigidos a sistemas de ML. La matriz está estructurada de manera muy similar a la modelo MITRE convencional, la conforma un eje de siete tácticas encuadradas en el ámbito del ML: Reconocimiento, Acceso Inicial, Ejecución, Persistencia, Evasión de Modelo, Exfiltración e Impacto. Debajo de ellas se encuadran las Técnicas, categorizadas en dos tipos: las técnicas en color naranja son exclusivas para sistemas de ML, mientras que las técnicas en color blanco pueden ser empleadas para sistemas de ML pero también para otros que no lo son y provienen directamente del marco común Enterprise.

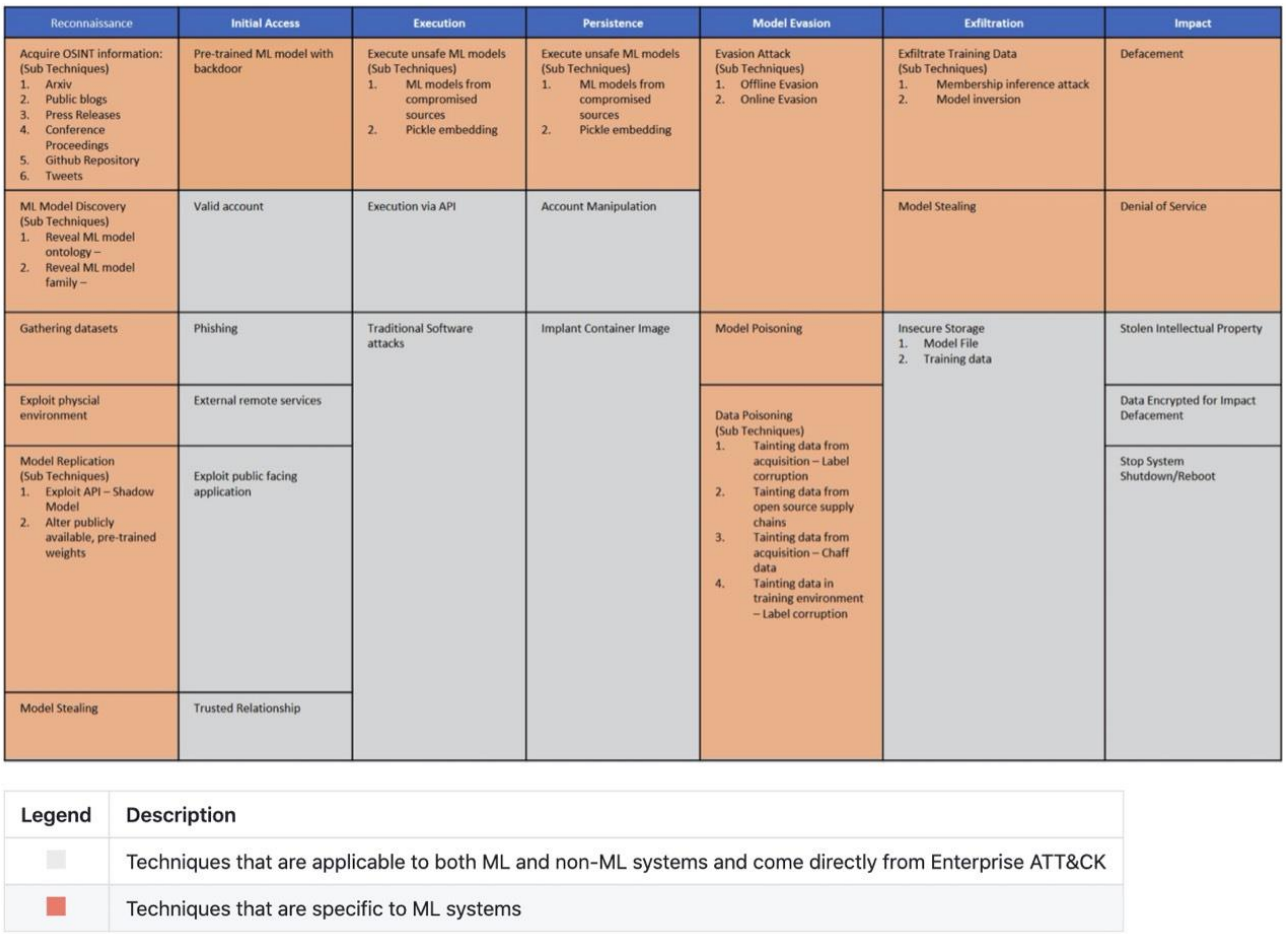

Figura 14: Matriz *Adversarial Machine Learning* de MITRE

Segu-Info - Ciberseguridad. (2020). MITRE lanza matriz de amenazas para Machine Learning. [Figura]. Recuperado de: https://blog.segu-info.com.ar/2020/10/mitre-lanza-matriz-de-amenazas-para.html

3.4 ML en Seguridad Defensiva

En la faz de seguridad defensiva, ML se aplica en áreas como las telecomunicaciones, seguridad de red, análisis de tráfico, detección y prevención de intrusiones, fuga de información, detección de fraudes, prevención de malware, entre otras áreas.

En los sistemas de detección y prevención de intrusiones (IDS/IPS), que identifican actividades maliciosas de la red y evitan que los intrusos accedan a los sistemas, los algoritmos de DL, con aplicaciones de RNA convolucionales y recurrentes, se aplican con éxito. Trabajan reduciendo el número de falsos positivos ayudando a los equipos de seguridad a diferenciar las actividades de red permitidas y no.

Con respecto al análisis del tráfico de la red DL se aplica con éxito en análisis del tráfico de red HTTPS para buscar actividades maliciosas como los ataques de *SQL Injection* y ataque de denegación de servicios DDoS.

Con respecto a la detección de malware, podemos apreciar que las soluciones tradicionales encuentran los malware con un sistema basado en firmas. Las empresas de seguridad gestionan bases de datos integradas de amenazas conocidas que se actualizan con frecuencia, siendo esto eficaz contra amenazas tradicionales, pero no suficiente frente a amenazas avanzadas. Soluciones basadas en DL permiten detectar amenazas avanzadas y no dependen de recordar firmas conocidas y patrones de ataque comunes.

En el análisis del comportamiento de los usuarios, ML se aplica con éxito, pueden detectar patrones de comportamiento normales de los empleados y reconocer actividades sospechosas, como por ejemplo acceso a los sistemas en horas no comunes.

En la faz defensiva existen proyectos de recopilación de información accesibles, esto dado el inconveniente de encontrar datos para el entrenamiento de los algoritmos de ML. Al respecto el proyecto MLSec (Mlsec, 2021) brinda una recopilación de software, datos de código abierto, proyectos y herramientas de IA dedicadas a seguridad defensiva, abordando las siguientes temáticas: ML, aprendizaje adversario, descubrimiento de vulnerabilidades, análisis de malware, análisis de datos, entre otros.

3.5 ML en Seguridad Ofensiva

En este tema se analiza los tipos de ataques que puede sufrir un sistema de ML, para ello se evalúa que puede hacer el atacante en función de las fases a las que tiene acceso, por ejemplo entradas, salidas, fases de entrenamiento o los modelos. Desde este punto de vista se puede ver la visión o modelos que usan los atacantes para vulnerar los sistemas de ML, que luego desde el punto de vista del defensor contribuyen para construir sistemas de ML más robustos.

3.5.1 Ataques clásicos mejorados

Existen ataques tradicionales que mejoran su rendimiento con la utilización de ML, a continuación, se describen algunos ejemplos:

El descifrado de contraseñas o *password cracking* consiste en el proceso de recuperación de contraseñas que se han almacenado en un equipo, esta actividad puede servir para ayudar a un usuario a recuperar alguna contraseña olvidada como también para obtener acceso no autorizado a un sistema

El almacenamiento de contraseñas de los sistemas de información evolucionó por cuestiones de seguridad, de tener las mismas en texto en claro en los repositorios, a usar algoritmos de hash para mejor resguardo. Con ataques como el de fuerza bruta y diccionario se intenta obtener la contraseña de la víctima de manera repetida, estos ataques se pueden realizar de forma *online* como *offline*. El ataque de fuerza bruta hace uso de una combinación de caracteres mientras que el de diccionario utiliza lista de palabras pre-generadas.

Una aplicación de ataques de contraseñas se refiere en el artículo *PassGAN: A Deep Learning Approach for Password Guessing* (Hitaj, Ateniese, Gasti & Perez-Cruz, 2019), en la investigación se diseñó la herramienta *PasGAn* que utiliza ML con una red neuronal GAN para aprender de manera autónoma la distribución de contraseñas. Con diccionarios de contraseñas el algoritmo puede generalizar una regla de *crackeo*, no solamente con las contraseñas que ha visto sino con otras similares. Se comparó *PasGAn* con herramientas como *Jhon the Ripper* y Hastcat, llegando a concluir que con ML se aumentó la velocidad de descifrado de la contraseña.

Los ataques de inyección de código SQL o *SQL Injection* consisten en que un atacante, realizando peticiones, inserta código malicioso en las cadenas que posteriormente se pasan a un intérprete de SQL para su análisis y ejecución. Luego de identificada la vulnerabilidad en el proceso de explotación, el atacante busca extraer base de datos, estructura, tablas, etc., realizando reiteradas peticiones al Servidor Sql. Aplicando ML se puede reducir las peticiones al servidor SQL, contribuyendo esto que el ataque no sea detectado. En vez de realizar ataque a ciegas o por fuerza bruta, se comprobó que utilizando ML se puede minimizar las peticiones.

3.5.2 Ataque de evasión a sistemas de ML

En los ataques de evasión (Hernández & Escribano, 2021) un adversario inserta una pequeña perturbación (en forma de ruido) en la entrada de un modelo de ML, con el objetivo que clasifique de forma incorrecta. Se denominan ataques exploratorios y comprometen la disponibilidad del modelo atacado, son los más estudiados y explotados durante los últimos años.

Las entradas maliciosas reciben el nombre de ejemplo adversario (*adversarial example*), la mayor cantidad de ataques usando ejemplos adversarios, se han realizado sobre imágenes, pero se pueden realizar sobre audio, vídeo, etc. En los ejemplos adversarios se busca la propiedad de que sean imperceptibles por un humano en el caso de imágenes, o que no cambien el contexto o el significado en audio o texto. En la figura 15 se muestran dos ejemplos adversarios, uno

en una imagen y otro en un audio, en los dos casos se agrega una pequeña perturbación en forma de ruido a la entrada.

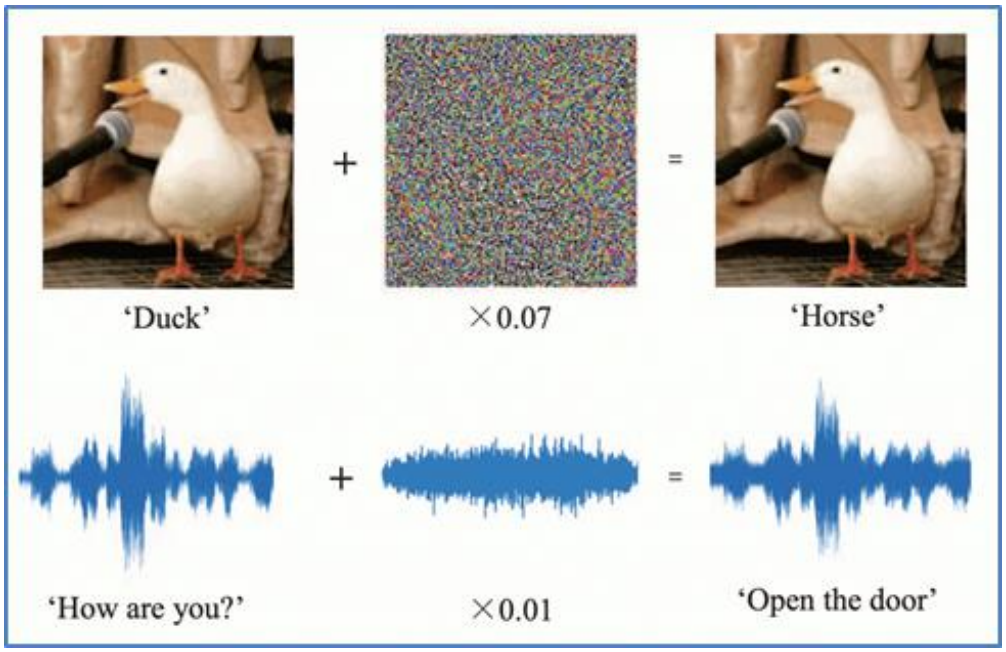

Figura 15: Ejemplos adversario

Gong Y., Poellabauer C. (2018). An Overview of Vulnerabilities of Voice Controlled Systems. [Figura]. Recuperado de: https://arxiv.org/pdf/1803.09156.pdf

El ataque de evasión se puede realizar de forma dirigida o no dirigida. De manera dirigida el adversario insta a que el modelo clasifique de acuerdo a una opción de su elección, mientras que de forma no dirigida el adversario quiere que el modelo clasifique mal.

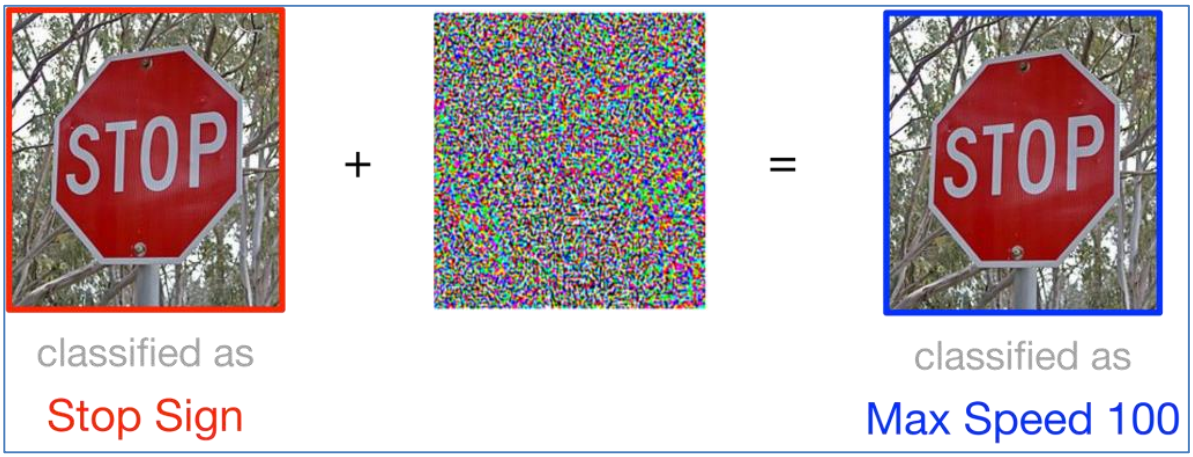

Figura 16: Ejemplo adversario sobre señal de tránsito

Chen J.(2019). Regularización sólida de atribuciones. [Figura]. Recuperado de: https://www.altacognita.com/robust-attribution/

En la figura 16 se dirige el ejemplo adversario sobre una señal de tránsito, el objetivo es que el modelo clasifique de forma incorrecta y además que clasifique un límite de velocidad de 100 km/h.

Los ataques de evasión se pueden realizar en escenarios de caja blanca como en caja negra. En caja blanca el adversario tiene acceso al modelo mientras que en caja negra se tiene un conocimiento limitado de la estructura interna del modelo. En ataques de caja blanca se puede obtener una perturbación a través de calcular gradientes o resolver un problema de optimización. En ataques de caja negra se puede realizar previamente un ataque de extracción, y mediante un modelo sustituto realizar un ataque de caja blanca; también a través de estimar gradientes para buscar ejemplos adversarios.

3.5.3 Ataque de envenenamiento a sistemas de ML

Los ataques de envenenamiento (Hernandez & Jose, 2021) buscan corromper el conjunto de entrenamiento, provocando que un modelo de ML reduzca su precisión siendo muy difícil de detectar. Se reconocieron los primeros ataques de este tipo sobre algoritmos clásicos de ML (SVM, Naïve Bayes, regresión logística y lineal, entre otros.), extendiéndose de forma posterior con el avance del DL.

Los ataques de envenenamiento se pueden realizar en escenarios de caja blanca como de caja negra, como objetivos de este tipo de ataque se reconocen:

- Destruir la disponibilidad del modelo modificando la frontera de decisión, afectando un modelo para que realice predicciones incorrectas. Por ejemplo, una señal de *Stop* que sea reconocida como límite de velocidad de 100 km/h.
- Crear una puerta trasera, de tal forma que el modelo se comporte de forma correcta en la mayoría de casos, pero fallando en ciertas entradas especialmente creadas por el adversario que producen resultados no deseados. El adversario puede manipular los resultados de las predicciones y lanzar ataques futuros.

El proceso que sigue un adversario para realizar un ataque de envenenamiento se puede ver en la Figura 17:

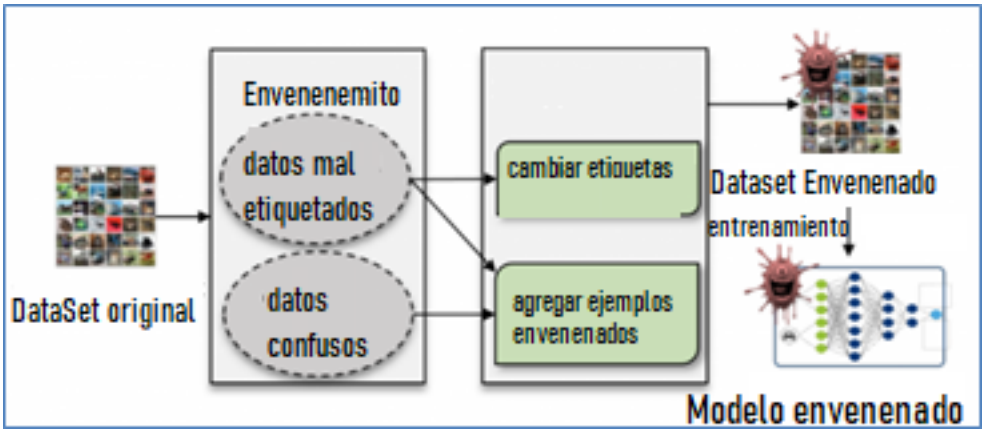

Figura 17: Proceso de un ataque de envenenamiento

Yingzhe He et al. (2020). Towards Security Threats of Deep Learning Systems: A Survey. [Figura]. Recuperado de: https://arxiv.org/abs/1911.12562

El adversario puede envenenar el conjunto de entrenamiento de dos maneras:

- Usando datos mal etiquetados, para ello selecciona ciertos datos de interés para el adversario en el conjunto de entrenamiento y cambia las etiquetas. Esta aproximación produce una modificación de la frontera de decisión para que las predicciones sean incorrectas.
- Creando datos confusos, para ello introduce características especiales que pueden ser aprendidas por el modelo, pero no forman parte del comportamiento del mismo. Estas características permiten a un adversario actuar como un disparador y producir una predicción incorrecta. Las características introducidas en el modelo se pueden usar como una puerta trasera.

En el artículo BadNets: *Identifying Vulnerabilities in the Machine Learning Model Supply Chain* (Tianyu Gu, Dolan-Gavitt & Garg, 2019), se analizó la aplicación de ML al caso de señales de tránsito utilizando el algoritmo Faster-RCNN, el objetivo era que detecte y reconozca objetos. El entrenamiento se realizó con el conjunto de datos que contenía tres clases de señales de tránsito: señales de stop, límites de velocidad y advertencias. Como disparador se usó un cuadrado amarillo, una imagen de una bomba y una imagen de una flor.

Figura 18: Disparadores empleados para el reconocimiento de imágenes de tráfico

Tianyu Gu et al. (2019). BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. [Figura]. Recuperado de: https://arxiv.org/abs/1708.06733

Los autores del artículo demostraron que el ataque de envenenamiento ser aplicado en el mundo real, como por ejemplo en sistemas de conducción autónoma obteniendo que la señal de stop fuera reconocida como señal de límite de velocidad, con una probabilidad del 94.7%.

En la figura 19 se aprecia que la puerta trasera se activa gracias a pegar una etiqueta con un cuadrado amarillo en la señal, este era uno de los disparadores empleados en el caso.

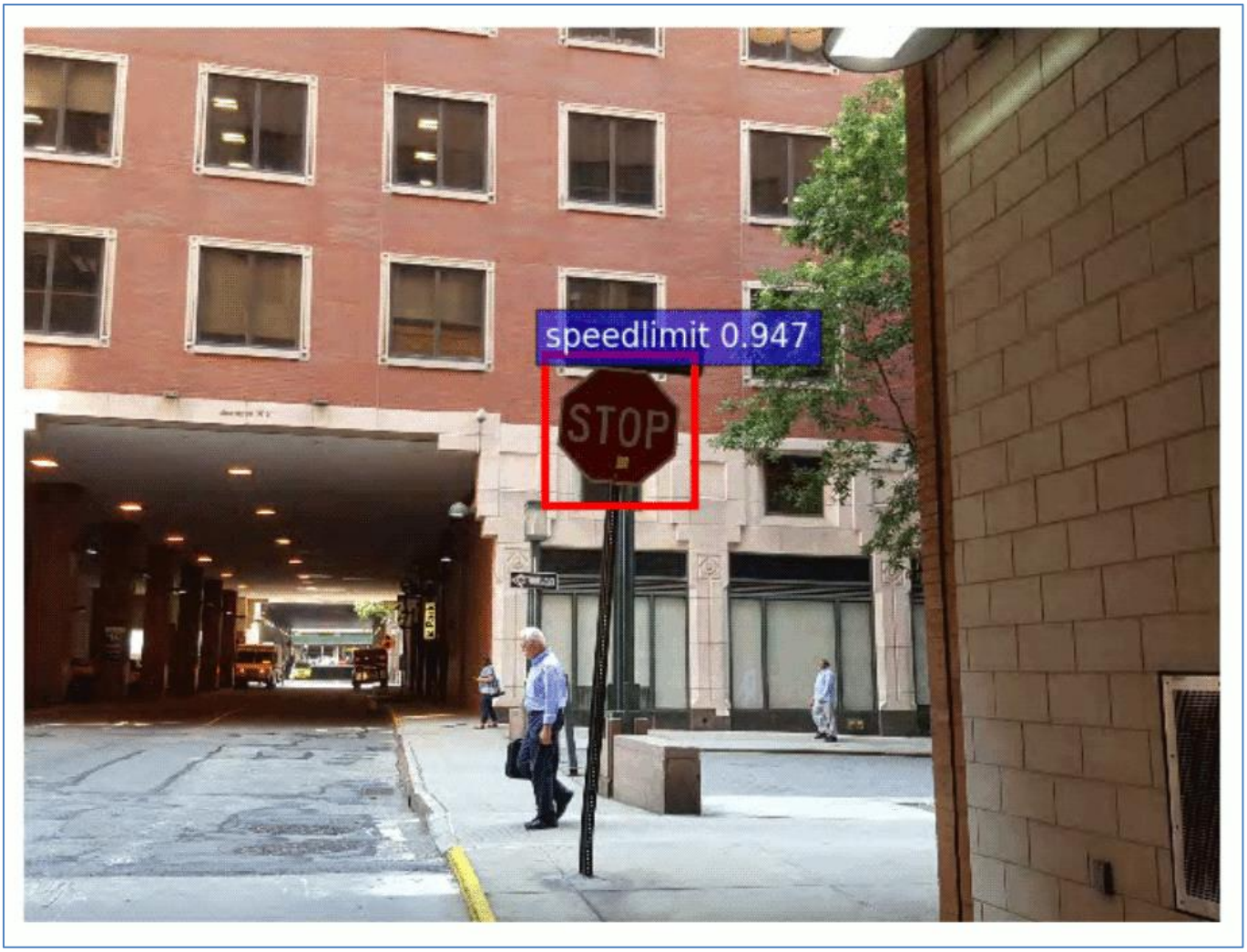

Figura 19: Ejemplo de señal de stop activada como límite de velocidad

Tianyu Gu et al. (2019). BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. [Figura]. Recuperado de: https://arxiv.org/abs/1708.06733

3.5.4 Ataque de extracción de modelos

Este tipo de ataque permite a un adversario robar los parámetros de un modelo de ML, para ello se hacen peticiones al modelo objetivo con entradas preparadas para extraer la mayor cantidad de información posible y con las respuestas obtenidas inferir los valores de los parámetros del modelo original. El ataque vulnera la confidencialidad del modelo y compromete la propiedad intelectual del mismo, por ejemplo, para evitar pagar por consumir un servicio de *Machine Learning as a Service (MLaaS)* como los disponibles en AWS, GCP, Azure. Se pueden realizar en escenarios de caja blanca o caja negra.

En el escenario donde el adversario sólo tiene acceso a las entradas que introduce al modelo que intenta robar y a las salidas que devuelve el mismo, se encuentra con las siguientes limitantes:

- No conoce los detalles internos la arquitectura interna del modelo o los hiperparámetros.
- Desconoce el proceso de entrenamiento y no tiene acceso a otro conjunto de datos con la misma distribución que los originales.
- Puede ser detectado si hace peticiones muy frecuentemente al modelo.

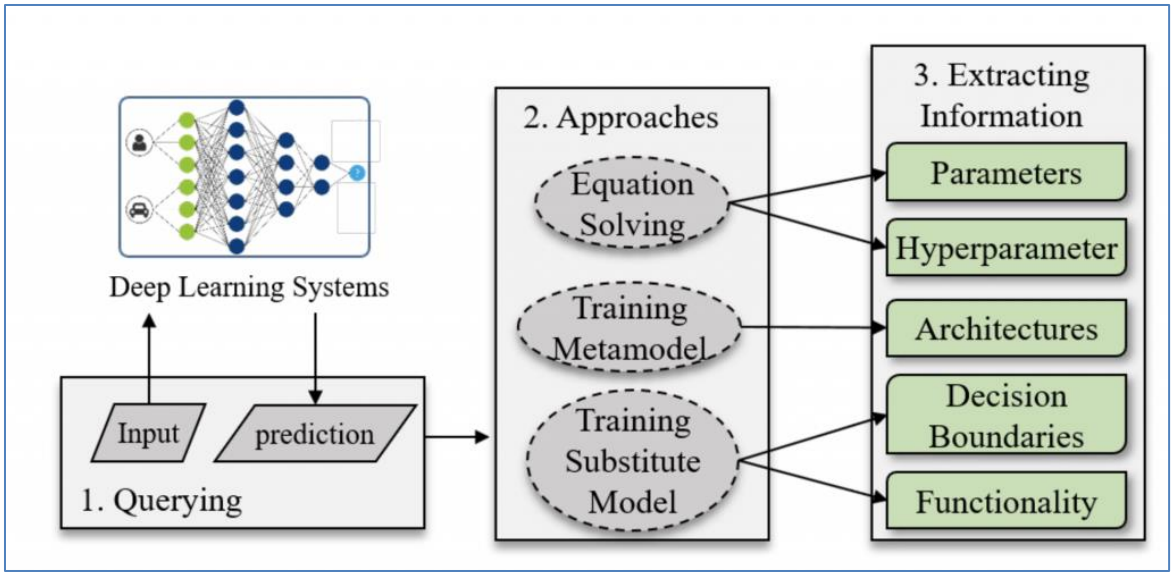

Figura 20: Proceso de extracción de modelos

Yingzhe He et al. (2020). Towards Security Threats of Deep Learning Systems: A Survey. [Figura]. Recuperado de: https://arxiv.org/abs/1911.12562

En el artículo de Towards Security *Threats of Deep Learning Systems: A Survey* (Yingzhe, Guozhu Meng, Kai Chen, Xingbo Hu & Jinwen He, 2020) se describe el proceso de extracción de modelos, como se muestra en la figura 20.

El adversario realiza peticiones al modelo que intenta robar y obtiene las salidas correspondientes del mismo. Posteriormente usa las entradas y salidas para extraer el modelo, eligiendo la aproximación que le convenga al adversario en su escenario. Se sustraen datos confidenciales entre los que se incluyen los parámetros, hiperparámetros, arquitectura, fronteras de decisión y funcionalidad del modelo.

Un ejemplo sencillo de este tipo de ataque se puede ver con un ejemplo de regresión logística, que es uno de los múltiples algoritmos que hay en inteligencia artificial. En la figura 21 se esquematiza y se puede proponer un procedimiento para obtener el modelo:

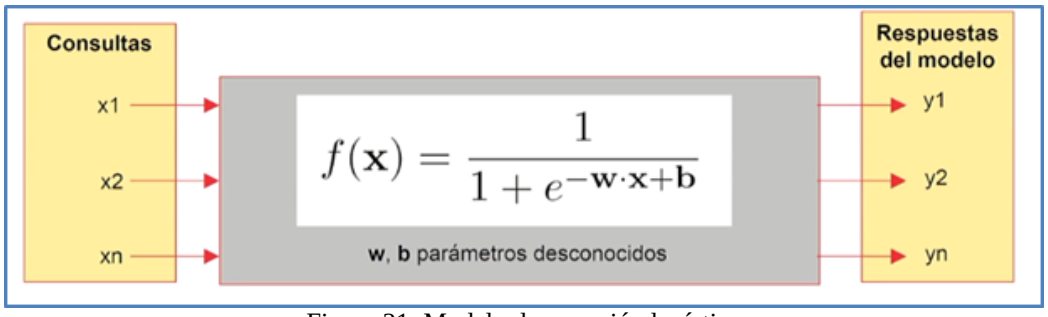

Figura 21: Modelo de regresión logística

Recuperado de: http://docs.aws.amazon.com/machine-learning/latest/dg/learning-algorithm.html

- Hacer peticiones con entradas x_i obteniendo $y_i=f(x_i)$. Si x_i tiene n dimensiones, solo necesitaríamos n+1 consultas.
- Sustituir para cada (x_i, y_i) en la ecuación
- Formar un sistema de ecuaciones con w,b y resolver obteniendo los valores de w y b. Sustituyendo esos valores en $f(x_i)$ se obtiene los parámetros de configuración del modelo.

3.6 Herramientas para evaluación de la seguridad de ML

Para comprobar la robustez de los modelos de ML frente a distintos ataques, se disponen de herramientas que permiten evaluar cómo se afectan los modelos y como poder defenderlos empleando las estrategias, a continuación, se describen algunas de ellas.

- *Cleverhans* es una librería para construir ataques adversarios y medir la robustez de un modelo de ML en modelos de imágenes. Está desarrollada en Python y se integra con los *frameworks Tensorflow, Torch y JAX*. Implementa numerosos ataques como L-BFGS, FSGM, JSMA, C&W, entre otros. El proyecto se encuentra activo dentro de la organización de *Tensorflow* en *GitHub*.
- *Foolbox* es un conjunto de herramientas para engañar redes neuronales usando ejemplos adversarios. Está implementada en Python y requiere Numpy y Scipy para funcionar. Soporta *frameworks* de DL como Tensorflow, Keras, PyTorch, JAX, Theano, MXNet y Lasagne.

- IBM *Adversarial Robustness Toolbox (*ART) es una librería desarrollada en Python para la defensa de sistemas ML. Permite la defensa de RNA, SVM, árboles de decisión, regresión logística, entre otros. Al ser de propósito general, no se centra en ataques adversarios, sino que contempla otros ataques: evasión, envenenamiento, extracción. Implementa un gran número de ataques, así como muchas defensas, soporta *frameworks* como Tensorflow, Keras, PyTorch, Scikit learn, entre otros.
- AdvBox es un conjunto de herramientas escritas en Python para generar ejemplos adversarios para atacar RNA, implementa numerosos ataques y defens. Soporta frameworks como PaddlePaddle, PyTorch, Caffe2, MxNet, Keras y Tensorflow.
- DeepRobust es una librería en Python, de ataque y defensa de adversarios para imágenes y grafos para PyTorch. Con imágenes implementa ataques típicos como FGSM, L-BFGS y C&W, entre otros. En defensa implementa diferentes técnicas de entrenamiento adversario.
- SecML es una librería implementada en Python que permite evaluar la robustez de algoritmos de ML. Soporta los algoritmos supervisados de Scikit-learn y RNA de PyTorch. Implementa ataques de envenenamiento y también de evasión, provee conectores con otras librerías de *Adversarial Machine Learning*. Cuenta con amplia documentación y numerosos ejemplos.
- Armory es una librería para probar de forma escalable evaluaciones de defensas frente a ataques *Adversarial Machine Learning* en imágenes. Soporta los *frameworks* PyTorch y *Tensorflow*. Implementa ataques de evasión y envenenamiento, y también defensas.

3.7 Formalización de seguridad en ML

En el artículo *Hidden Technical Debt in Machine Learning Systems* (Sculley, Holt, Golovin, Dvydov & Phillips, 2014), se hace referencia a una "Deuda técnica oculta" que tienen los sistemas ML, debido al elevado número de componentes relacionados con el proceso de aprendizaje. Solo una pequeña fracción de los sistemas de ML del mundo real está compuesta solo por el código, y como se muestra en la figura 22, existen numerosas tareas y elementos adicionales a tener en cuenta.

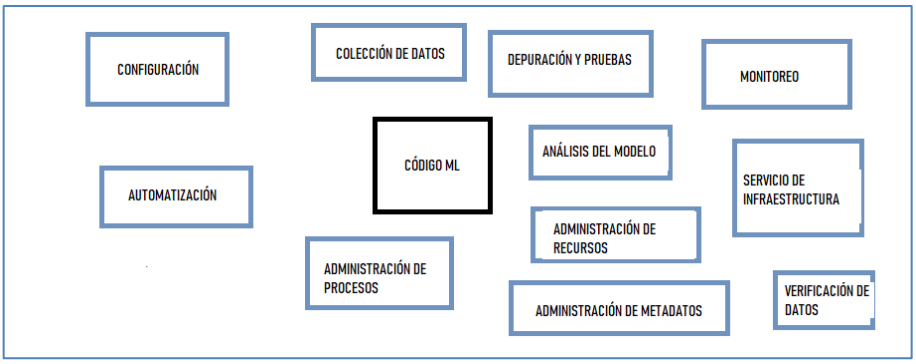

Figura 22: Deuda técnica oculta en modelos de ML

Sculley, D. et al. (2020). Adaptación de Deuda técnica oculta en sistemas de ML. [Figura].

Bajo un enfoque de ingeniería de software se visualiza la necesidad de explorar varios factores de riesgo a tener en cuenta en el diseño del sistema ML, esto ya que los sistemas de ML tienen todos los problemas de mantenimiento del código tradicional y se le agrega un conjunto enfocado a cuestiones específicas como se puede apreciar en la figura 22.

3.8 Machine Learning Operation(MLDevOps)

Antes de abordar MLDevOps vamos a referirnos a DevOps y SecDevOps, paradigmas relacionados que suponen un gran avance en el diseño y la programación de código seguro.

DevOps es un paradigma que propone un cambio en la forma y la cultura de trabajo entre programadores o desarrolladores (“Dev” *developers*) y el personal encargado de las operaciones (“Ops” *operations*).

En DevOps se propone acercar los grupos de desarrolladores y personal de operaciones, esto para que trabajen colaborativamente. De esta forma el desarrollador tiene un mayor conocimiento sobre la infraestructura donde correrá su aplicación, y por otro lado el personal de operaciones toma un conocimiento más preciso de los recursos que la aplicación requiere.

DevOps propone aplicar la automatización a todo el proceso, con una retroalimentación continua para permitir observar no solo errores, sino también la evolución de la aplicación. Bajo el paradigma DevOps, el desarrollo ya no es cuestión de un grupo, sino que entran otros actores en el proceso. En la siguiente figura 23 se puede ver el ciclo de vida clásico DevOps con todas las fases a llevar delante de forma cíclica.

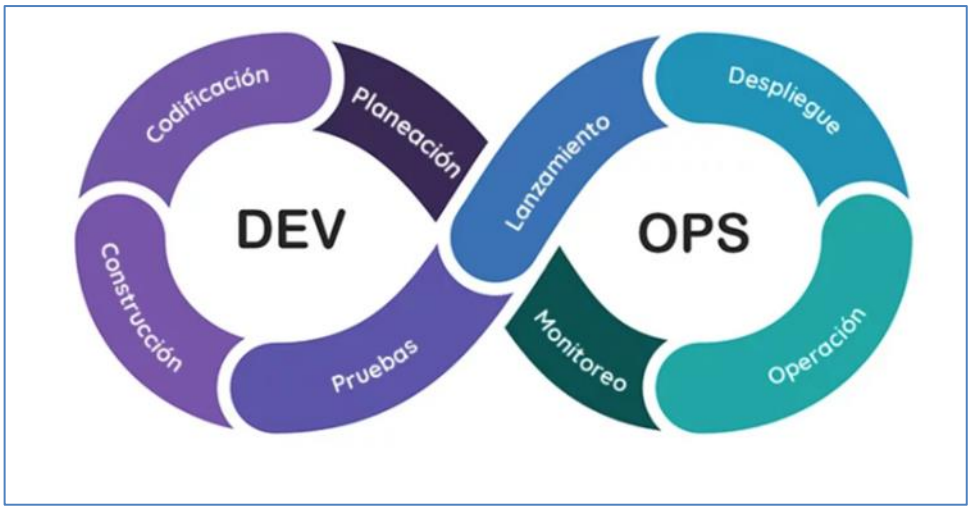

Figura 23: Ciclo de vida clásico DevOps

SES Digital (2022). Arquitectura de Software y Cultura Devops. [Figura]. Recuperado de: https://sesitdigital.com/arquitectura-de-software-y-cultura-devops/

En Planeación se recogen los requerimientos de la aplicación, en Codificación se comienza a escribir código y continúa en Construcción, en Pruebas se realizan la ejecución de las mismas, en la fase de Lanzamiento se empaqueta la aplicación para su despliegue (por ejemplo, en un contenedor *Docker*) y se sube el artefacto. Recién en la fase Despliegue se comienzan a aplicar los controles de seguridad.

Los beneficios de aplicar *DevOps* se orientan a mejorar la calidad del software, reducir el tiempo de entrega de funcionalidad y ahorrar esfuerzos y costos en el proceso completo, alineando las estrategias de desarrollo y operaciones.

Relacionado con la seguridad se puede apreciar en el modelo clásico *DevOps*, los controles de seguridad se aplican de forma tardía, interviniendo el equipo encargado de la seguridad recién en la fase de Despliegue. Si se encuentran problemas se los reporta al equipo de desarrollo, el cual tendrá que codificar y probar las soluciones. La intervención tardía del equipo de seguridad afecta significativamente los tiempos de entrega y los costos asociados siendo una solución la que se plantea en *SecDevOps*.

El paradigma *SecDevOps* (Alonso, 2019) propone tener controles de seguridad desde la primera fase.

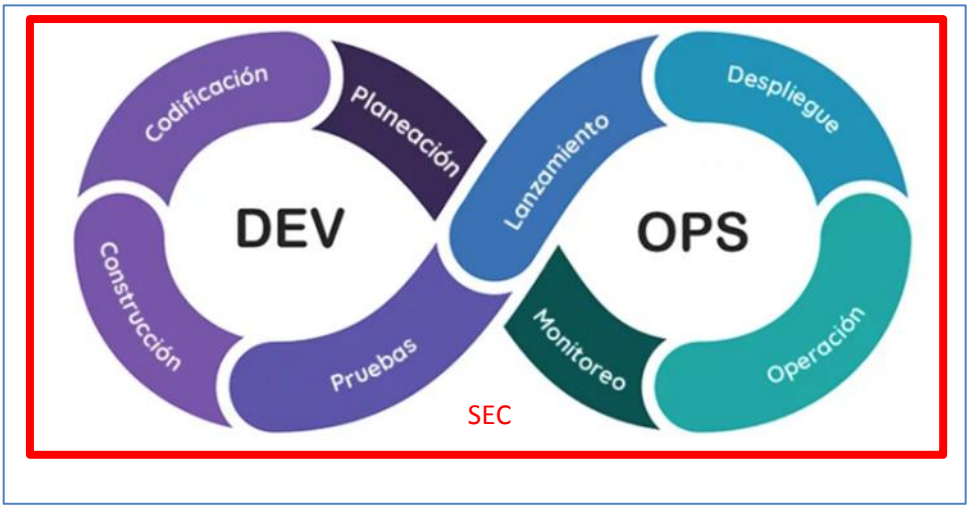

Figura 24: Ciclo de vida *SecDevOps*

SES Digital (2022). Arquitectura de Software y Cultura Devops. Recuperado de: https://sesitdigital.com/arquitectura-de-software-y-cultura-devops/

A continuación, se describen funciones del equipo de seguridad en el ciclo de vida:

- En la fase de planeación se analiza el modelo de amenazas.
- En las fases de codificación y construcción, el equipo de seguridad capacita y provee de herramientas de seguridad al equipo de desarrollo, para que se familiarice con términos de seguridad. Por ejemplo, se los ayuda a reconocer ataques SQLi, XSS, DDoS, etc.
- En la fase de pruebas se ejecutan las respectivas pruebas unitarias y de integración.
- En la fase de lanzamiento se unifica todo el código en un sólo paquete o artefacto. De corresponder se realizan escaneos de seguridad para las librerías externas que permitan detectar vulnerabilidades. Se ubica el artefacto en algún repositorio central donde posteriormente será consumido en la siguiente fase.
- En la fase de despliegue se instala la aplicación en un entorno preparado para prueba. Podrán existir varios entornos de pruebas específicos, para que cada equipo pueda trabajar en una parte concreta de la aplicación. El equipo de QA realiza los test de calidad y en esta fase es donde se realiza análisis dinámico de la aplicación (XSS, SQLi,

SSL habilitado, etc.). Se utilizan herramientas típicas de *Ethical Hacking* como *Nmap, Sqlmap, Nikto*, entre otras.

- En la fase de operación la aplicación ya está desplegada y funcionando. Se realizan pruebas de seguridad *RedTeam*, resiliencia, etc.
- En la fase de monitoreo considerando que tenemos *logs* centralizado, se analizan los mismos de forma automática para detectar posibles ataques como XSS, SQLi, DoS, DDoS, etc.

MLDevOps deriva del concepto de DevOps que tiene en cuenta el conjunto de acciones que se realizan de forma cíclica desde el punto de vista del desarrollo del software. Cuando se comenzaron a desarrollar los modelos de ML, los investigadores se dieron cuenta que había actividades que no estaban incluidas en el proceso DevOps o SecDevOps, por ello se tuvieron que incluir tareas adicionales.

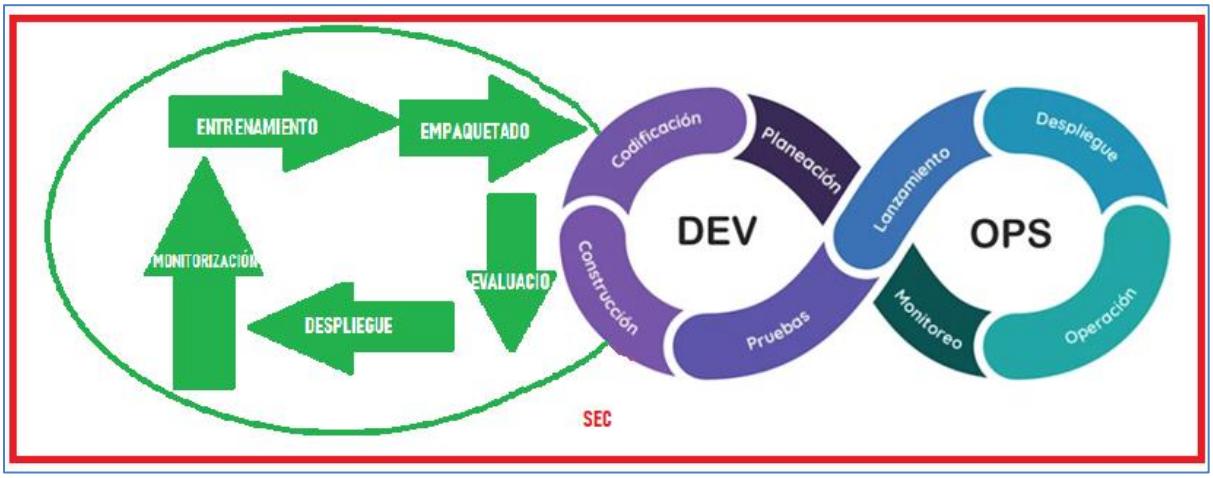

Figura 25: Ciclo de vida MLDevOps

SES Digital (2022). Arquitectura de Software y Cultura Devops. [Figura]. Recuperado de: https://sesitdigital.com/arquitectura-de-software-y-cultura-devops/

Las 5 (cinco) operaciones propias del ML que se incorporan son: evaluación, despliegue, monitorización, entrenamiento y empaquetado, con ellas se completa el ciclo de MLDevOps.

El agregado de las nuevas operaciones se debe a que la construcción de sistema ML incluye particularidades, que se agregan a la de cualquier software tradicional. El proceso de entrenamiento y despliegue (inferencia) de un modelo está formado por cinco operaciones básicas.

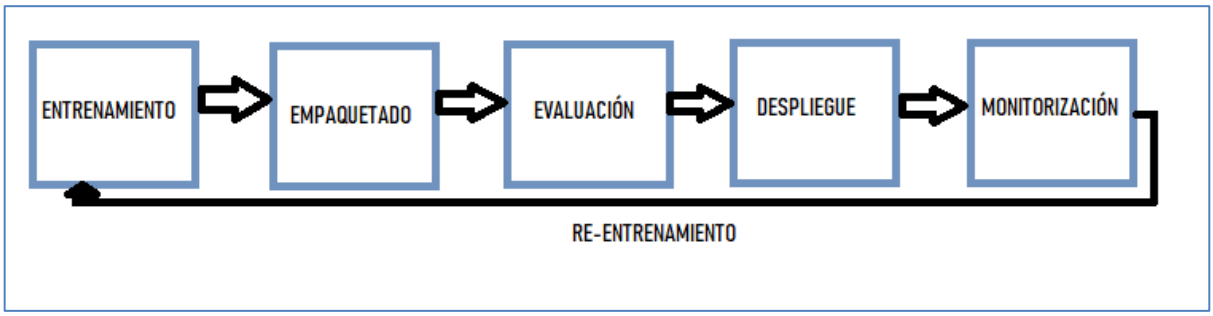

Figura 26: Operaciones básicas de entrenamiento y despliegue

Recuperado de: https://docs.microsoft.com/es-es/azure/architecture/example-scenario/mlops/mlops-technical-paper

- Entrenamiento: preparación de los datos y la ejecución del proceso de entrenamiento. El proceso debe ser reproducible para que se ejecute de igual manera a excepción de los datos.
- Empaquetado: nos permite empaquetar el modelo y facilitar el despliegue. En esta actividad se debe contar con todo el software necesario para la ejecución, sin necesidad de instalar de manera manual ningún tipo de software para realizar el proceso de inferencia o de predicción.
- Evaluación: Además de la evaluación que se realiza en el proceso de entrenamiento, tenemos que hacer una evaluación del paquete, es decir comprobar que ese paquete funciona correctamente y se puede desplegar de manera correcta en los entornos operacionales donde se va a ejecutar.
- Despliegue: Se realiza en un entorno de ejecución válido, que haya sido evaluado previamente en la fase anterior. El despliegue tiene que estar preparado para funcionar en tiempo real y aceptar todas las peticiones que sean necesarias.
- Monitorización: Todo este proceso de desarrollo y el de inferencia debe estar totalmente monitorizado, se tiene que tener información de cada etapa y analizar lo que está ocurriendo.
- Una vez que están contempladas todas las etapas el ciclo se repite para producir un reentrenamiento para mejorar el modelo, esto con la información obtenida de los usuarios que utilizan el modelo o la información que se va incrementando para el proceso de entrenamiento.

A continuación, se mencionan algunas tecnologías y herramientas que se utilizan en el proceso MLDevOps:

- Para entrenamiento: Tensor Flow Extended, Frameworks de desarrollo como: Tensor Flow, Caffee, Keras, SkyLearn, entre otros.
- Para empaquetado: Docker.
- En evaluación: Tensor Flow Extended, Google Cloud.
- En despliegue: Computer Team, Ubernet, Tensor Flow Extended.
- Para monitorización: Dashboard de Data Studio, Tensor Board.

4. Capítulo IV – Implementación de caso de estudio aplicando ML

En el trabajo de tesis se analizaron y realizaron implementaciones de casos de estudio relacionados con ciberseguridad. Para ello se analizaron algunos ejemplos planteados: Filtrado de Spam con ML (Kumar, 2017) y Detección de URL Maliciosas (Schiappa, 2017).

Para la implementación de los ejemplos se utilizó el paquete Anaconda, la herramienta *Jupyter Notebook* con el lenguaje Python, adicionalmente se trabajó en algunos casos con la herramienta Colaboraty de Google. Como actividad técnica para preparar el entorno de desarrollo fue necesario actualizar las herramientas a la última versión, para ello se ejecutaron los siguientes comandos en anaconda prompt:

- Para actualizar a la versión más reciente de la suite: conda update anaconda
- Para actualizar librería scikit-learn: conda update scikit-learn
- Para instalar librerias para DL: conda install -c conda-forge tensorflow pip install keras
- Para instalar opencv: conda install -c conda-forge opencv
- Para instalar wget: conda install -c conda-forge python-wget

4.1 Filtrado de Spam

El filtrado de SPAM es una aplicación relacionada con la clasificación de documentos, consiste en clasificar un email como Spam o No-Spam. Para la implementación de este caso se utilizó el dataset *Ling-spam*. A continuación, se describen las actividades realizadas para implementar el modelo.

4.1.1 Preparación de los datos de texto

Se dividió los datos en dos conjuntos, uno de entrenamiento con 702 correos y otro de prueba con 206 correos, los correos Spam se reconocen ya que contiene *spmsg* en su nombre.

Para realizar la limpieza de datos se quitó palabras de la muestra que no aportaban a la información buscada. Se retiraron palabras como “y", "el", "de", entre otras, que son frecuentes en frases de inglés y no significativos a la hora de decidir si un mensaje es Spam. Otra acción realizada fue la lematización, que consiste en agrupar diferentes derivaciones de una palabra por lo que puede ser analizado como un elemento único. Por ejemplo, "incluir", "incluye" y "incluidos" todos se representarían como "incluir".

4.1.2 Creación de un diccionario

Se procedió a crear un diccionario de palabras con su frecuencia, para ello se utilizó conjunto de datos de entrenamiento de 702 y la función Phyton que se muestra en la Figura 27.

```
In [2]: def make_Dictionary(train_dir):
            emails = [os.path.join(train_dir, f) for f in os.listdir(train_dir)]
            all_words = []
            for mail in emails:
                with open(mail) as m:
                    for i, line in enumerate(m):
                        if i == 2:
                            words = line.split()
                            all_words += words

            dictionary = Counter(all_words)

            #funcion quita palabras y caracteres inservibles
            #list_to_remove = dictionary.keys()
            ml = list(dictionary)


            for item in ml:
                if item.isalpha() == False:
                    del dictionary[item]
                elif len(item) == 1:
                    del dictionary[item]
            dictionary = dictionary.most_common(3000)
            return dictionary
```

Figura 27: código para creación de diccionario. Imagen Propia.

Con el diccionario creado se puede ejecutar tareas de limpieza de código, se muestra su implementación en la misma figura.

4.1.3 Proceso de extracción de características

Con el diccionario listo, se extra un vector que cuenta palabras, de 3000 dimensiones por cada email del conjunto de entrenamiento.

```
In [3]: def extract_features(mail_dir):
            files = [os.path.join(mail_dir, fi) for fi in os.listdir(mail_dir)]
            features_matrix = np.zeros((len(files), 3000))
            docID = 0;
            for fil in files:
                with open(fil) as fi:
                    for i, line in enumerate(fi):
                        if i == 2:
                            words = line.split()
                            for word in words:
                                wordID = 0
                                for i, d in enumerate(dictionary):
                                    if d[0] == word:
                                        wordID = i
                                        features_matrix[docID, wordID] = words.count(word)
                    docID = docID + 1
            return features_matrix
```

Figura 28: código para extracción de características. Imagen Propia.

4.1.4 Entrenamiento de los clasificadores

Con la librería de ML *Scikit-Learn* se realizó el entrenamiento de clasificadores. Se utilizó un clasificador bayesiano ingenuo (*Naive Bayes*) y Máquinas de Vectores de Soporte (SVM).

El Clasificador bayesiano ingenuo es muy utilizado en ML por su simplicidad y rapidez, se usa en clasificación de texto, detección de Spam, análisis de sentimientos, sistemas de recomendaciones, entre otras aplicaciones. Es un clasificador probabilístico supervisado que asume la independencia entre varios pares de características.

Por otro lado, los SVM son clasificadores binarios supervisados muy efectivos cuando se tiene un número grande de características. El objetivo de SVM es separar un subconjunto de datos de entrenamiento del resto llamado vectores de soporte (límite de separación del hiperplano). La función de decisión del modelo SVM que predice la clase de los datos de prueba se basa en vectores de soporte. En la Figura 29 se muestra la codificación:

```
In [1]: import os
        import numpy as np
        import matplotlib.pyplot as plt
        from collections import Counter
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.svm import LinearSVC
        from sklearn.metrics import confusion_matrix
```

```
In [2]: def make_Dictionary(train_dir):
            emails = [os.path.join(train_dir, f) for f in os.listdir(train_dir)]
            all_words = []
            for mail in emails:
                with open(mail) as m:
                    for i, line in enumerate(m):
                        if i == 2:
                            words = line.split()
                            all_words += words

            dictionary = Counter(all_words)

            #funcion quita palabras y caracteres inservibles
            #list_to_remove = dictionary.keys()
            ml = list(dictionary)


            for item in ml:
                if item.isalpha() == False:
                    del dictionary[item]
                elif len(item) == 1:
                    del dictionary[item]
            dictionary = dictionary.most_common(3000)
            return dictionary
```

```
In [3]: def extract_features(mail_dir):
            files = [os.path.join(mail_dir, fi) for fi in os.listdir(mail_dir)]
            features_matrix = np.zeros((len(files), 3000))
            docID = 0;
            for fil in files:
                with open(fil) as fi:
                    for i, line in enumerate(fi):
                        if i == 2:
                            words = line.split()
                            for word in words:
                                wordID = 0
                                for i, d in enumerate(dictionary):
                                    if d[0] == word:
                                        wordID = i
                                        features_matrix[docID, wordID] = words.count(word)
                    docID = docID + 1
            return features_matrix
```

```
In [4]: # Crea un diccionario de palabras con su frecuencia.
        train_dir = 'C:\\programacion\\python\\Mail-Spam-Filtering-master\\train-mails'
        dictionary = make_Dictionary(train_dir)
```

```
In [ ]: # Prepare vectores de características por correo de capacitación y sus etiquetas

        train_labels = np.zeros(702)
        train_labels[351:701] = 1
        train_matrix = extract_features(train_dir)
```

```
In [ ]: # Entrenamiento SVM y clasificador Naive bayes y sus variantes

        model1 = LinearSVC()
        model2 = MultinomialNB()

        model1.fit(train_matrix, train_labels)
        model2.fit(train_matrix, train_labels)
```

```
In [ ]: # Pruebe los correos no vistos en busca de spam

        test_dir = 'C:\\programacion\\python\\Mail-Spam-Filtering-master\\test-mails
        test_matrix = extract_features(test_dir)
        test_labels = np.zeros(260)
        test_labels[130:260] = 1

        result1 = model1.predict(test_matrix)
        result2 = model2.predict(test_matrix)
```

```
In [ ]: print confusion_matrix(test_labels,result1)
        print confusion_matrix(test_labels,result2)
```

Figura 29: Entrenador de clasificadores. Imagen Propia.

4.1.5 Comprobación del funcionamiento

Tomando un conjunto de entrenamiento tiene 130 emails spam y 130 correos electrónicos válidos, en la Tabla 1 muestra los correos Spam detectados correctamente y los correos válidos detectados correctamente discriminados por los algoritmos utilizados.

SVM		
	No Spam	**Spam**
No Spam	126	4
Spam	6	124
Bayes		
	No Spam	**Spam**
No Spam	129	1
Spam	9	121

Tabla 1: Comprobación de funcionamiento Filtrado de Spam con ML

Ambos modelos tuvieron un rendimiento similar, SVM tuvo mayor número de falsos positivos SPAM (4) en el conjunto de datos de correo deseado y mayor acierto (124) al detectar los SPAM.

4.2 Detección de URL Maliciosas con DL

De manera tradicional hay dos métodos de detección de url maliciosas. El primero es por firmas donde se manejan listas negras RBLs que son consultadas cada vez que un cliente quiere acceder a ella; el segundo es a través de expresiones regulares destinadas a crear url maliciosas y sus variantes. Estos métodos involucran la participación de analistas humanos que constantemente revisan las actualizaciones de firmas o expresiones regulares, es un método reactivo ya que antes de pasar a una lista negra tiene que haber afectado a algunos usuarios. Para la implementación de la aplicación de DL se usaron las herramientas Keras, Tensorflow y scikit-learn y los dataset con ejemplos de url maliciosas dirty.csv y url limpias clean.csv con miles de ejemplos en las mismas.

4.2.1 Preparación de los datos de entrada

Para esta tarea se utilizó la librería SkLearn que tiene algoritmos para la extracción de características que se puede dividir en vectores. También NLTK (un conjunto de herramientas de lenguaje natural) y mmh3 para codificar cada URL en un vector (tensor) numérico de longitud 1000, se divide la url en ngramas.

```
In [4]: # Esta función tomará los 3 n-gramas de la URL y los convertirá en un vector de longitud 1000
        def eng_hash(data, vdim=1000):
            final = []
            for url in data:
                v = [0] * vdim
                new = list(ngrams(url, 3))
                for i in new:
                    new_ = ''.join(i)
                    idx = mmh3.hash(new_) % vdim
                    v[idx] += 1
                final.append([np.array(v)])
            return final
```

Figura 30: Extracción de características. Imagen Propia.

Una vez que los vectores de características están en un lenguaje que la RNA puede entender, se divide a los datos en un conjunto de entrenamiento y un conjunto de pruebas, para ello se utiliza SKLearn.

```
# ahora divide el tiempo
p_cut = 70.0
percentile = np.min((np.percentile(first_seen[y_label==0], p_cut), np.percentile(first_seen[y_label==1], p_cut)))

train = []
test = []
for i, v in enumerate(first_seen):
    if v < percentile and y_label[0][i] >= 0:
        train.append(i)
    elif v >= percentile and y_label[0][i] >= 0:
        test.append(i)

cv = [[np.array(train), np.array(test)]]
```

Figura 31: División de los conjunto de datos. Imagen Propia.

Para la división de los datos para el training/validación de los archivos .csv se pueden optar por los siguientes criterios. Batch: cantidad de datos a tomar (número de tensores/vectores); Epochs: repeticiones con el mismo dataset. El modelo aprende por realimentación optimizando los pesos con las repeticiones.

4.2.2 Construcción del modelo

En la construcción del modelo, para el desarrollo de la RNA se utiliza Keras que permite construir un modelo capa por capa.

```
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, Activation
```

Figura 32: Definición del modelo con Keras. Imagen Propia.

En cada capa se determina que funciones va a utilizar y el número de neuronas que tiene. Keras nos permite abstraernos de las distintas capas neuronales. En este ejemplo con un modelo de un vector (tensor) de longitud 1000, se propone hacer 128 capas (densas) y una normalización aleatoria del 15% usando la función de activación ReLu.

```
def construct_model(model_type):
    log.info("Constructing model")
    if model_type=='deep':
        model = Sequential()

        # capas ocultas
        model.add(Dense(128, input_dim=1000))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        model.add(Dropout(.15))
```

Figura 33: Definición de capas. Imagen Propia.

El Dropout del 15% significa quitar el 15% de los enlaces entre neuronas de forma aleatoria.

```
model.compile(loss='binary_crossentropy',
              optimizer='SGD',
              metrics=['accuracy'])
```

Figura 34: Definición de métricas. Imagen Propia.

4.2.3 Datos de entrenamiento y validación del modelo

Para esta etapa las muestras se introducen en el modelo en lotes de 128 vectores y pasan a través de todo el conjunto de entrenamiento 20 veces. Se alimenta el modelo de los vectores URL para comparar el resultado del modelo a las etiquetas (- 1) para malicioso y (0) para benigno.

```
In [7]: def train_model(X_train, y_train, model):
            log.info("Beginning training model")
            loss = LossHistory()
            model.fit(X_train, y_train,
                      epochs=20,
                      batch_size=128, verbose=1, callbacks=[loss])
            return model, loss
```

Figura 35: Entrenamiento del modelo. Imagen Propia.

4.2.4 Evaluación el rendimiento del modelo

El modelo aprende a través de *backpropagation* minimizando el error de entropía cruzada para encontrar los pesos óptimos. Empleamos las siguientes funciones matemáticas para medir la calidad del sistema.

TP= *True Positive* → URL maliciosa detectada como maliciosa

TN: *True Negative* → URL benigna detectada como benigna

FP: *False Positive* → URL benigna detectada como maliciosa

FN: *False Negative* → URL maliciosa detectada como benigna

Precisión= (TP/(TP+F)

Tasa de true positive → TPR= (TP/(TP+FN)

Tasa de false positive → FPR=(FP/(FP+TN)

Cuando se ejecuta con cada lote de datos (EPOC) se ve el desempeño.

Al final se debe analizar la tasa de TPR/FPR y para ello se evalúa el gráfico ROC para evaluar el rendimiento, cuanto más se alimente al sistema mejores tasas de detecciones se van a tener, esto variando la tasa de detecciones.

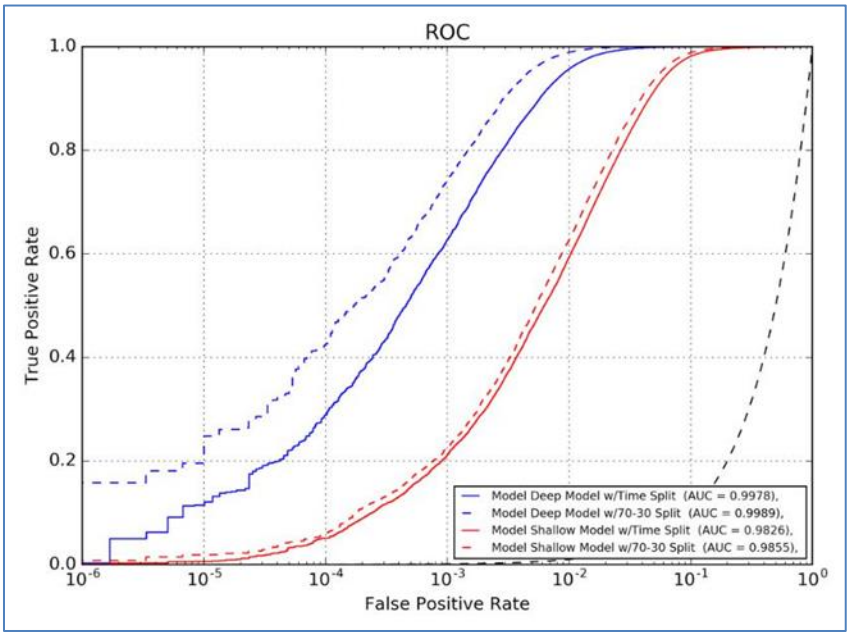

Figura 36: Grafico ROC (Shopos Iberia)

Recuperado de: https://www.youtube.com/watch?v=wdG3NDd5fY0&t=1466s

5. Conclusiones y Trabajos Futuros

Mediante la exposición del marco teórico se repasó los hitos de la IA hasta sus aplicaciones actuales relacionada con ML. Con el análisis del Estado del Arte de la aplicación de ML en el campo de la ciberdefensa y ciberseguridad, se pudo comprobar y contrastar su aplicación en diferentes áreas, junto con el esfuerzo de empresas y gobiernos en realizar avances en la materia para lo cual destinan cuantiosos recursos económicos para investigación y desarrollo.

La continua evolución de amenazas cibernéticas proporciona un ámbito propicio para la aplicación de ML como complemento de herramientas existentes, esto por sus capacidades de adaptación y de aprendizaje. Del análisis de productos de empresas líderes, se pudo observar la creciente aplicación de ML en sistemas, servicios o productos, tanto para la faz ofensiva como defensiva.

Considero que ML se aplica con éxito en la solución de problemas reales, esto se ve reflejado por ejemplo en motores de exploración, velocidad de detección y en la capacidad de identificar irregularidades, esto entre otros tantos ejemplos. Todo esto contribuye a proteger mejor las infraestructuras, en especial frente a las amenazas nuevas y emergentes, así como las APT.

Si bien ML contribuye a potenciar soluciones tradicionales y se observa un aumento creciente de su aplicación, cuenta con limitaciones que se deben tener en cuenta. Una limitación es que se necesita disponer de una gran cantidad de datos representativos del dominio que se está analizando, que no siempre es fácil. Otra limitación es la necesidad de contar con importante poder de cómputo necesario para su entrenamiento y ejecución.

Considero que ML no se debe considerar como elemento exclusivo, sino que lo más adecuado es desplegar una solución en varias capas, para aprovechar el poder y potencial de IA y ML, pero con el respaldo de otras tecnologías de detección y protección, siendo en este punto la componente humana un factor diferencial e imprescindible.

ML es de utilidad en la faz ofensiva como defensiva. En la faz de seguridad defensiva ML se aplica en áreas como las telecomunicaciones, seguridad de red, en análisis de tráfico, fuga de información, detección de malware, detección de anomalías de dispositivos IoT, análisis de comportamiento de usuarios, criptografía, detección de *Spam* y *antiphishin*g, biometría, entre otras áreas. En el relevamiento efectuado en el trabajo final se pudo apreciar que se dispone de *frameworks*, herramientas y librerías que permiten analizar la robustez de los sistemas ML con sus características particulares, lo que permite reforzar los esquemas de protección con un panorama de amenazas creciente.

En la faz ofensiva, habiendo analizado los tipos de ataques que puede sufrir un sistema ML, se pudo comprobar que se puede usar para mejorar ataques clásicos como SQLi, descifrado de contraseñas, entre otros. Los atacantes ataques que busquen afectar vulnerabilidades propias de los sistemas ML, por ejemplo, de evasión, envenenamiento, extracción de modelos, entre otros.

En ciberseguridad los atacantes no siguen pautas ni aceptan limitaciones, incluso pueden cambiar las reglas de juego completo sin ninguna advertencia. Desde el punto de vista de ingeniería de software se debe considerar la incorporación de la seguridad en todas las fases del ciclo de vida de desarrollo de un sistema ML. Considero que tener conocimiento de avances en metodologías, técnicas y herramientas que propicien desarrollo y operación segura de sistemas ML con sus características especiales que difieren de sistemas convencionales, es un objetivo a tener en cuenta. Y sumados a la creciente aplicación de ML en diferentes áreas, es necesario fortalecer la formación en IA de profesionales del área. Se destaca una tendencia de migración de investigadores formados en IA del sector académico al sector privado.

Considero que el trabajo final constituye un punto de partida para visualizar las aplicaciones de ML en ciberseguridad y ciberdefensa, al respecto recomiendo para los que deseen seguir esta línea de investigación profundizar en las aplicaciones relacionadas con DL y otras tecnologías de IA.

6. Bibliografía y Referencias

10th International Conference on Cyber Conflict. (2018). *On the Effectiveness of Machine and Deep Learning for Cyber Security*. NATO CCD COE *Publications*, Tallinn.

Abdelhamid, N., Ayesh, A. y Thabtah, F. (2014). Phishing detection based associative classification data Mining. Expert Systems with Applications.

Alonso, C. (11 de Febrero de 2019). SecDevOps. Recuperado el 2 de octubre de 2021, de https://www.elladodelmal.com/2019/02/secdevops-una-explicacion-en-cinco.html

Andrew Ng. (2018). *Machine Learning Yearning*. *Stanford University*. Deeplearnin.ai.

Argentina.gob.ar. Resolución 1523/2019 (2019). Definición de Infraestructuras Críticas de Información y Glosario de Términos de Ciberseguridad. Jefatura de Gabinetes de Ministros de la República Argentina. Recuperado de: https://www.argentina.gob.ar/normativa/nacional/resoluci%C3%B3n-1523-2019-328599

Ayerbe, A. (2020). La ciberseguridad y su relación con la inteligencia artificial. ARI 128/2020 - Real Instituto Elcano, 8.

Barr, A. Feigenbaum, E. (1981). Handbook of Artificial Intelligence. Butterworth-Heinemann.

Borowiec, S. y Lien, T. (17 de octubre de 2019). AlphaGo beats human Go champ in milestone for artificial intelligence. Los Angeles Times. Recuperado de: http://www. latimes.com/world/asia/la-fg-korea-alphago-20160312-story.html

Bourciere, D. (2003). Inteligencia Artificial y Derecho. Editorial UOC.

CISCO. (Noviembre de 2021). AI and Machine Learning. Recuperado de: https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/digital-network-architecture/nb-06-cisco-dna-ai-ml-primer-cte-en.html

Citomic. (6 de Noviembre de 2020). Adversarial ML Threat Matrix: la respuesta de MITRE ante las amenazas contra la IA. Recuperado el 6 de Noviembre de 2021, de https://www.cytomic.ai/es/tendencias/adversarial-ml-threat-matrix/

Darktrace. (7 de Noviembre de 2021). Seguridad basada en autoaprendizaje. Recuperado el 7 de Noviembre de 2021, de https://www.darktrace.com

DARPA. (7 de Noviembre de 2021). Lifelong Learning Machines (L2M). Recuperado el 7 de Noviembre de 2021, de https://www.darpa.mil/program/lifelong-learning-machines

FATE, S. (2017). The Modern Mercenary. Oxford University Press.

Ferrado L. y Cuenca M. (Octubre de 2021). Filtrando eventos en seguridad en forma conservativa mediante deep learning. Recuperado de: http://sedici.unlp.edu.ar/bitstream/handle/10915/56884/Documento_completo.pdf-PDFA.pdf?sequence=1

Fortinet. (Noviembre de 2021). Operaciones de seguridad. Recuperado de: https://www.fortinet.com/lat/solutions/enterprise-midsize-business/security-operations

Futuro FHI. (2022). Instituto del Futuro de la Humanidad. Recuperado de: https://www.fhi.ox.ac.uk/.

HAI Stanford University Human Centered Artificial Intelligence. Recuperado 20/06/2022 de: https://hai.stanford.edu/ai-index-2021

Gala Barxa Iria y Alvarez J. (Noviembre de 2021). Tecnología de IA para determinar la necesidad de respuesta frente a ciberataques. Recuperado de Fujitsu EMEIA: https://www.fujitsu.com/es/about/resources/news/press-releases/2019/spain-fujitsu-desarrolla-una-tecnolog-a-de-ia-para.html

Gallegos, J. y Soto, A. (2014). Inteligencia Artificial. Iniciativa Latinoamericana de textos abiertos.

Gil, J.M. (2019 de 7 de 7). La Integración Del Ciberespacio En El Ámbito Militar. Seguridadinternacional.es. Recuperado de: http://www.seguridadinternacional.es/?q=es/content/la-integraci%C3%B3n-del-ciberespacio-en-el-%C3%A1mbito-militar

Google, D. (Enero de 2019). Machine Learning Crash Course. Recuperado de: https://developers.google.com/machine-learning/crash-course/ml-intro

Hernandez M. y Escribano J. (Noviembre de 2021). Adversarial Machine Learning (parte IV): ataques de envenenamiento. Recuperado de: https://www.bbvanexttechnologies.com/pills/adversarial-machine-learning-ataques-de-envenenamiento/

Hernández M. y Escribano J. (Noviembre de 2021). Adversarial Machine Learning (parte V): ataques de evasión. Recuperado de: https://www.bbvanexttechnologies.com/pills/adversarial-machine-learning-parte-v-ataques-de-evasion/

Hitaj B., Ateniese, G., Gasti, P. y Perez-Cruz, F. (2019). PassGAN: A Deep Learning Approach for Password Guessing. Arxiv.

Hsiao, W. y Chang T. (2008). An incremental cluster-based approach to spam filtering. Expert Systems with Applications.

Incibe. (Enero de 2019). Glosario de términos de ciberseguridad. Recuperado de: https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia_glosario_ciberseguridad_metad.pdf

IT digital Security. (24 de Junio de 2020). Palo Alto lanza su primer firewall con machine learning integrado. Recuperado el 12 de febrero de 2021, de https://www.itdigitalsecurity.es/endpoint/2020/06/palo-alto-lanza-su-primer-firewall-con-machine-learning-integrado

Karspersky MLAD. (6 de Noviembre de 2021). Kaspersky Machine Learning. Recuperado el 6 de Noviembre de 2021, de https://mlad.kaspersky.com/

Kubovic, O. (2021). ¿Es posible que la inteligencia artificial potencie el malware en el futuro? ESET.

Kumar, A. (23 de Junio de 2017). Machine learning in action. Recuperado el 21 de 08 de 2019, de Machine learning in action: https://appliedmachinelearning.blog/2017/01/23/email-spam-filter-python-scikit-learn/#comments

Lopez Briega, R. (2019). Libro Online de IIAR. Recuperado el 6 de Noviembre de 2021, de https://iaarbook.github.io/

McCarthy, J., Marvin L. M., Rochester, N. y Claude E. S. (17 de octubre de 2019). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. Recuperado de http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html.

Middleton, B. (2017). A History of Cyber Security Attacks 1980 to Present. Taylor & Francis Group

Mlsec. (7 de Noviembre de 2021). Machine Learning. Recuperado el 2021 de enero de 2021, de http://www.mlsec.org/

Muñoz, A. y Escribano, J. (octubre de 2021). Criptografía adversaria usando deep learning, limitaciones y oportunidades. Recuperado de BBVA: https://www.bbvanexttechnologies.com/wp-content/uploads/2018/07/amunoz-jescribano-criptograf%C3%ADa-adversaria-recsi-2018.pdf

Ottis R. (7 de 7 de 2019). Analysis of the 2007 Cyber Attacks Against Estonia from the Information Warfare Perspective. Recuperado de: CCDCOE https://ccdcoe.org/uploads/2018/10/Ottis2008_AnalysisOf2007FromTheInformationWarfarePerspective.pdf

Owain E. (2018). Predicting Human Deliberative Judgments. FHI Oxford Technical Report #2018-2, 21.

Owain E., Stuhlmüllery, A., Cundy, C., Careyy, R. y Zachary. (2018). Predicting Human Deliberative Judgments. FHI Oxford Technical Report #2018-2, Junio.

Pacheco, V. G. (18 de enero de 2019). LUCA Think Big. Obtenido de Víctor González Pacheco

Point, C. (16 de Octubre de 2018). La inteligencia artificial al servicio de la ciberseguridad. Recuperado el 5 de Febrero de 2020, de https://www.checkpoint.com/es/press/2018/la-inteligencia-artificial-al-servicio-de-la-ciberseguridad/

Rieck, K., Trinius, P., Willems, C. y Holz, T. (2011). Automatic Analysis of Malware Behavior Using Machine Learning. Journal of Computer Security.

Sampieri, R. (2014). Metodología de la Investigación sexta edición. Mc Graw Hill Education.

Schiappa, M. (2017). Machine Learning: How to Build a Better Threat Detection Model. Shopos Thecnical papers, 16.

Sculley, D., Holt,G., Golovin, D., Dvydov, E. y Phillips, T. (2014). Hidden Technical Debt in Machine Learning Systems. SE4ML whorshop, 9.

Stanford University. (17 de Octubre de 2019). One Hundred Year Study on Artificial Intelligence (AI100). Recuperado de: https://ai100.stanford.edu

STANLEY, B.E. (2015). Outsourcing Security – Private Military Contractors and U.S. Foreign Policy. Potomac Books.

Sullivan, E. (19 de 12 de 2018). Contratista de seguridad de Blackwater declarado culpable. The New York Times. Recuperado de: https://www.nytimes.com/2018/12/19/us/politics/blackwater-security-contractor-iraq-shooting.html

Tianyu Gu, Brendan Dolan-Gavitt y Siddharth Garg. (2019). BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. Arxiv, 13.

Trama G. A. (2017). Operaciones Cibernéticas. Revista Visión Conjunta N°17. Escuela Superior de Guerra conjunta de las Fuerzas Armadas.

Uzal, R., Riesco, D., Montejano, G., Agüero, W. (2015). Lavado transnacional de activos en el ciberespacio. Sociedad Argentina de Informática e Investigación Operativa (SADIO).

Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu y Jinwen He. (2020). Towards Security Threats of Deep Learning. IEEE transactions on software engineering.