Implementación de un sistema de bloqueo para URLs de pornografía infantil en Colombia con el uso de herramientas de Machine Learning

JOHN MAURICIO FREYRE

Director:
Ing. Gustavo Chica, PhD / MSc

Codirector:
Ing. Jesús Guzmán, PhD / MSc

UNIVERSIDAD SANTO TOMAS
FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES
MAESTRÍA EN TELECOMUNICACIONES Y REGULACIÓN TIC
BOGOTÁ, 2025

Dedicatoria

A mi madre, **Fabiola Freyre**, por su amor, fortaleza y apoyo incondicional en cada etapa de mi vida.

A mi abuela, **Blanca Benavides**, por su sabiduría, cariño y ejemplo constante de perseverancia.

A **Karen Zarama**, por su amor, compañía y apoyo incondicional.

Este logro es tan mío como suyo.

AGRADECIMIENTOS

Agradezco profundamente a la comunidad universitaria de la **Universidad Santo Tomás**, por brindarme un espacio de formación integral, reflexión crítica y crecimiento profesional a lo largo de este proceso académico.

Al **Ingeniero Gustavo Chica, PhD**, director de este trabajo, por su guía, conocimiento y compromiso, los cuales fueron fundamentales para la culminación exitosa de esta investigación.

A todos ustedes, gracias por hacer parte de este logro.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ACRÓNIMOS

Accuracy	Métrica que indica la proporción de predicciones correctas realizadas por un modelo sobre el total evaluado.
API	*Application Programming Interface*. Conjunto de funciones que permiten la interacción entre aplicaciones.
Backend	Parte del desarrollo web encargada del procesamiento en el servidor, bases de datos y lógica de negocio.
CONPES	*Consejo Nacional de Política Económica y Social*. Documento técnico del gobierno colombiano que establece políticas públicas.
CSMA	*Carrier Sense Multiple Access*. Protocolo de red que evita colisiones al detectar si el canal está libre.
DataFrame	Estructura de datos bidimensional en Python (Pandas) usada para manipular datos tabulares.
DDoS	*Distributed Denial of Service*. Ataque que satura un sistema con múltiples solicitudes, provocando su caída.
DNN	*Deep Neural Network*. Red neuronal profunda con múltiples capas ocultas para el aprendizaje no lineal.
Frontend	Parte visual de una aplicación web con la que interactúa el usuario final.
GitHub	Plataforma de control de versiones para alojar y colaborar en proyectos de desarrollo con Git.
Grafos de flujo	Representaciones gráficas de procesos computacionales mediante nodos y conexiones (ej. TensorFlow).
GPU	*Graphics Processing Unit*. Procesador especializado en cálculos paralelos, útil en IA y aprendizaje profundo.
HTTP	*HyperText Transfer Protocol*. Protocolo de transferencia de datos utilizado en la web.
IA	*Inteligencia Artificial*. Rama de la informática enfocada en crear sistemas que simulan inteligencia humana.
IPv4	*Internet Protocol version 4*. Protocolo de direccionamiento con direcciones de 32 bits.
IPv6	*Internet Protocol version 6*. Versión mejorada del protocolo IP, con direcciones de 128 bits.
ISP	*Internet Service Provider*. Empresa que proporciona acceso a Internet a usuarios y organizaciones.
Jupyter Lab	Entorno interactivo en Python para análisis de datos, pruebas de código y documentación integrada.
Kaggle	Plataforma para ciencia de datos que ofrece datasets, retos de modelado y colaboración entre usuarios.
Kernel	Función matemática utilizada en algoritmos como SVM para transformar los datos a espacios más complejos.
Post-poda	Técnica en árboles de decisión que elimina ramas no significativas tras el entrenamiento, evitando sobreajuste.
RFC 1808	*Request for Comments 1808*. Documento técnico que define la sintaxis de URLs relativas.
RMSprop	*Root Mean Square Propagation*. Algoritmo de optimización que ajusta la tasa de aprendizaje en redes neuronales.
SGD	*Stochastic Gradient Descent*. Algoritmo de optimización que ajusta parámetros del modelo en función de muestras aleatorias.
URL	*Uniform Resource Locator*. Dirección web que identifica un recurso en Internet.

RESUMEN

La difusión de la pornografía infantil en línea es una grave problemática que requiere de soluciones tecnológicas para mitigar su expansión en Colombia. El interés de este estudio tiene como objetivo principal en diseñar e implementar un modelo de aprendizaje automático que bloquee URLs con información ilegal por medio de su enfoque en el uso de herramientas de Machine Learning, pertinentes para identificar y bloquear contenido con patrones asociados a la pornografía de menores, en tiempo real.

El alcance de esta investigación tiene la intención de garantizar precisión y efectividad en la detección de URLs ilícitas, controlar el acceso a las mismas, luchar contra su distribución, proteger los derechos del niño en el entorno digital y contribuir a los espacios virtuales seguros.

Palabras clave: modelo; machine Learning; Python; URLs: pornografía infantil.

ABSTRACT

The spread of child pornography online is a serious problem that requires technological solutions to mitigate its expansion in Colombia. The main objective of this study is to design and implement a machine learning model that blocks URLs with illegal information through its focus on the use of Machine Learning tools, relevant to identify and block content with patterns associated with child pornography, in real time.

The scope of this research is intended to guarantee precision and effectiveness in the detection of illicit URLs, control access to them, fight against their distribution, protect the rights of children in the digital environment and contribute to safe virtual spaces.

Keywords: model; Machine Learning; Python; URLs; child pornography.

INTRODUCCIÓN

El tema sobre la pornografía infantil en línea se ha convertido en una prioridad, no solo para Colombia, sino para los gobiernos y organizaciones internacionales. La creciente expansión del internet y el fácil acceso a sitios web, propende a que se distribuya y se comercialice contenidos ilegales de todo tipo, al punto de convertirse en una gran preocupación.

En lo que respecta al país, esta problemática ha impulsado a la búsqueda de soluciones tecnológicas que permitan la detección y bloqueo que sean eficientes al momento de detectar material relacionado con la explotación a menores. En este caso, el uso de herramientas como Machine Learning serán las alternativas que favorecerán en gran medida, a las estrategias de protección digital e identificación automática en la localización de contenido sensible.

La propuesta de implementar un sistema de bloqueo de URLs de pornografía infantil basado en el uso de algoritmos de Machine Learning, tiene el objetivo de identificar, filtrar y bloquear por medio de un modelo automático y efectivo, todo índice ilegal relacionado con la vulneración del niño. De manera general, este patrón de aprendizaje busca integrar un enfoque innovador y con potencial para mejorar las estrategias de ciberseguridad, para contribuir a la protección de los derechos de los infantes y garantizar la seguridad en los entornos digitales.

1 MARCO GENERAL DEL PROYECTO

En este capítulo se señalarán los objetivos con los cuales se abordará la problemática relacionada con la difusión en línea de pornografía infantil en Colombia, una modalidad grave que atenta y vulnera el derecho a la integridad de la niñez, mediante el uso de herramientas digitales. De igual modo, el desarrollo de cada propósito determinará el óptimo alcance del proceso de filtrado Firewall, el cual integrará el software para gestionar el bloqueo de contenidos de internet de esta índole, en aras de contribuir a la prevención de la explotación sexual infantil.

Problema: Para llevar a cabo el bloqueo de este tipo de contenido, se debe recurrir a medios tecnológicos como sistemas de firewall que puedan contener el acceso a contenido sensible el cual reporta MINTIC a todos los ISP. Sin embargo, algunos administradores de estas páginas se las han arreglado para pasar desapercibidos por el filtro Firewall. Lo que lleva a la pregunta: ¿Cómo podemos mejorar el procedimiento de filtrado?

Justificación: Nace de la necesidad de fortalecer los mecanismos actuales de bloqueo de contenido de pornografía infantil, lo cual vuelve vulnerable los derechos de los niños, e infringe la ley 599 de 2000, de igual manera la ley 679 de 2001. Articulo 8 Deberes. A pesar de que existen dichas leyes y medidas de protección contra la pornografía infantil, este problema sigue siendo una amenaza real y creciente en la sociedad. Por tal motivo es necesario el desarrollo de un software que permita mejorar el proceso de bloqueo de este tipo de contenidos en internet, y pueda contribuir de manera significativa a la protección de los menores y a la prevención de la explotación sexual infantil.

1.1 OBJETIVOS

1.1.1 Objetivo General

Implementar un sistema de bloqueo para URLs de pornografía infantil en Colombia con el uso de herramientas de Machine Learning.

1.1.2 Objetivos Específicos

- Identificar el estado actual de las técnicas de ciberseguridad utilizadas para controlar la pornografía infantil en Colombia, por medio de la revisión sistemática de literatura.

- Proponer el nuevo algoritmo o modelo de aprendizaje automático que identifique el contenido de pornografía infantil en línea y bloquee las URLs que no obedecen a los filtros de seguridad de la plataforma Allot en Colombia.
- Desarrollar un programa en Python que ponga en marcha el algoritmo o modelo de aprendizaje propuesto en el anterior objetivo.

1.2 ALCANCE

Para bloquear el contenido de pornografía infantil en Colombia, se desarrollará un sistema de filtrado con el uso de herramientas de Machine Learning para URLs que solamente MINTIC reporta en su plataforma virtual y donde únicamente tienen acceso y autorización los ISP. En este sentido, del resultado del primer filtrado con ALLOT, se implementará un segundo, bajo el diseño de un sistema de bloqueo mejorado que utilice herramientas de scripting para detectar y bloquear las URLs que el Firewall no filtró en su momento.

Teniendo en cuenta lo anterior, se recopilarán datos de contenido explicito para el entrenamiento del modelo de aprendizaje, el cual permitirá identificar los patrones de conducta similares en las URL que lograron pasar el primer filtro y así poder usar este mecanismo para el filtrado de futuras URLs. Como acto seguido, se evaluará la efectividad del sistema del bloqueo mejorado y se determinarán ajustes adicionales de ser necesario; y luego, se documentará todo el proceso de investigación, los resultados, las recomendaciones y demás aspectos, como fuentes de información de interés, para futuras investigaciones.

Las técnicas de ciberseguridad que se van a consultar para el proceso de lucha contra la pornografía infantil son un compilado de trabajo de varias entidades como: NCMEC, ICMEC, IWF y ECPAT, de las que se tendrá en cuenta su modelo de identificación y el bloqueo de páginas con este tipo de contenido y, además, se resaltará las funciones de las IA como Thorn y NetClean en relación al proceso de filtrado de cada una de ellas.

Respecto al modelo de aprendizaje propuesto, la usabilidad de este tendrá la función de analizar el segundo listado que comprende entre 10 y 15 URLs con material pornográfico, de las cuales se pretende bloquear un 30%. Además, este nuevo algoritmo, abordará técnicas de ML que se llevarán a cabo en un entorno controlado y a nivel local, con la intención de contrarrestar la problemática sin ningún fin lucrativo o que sea referido como producto.

Por otro lado, el resultado de bloquear el 30 % de las URLs del segundo listado en mención, proyectará un banner que proporcionará información sobre el estado del bloqueo y finalmente, cabe resaltar que el patrón de comportamiento puede variar por las actualizaciones de

redireccionamiento, las cuales harán que el código aprenda el nuevo comportamiento para poder bloquear este tipo de contenido.

1.3 METODOLOGIA

El tipo de contenido que se va a bloquear es todo aquel que estén involucrados menores de edad en actos ilegales como la pornografía infantil. El filtrado se realizará por medio del informe que genera MINTIC en su plataforma, el cual adjunta un listado con todas las URLs que se deben bloquear para los ISP en todo el país. Una vez identificada la actualidad de las técnicas de ciberseguridad utilizadas para controlar la pornografía infantil en Colombia (primera Etapa de la metodología), el desarrollo de las etapas dos y tres, correspondientes a la construcción del algoritmo y de la implementación del software, se desarrollarán en 6 fases:

La primera fase es el levantamiento de requerimientos para el desarrollo del software; Es necesario obtener información a través de requerimientos funcionales y no funcionales. La segunda fase es el análisis de los patrones de comportamiento, para lo cual se pretende que toda la información brindada por los pasos anteriores se pueda integrar un script por medio de IA el cual pueda identificar el patrón de comportamiento sobre las URLs que el Firewall no pudo filtrar. La tercera fase es el desarrollo del programa; una vez que se tiene identificado el patrón de comportamiento y el modelo de IA entrenado, se puede comenzar a desarrollar el programa de bloqueo de contenido de pornografía infantil. La cuarta fase son las pruebas y evaluación para poder probar el programa en un entorno controlado para evaluar su eficiencia. La quinta fase es la documentación y presentación; para lo cual, todo el proceso e investigación será documentado. La última fase será la implementación y el mantenimiento; Una vez que se ha desarrollado y probado el programa, se puede implementar en un entorno de producción. En este caso se manejará en un entorno controlado.

2 IDENTIFICACIÓN DEL ESTADO ACTUAL DE LAS TÉCNICAS DE CIBERSEGURIDAD UTILIZADAS PARA CONTROLAR LA PORNOGRAFÍA INFANTIL EN COLOMBIA, POR MEDIO DE LA REVISIÓN SISTEMÁTICA DE LITERATURA.

La pornografía infantil es un fenómeno delicado que se ha expandido y convertido en objeto de difusión digital a nivel mundial, destacándose como la situación más preocupante y alarmante para los gobiernos. Frente a esta problemática, la cooperación internacional de casi todas las naciones del mundo, han establecido estrategias para poder minimizar el impacto de este delito mediante sus sistemas legislativos con la de prohibición de la producción, distribución, circulación y posesión de este tipo de contenido.

Actualmente, gracias a la tecnología, se ha logrado mitigar temáticas de esta índole, por medio del uso de herramientas y sistemas de filtrados digitales que tienen la tarea de detectar y bloquear contenido ilegal desde los ISP, garantizando en gran medida, la seguridad. Por otro lado, existen varias organizaciones a nivel global que trabajan por la integridad de la niñez, como por ejemplo: National Center for Missing and Exploited Children (NCMEC) de USA [1], International Centre for Missing and Exploited Children [2], Internet Watch Foundation (IWF) del reino Unido [3] y ECPAT International [4], las cuales son organismos sin fines de lucro que trabajan arduamente por la protección la niñez en relación con la amenaza latente de la explotación sexual y pornografía infantil en línea.

Cabe resaltar, que, entre los aspectos para reducir el impacto y circulación de la pornografía infantil, se debe considerar el de establecer programas de educación y concientización dirigidos a padres, maestros y niños, en los que se enseñe a identificar y a denunciar esta modalidad de delito. Del mismo modo, esta inclinación investigativa es fundamental para desarrollar estrategias efectivas encaminadas a la identificación de patrones y tendencias en la producción y distribución de este fenómeno, lo que sería un aporte significativo para la reducción de dicha problemática.

En consecuencia, para comprender a profundidad este tipo de contenidos ilegales presentes en internet, es pertinente establecer un marco teórico conceptual que permita conocer el estado actual de los métodos de la ciberseguridad usados para controlar el tema de pornografía infantil específicamente en Colombia, mediante una estricta revisión documental que resalta conceptos importantes y consolida lo ya expuesto. Para ello se tendrán en cuenta las siguientes nociones:

2.1 IA para detectar pornografía infantil

Debido a la naturaleza del problema sobre la pornografía infantil, no es común que las organizaciones encargadas de desarrollar sistemas de Inteligencia Artificial (IA) para el bloqueo de contenido de pornografía infantil, divulguen públicamente sus nombres. Sin embargo, existen algunas instituciones que ofrecen soluciones de software de IA para la detección y el bloqueo de contenido de pornografía infantil en línea, como son:

Thorn: Esta organización sin fines de lucro ha desarrollado una herramienta de IA llamada Spotlight que utiliza un aprendizaje automático para detectar imágenes de abuso infantil en línea. La herramienta funciona mediante la identificación de características específicas de las imágenes y la comparación con una base de datos de imágenes conocidas de abuso infantil [5].

NetClean: Es una empresa que ofrece soluciones de software para la detección y el bloqueo de contenido de pornografía infantil. Sus herramientas utilizan algoritmos de aprendizaje automático para analizar el contenido y detectar imágenes de abuso infantil en tiempo real [6].

Safer Networking: Es una empresa que desarrolla software antivirus que incluye una función de bloqueo de contenido de pornografía infantil. Su solución utiliza tecnología de IA para analizar el contenido y detectar sitios web con imágenes de abuso infantil [7].

Según lo anterior, es importante tener en cuenta que la implementación de sistemas de IA para el bloqueo de contenido de pornografía infantil es solamente es una pequeña parte de la solución y por lo tanto se requiere implementar un enfoque multifacético que permita abordar la explotación infantil en línea, de manera significativa.

2.2 Tipos de inteligencia artificial

En este apartado, se ahondará sobre los tipos de IA fundamentales, con las que se abordará el problema. En este caso, se subrayan los siguientes aspectos:

2.2.1 Aprendizaje Autónomo (Machine Learning)

El Machine Learning está orientado al aprendizaje por medio del entrenamiento del ser humano en la máquina para reconocer patrones basándose en datos, estos pueden partir de una imagen, un video, comportamiento, rutina entre otras acciones que permitan a la máquina predecir el siguiente movimiento o acción a realizar[8].

2.2.2 Sistemas expertos

“Los sistemas expertos son una de las formas más simples de inteligencia artificial y, aunque generalmente no somos conscientes de ello, son tecnologías que nos acompañan todo el tiempo y forman parte de nuestro día a día”[9].

“Estas tecnologías pueden definirse como herramientas construidas para resolver tareas específicas del mismo modo en que lo haría el razonamiento humano…se les conoce así porque están diseñadas para funciones y tareas concretas que requieren un buen nivel de conocimiento en la materia, tal como se exigiría de un agente humano”[9].

“Su funcionamiento consiste en la estructuración de reglas que le indican a la máquina qué decisión tomar frente a un escenario concreto. Esto hace que los sistemas complejos sean muy precisos, pero que tengan dificultades para resolver problemas imprevistos”[9].

2.2.3 Redes neuronales artificiales

“Las redes neuronales artificiales son un tipo de algoritmo de inteligencia computacional que superan algunas limitaciones de los sistemas expertos (como su incapacidad de resolver casos nuevos o que están fuera de sus reglas de programación). Su funcionamiento se basa —tal como en el cerebro humano— en pequeñas unidades de información que, en colaboración con otras, procesan información”[9].

“Su principal función es permitir que las máquinas extraigan información nueva a partir de aquella con la que se le alimenta, con el fin de obtener un aprendizaje de ella. Por ello, a los procesos que estas tecnologías llevan a cabo comúnmente se les conoce como machine learning, o aprendizaje de máquinas. Esto permite que la máquina optimice progresivamente sus funciones, ofrezca mejores resultados y trabaje de forma eficiente”[9].

“Estos algoritmos son realmente populares en los negocios, ya que posibilitan analizar información de forma automática y obtener algunas tendencias, estadísticas o predicciones sobre un tema específico”[9].

2.2.4 Deep learning

“El deep learning, conocido en español como aprendizaje profundo, es un tipo de algoritmo de redes neuronales artificiales que, como su nombre lo indica, son profundas. Al hablar de aprendizaje profundo nos referimos a algoritmos con un alto nivel de complejidad que permiten llevar a cabo tareas más complicadas y con requisitos computacionales elevados. Estas tecnologías destacan por tener un código complicado y por ser alimentadas con extensas bases de datos. Es por ello que generalmente el deep learning va de la mano con la minería de

datos, una disciplina de la estadística que busca encontrar patrones en ingentes volúmenes de información”[9].

“Estas tecnologías son ideales para llevar a cabo tareas más complicadas, pero sobre todo para realizar actividades que trascienden las capacidades de los agentes humanos (o para las que no se dispone de personal o de tiempo)”[9].

2.2.5 Robótica

”La robótica es una rama computacional independiente de la inteligencia artificial, que se ha nutrido de forma importante mediante estos recursos digitales. Esto ha hecho que una gran cantidad de soluciones de robótica estén impulsadas por redes neuronales artificiales”[9].

“Hoy en día existen robots capaces de caminar, hacer tareas complejas e incluso jugar ajedrez gracias a sistemas de visión computacional, algoritmos de aprendizaje y código que les permite tomar decisiones. La inteligencia artificial está detrás de todo ello”[9].

“En los negocios hay algunos casos de uso de estas tecnologías, especialmente en el rubro de la producción y de la atención al cliente”[9].

2.2.6 Agentes inteligentes

“Como su nombre sugiere, estos sistemas tienen la capacidad de tomar decisiones y actuar con base en ellas mediante razonamientos similares a los de los seres humanos. Esto significa que los agentes inteligentes deben poseer un buen margen de autonomía y libertad para aprender y ejecutar decisiones”[9].

“Actualmente, estas tecnologías están en desarrollo y prometen hacer que la gestión de tareas empresariales se vuelva más sencilla. Sin embargo, hoy en día su uso es reservado, ya que implica consideraciones éticas de importancia”[9].

2.3 Plataforma Allot

Es una tecnología robusta en seguridad que ofrece todo un esquema de protección de base de datos, acceso a la web y tratamiento de datos, está basada en DPI (Deep Packet Inspection) la cual identifica y proporciona inteligencia en la red a través del análisis y protección. El consumo de video y voz genera mayor tráfico, además de presentar descargas directas o ataques DDoS, la herramienta Allot brinda mejoría en la red dotándola de inteligencia [10].

Con relación a lo anterior, esta plataforma es la encargada de bloquear todas las URLs con contenido ilegal que se suban en su base de datos, por medio de comandos específicos y se actualiza cuando Mintic publica nuevos listados con las URLs de dicha índole, para su

respectivo bloqueo. Es conveniente resaltar que solamente el personal especializado en la compañía es el responsable de cargar el listado a la plataforma, por temas de seguridad.
Cabe señalar, que el servicio de protección que bloquea todo el contenido de páginas de pornografía infantil no es altamente efectivo, pues en ocasiones suelen filtrarse varias URLs que por motivos de encriptación en la resolución de la página, se abren. Por esta razón, es conveniente que se lleve a cabo un segundo filtro, soportado mediante un lenguaje de programación capaz de bloquear las páginas que el firewall no logró y en este sentido, abordar códigos de alto nivel como los de Python.

2.4 Python

"Es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de *software*, la ciencia de datos y el *machine learning* (ML). Además de su eficiencia, tiene un entorno amigable y se puede ejecutar en muchas plataformas diferentes. Este lenguaje es usado para aplicaciones como Instagram, Netflix, Spotify entre otros, soporta programación orientada a objetos, programación imperativa y de menor medida y también es dinámico y multiplataforma"[11].
Su versatilidad permite acceder a una gran variedad de archivos y herramientas para distintas tareas, incluyendo el bloqueo de URLs de contenido de pornografía infantil. Las bibliotecas principales para trabajar en redes son: urllib, requests, o socket, las cuales permiten realizar peticiones HTTP, fundamentales para acceder a los sitios web que se desean bloquear.
Su mantenimiento es fácil ya que su sintaxis es clara y concisa, además, existen herramientas como pylint que ayudan a detectar errores y mejorar la calidad del código. También cuenta con una gran comunidad de desarrolladores enfocados en crear nuevas bibliotecas y herramientas para el lenguaje. Su código está disponible para todo el público y puede ser utilizado bajo la licencia conocida como Python Software Foundation License [12] lo que significa que siempre habrá solución a problemas relacionados con la ineficacia en el bloqueo de ciertas URLs.

2.5 Lenguaje Python y su potencial en el desarrollo de software IA.

"El presente estudio, se centra en profundizar en las características técnicas y el potencial del lenguaje Python, en relación con la Inteligencia Artificial. En este caso, se subraya su importancia en el involucramiento directo con diferentes tipos de tecnologías como la robótica, el Internet de las cosas y la IA"[8].
Por lo anterior, es necesario entender que la Inteligencia Artificial (IA) en combinación de algoritmos planteados para el funcionamiento de máquinas, presentan capacidades similares a

las del ser humano, pues se resalta que: "*Normalmente, un sistema de IA es capaz de analizar datos en grandes cantidades (big data), identificar patrones y tendencias y, por lo tanto, formular predicciones de forma automática*"[13], por otro lado, El portal SAS INSIGHTS destaca que de esta forma es posible que "*las máquinas aprendan de la experiencia, se ajusten a nuevas aportaciones y realicen tareas como seres humanos*"[14].

En síntesis, Python es un lenguaje apropiado en el desarrollo de IA, no solamente por su simplicidad, sino por la capacidad de manejar otras tareas como procesamiento de datos, aprendizaje automático, entre otras, lo que indica que es una herramienta esencial para quienes pretendan trabajar en el campo de la inteligencia artificial.

2.6 Visual Studio

"Es un entorno de desarrollo integrado (IDE) creado por Microsoft, es una herramienta completa la cual ofrece edición, depuración y compilación de código para poder después publicar la aplicación, en su actualización en 2022 ha logrado por primera vez un entorno de 64 bits nativo, lo cual permite mayor rendimiento y fiabilidad para soluciones a gran escala. Mayor rendimiento en el desarrollo con un IntelliCode mejorado, que ha incorporado Inteligencia Artificial. Si antes, según se escribía el código, aparecían todas las posibilidades por orden alfabético, ahora aparecen auténticas sugerencias de cómo seguir escribiéndolo, y no solo del elemento en el que se está, sino para toda la línea de código, lo que es especialmente útil para quienes están aprendiendo un lenguaje. Las sugerencias están basadas en modelos de aprendizaje automático, (Machine learning), que se ejecutan localmente"[15].

"Sus funciones principales es la codificación con confianza, lo cual brinda asistencia en el proceso de codificación sin importar el lenguaje, desde C#/VB y C++ a JavaScript y Python. Visual ayuda a mantener el contexto del Código con el cual está estructurado, con funcionalidades como Peek to Definition e Ir a (memoria), lo cual permite filtrar fácilmente y buscar todo tipo de elemento. El entorno resolutivo es más manejable con los iconos de "bombilla" lo cual ayuda a relacionar y corregir problemas de codificación habituales. Visual Studio refactoriza a medida que el código va tomando mayor envergadura, lo cual lo vuelve más sencillo la reestructuración"[15].

Para el despliegue del código realizado en Python a la aplicación web que se va a implementar en el proyecto propuesto, Visual Studio puede integrarse con el framework web Django para la

creación de sitios web y aplicaciones de manera rápida segura y eficiente el cual se describe detalladamente a continuación.

2.7 Django

Según [16] “Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que el usuario puede concentrarse en escribir su aplicación sin necesidad de reinventar la estructuración del programa. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago”.

“Django puede ser (y ha sido) usado para construir casi cualquier tipo de sitio web, desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc)”[16].

“Django es seguro, ayuda a los desarrolladores evitar varios errores comunes de seguridad al proveer un framework que ha sido diseñado para "hacer lo correcto" para proteger el sitio web automáticamente. Por ejemplo, Django, proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando así errores comunes como colocar informaciones de sesión en cookies donde es vulnerable (en lugar de eso las cookies solo contienen una clave y los datos se almacenan en la base de datos) o se almacenan directamente las contraseñas en un hash de contraseñas”[16].

“Django es escalable, usa un componente basado en la arquitectura "shared-nothing" (cada parte de la arquitectura es independiente de las otras, y por lo tanto puede ser reemplazado o cambiado si es necesario). Teniendo en cuenta una clara separación entre las diferentes partes significa que puede escalar para aumentar el tráfico al agregar hardware en cualquier nivel: servidores de cache, servidores de bases de datos o servidores de aplicación. Algunos de los sitios más concurridos han escalado a Django para satisfacer sus demandas, por ejemplo, Instagram y Disqus”[16].

Para entender el código de Django,[16] “en un sitio web tradicional basado en datos, una aplicación web espera peticiones HTTP del explorador web (o de otro cliente). Cuando se

recibe una petición la aplicación elabora lo que se necesita basándose en la URL y posiblemente en la información incluida en los datos POST o GET. Dependiendo de qué se necesita quizás pueda entonces leer o escribir información desde una base de datos o realizar otras tareas requeridas para satisfacer la petición. La aplicación devolverá a continuación una respuesta al explorador web, con frecuencia creando dinámicamente una página HTML para que el explorador la presente insertando los datos recuperados en marcadores de posición dentro de una plantilla HTML".

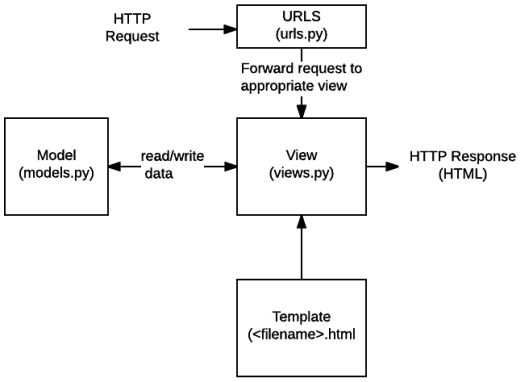


Figura 1. Diagrama Código Django. Fuente[16]

2.8 Marco regulatorio y ciberseguridad

En los últimos años, el internet ha sido aprovechado para fines delictivos, gracias a la navegación y conexión global que permite la fácil transferencia de archivos ilegales, a tal punto de considerarse como una actividad de difusión, frecuente en la red. En este caso, a las conductas de propagación de la pornografía infantil, también se le asocian otros contenidos como la xenofobia, la incitación al odio, la violencia y el ciberterrorismo, donde sus principales factores denominadores son la difusión, la divulgación, la distribución y la exhibición de dicho contenido.

Lo anterior indica que aquellas conductas, claramente son ilícitas, pues es irrelevante el consentimiento del titular o responsable del mismo, al igual que la magnitud de la gravedad de la distribución de contenido pornográfico entre menores[17]

En cuanto al campo legal, se puede resaltar que el riesgo a la impunidad no representa un problema para los ciberdelitos en la difusión de contenido en lo que respecta en la ley nacional, esto hace que el medio de comunicación por el cual se distribuye el contenido sea su fortaleza a tal punto de convertirse en un delito universalizado como resultado de su multilocalización. De este modo, se subraya entonces que la Internet se constituye en un entorno de información

y comunicación masiva y universal en la que no existen fronteras espaciales que cuenta con un espacio virtual transnacional[17]

Por otro lado, es preciso señalar que a nivel geográfico no es posible que ocurra una traslación del lugar de comisión del delito, por consiguiente, se ejecuta en múltiples lugares, de ahí su internacionalización o universalización[18] Así mismo, esta es una característica relevante de cómo actúa la criminalidad en el ciberespacio, de aquí parte la tentativa de los estados en convertirse en guardianes del ciberespacio defendiendo la necesidad de establecer un derecho penal de alcance internacional para los ciberdelitos como los de contenido explicito[19].

2.9 Justicia digital en Colombia

Desde los años 90, son varias las iniciativas de la incorporación de las TIC en la administración de justicia. En ese sentido, se entiende que la justicia digital hace parte del gobierno electrónico, entendido como el uso de las TIC para el desarrollo de una administración pública eficiente, en la prestación de servicios e información a los ciudadanos y empresas[20].

Del mismo modo, la justicia digital constituye un sector de la sociedad de la información. Una definición amplia de la justicia digital contempla el uso de las TIC para prevenir el crimen, mejorar la administración de justicia y el sistema legislativo[21]. Además, se establece que la justicia digital busca mejorar el acceso de los ciudadanos a la justicia y a la acción judicial efectiva, que consiste en la solución de controversias o la imposición de sanciones penales…también es conocida como justicia electrónica, e-justicia, justicia *on-line* o ciber justicia[22].

En conclusión, la justicia digital involucra desde los métodos de comunicación como el correo electrónico, las videoconferencias, los sistemas de resolución de conflictos en línea y hasta los sistemas de información para la gestión de los procesos, pasando por las tecnologías para los tribunales (salas de audiencia), los servicios en línea para consulta de los ciudadanos, entre otros[21][23].

2.10 Ciberseguridad y el marco de ciberseguridad NIST

La Unión Internacional de Telecomunicaciones (UIT) la define como el conjunto de herramientas, políticas, conceptos de seguridad, salvaguardas de seguridad, directrices, métodos de gestión de riesgos, acciones, formación, prácticas idóneas, seguros y tecnologías que pueden utilizarse para proteger los activos de la organización y los usuarios en el ciber

entorno. Las propiedades de seguridad incluyen uno o más de las siguientes: disponibilidad, integridad (que puede incluir autenticidad y el no repudio) y confidencialidad[24].

Conforme a la norma ISO/IEC 27032:2012 se señala que la ciberseguridad se trata de la preservación de la confidencialidad, integridad y disponibilidad de la información en el ciberespacio, definiéndolo, a su vez, como el entorno complejo resultante de la interacción de personas, *software* y servicios en internet, a través de dispositivos tecnológicos y redes conectadas a él, que no existen en ninguna forma física[25].

El Gobierno colombiano, en documento del Consejo Nacional de Política Económica y Social CONPES- 3854 de 2016 y en documento CONPES 3995 de 2020, define la ciberseguridad como la capacidad del Estado para minimizar el nivel de riesgo al que están expuestos sus ciudadanos, ante amenazas o incidentes de naturaleza cibernética, buscando la disponibilidad, integridad, autenticación, confidencialidad y no repudio de las interacciones digitales[26][27].

2.11 Aspecto legal para la detección de contenido de abuso sexual infantil en línea

Actualmente, las tecnologías para la identificación de CSMA[28] son más efectivas si tienen estrecha relación con el tema legal, pues dicho enlace, permite tomar medidas contra los perpetradores que difunden este contenido. En este aspecto, es necesaria una mejor legislación a nivel global para poder combatir eficazmente contra el CSMA, además de examinar leyes nacionales para identificar las brechas que se necesiten mejorar y proteger a las víctimas.[29]

Según el reporte anual presentado por la (ICMEC)[2] donde se evalúa un total de 196 países, se establecen que estos pueden cumplir con los siguientes criterios:

- Existencia de legislaciones nacionales respecto al CSMA[28].
- Definición de CSAM[28].
- Penalización de delitos CSAM por medio de la tecnología.
- Posesión con conocimiento independientemente de la intención de distribuir.
- Responsabilidad de proveedores de servicios de Internet (ISP).

Se subraya además que[29] “En 2018, se informó que 118 de 196 países tenían legislación suficiente que cumpla al menos cuatro de los cinco criterios. De estos, 21 países pudieron

cumplir los cinco criterios. Sin embargo, 16 países de 196 no tienen ninguna legislación que específicamente aborda el CSAM [28]. Los 62 países restantes carecen de una cobertura completa de criterios y cumplir con cualquier cosa, de uno a tres puntos señalados. Aunque hay margen de mejora, la evolución del alcance de la legislación a lo largo de los años muestra progreso. En 2006, sólo 27 de 184 países tenían suficiente legislación, lo que representa aproximadamente un aumento del 45% en comparación con los resultados de 2018".

2.12 Aplicaciones para la detección de CSAM

Las aplicaciones para la detección de Material de Abuso Sexual Infantil (CSAM)[28] son esenciales para combatir la difusión y distribución de contenido ilegal en línea. Estas herramientas tienen la principal función de utilizar tecnologías como el análisis de imágenes, videos y la IA para identificar y reportar contenidos de esta modalidad. Entre las principales aplicaciones se pueden resaltar las siguientes:

Image Hash Database: Utilizados para la verificación de datos, cifrado de contraseñas y otros datos sensibles. El hash de imágenes es la tecnología primaria para detectar CSAM[30] y puede usarse para identificar actividades desconocidas, en este método cada imagen previamente identificada como cada CSAM se combina con un valor hash único, el cual se deriva a través de un algoritmo matemático, convirtiéndose en datos más cortos, de longitud fija y en formato de 24 códigos hexadecimal[31].

El método para detectar este material ilícito es a través de funciones criptográficas, las más conocidas son MD5 (Algoritmo 5 de resumen de mensajes) y SHA (Algoritmo Hash seguro), donde MD% produce un valor hash de 128 bits el SHA produce un valor hash de 160 bits[27]. Este método ha mostrado ciertas limitaciones. En primer lugar la BD solo cubre las imágenes de CSA, sin embargo, otros formatos como videos deben integrarse en la búsqueda de CSAM, otra desventaja es la eficacia de la BD para encontrar CSAM que no ha sido identificado en el pasado y el contenido reciente no es detectable en la BD de hash imágenes[29].

Web-Crawler: Conocido como buscador de bots, está diseñado para navegar automáticamente sobre la web y su objetivo es coleccionar todos los datos de cada página que pueda contener material sensible, existen varios tipos de web-crawlers los cuales tienen el mismo patrón de trabajo que se encargan de rastrear todos los datos a una base de datos[27]. Esta aplicación es capaz de identificar el contenido CSAM al proporcionar características específicas de los sitios

web que contienen CSAM, para ello es fundamental comprender las diferencias entre los sitios que contengan CSAM y los que no[32].

Su mecanismo de trabajo es identificar diferentes tipos de palabras, los cuales se dividen en diferentes categorías. La primera categoría, incluye palabras que son usadas entre los traficantes de CSAM, cada sitio que contengan estos términos será tratado con mayor atención ya que es más probable que contenga CSAM. La segunda categoría, incluye términos que son usados para buscar CSAM, pero también se encontraría el contexto que no esté relacionado a CSAM, esto ayudaría a proporcionar el contexto adecuado del contenido. La última categoría identifica las palabras clave las cuales los usuarios las utilizan, pero no pueden estar relacionadas estrictamente con CSAM[27].

Detección Basada en Nombre de Archivo y Metadata: Propone un sistema capaz de detectar CSAM en redes P2P[33] y engloba un aprendizaje automático que sirve como procesamiento de lenguaje que reconoce consultas de personas que intentan acceder a contenido CSAM, además, clasifica entre el contenido de pornografía legal con el contenido CSAM comparando información con los archivos CSAM conocidos y archivos no conocidos. Su proceso es extraer características de la metadata del archivo segmentado y clasificarlo a través de una Machine Learning[30]&[34].5

Detección Visual: Según[34] Tiene la similar distinción de contenido CSAM como el contenido que no es ilegal, al igual que el anterior método, una vez se extrae el contenido ilegal, debe ser examinado detalladamente y ser clasificado de manera visual; este sistema es más robusto que los anteriores métodos ya que su principal desafío es la detección y la clasificación de CSAM de forma visual. Su proceso de aplicación es detectar CSAM a través de un conjunto actualizado de rasgos faciales y un filtro de tono de piel, el cual, posteriormente se encarga de clasificar si el tipo de contenido CSAM es de bajo o alto nivel, un claro ejemplo de ello es la identificación de niños y el estado que muestra la imagen con una tasa de precisión del 74,19%.

En lo que respecta al territorio colombiano, se cuenta con una línea de denuncia online para la lucha contra la pornografía infantil creado por La Fundación INHOPE y Foro Generaciones Interactivas, donde se puede reportar abusos sexuales contra menores.

Una “hotline” es un canal de denuncia anónimo que se ofrece a los usuarios de Internet, para que puedan informar sobre las páginas que generen y alberguen contenidos potencialmente

ilegales en la Red. Cuenta con un fuerte apoyo nacional y cumple con todas las normas de funcionamiento, protocolos y mejores prácticas exigidas por el INHOPE[28].

3 PROPUESTA DEL NUEVO ALGORITMO O MODELO DE APRENDIZAJE AUTOMÁTICO QUE IDENTIFICARÁ EL CONTENIDO DE PORNOGRAFIA INFANTIL EN LINEA

El nuevo algoritmo o modelo de aprendizaje automático propuesto, tiene la función principal de garantizar la legalidad y la ética como aspectos esenciales presentes en la estructura de esta investigación. Los mecanismos de machine learning serán los aliados para favorecer el bloqueo de URLs que contengan material de pornografía infantil, donde su principal preocupación será la de solventar falencias, mediante objetivos complementarios como la protección de privacidad en línea, el uso de las tecnologías de la información y la integridad de los datos para promover el respeto de los derechos humanos. En este sentido, es pertinente establecer aspectos importantes que permitirán reforzar esta propuesta, mediante un marco legal, un contexto normativo y un elemento técnico, necesarios para el desarrollo de este capítulo, tal como se muestra a continuación.

3.1 Contexto legal

En base a la constitución política de la república de Colombia de 1991 se manejan artículos relevantes que protegen los derechos fundamentales de los niños y su integridad, entre ellos están:

Artículo 44: “Son derechos fundamentales de los niños: la vida, la integridad física, la salud y la seguridad social, la alimentación equilibrada, su nombre y nacionalidad, tener una familia y no ser separados de ella, el cuidado y amor, la educación y la cultura, la recreación y la libre expresión de su opinión. Serán protegidos contra toda forma de abandono, violencia física o moral, secuestro, venta, abuso sexual, explotación laboral o económica y trabajos riesgosos. Gozarán también de los demás derechos consagrados en la Constitución, en las leyes y en los tratados internacionales ratificados por Colombia”[35].

Artículo 13: “…El Estado protegerá especialmente a aquellas personas que, por su condición económica, física o mental, se encuentren en circunstancia de debilidad manifiesta y sancionará los abusos o maltratos que contra ellas se cometan”[35].

Ley 1273 de 2009 la cual se modifica en el código penal y se crea un nuevo bien jurídico tutelado, denominado “De la protección de la información y de los datos” la cual preserva integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones[36].

Ley 1336 de 2009 Articulo 24 (modificada de la ley 599 de 2000 la cual, en el artículo 218). “Demanda que todo acto de pornografía con personas menores de 18 años estará sujeto a penalidades, además se aplicará una condena de 10 a 20 años y multa de 150 a 1.500 salarios mínimos legales mensuales vigentes”[37].

Ley 679 de 2001 Articulo 8 y el decreto 1524 de 2002 Articulo 5 Denominado Deberes: “Denuncia ante las autoridades competentes cualquier acto criminal contra menores de edad que tengan conocimiento, incluso de la difusión de material pornográfico asociado a menores. Además de combatir con todos los medios técnicos a su alcance la difusión de material pornográfico con menores de edad”[38]

3.2 Comercio ilegal y electrónico

Esta modalidad de pornografía infantil es un delito grave que implica la producción, distribución, posesión o intercambio de material que representa a menores de edad involucrados en actividades sexuales y de contenido de índole explícitos. Este tipo de acciones ilegales se llevan a cabo en internet y en otras plataformas digitales, que permiten distinguir los siguientes aspectos:

- Producción y Distribución, donde perpetradores crean contenido pornográfico infantil, alientan la explotación sexual de menores y luego distribuyen e intercambian archivos por medio de la red.
- Dark Web y Redes Encubiertas que son sitios encriptados, en los cuales se encuentra la mayoría del comercio ilegal de pornografía infantil y difícilmente permiten la identificación y la persecución de los delincuentes.
- Criptomonedas que son monedas digitales usadas por los ciberdelincuentes para sus actividades financieras, las cuales proporcionan cierto nivel de anonimato y dificultan el rastreo de las transacciones.

- Foros y Comunidades Encubiertas los cuales son grupos de interés donde los perpetradores comparten este contenido, intercambian consejos de cómo evitar la detección y colaboran en la obtención de más material.

Ética de la investigación

La lucha contra el comercio ilegal de pornografía infantil implica la cooperación de agencias dedicadas a la aplicación de leyes a nivel nacional e internacional, junto con legislaciones específicas que persiguen y penalizan este tipo de actividad. Una de las herramientas eficaces para ayudar con la detección y bloqueo de este acto ilegal, son los algoritmos de reconocimiento de imágenes y el software de análisis de contenido para detectar y bloquear la distribución de contenido sexual infantil.

Por otro lado, los gobiernos, las organizaciones internacionales, las empresas de tecnología y la sociedad trabajan en conjunto para combatir este tipo de comercio ilegal, donde se han tomado medidas para identificar a los perpetradores, rescatar las víctimas y prevenir la propagación de material ilícito.

Lo anteriormente señalado, es un trabajo preventivo que tiene la finalidad de concientizar a todos los usuarios de la red, frente a delitos y actos delincuenciales relacionados con el tráfico de armas, la extorción y el contenido pornográfico donde sus principales víctimas son los niños. Sensibilizar sobre el consumo este tipo de contenido tiene como objetivo fundamental el de mitigar actos ilegales a nivel nacional, a través de herramientas tecnológicas que permitan el bloqueo de URLs con información de esta índole.

3.3 Implementación técnica

El sistema de bloqueo de URLs de Pornografía Infantil se aplicará según el marco regulatorio colombiano, al listado de URLs suministrado por MINTIC, las cuales se filtrarán de acuerdo con el sistema de seguridad de la entidad prestadora de servicio de internet (ISP). En este caso, el uso de herramientas de Machine Learning y la revisión sistemática de literatura correspondiente al tema, permitirán identificar las técnicas de ciberseguridad, utilizadas para controlar el movimiento de pornografía infantil a nivel nacional.

Respecto al bloqueo de contenido de pornografía infantil, se desarrollará un sistema de filtrado en el que se usará la programación de Python, la cual es la herramienta adecuada para realizar el código, permite mitigar posibles sesgos y es un software libre que se adapta al requerimiento del aprendizaje automático propuesto.

En cuanto a la elaboración global del modelo de aprendizaje, es pertinente destacar que su desarrollo final, conlleva un proceso complejo y delicado, pues requiere de la participación de

expertos en el tema y las consideraciones éticas y legales correspondientes para la detección y el bloqueo de contenido de pornografía infantil. Por lo tanto, se presentará un apartado con la propuesta del sistema de filtrado de contenido de pornografía infantil que haya pasado por el primer filtro de Allot.

3.4 Diseño del sistema

La finalidad de este diseño es bloquear todas las direcciones de contenido pornográfico que involucren menores de edad. El filtrado de las URLs se realizará por medio del informe que genera MINTIC en su plataforma, el cual adjunta un listado con todas las URLs que se deben bloquear para los ISP, en todo el país.

Una vez identificado el estado actual de las técnicas de ciberseguridad y los procesos de codificación, como primera etapa de la metodología, se procederá a llevar a cabo el control del bloqueo de pornografía infantil en Colombia, y en relación con las etapas dos y tres correspondientes a la construcción del algoritmo y a la implementación del software, respectivamente, se desarrollarán en seis pasos, de la siguiente manera:

3.5 Levantamiento de requerimientos

En esta fase es conveniente resaltar que las partes interesadas, denominadas como Stakeholders, son todos los usuarios de la web, y son a quienes se deben proteger de información con contenido ilegal. Por otro lado, defender la identidad de las víctimas es mucho más importante, pues la intención es filtrar todas las URLs posibles que provee Mintic para el respectivo bloqueo de sitios que promueven contenido de pornografía infantil.

La idea anterior, no solamente contribuirá a disminuir el consumo de estas páginas en el territorio colombiano, sino que además es una propuesta que puede servir de modelo para grupos de autoridades legales y demás usuarios, interesados en contrarrestar acciones ilegales en línea. Cabe aclarar que es sumamente necesario, comprender a profundidad la problemática expuesta, porque los niños son vulnerados y sometidos a actos inmorales por toda la red, lo cual es un indicador alarmante que exige mejorar el filtrado de URLs por medio de un lenguaje de programación que sea libre y escalable, como lo es Python.

Respecto al levantamiento de requerimientos, estos se deliberan en unas técnicas en donde se consideran los siguientes aspectos:

- **Requerimientos Funcionales:** El software propuesto debe tener la capacidad de analizar las URLs, la cual se va a ir actualizando por medio del listado que provee Mintic para el respectivo filtrado. Este debe clasificar páginas que sean seguras o peligrosas y según el análisis, debe ser apto para bloquear el acceso al contenido ilegal.
- **Requerimientos No Funcionales:** Una vez terminado el software propuesto, se espera obtener un rendimiento que se acomode a su funcionalidad. Su sistema debe ser seguro para que el código no sea manipulado por personal extraño y necesariamente, tener la capacidad de actualizar y clasificar la base de datos de las URLs que provee Mintic.
- **Restricciones y Limitaciones:** Se realizarán anotaciones de cualquier restricción o limitación que pueda afectar el desarrollo y/o la implementación del sistema.
- **Seguridad y Privacidad:** Toda información que contenga datos como imágenes y URLs, serán protegidas y solo la persona encargada de la implementación, podrá tener acceso para bloquear contenido de pornografía infantil.
- **Escenarios de Uso y Casos de Uso:** son contextos que describen la manera en que el usuario puede interactuar con el sistema de filtrado propuesto a través del software, para ello, se establece el siguiente ejemplo. “El usuario enciende su dispositivo y abre el navegador web. El usuario intenta acceder a una página web ingresando la URL en su navegador. Antes de cargar la página, el software verifica la URL utilizando el modelo de clasificación de contenido. El sistema utiliza el modelo de clasificación ya entrenado para determinar si la URL tiene contenido de pornografía infantil. Si la URL se clasifica como peligrosa, el software bloquea el acceso a la página y muestra el banner de bloqueo al usuario.”

Precondiciones: El software está instalado y en ejecución, el modelo de clasificación esta actualizado.

Postcondiciones: La página se cargará normalmente siempre y cuando no contenga información relacionada con pornografía infantil, en caso de bloqueo, el sistema arrojará un mensaje informativo al usuario.

- Métodos de Evaluación: Se realizará revisión periódica y se probará el programa en determinadas fechas para garantizar su operabilidad.
- Documentación y mantenimiento**:** Se considera como requisito importante, relacionar toda la documentación del software y su capacidad de mantenimiento a largo plazo.

3.6 Patrones de comportamiento

En machine learning, los patrones de comportamiento se refieren a las tendencias predecibles en los datos, es decir, que, para poder construir modelos predictivos, es importante identificar esquemas repetitivos en los datos y que, en cierto modo, estén relacionados entre sí. Dicho de otra manera, lo que se busca es que la tendencia se identifique mediante algoritmos de aprendizaje automático y le permita realizar tareas específicas, tal como en este caso, donde se hará uso de un modelo de clasificación, el cual se detallará en páginas posteriores.

En cuanto a la identificación del modelo de comportamiento, será necesario recolectar una cantidad considerable de URLs que tengan un cierto patrón en su direccionamiento. En esta situación, se ha evidenciado que la mayoría de URLs son extensas y decodificadas, al punto de que los sistemas de seguridad pasen desapercibidos.

La tarea fundamental, es poder entrenar el sistema automático, por medio de una base de datos que contenga todas las URLs con información sensible, donde solo el personal autorizado pueda realizar el análisis pertinente para poder entrenar el código y posteriormente, proceder a aplicar el filtrado que determina Mintic.

3.7 Desarrollo del programa

Una vez que se haya recopilado los datos sobre las diferentes formas de direccionamiento anteriormente descritas, se procederá a realizar la construcción del modelo de aprendizaje que será manejado con la legalidad y ética del caso. Las técnicas de aprendizaje y las herramientas fundamentales que se abordaran en este proyecto estarán mediadas por machine learning, la cual será la disciplina madre para entrenar el modelo que identificará el porcentaje de pornografía infantil que no puede ser filtrado en su momento.

Con relación a lo anterior, es conveniente resaltar una serie de aspectos fundamentales concernientes al modelo, tal como se especifica a continuación:

- **Características del modelo:** se tendrán en cuenta elementos que ayudarán a identificar el contenido inapropiado para aplicar el filtro a través del modelo de aprendizaje. Entre las particularidades que se tendrán en cuenta están patrones visuales, texto o metadatos. En este caso, se usarán los modelos de direccionamiento que resuelvan dichas páginas a través del listado de URLs.
- **Validación del Modelo**: Se pretende usar un conjunto de datos equilibrados para realizar una validación exhaustiva y así asegurar que el modelo no tenga sesgos.
- **Integración del Sistema:** Se iniciará el proceso de codificación del algoritmo de machine learning el cual se realizará en Python, que a su vez estará apoyado de

técnicas de clasificación, del mismo modo, se pretende entrenar el modelo para mostrar una predicción de las URLs, las cuales contengan información ilícita y así aplicar el bloqueo.

- **Privacidad y Seguridad:** Se garantizará que la privacidad de los usuarios se mantenga invulnerable, al igual que la posibilidad de anonimizar los datos de entrada y salida, los cuales requieren de la implementación de medidas de seguridad para proteger la identidad y confidencialidad del modelo y los datos.
- **Legalidad y Ética:** Se asegurará cumplir con todas las leyes y regulaciones nacionales relacionadas con la privacidad y la seguridad de datos, además de considerar la posibilidad de proporcionar información clara a los usuarios, sobre cómo se utilizará la información y cómo funcionará el sistema.

3.8 Pruebas y evaluación

En esta investigación, es fundamental asegurar la eficacia, la integridad y el funcionamiento del modelo de aprendizaje por medio del entrenamiento continuo. Para este caso, la etapa de pruebas se llevará a cabo una vez se haya finalizado la elaboración del código y para ello, se tendrán en cuenta los siguientes pasos:

- **Diseño de Casos de Prueba:** Se cubrirán varios escenarios relacionados con diferentes casos donde el usuario intente acceder al contenido ilegal para que la herramienta proceda a filtrar la URL que lo contiene, además, se harán simulacros de ciertas situaciones en las que el software deba bloquear o permitir el acceso.
- **Pruebas Funcionales:** Se realizarán ensayos para asegurar que el modelo cumpla con los requisitos funcionales especificados, se validará la precisión del bloqueo sobre las URLs que se deban cerrar y se garantizará el acceso seguro a sitios web que no tengan información ilícita.
- **Pruebas de Rendimiento:** Se evaluará el rendimiento del sistema con diferentes cargas y garantizará que el bloqueo funcione de manera eficiente sin afectar su utilidad.
- **Pruebas de escalabilidad**: se evaluará la capacidad del sistema para que pueda escalar según los cambios que el código requiera, logre ser cada vez más robusto y cumpla con sus funcionalidades.
- **Evaluación de Falsos Positivos:** Se realizarán pruebas que permitan analizar casos en los que el software bloquee incorrectamente un sitio web legal (Falso Positivo) o permita el acceso a una URLs la cual si deba ser bloqueada (falso negativo).

3.9 Documentación y presentación

Una vez se tenga los resultados de las pruebas, se procederá a organizar y a describir la documentación junto con cualquier novedad o problema resultante de la ejecución del modelo de aprendizaje. Toda información recopilada sobre el tema será de gran importancia para futuras mejoras y actualizaciones de este. Respecto al material literario, estado actual del tema, autores y demás fuentes, estos han sido previamente consultados, estudiados y referenciados de manera correcta a lo largo del proyecto, para reforzar las ideas expuestas y enriquecer la investigación.

3.10 Implementación y mantenimiento

Justo después de probado y desarrollado el modelo de aprendizaje, se procederá a implementarlo como tal, en un entorno de producción para su determinada ejecución. En este caso, es conveniente destacar que un programa de este tipo, conlleva un proceso complejo y delicado que requiere de la participación de expertos y la consideración de aspectos éticos y legales relacionados con la detección y bloqueo de contenido de pornografía infantil, por tal razón, el proceso introductorio que se va a abordar, presentará un sistema para el segundo difusor de contenido de pornografía infantil que haya pasado el primer filtro de Allot, además, como se menciona en el alcance de este proyecto, este proceso se manejará en un entorno controlado(de manera local).

3.11 Arquitectura del sistema

Para entender el procedimiento del software a implementar, se presenta a continuación un diagrama de flujo que explica los pasos, los recursos a utilizar y los procesos que este implica, tal como se puede observar el la Figura

Figura 2. Diagrama de flujo. Modelo de aprendizaje propuesto. Fuente: este trabajo.

3.12 Filtrado de contenido

El algoritmo se llevará a cabo por medio de la subcategoría de ML denominada “Aprendizaje Supervisado” que se define por el uso del conjunto de datos etiquetados para entrenar algoritmos que clasifican o prevén dichos datos con precisión. A medida que se introduce nueva información, el sistema irá obteniendo una mejor adaptabilidad, en este caso, de la base de datos con 48.619 URLs, 23.620 de ellas son de contenido explícito y 24999 son benignas. Cabe mencionar que esta carpeta entrará en el proceso de entrenamiento y permitirá una comparación con las URLs que se vayan ingresando al sistema[39].

Respecto al funcionamiento del aprendizaje supervisado, este se determina mediante el siguiente modelamiento:

Clasificación: Este tipo de aprendizaje asigna con precisión, los datos de prueba a categorías específicas que se reconocen en un conjunto de entidades, las cuales son definidas o etiquetadas a manera de conclusión por dicho modelo. Este proceso es útil para poder identificar imágenes que expongan a niños en eventos pornográficos, pues el algoritmo procesa la información y la compara con las etiquetas estructuradas del sistema.

Es pertinente señalar que “La clasificación en machine Learning intenta predecir con exactitud a qué grupo pertenece cada muestra de datos, como en este caso, que se trata de una base de información que contiene URLs etiquetadas como Benign y Malicious. A patir de esta información, se ajustará el modelo para poder diferenciar los patrones entre las URLs que sean benignas y las que son de contenido explícito, por lo tanto, una vez creado el modelo, cuando una nueva URL llegue, este la identificará y la clasificará según corresponda (Benign o Malicious).

Respecto a lo anterior, es preciso señalar que los algoritmos de clasificación trabajan con datos etiquetados que, en muchos casos, son categóricos. Los métodos de clasificación no predicen un valor continuo (como hacen los algoritmos de regresión), si no que predicen un grupo (clase) de un conjunto existente”[40].

Cabe mencionar que el proceso de almacenamiento de datos que se va a utilizar podría cambiar dependiendo de cómo se adapte al sistema, inicialmente todas las URLs se manejarían en una hoja de cálculo de Excel para que se pueda sincronizar con el algoritmo. Por otro lado, también se podría alojar la información en un entorno web, tal como lo hace Mintic, para así poder presentar a los ISP, las URLs que deben bloquearse.

3.13 Tecnologías utilizadas

En el modelo de aprendizaje automático, las tecnologías juegan un papel fundamental, pues son herramientas efectivas para hacer proyecciones precisas y estratégicas. En este caso, se pondrán en consideración 3 métodos: K Vecinos más cercanos (K Nearest Neighbours), Random Forest y el modelamiento de Redes Neuronales Profundas, de los que se podrá identificar, cuál es más factible en relación con el comportamiento y la eficacia en respuesta a los requerimientos necesarios. En este contexto, se ejemplifican de la siguiente manera:

- **K Vecinos mas cercanos (K Nearest Neighbours):**

“Este algoritmo es un método de clasificación supervisada que compara un nuevo dato con los ya existentes para encontrar los “K” (puntos de datos más cercanos) que se relacionan con la métrica de distancia Euclidiana. La clase del nuevo dato se asigna según la mayoría de las

clases presentes en los “K” vecinos más cercanos. Este proceso en vez de devolver un valor numérico, devuelve una etiqueta que clasifica la muestra. La etiqueta se puede obtener contando cada formula del conjunto de vecinos y seleccionando la ganadora. Por otro lado, también se puede realizar una agregación ponderada de etiquetas, según la distancia normalizada de cada vecino”[40].

“A pesar de su simplicidad, los vecinos más cercanos han tenido éxito en una gran cantidad de problemas de clasificación y regresión, como en áreas de salud, análisis de imágenes satelitales, finanzas, ciencias políticas, reconocimiento de video y más. Al ser un método no paramétrico, a menudo tiene éxito en situaciones de tipificación donde el límite de decisiones es muy irregular. Además, KNN tiene la característica de ser un algoritmo de aprendizaje no paramétrico y perezoso”[41].

Es preciso resaltar que el término “No paramétrico”, se refiere a que no hay suposiciones para la distribución de datos subyacentes, por lo que es muy útil en la práctica donde la mayoría de los conjuntos de datos del mundo real, no siguen supuestos teóricos matemáticos. En cuanto a la expresión “perezoso”, quiere decir que no necesita datos de entrenamiento para la generación del modelo. Toda la información de entrenamiento será utilizada en la fase de prueba. Esto hace que el periodo de entrenamiento sea más rápido y la prueba como tal, sea más lenta y costosa.[41].

Del mismo modo, “La clasificación básica de vecinos cercanos utiliza ponderaciones uniformes, es decir, que el valor asignado a un punto de consultas, se calcula a partir de una mayoría simple de votos de los vecinos más cercanos. Cabe señalar que en algunas circunstancias, es mejor ponderar a los vecinos, de modo que los más cercanos contribuyan al ajuste. Esto se puede lograr a través de la palabra **weights**”[41], de la siguiente manera:

- **Weights** =”Uniform”.Asigna pesos uniformes a cada vecino.
- **Weights =** “Distance”. Asigna pesos proporcionales al inverso de la distancia desde el punto de consulta. Alternativamente, se puede suministrar una función de la distancia definida por el usuario para calcular los pesos.

“A modo de ejemplo, se tiene una muestra clasificada en dos clases (1 y 2), tal como Figura se muestra en la Figura 3, para poder tomar el valor de K=3”[41].

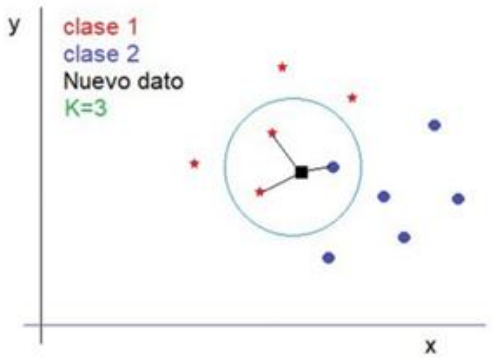


Figura 3. Aplicación KNN. Fuente[41]

En el anterior ejemplo se tiene un nuevo punto a clasificar, por tal motivo se realiza los siguientes pasos:

- Se calcula la distancia a los datos.
- Se encuentran los vecinos más cercanos.
- Se clasifica según categoría de la mayoría.

Por otro lado, “En la Figura 4, vemos que, de la muestra a clasificar, en primer lugar, se calcula la distancia de este elemento al más cercano. En este caso, sus vecinos más cercanos son 3 de la clase círculo y 2 de la clase triangulo. Por lo tanto, el nuevo elemento se clasificaría como de la clase circulo, ya que es la clase mayoritaria”[41].

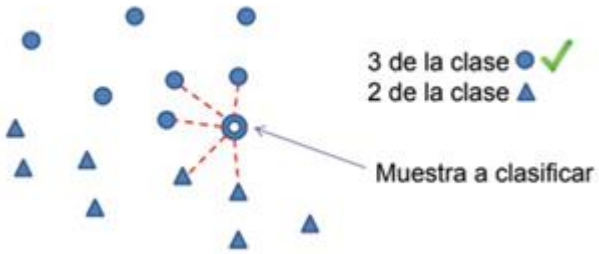


Figura 4. Aplicación KNN. Fuente[41]

No obstante, “A la hora de clasificar un nuevo elemento, se busca en el conjunto de datos disponible los K Puntos más cercanos y se asigna su categoría”[41], tal como se puede observar en Figura 5.

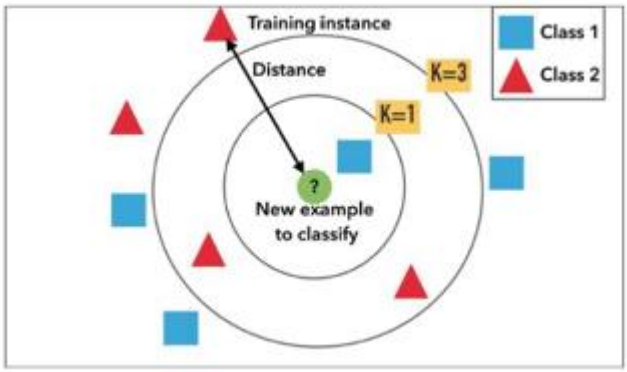


Figura 5. Aplicación KNN. Fuente[41]

Para clasificar URLs Benign o Malicious con KNN, se necesitarán las siguientes librerías en Python:

- **Scikit-learn:** Librería usada para machine Learning, sirve para implementar KNN y otras utilidades de preprocesamiento.
- **KNeighborsClassifier:** "Es un clasificador que implementa el voto de K-Vecinos más cercanos de cada punto de consulta, donde K es un valor entero especificado por el usuario."[41]
- **RadiusNeighborsClassifier:** "Implementa el aprendizaje basado en el número de vecinos dentro de un radio fijo r de cada punto de entrenamiento, donde r es un valor de punto flotante especificado por el usuario."[41]
- **Sklearn.metrics:** "Es un módulo que implementa varias funciones de perdida, puntuación y utilidad para medir el rendimiento de la clasificación, lo cual puede requerir de estimaciones de probabilidad de clase positiva, valores de confianza o decisiones binarias."[41]
- **Sklearn.model_selection.train_test_split:** "Separa arreglos o matrices en subconjuntos aleatorios de entrenamiento y prueba, es una fase importante ya que se necesita evaluar el modelo con datos que no ha visto en el entrenamiento, por lo cual se debe usar un conjunto de pruebas separado."[41]

  Para entender el flujo del algoritmo, inicialmente se preprocesa los datos, en este paso se debe convertir las URLs en características numéricas, después se procede a dividir el conjunto de datos en entrenamiento y en prueba para aplicar el clasificador KNN. Posteriormente devolverá una etiqueta que clasificará la muestra y por último permitirá evaluar el rendimiento del modelo en cuanto a precisión, matriz de confusión y accuracy.

- **Árboles de Decisión (Random Forest):** “Es un tipo de aprendizaje supervisado utilizado tanto para regresión como para clasificación, el cual se basa en un conjunto de condiciones que se organizan de manera jerárquica. El objetivo es crear un modelo que prediga el valor de una variable mediante el aprendizaje de reglas simples de decisión, a partir de datos insertados, es decir, que el árbol inicia desde la raíz hasta sus hojas o cajas, para llegar a la decisión final”[41].

  Del mismo modo, “estos modelos consisten en estimar una variable endógena mediante secuencias de decisión en forma de particiones en las variables predictoras. Los árboles de decisión se entrenan desde la exploración de diferentes umbrales de corte dentro del conjunto de variables y además desde la selección de aquellos que mejor discriminen. Tras hacer esto de forma recursiva hasta alcanzar algún criterio de parada, se aplican algoritmos de lo que se llama “post-poda”, para reducir el sobreajuste, o se va repitiendo el mecanismo de creación del árbol ensamblando iterativamente modelos simples mediante técnicas de remuestreo. De este modo quedan conformados modelos complejos y precisos, partiendo de árboles de decisión sencillos”[41].

  De la misma manera, “los modelos de árbol donde el objetivo puede tomar un conjunto discreto de valores, se denominan árboles de clasificación. Los árboles de decisión donde el objetivo puede tomar valores continuos, se llaman árboles de regresión y tienen como funciones: segregar los datos en grupos según valores de variables, de evaluar cada grupo y de repetir el proceso en cada uno de ellos, hasta que el proceso de evaluación determine que no es necesario segregar más grupos. Cuando usamos más de un árbol hablamos de Bosque (Forest)”[41].

  Así mismo, “Random Forest es una combinación de predictores de árbol donde cada árbol depende de los valores de una muestra de vector aleatorio de forma independiente y con la misma distribución que el resto de los árboles del bosque. Random Forest comienza con una técnica de aprendizaje automático estándar llamada “árbol de decisiones”, que corresponde a un aprendizaje. En un árbol de decisión, una entrada se introduce en la parte superior y hacia abajo a medida que atraviesa el árbol, donde los datos se acumulan en conjuntos cada vez más pequeños. En la Figura 6 se puede observar el principal funcionamiento de dicho algoritmo”[41].

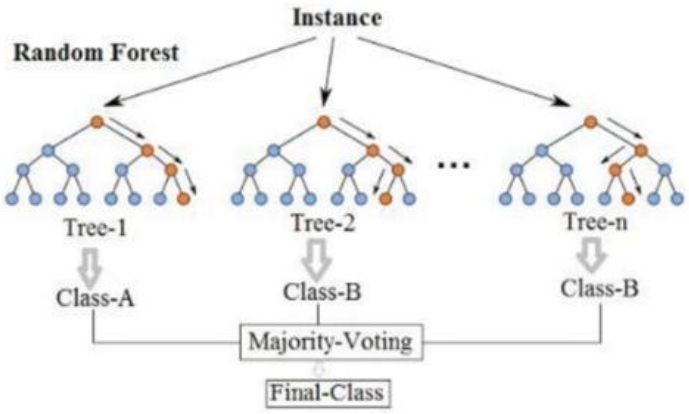


Figura 6. Representación del Algoritmo Random Forest. Fuente [41]

Para clasificar URLs Benign o Malicious con Random Forest, se necesitarán las siguientes librerías en Python:

- **RandomForestClassifier del módulo sklearn.ensemble:**" incluye dos algoritmos de promedio basados en arboles de decisión: el RandomForest y el método Extra-Trees. Ambos algoritmos son técnicas de perturbar-y-combinar específicamente diseñadas para árboles. Esto significa que un diverso conjunto de clasificadores es creado introduciendo aleatoriedad en la construcción de los clasificadores. La predicción del conjunto es dada como la predicción promediada de los clasificadores individuales."[49].
- **StandardScaler del módulo sklearn.preprocessing**:"Modelamiento que estandariza las características eliminando la media y escalando a la varianza unitaria. La puntuación estándar de una muestra X se calcula como:

$$Z = (x - u)/s$$

Donde "u" es la media de las muestras de entrenamiento o cero si with_mean=False, y "s" es la desviación estándar de las muestras de entrenamiento o uno si with_std=False. El centrado y el escalado se realizan de forma independiente en cada característica, calculando los estadísticos pertinentes en las muestras del conjunto de entrenamiento. La media y la desviación estándar se almacenan para ser utilizadas en datos posteriores utilizando transform."[42]

Su flujo inicial es el preprocesamiento de datos, para lo cual se debe convertir las URLs en características numéricas (por ejemplo, a partir de la longitud de la URL, el número de subdominios, caracteres especiales). Después se divide el conjunto de datos de entrenamiento y prueba, una vez se tenga segmentado, se procede a entrenar con Random forest a partir de dichos datos de entrenamiento y por último, se evalúa el rendimiento a través de métricas como la precisión, la matriz de confusión y el informe de clasificación.

- **Redes Neuronales Profundas (Deep NN):**

  La neurona biológica: "Las redes neuronales artificiales tratan de imitar el funcionamiento del cerebro. Su elemento básico son las neuronas, por lo que resulta necesario conocer el funcionamiento de una neurona biológica para entender el modelo simplificado de neurona en el que se basan las redes neuronales artificiales. En la Figura se aprecia que el elemento central de una neurona es su núcleo; el núcleo se encarga de realizar el procesamiento de la información que le llega a la neurona. Esta información entrante proviene de las dendritas; podemos suponer que normalmente la información llega desde otras neuronas o, en algunas ocasiones, desde estímulos exteriores: nervio óptico, auditivo, etc. La información procesada se transmite a lo largo del axón hacia otras neuronas o hacia elementos biológicos del cuerpo, que finalmente provocan las reacciones a los estímulos de entrada: movimientos musculares, mecanismos de defensa, etc. Los terminales del axón se denominan sinapsis; son los encargados de realizar esta transferencia hacia las siguientes neuronas."[40].

  "Un cerebro humano tiene unos 80 mil millones de neuronas y unos 100 millones de millones de conexiones. Cada neurona transmite la señal que procesa a potencialmente cientos de neuronas, que a su vez hacen lo mismo. Desde que se produce un estímulo (visual, auditivo, sensitivo, etc.) hasta que se genera la respuesta (p.e.: movimiento muscular) el procesamiento acumulado por las neuronas involucradas es ingente: capaz de resolver las complejas situaciones de la vida que nuestro desarrollo evolutivo ha podido solventar. La velocidad de procesamiento de las neuronas está entre 1 y 2 milisegundos. En definitiva, el cerebro tiene una enorme capacidad de procesamiento paralelo basada en grano fino: unidades de procesamiento sencillas con un gran número de canales de comunicación entre ellas."[40].

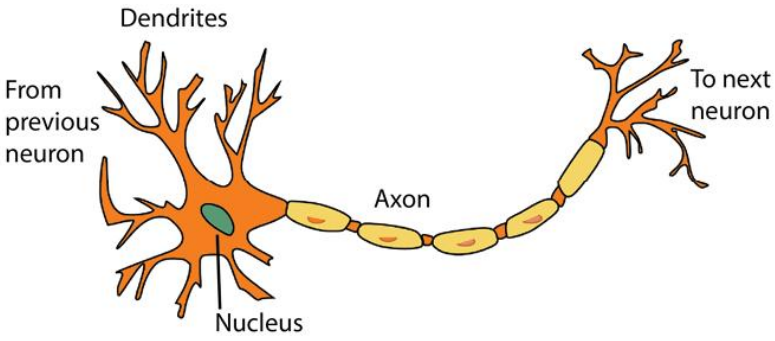


Figura 7. Representación de la Neurona Biológica. Fuente[40]

Según lo citado anteriormente [40], El procesamiento simplificado de una neurona biológica es el que se describe a continuación y se ejemplifica en la Figura :

- Inicialmente, el núcleo de la célula se encuentra en reposo, con una carga potencial de unos -70mV.
- Cuando llegan señales de las sinapsis de otras neuronas, el cuerpo de la célula va perdiendo carga negativa.
- Siguiendo el proceso anterior, el cuerpo de la célula llega a situarse con un potencial de unos +30mV
- Interviene un proceso que descarga esta tensión, devolviendo a la célula a su estado de reposo.

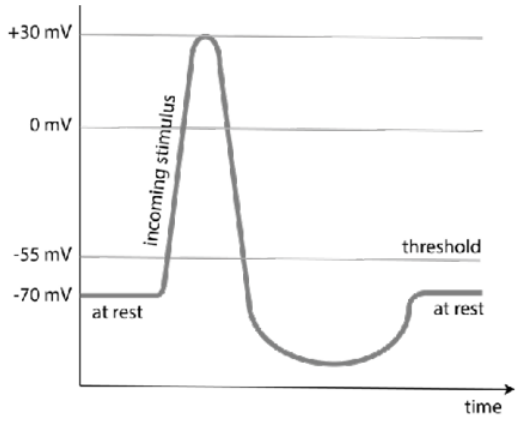


Figura 8. Procesamiento simplificado de una Neurona Biológica. Fuente[40]

La Neurona Artificial: "Las neuronas artificiales presentan una estructura y siguen unas pautas de funcionamiento muy similares a sus correspondientes en el modelo

natural presentado en el apartado anterior. La Figura , muestra el esquema de una neurona artificial. Sus elementos son:

- **Ui:** Representa la neurona i-ésima.
- **Yi:** Representa el resultado que genera la neurona i, equivalente a la salida que se produce en el axón de la neurona biológica.
- **Wji:** Representa el valor de inhibición o excitación entre las neuronas uj y ui. Es el equivalente al efecto de los neurotransmisores sobre la sinapsis que une las neuronas uj y ui. Cuando wji>0 se modela una sinapsis excitadora; cuando wji es menor a 0, se modela una sinapsis inhibidora; cuando wji es cero se modela ausencia de conexión.
- **Neti**: Representa la suma de las señales que le llegan a la neurona ui.

$$Net_i = \sum_j y_j w_{ji}$$

- **f(Neti):** para representar la función de salida o transferencia. Es el equivalente al proceso que realiza el núcleo de la neurona biológica en función de la acumulación de los voltajes que le llegan: Neti

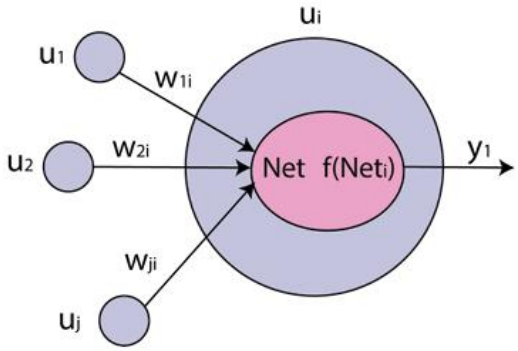


Figura 9. Representación de la Neurona Artificial. Fuente[40]

“La función de transferencia es necesaria para evitar la linealidad del cómputo. Si encadenamos un conjunto de neuronas realizando un proceso cada una, pero sin aplicar funciones de transferencia no lineal, entonces el cómputo, conjunto de esas neuronas podría haber sido realizado por una sola de ellas. La formulación de este concepto en el que se usan tres neuronas es el siguiente”[40].

$$Net_i = \sum_l \left[\sum_k \left(\sum_j y_j w_{ji} \right) w`_{ki} \right] w``_{li}$$

La cual se simplifica de la siguiente manera:

$$Net_i = \sum_l y_l \left(w_{ji} * w`_{ki} * w``_{li} \right) = \sum_l y_l (w_{li})$$

“En definitiva, para dotar de una adecuada capacidad de cómputo a una red neuronal, es necesario establecer una relación no lineal entre la entrada de cada neurona y su salida. Entre las funciones de transferencia f más corrientes que se utilizan en Deep Learning se considera”[40]:

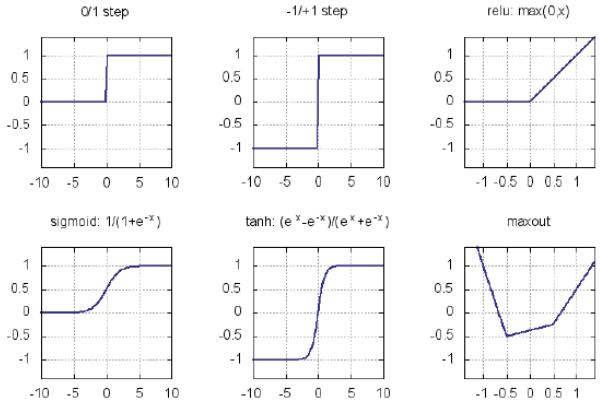


Figura 10. Representación No linealidad. Fuente[40]

El perceptrón: “Es el caso más sencillo de red neuronal. En el perceptrón solo existe una neurona de cómputo, y por tanto solo una salida de resultado. Esta red admite un número no restringido de datos de entrada a la neurona artificial (el equivalente a un número ilimitado de dendritas). En su formulación, al existir una única neurona de cómputo, podemos eliminar el índice i de los pesos w, tal como se observa en Figura 111 que muestra el esquema de un perceptrón con N entradas”[40].

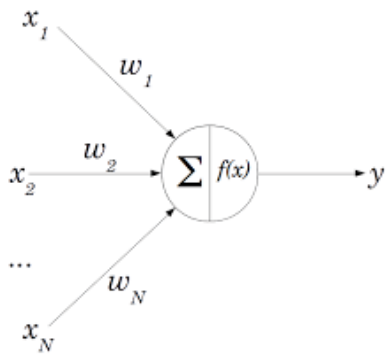


Figura 11. Perceptrón con N entradas. Fuente[40]

Donde:

$$y = f\left[\sum_{i=1}^{N} x_i w_i - \theta\right]$$

En el caso didáctico en que N=2:

$$y = f(x_l w_l + x_2 w_2 - \theta)$$

“En la formulación de las neuronas artificiales añadimos el umbral theta para añadir un grado de libertad en su respuesta. Esta necesidad se entiende mejor si reformulamos la ecuación y elegimos la función escalón, como función de transferencia”[40].

$$f(x)\begin{cases} 0\ if\ 0 > x \\ 1\ if\ x\ \geq 0 \end{cases}$$

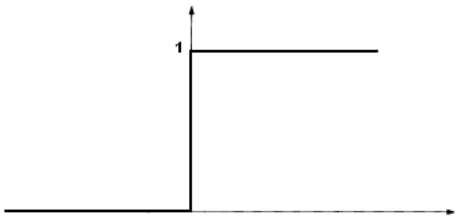


Figura 12. Unit Step (Threshold). Fuente [40]

$$y = 0 \rightarrow x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$

“Como se puede apreciar en la anterior Figura 12 y con la ecuación de una recta en dos dimensiones. Variando los pesos w1 y w2 modificamos la pendiente de la recta. Variando w2 y theta modificamos el punto de corte de la recta en el eje y. De esta manera, un perceptrón es capaz de realizar separaciones lineales en las muestras de entrada”[40]. Como puede observar a continuación según la Figura 13.

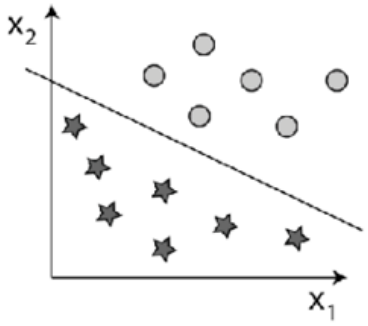


Figura 13. Esquema Separación Lineal. Fuente[40]

Redes Multicapa y Algoritmo Back Propagation: “Puesto que el perceptrón solamente permite realizar separaciones lineales, es preciso diseñar redes neuronales que consigan clasificar muestras que no cumplen la condición de separabilidad lineal. Las funciones AND y OR son separables linealmente, por lo que el perceptrón es capaz de realizar dichas funciones. Podemos crear una red neuronal en la que varios perceptores realicen diferentes separaciones lineales y, posteriormente, un último perceptrón realice el AND o el OR de las separaciones anteriores”[40]. A continuación, se ejemplifica la siguiente imagen:

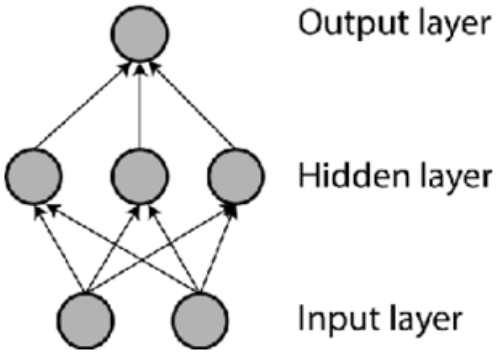


Figura 14. Esquema Red Neuronal. Fuente[40]

“El tipo de separación entre dos clases que podría generar la red neuronal de la Figura 14, se muestra en la siguiente Figura 15. Nótese que cuantas más neuronas contenga la capa oculta más compleja puede ser la frontera de separación entre las clases. En la mayor parte de los casos resulta necesario trabajar con redes de varias capas que presenten un número suficiente de neuronas en cada una de estas capas”[40].

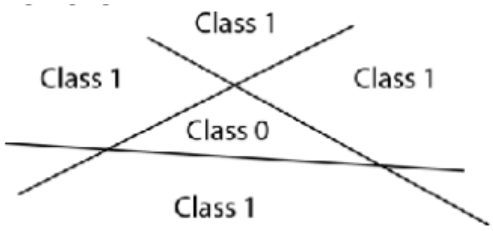


Figura 15. Separación de clases. Fuente[40]

Por consiguiente, las Redes Neuronales Profundas son un enfoque avanzado de aprendizaje supervisado que consiste en múltiples capas de neuronas conectadas entre sí, lo que permite al modelo aprender patrones complejos en los datos. Son particularmente efectivas en la clasificación de datos con características complejas y no lineales, lo cual es útil en tareas como la detección de URLs maliciosas.

Los siguientes conceptos son claves para el manejo de DNN:

Neuronas y Capas: Una red neuronal profunda tiene varias capas (entradas, ocultas y salidas), donde cada una de ellas está formada por neuronas. Cada neurona recibe una entrada, la procesa con una función de activación, y transmite la salida a la siguiente capa.

Propagación Hacia Adelante (Forward Propagation): se refiere al proceso de pasar los datos de entrada a través de las capas, hasta llegar a la salida.

Retropropagación (Backpropagation): es el ajuste de los pesos de las conexiones para minimizar el error en la predicción. Esto se logra calculando el gradiente del error con respecto a los pesos y actualizándolos, utilizando un algoritmo de optimización, como el descenso de gradiente.

Función de Activación: Ayuda a introducir no linealidad en el modelo, permitiendo que la red aprenda patrones complejos. Los ejemplos más comunes son:

Función sigmoide: “Es una función capaz de activar las capas ocultas, la cual toma un valor de entrada y lo comprime en un rango comprendido entre 0 y 1, útil para modelar probabilidades y realizar clasificaciones binarias, tiene la característica S y su fórmula es la siguiente:”[43]

$$f(x) = \frac{1}{1 + e^{-x}}$$

“Donde e es la constante, X es el valor de entrada, Cuando X es grande y positivo se aproxima a 1, si es grande y negativo se aproxima a 0 y tiene una media de 0.5.

La función sigmoide se utiliza para calcular la derivada de la función de pérdida en función de los pesos, lo que permite que el algoritmo de aprendizaje actualice los pesos de manera eficiente, tal como se puede observar en la Figura .[43]

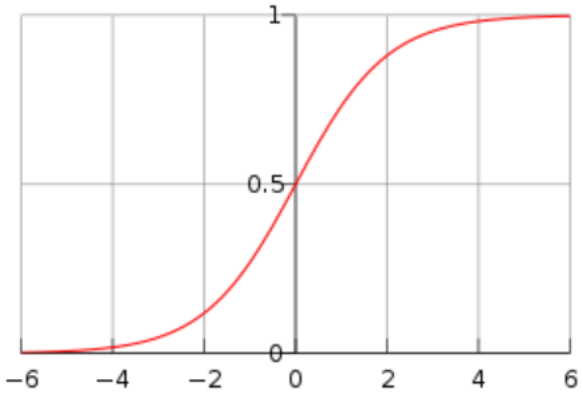


Figura 16. Función Sigmoide, valor medio 0,5. Fuente[43]

Función Tanh: (Tangente hiperbolica) “Tiene el comportamiento muy similar a la sigmoidal, con la diferencia de que los valores de salida estarán en el rango de -1 a 1.” Tal como se puede apreciar en Figura [44]

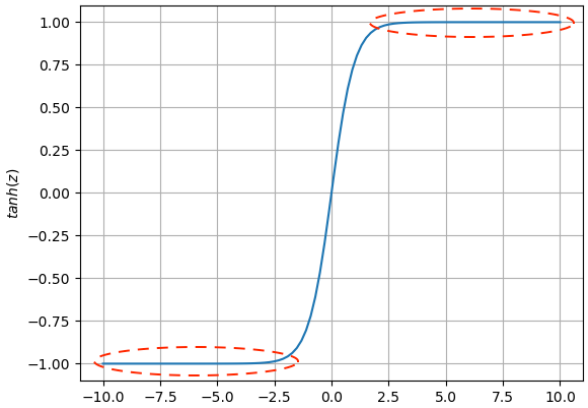


Figura 17. Función Tanh, donde su rango esta entre -1 y1. Fuente[44]

“En la gráfica anterior, se observa la presentación de problemas de saturación, una de sus ventajas es que ofrece una salida simétrica, lo cual facilita el proceso de entrenamiento.”[44]

Función ReLU: (Rectified Linear Unit): “Esta función genera una salida igual a 0 cuando la entrada (z) sea negativa, y una salida igual a la entrada, cuando la última sea positiva, esta función es la más usada ya que evita la saturación como si ocurre en Sigmoidal y Tanh, lo cual hace que el algoritmo del gradiente descendiente converja mucho más rápido y facilite el entrenamiento. Otra ventaja es que es más sencillo de implementar en comparación con otras situaciones, las cuales requieren funciones matemáticas más complejas como la exponencial y su fórmula es la siguiente:”[44]

$$\boldsymbol{f(x) = max(0, x)}$$

Donde X es la entrada de la neurona la cual es análoga a la rectificación de media onda en electrónica, activa los pesos según el resultado de dicha neurona en la IA, como se aprecia a continuación en Figura .

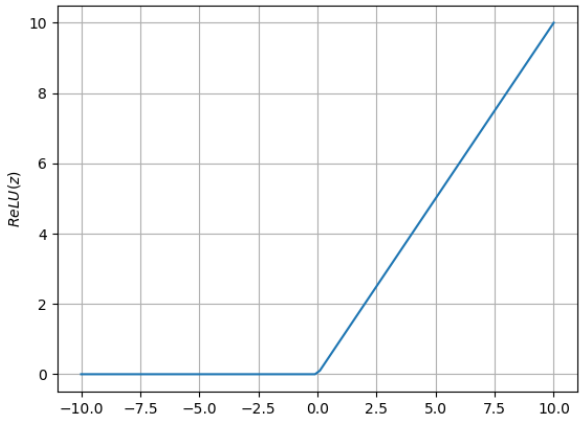


Figura 18. Función Relu, rectifica datos Z negativos y los devuelve 0. Valores Positivos no sufren cambios. Fuente[44]

Activación SoftMax: “Es especialmente útil en el contexto de los problemas de clasificación multiclase. Esta función opera sobre un vector, a menudo denominado logits, que representa las previsiones o las puntuaciones brutas de cada clase calculadas por las capas anteriores de una red neuronal. Para un vector de entrada x con elementos x1, x2,..., xC, la función softmax se define como:”[45]

$$f(xi) = \frac{e^{xi}}{\sum j * e^{xj}}$$

“El resultado de la función softmax es una distribución de probabilidad cuya suma es uno. Cada elemento del resultado representa la probabilidad de que la entrada pertenezca a una clase determinada. El uso de la función exponencial garantiza que todos los valores de salida sean no negativos. Esto es crucial porque las probabilidades no pueden ser negativas.”[45]

“Softmax se suele utilizar en la capa de salida de una red neuronal cuando la tarea consiste en clasificar una entrada en una de varias (más de dos) categorías posibles (clasificación multiclase). Como softmax amplifica las diferencias, puede ser sensible a los valores atípicos o extremos. Por ejemplo, si el vector de entrada tiene un valor muy grande, softmax puede "reducir" las probabilidades de otras clases, dando lugar a un modelo demasiado confiado.”[45]. Esta función se ejemplifica como se observa en la figura 19.

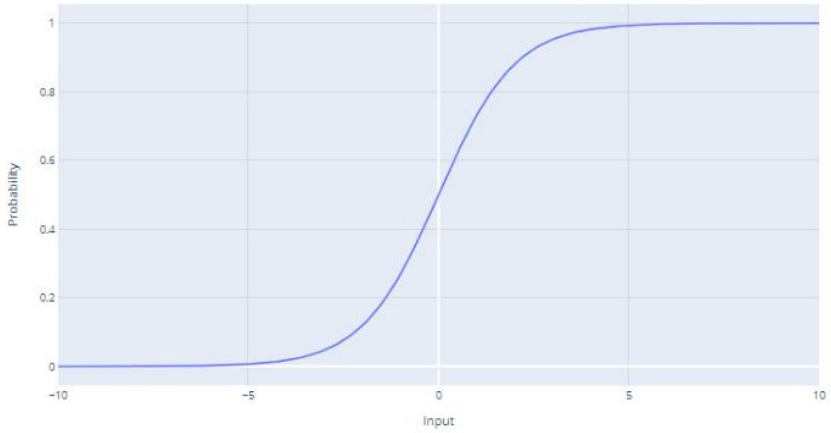


Figura 19. Función Softmax. Garantiza probabilidades mayores a 0. Fuente[45]

Para clasificar URLs Benign o Malicious con DNN, se necesitarán las siguientes librerías en Python:

- **TensorFlow: "**Es una de las librerías más principales para construir redes neuronales, su estructura permite usar todo tipo de operaciones de multiplicaciones matriciales, además tiene mejor rendimiento usando los procesadores grafico (GPU) enfocados en esas tareas. Funciona en un entorno de gran escala y de forma heterogénea. Tensorflow usa como forma de programación grafos de flujo de datos el cual nace bajo e trabajo de Google Brain que corresponde a un grupo de investigadores dedicados a la IA. Al estar basados en grafos, los nodos representan operaciones matemáticas, por otro lado, las conexiones del grafo representan los conjuntos de datos multidimensionales llamados tensores. Un tensor es un conjunto de datos primitivos los cuales suponen números flotantes o enteros los cuales se organizan en un array de 1 o N dimensiones."[41]
- **PyTorch:** "Creado para ser flexible y modular para la investigación, con la estabilidad y el soporte necesarios para el despliegue de producción. PyTorch proporciona un paquete de Python para funciones de alto nivel como el cálculo de tensor (como NumPy) con una fuerte aceleración de GPU y TorchScript para una transición fácil entre el modo «eager» y el modo gráfico. Con la última versión de PyTorch, el framework proporciona ejecución basada en gráficos, capacitación distribuida, implementación móvil y cuantización. La sencillez de su interfaz, y su capacidad para ejecutarse en GPUs (lo que acelera el entrenamiento de los modelos), lo convierten en la opción más asequible para crear redes neuronales artificiales. PyTorch tiene una

característica muy útil conocida como paralelismo de datos. Con esta función, PyTorch puede distribuir el trabajo computacional entre múltiples núcleos de CPU o GPU."[46]

- **Librería Keras en Deep Learning:** "Esta librería es una API para redes neuronales la cual permite ejecutarse en varias herramientas ML. Se importa mediante el paquete tf.keras, haciendo que tensorFlow sea más fácil de usar. Para poder construir un modelo en base Keras, se debe usar de forma secuencial y así permitir ensamblar diferentes capas para formar el modelo que se crea a partir de tf.keras.Sequential, en el que se toma 2 decisiones: Cuántas capas usar y Cuántas unidades ocultas habría que usar para cada capa."[41]
- **Dense:** Es el tipo más común de capa en DNN, donde cada neurona está conectada a todas las neuronas de capa anterior y la capa siguiente, al definir Dense, se especifica el número de Neuronas que se desea en ella y la función de activación que determinará como se propagan los valores de la próxima. De esta manera, el siguiente código muestra cómo se crea una capa con 64 neuronas y la función de activación relu:

```
Dense(64, activation='relu')
```

- **Dropout:** Es una técnica de regularización que se usa para prevenir el sobreajuste (Overtitting). Durante el entrenamiento, la capa Dropout desactiva aleatoriamente una fracción de las neuronas de la capa anterior en cada paso, lo que obliga a la red a aprender características más robustas y evitar que se ajuste demasiado a los datos de entrenamiento.
- **BatchNormalization:** Esta capa normaliza las salidas de la anterior para que tenga una media de 0 y una desviación estándar de 1 dentro de cada mini-batch, lo cual ayuda a acelerar el entrenamiento y estabilizar la red neuronal.
- **Optimizador Adam: "(**Adaptive Moment Estimation) Es un algoritmo de optimización adaptativo que se creó específicamente para el entrenamiento redes neuronales. Puede considerarse como una fusión del descenso de gradiente estocástico basado en el momento y RMSprop. Escala la tasa de aprendizaje utilizando gradientes al cuadrado, de forma similar a RMSprop, y aprovecha el momento utilizando el promedio móvil del gradiente en lugar del gradiente en sí, de forma similar a SGD con momento. Para estimar el momento, Adam utiliza promedios móviles exponenciales calculados sobre los gradientes evaluados en el minilote actual. Matemáticamente, esto se puede escribir como:"[47]

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

"Donde m y v son las medias móviles y g es el valor del gradiente. Las betas son hiperparámetros cuyos valores por defecto son, como se sugiere en el artículo, 0,9 y 0,999 respectivamente. Una vez se haya usado toda esta Info, Adam actualiza los pesos usando la siguiente fórmula similar a RMSprop:"[47]

$$w_t = w_{t-1} - n \frac{m_t}{\sqrt{v_t} + \varepsilon}$$

"Donde w es el peso, eta es la tasa de aprendizaje y epsilon es un valor infinitamente pequeño, normalmente 10^{-8}, que usamos para evitar la división por cero."[47]

3.14 Descomposición Urls:

"Para poder analizar el path de las URLs se debe usar el módulo urllib.parse el cual define una interfaz estándar para separar cadenas de texto del Localizador de recursos uniforme (más conocido por las siglas URL, del inglés *Uniform Resource Locator*) en componentes (esquema de dirección, ubicación de red, ruta de acceso, etc.), para poder combinar los componentes nuevamente en una cadena de texto URL, y convertir una «URL relativa» a una URL absoluta a partir de una «URL base»"[48]

"Este módulo ha sido diseñado para coincidir con el estándar de Internet RFC de los Localizadores de recursos uniformes relativos. Este módulo soporta los siguientes esquemasURL: file, ftp, gopher, hdl, http, https, imap, mailto, mms, news, nntp, prospero, rsync, rtsp, rtspu, sftp, shttp, sip, sips, snews, svn, svn+ssh, telnet, wais, ws, wss"[48]

Análisis de URL

urllib.parse.**urlparse**(*urlstring*, *scheme=''*, *allow_fragments=True*)

"Analiza una dirección URL en seis componentes, retornando una tupla nombrada de 6 elementos. Esto corresponde a la estructura general de una URL: scheme://netloc/path;parameters?query#fragment. Cada elemento de la tupla es una

cadena, posiblemente vacía. Los componentes no se dividen en piezas más pequeñas (por ejemplo, la ubicación de red es una sola cadena) y los caracteres % de escape no se expanden. Los delimitadores, como se muestra anteriormente, no forman parte del resultado, excepto una barra diagonal inicial en el componente *path*, que se conserva si está presente. Lo anterior se puede apreciar en la Figura ” [48]

```
>>> from urllib.parse import urlparse
>>> urlparse("scheme://netloc/path;parameters?query#fragment")
ParseResult(scheme='scheme', netloc='netloc', path='/path;parameters', params='',
            query='query', fragment='fragment')
>>> o = urlparse("http://docs.python.org:80/3/library/urllib.parse.html?"
...              "highlight=params#url-parsing")
>>> o
ParseResult(scheme='http', netloc='docs.python.org:80',
            path='/3/library/urllib.parse.html', params='',
            query='highlight=params', fragment='url-parsing')
>>> o.scheme
'http'
>>> o.netloc
'docs.python.org:80'
>>> o.hostname
'docs.python.org'
>>> o.port
80
>>> o._replace(fragment="").geturl()
'http://docs.python.org:80/3/library/urllib.parse.html?highlight=params'
```

Figura 20. Ejemplo Función urllib.parse. Fuente[48]

“Siguiendo las especificaciones de sintaxis en **RFC 1808**, urlparse reconoce un netloc sólo si es introducido correctamente por “//”. De lo contrario, se supone que la entrada es una dirección URL relativa, y por lo tanto, comienza con un componente de ruta de acceso.”[48]. Como se ejemplifica en la Figura 21.

```
>>> from urllib.parse import urlparse
>>> urlparse('//www.cwi.nl:80/%7Eguido/Python.html')
ParseResult(scheme='', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
            params='', query='', fragment='')
>>> urlparse('www.cwi.nl/%7Eguido/Python.html')
ParseResult(scheme='', netloc='', path='www.cwi.nl/%7Eguido/Python.html',
            params='', query='', fragment='')
>>> urlparse('help/Python.html')
ParseResult(scheme='', netloc='', path='help/Python.html', params='',
            query='', fragment='')
```

Figura 21. Ejemplo Función urllib.parse. Fuente[48]

“Los caracteres del atributo netloc que se descomponen en la normalización de NFKC (según lo utilizado por la codificación IDNA) en cualquiera de /, ?, #, @ o : lanzará un ValueError. Si la dirección URL se descompone antes del análisis, no se producirá ningún error.”[48]

“Como es el caso con todas las tuplas con nombre, la subclase tiene algunos métodos y atributos adicionales que son particularmente útiles. Uno de estos métodos es _replace(). El método _replace() retornará un nuevo objeto ParseResult, reemplazando los campos especificados por nuevos valores.”[48] como en la Figura 22.

```
>>> from urllib.parse import urlparse
>>> u = urlparse('//www.cwi.nl:80/%7Eguido/Python.html')
>>> u
ParseResult(scheme='', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
            params='', query='', fragment='')
>>> u._replace(scheme='http')
ParseResult(scheme='http', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
            params='', query='', fragment='')
```

Figura 22. Ejemplo Función urllib.parse. Fuente[48]

3.15 Medición de la Calidad:

“Se invierte gran cantidad de tiempo para poder mejorar la calidad de los resultados de los modelos obtenidos, por lo general no es posible saber visualmente si un resultado es mejor que otro, en este caso, resulta necesario procesar medidas de calidad para conocer con exactitud, los resultados. Así mismo, se emplea diferentes medidas de calidad las cuales difieren para qué tipo de aprendizaje se está usando, que en definitiva sería clasificación, en cuales se va a presentar a continuación, no sin antes presentar su metodología”.

“La regla de oro en metodología de la validación es: no medir la calidad con el mismo conjunto de datos empleado para obtener el modelo. Lo que se quiere es evitar situaciones en las que el modelo aprende cada una de las muestras de datos, pero es incapaz de tratar datos nuevos. Esta es la situación de sobreajuste (overfitting) que se ha explicado anteriormente, al principio de la sección precedente (pastores alemanes y gatos). Es necesario dividir las muestras de datos y los valores objetivo en varios conjuntos:”[40]

- **Entrenamiento**
- **Validación**
- **Test**

“El conjunto de datos de entrenamiento es el mayor: un valor típico es que sea el 80% del conjunto total de muestras. Se usa para entrenar el modelo. El conjunto de validación contiene las muestras usadas para mejorar el modelo a base de realizar un ajuste fino en los híper-parámetros (valores que controlan diferentes variaciones en el funcionamiento de los algoritmos). Por último, el conjunto de test (o pruebas) permitirá medir la calidad del modelo utilizando las muestras que no hayan sido usadas para entrenar el modelo o para mejorarlo. Del mismo modo, se debe asegurar de que los tres conjuntos tengan una distribución similar de los datos que son representativos y que su intersección sea nula. Una vez que entrenado y mejorado el modelo con el conjunto de muestras junto con el grupo de muestras de validación, podremos usar las muestras de test para hacer predicciones con datos que no han sido previamente procesados. Los indicadores de la calidad serán mayores a medida que se asemejen las predicciones obtenidas, a los valores objetivo-existentes.”[40]

“Existe un enfoque que hace posible diseñar medidas de calidad más específicas para la clasificación. Si se centra la atención en la clasificación binaria, en la matriz de confusión, solamente pueden existir dos clases (grupos)”[40]. Tal como se puede observar en la Tabla 1:

Predicción	Clase 0 (Positivo)	Clase 1 (Negativo)
Clase 0	Verdadero Positivo (TP)	Falso Negativo (FN)
Clase 1	Falso Positivo (FP)	Verdadero Negativo (TN)

Tabla 1. Clase 0 y 1. Matriz de confusión. Fuente: este trabajo.

“Para mostrar un ejemplo ilustrativo, supongamos que tenemos 100 URLs de test etiquetadas: (‘Malicious’, ‘Benign’). 45 de ellas han sido predichas correctamente como Benign, 35 han sido predichas correctamente como Malicious, 5 han sido predichas incorrectamente como Benign y 15 han sido incorrectamente predichas como Malicious.”[40]

Predicción URL	Benign	Malicious
Benign	45	15
Malicious	5	35

Tabla 2. Predicción URL (Malicious, Benign). Fuente: este trabajo.

Según[40]“La medida de calidad denominada Precisión, nos muestra la proporción de aciertos en la predicción: ¿Cuántas de sus predicciones son correctas? Se expresa de la siguiente manera:

$$Precisión = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

La medida de calidad Recall centra su atención en el número relativo de muestras positivas (en nuestro ejemplo, tumores benignos): ¿Cuántas de sus predicciones son correctas de manera relativa al número total de muestras positivas?

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Mientras que Precisión nos proporciona un valor de calidad relativo al número total de predicciones realizadas, Recall nos proporciona un valor de calidad relativo al número total de muestras positivas. Merece la pena mantener niveles altos de Precisión cuando el número de predicciones crece. Merece la pena mantener niveles altos de Recall cuando el número de muestras positivas es bajo. Dependiendo de la semántica de cada problema de inteligencia artificial, los valores de la matriz de confusión cobran mayor o menor importancia. En este ejemplo, es vital mantener el número de falsos positivos (URLs Malicious no detectadas) tan bajo como sea posible y resulta muy importante predecir de manera correcta, los negativos verdaderos (URLs Malicious detectadas)”.

Existe una medida de calidad de clasificación que une Precisión y Recall; esta corresponde a la F1, de la siguiente manera:

$$F1 = 2\frac{\mathrm{Precisión} * \mathrm{Recall}}{\mathrm{Precisión} + \mathrm{Recall}}$$

“En el caso de que todas las clases sean igualmente importantes, se puede usar la siguiente medida de calidad de exactitud (accuracy)”[38]:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Para medir la calidad de la clasificación también se usa la curva ROC: combina los valores verdaderos positivos con los falsos positivos, definida como:

$$TPR = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, FPR = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

Cuanto mayor sea el área bajo la curva ROC, mejor será el modelo. Los valores obtenidos deben ser mayores de 0.5, ya que 0.5 corresponde con un clasificador aleatorio.

3.16 Estrategias de escalabilidad

La actualización de la base de datos se incrementará con la carga de información que se introduzca, haciendo que el mecanismo de aprendizaje sea más robusto y seguro. Por el momento, el algoritmo se pondrá a prueba en un entorno controlado, además se realizará un preámbulo al proceso de bloqueo, no se contaría con un sistema de servidores cloud y solo en caso de que sea implementado y puesto en marcha en un entorno laboral, habría la posibilidad de optimizar el algoritmo, conforme al avance en el manejo del sistema de aprendizaje.

4 DESARROLLO DEL PROGRAMA PYTHON PARA LA PUESTA EN MARCHA DEL MODELO DE APRENDIZAJE PROPUESTO

Como se menciona anteriormente, el código que se va a llevar a cabo en el sistema de detección y bloqueo de URLs, será en Python, ya que se trata de un lenguaje de alto nivel, el cual permite, por medio de sus diversas librerías, emplear métodos para analizar cada URLs que está almacenada en la base de datos de Excel, llamada DatasetMintic.csv. En este archivo están las URLs de contenido malicioso que reporta MINTIC, además de las URLs benignas. Posteriormente, se agrupará en 2 categorías, en un mismo archivo para que el modelo se pueda entrenar y sea capaz de diferenciar entre una URLs Maliciosa y una Benigna. En el desarrollo de este código se utilizaron las siguientes plataformas:

4.1 Jupyter Lab

“Es una de las herramientas más útiles en el análisis de datos con Python, en esta plataforma se puede cargar datos, transformarlos y modelarlos dentro de una única ventana, además, permite probar el código y explorar ideas de forma rápida y fácil. Del mismo modo, aprueba

documentar el código, pues agrega texto con formato de forma, incluso para generar un reporte del código realizado.”[41]

“Jupyter Lab maneja 2 componentes principales que son un conjunto de núcleos llamados: (Interpreter) y Dashboard. Cada núcleo o kernel es un motor de ejecución para un lenguaje que se encarga de procesar las solicitudes y devolver las respuestas apropiadas, Jupyter trabaja con el kernel, por defecto IPython que es un intérprete de líneas de comando, permite trabajar con lenguaje Python.”[41]

“Jupyter Lab tiene la capacidad de asociarse con otros lenguajes como C++, R, Julia, Ruby, JavaScript, CoffeeScript, PHP o Java. Por un lado, el dashboard (Panel de Control), funciona como una interfaz de administración de cada uno de los kernels y por otro lado, el centro de control permite crear nuevos documentos o abrir proyectos existentes.”[41]

“El programa se ejecuta en un entorno web y funciona para cualquier navegador, inicialmente se debe instalar y ejecutar en el sistema el servidor Jupyter Lab. Toda la documentación se puede exportar a diferentes formatos HTML, PDF, Markdown o Python, además se puede compartir con otros usuarios vía E-Mail. Una vez se haya instalado el software, se puede iniciar el servidor Notebook junto con la línea de comando y luego el dashboard en el navegador predeterminado mediante la URL: http://localhost:8888”[41]

Para este trabajo, se ha creado la carpeta llamada “Tesis” que contiene los 5 códigos de programación y 2 archivos .CSV que son los Datasets, tal como se puede apreciar en la Figura 23. Archivos alojados en carpeta "Tesis". Fuente: este trabajo.

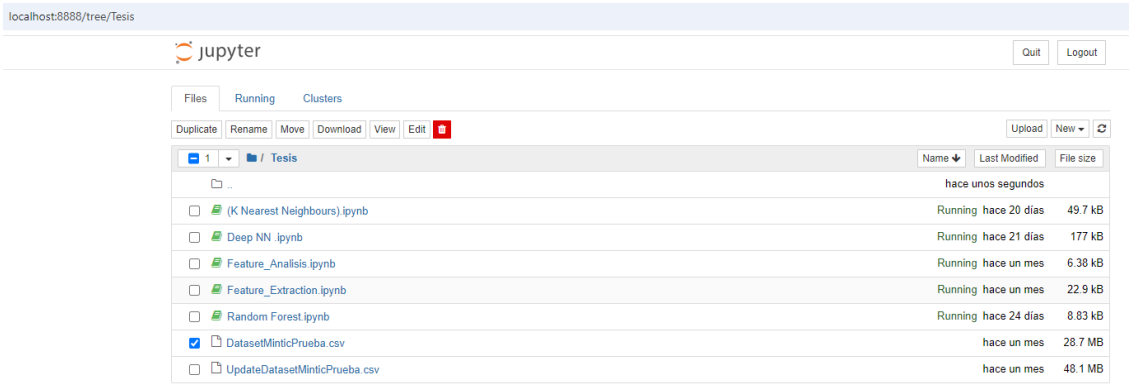
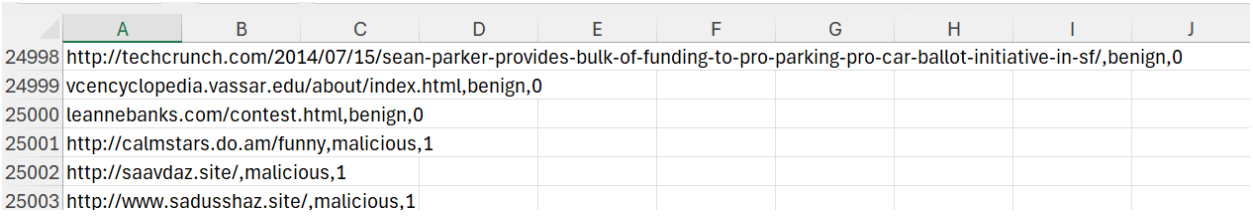


Figura 23. Archivos alojados en carpeta "Tesis". Fuente: este trabajo.

Para poder ejecutar el primer código llamado Feature_Extraction.ipynb, se debe tener el dataset con las URLs agregadas tanto Benignas como maliciosas en un mismo archivo, esto se realizó bajo los siguientes parámetros:

4.2 Ordenamiento de URLs en Archivo .CSV

Para consolidar el dataset con los 2 conjuntos de URLs etiquetadas como “Benign,0” y “Malicious,1” se tomaron las 23.620 URLs de contenido de pornografía infantil a través de la página de Mintic[50], la cual únicamente tiene acceso los ISP que operan en territorio colombiano. Luego, desde la página Kaggle[51], se tomó el dataset llamado Malicious_URL`s_Dataset, donde almacena 651.191 URLs. De toda esta cantidad vamos a necesitar 24.999 URLs las cuales están etiquetadas como “Benign”. Una vez filtradas las URLs con etiqueta “Benign,0”, se procede a crear un nuevo archivo llamado “DatasetMintic.csv” y se consolida las 2 categorías mencionadas, para poder cargarlo en el primer código propuesto. Lo anteriormente señalado, se puede observar en la Figura 24:



	A	B	C	D	E	F	G	H	I	J
24998	http://techcrunch.com/2014/07/15/sean-parker-provides-bulk-of-funding-to-pro-parking-pro-car-ballot-initiative-in-sf/,benign,0									
24999	vcencyclopedia.vassar.edu/about/index.html,benign,0									
25000	leannebanks.com/contest.html,benign,0									
25001	http://calmstars.do.am/funny,malicious,1									
25002	http://saavdaz.site/,malicious,1									
25003	http://www.sadusshaz.site/,malicious,1									

Figura 24. Archivo DatasetMintic.csv con sus etiquetas "Benign,0 y Malicious,1". Fuente: este trabajo

4.3 Feature Engineering de URLs para Detección de Sitios Maliciosos usando Python

Se instala las librerías principales que utiliza el sistema para el análisis de las URLs a tratar, la librería pandas maneja los datos en forma de tabla en este caso el documento Excel DatasetMintic.csv, librería seaborn para visualilzación en matplotlib como las gráficas que va a representar la matriz de confusión, librería re para trabajar expresiones regulares, útil para extraer patrones en textos como partes de una url, librería warning para no llenar la consola de mensajes innecesarios.

Para el procesamiento de URLs se va a manejar la librería tld la cual extrae el dominio de nivel superior (.com, .org) de una URL, la librería urlparse extrae partes de una URL como el

dominio, patch, etc. Para la ejecución de modelos machine learning se va a utilizar la librería SVC que es clasificador de máquinas de vectores de soporte, librería confusion_matrix que mide el rendimiento del modelo el cual fue predicho de manera correcta y de cual no lo fue, librería classification_report la cual da un resumen de métricas como precisión, recall, librería RandomForestClassifier uno de los métodos ML que se usará para el análisis de URLs a clasificar, librería train_test_split divide los datos de entrenamiento y prueba, librería TfidfVectorizer transforma el texto en números usando TF-IDF, tal como se aprecia a continuación en la Figura 25.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

#Importación de Dependencias
import re
import warnings
warnings.filterwarnings('ignore')
from tld import get_tld
from urllib.parse import urlparse

#Importación pip install scikit-learn
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
```

Figura 25. Importación de Módulos y librerías. Fuente este trabajo.

Se carga el archivo DatasetMintic.csv y se ignora los errores que pueda tener el archivo como algúna columna faltante o si hay líneas con ciertos errores las ignora, después selecciona la columna result la cual esta etiquetada con Benign(0) Malicious(1) y se convierte en un valor entero de 64bits para la clasificación binaria, como en la Figura 26:

```
[3]: # Cargar el dataset
     dataset_url = pd.read_csv("./DatasetMintic.csv", on_bad_lines='skip')

[457]: dataset_url['result'] = dataset_url['result'].fillna(0).astype('int64')

[459]: dataset_url.head()
```

[459]:

	url	type	result
0	https://www.eltiempo.com	benign	0
1	https://www.elespectador.com	benign	0
2	https://www.semana.com	benign	0
3	https://www.caracoltv.com	benign	0
4	https://www.rcnradio.com	benign	0

Figura 26. Cargue y limpieza de los datos. Fuente este trabajo.

Se agrupa los datos del Dataframe denominado dataset_url conforme la columna “type” para poder contabilizar las filas que hay para los grupos de Benig y Malicious. Ver Figura 27:

```
[463]: dataset_url.groupby("type").count()
```

[463]:

type	url	result
benign	24999	24999
malicious	23502	23502

```
[465]: dataset_url.shape

[465]: (48526, 3)
```

Figura 27. Agrupamiento de datos Benig-Malicious. Fuente: este trabajo.

Se genera la columna denominada url_length la cual almacena la longitud de los caracteres de cada URL la cual es útil para identificar si la URL a analizar tiene muchos caracteres, la mayoría de las veces suele ser denominada “Malicious” Además se implementa la función get_hostname_length la cual extrae las características útiles de las URLs, urlparse separa la URL en partes como el dominio, patch, etc. Posteriormente calcula cuantos caracteres tiene dicho dominio. También se agrega la linea path_length para calcula la longitud del patch(caracteres antecedidos por /)

```
•[477]: # Función para extraer la longitud
def get_hostname_length(url):
    try:
        parsed_url = urlparse(url)
        return len(parsed_url.netloc)
    except Exception as e:
        print(f"Error parsing URL '{url}': {e}")
        return -1  # Indicate error with a sentinel value

# Apply the function to create the 'hostname_length' column
dataset_url['hostname_length'] = dataset_url['url'].apply(get_hostname_length)

•[479]: #Longitud del Path
dataset_url['path_length'] = dataset_url['url'].apply(lambda i: len(urlparse(i).path))
```

Figura 28. Función para extracción de longitud de URL. Fuente: este trabajo.

Se aplican funciones adicionales como fd_length que sirve para extraer una feature muy específica del path de la URL, toma el primer directorio después del / y cuenta cuantos caracteres tiene, si no hay directorio devuelve 0. Se extrae y se analiza el dominio de nivel superior(TLD) bajo la función tld_length toma cada URL y extrae su dominio (.com, .info, .co, etc) crea una nueva columna llamada “tld” y cuenta cada carácter a partir de dicho dominio. Esta función es muy útil para determinar si el TLD es muy largo o inusual como, por ejemplo .tk, .xyz, .info pueda ser denominado como “Malicious”. Según la Figura 29.

```
#Longitud del Primer Directorio.

def fd_length(url):
    try:
        urlpath = urlparse(url).path
        if len(urlpath.split('/')) > 1:
            return len(urlpath.split('/')[1])
        else:
            return 0
    except ValueError:
        return 0

dataset_url['fd_length'] = dataset_url['url'].apply(fd_length)
```

```
#Longitud de dominio de nivel superior.
dataset_url['tld'] = dataset_url['url'].apply(lambda i: get_tld(i,fail_silently=True))
def tld_length(tld):
    try:
        return len(tld)
    except:
        return -1

dataset_url['tld_length'] = dataset_url['tld'].apply(lambda i: tld_length(i))
```

Figura 29. Funciones primer directorio y nivel superior. Fuente: este trabajo.

Se procede a extraer caracteres especiales y patrones comunes de las URLs, la cual se usa en procesos de ciberseguridad para detección de phishing o malware. Cada línea cuenta el

número de caracteres de cada URL, como por ejemplo la primera línea cuenta los guiones que son usados para poder confundir o alargar nombres de dominio, la segunda línea cuenta las @ que suelen ser usadas para redirigir o disfrazar URLs, la tercera línea se marca la ? como el inicio de los parámetros de cada URL, la linea que cuenta % es usada frecuentemente para codificar caracteres especiales lo cual, los atacantes lo usan para evadir filtros, los puntos (.) se los usa para subdominios, la linea que cuenta = pueden ser usados para indicar manipulación o tracking. Las líneas que cuentan count-http - count-https suelen ser usadas las 2 en una sola URL, lo que indica que puede estar incrustada una segunda URL y puede ser redireccionado a sitios sospechosos. Por último, la línea que cuenta www. En caso de aparecer más de una vez en una misma URL se denominaría como sospechosa según la Figura 30.

```
dataset_url['count-'] = dataset_url['url'].apply(lambda i: i.count('-'))
dataset_url['count@'] = dataset_url['url'].apply(lambda i: i.count('@'))
dataset_url['count?'] = dataset_url['url'].apply(lambda i: i.count('?'))
dataset_url['count%'] = dataset_url['url'].apply(lambda i: i.count('%'))
dataset_url['count.'] = dataset_url['url'].apply(lambda i: i.count('.'))
dataset_url['count='] = dataset_url['url'].apply(lambda i: i.count('='))
dataset_url['count-http'] = dataset_url['url'].apply(lambda i : i.count('http'))
dataset_url['count-https'] = dataset_url['url'].apply(lambda i : i.count('https'))
dataset_url['count-www'] = dataset_url['url'].apply(lambda i: i.count('www'))
```

Figura 30. Extracción de caracteres especiales. Fuente: este trabajo.

Se implemente la función having_ip_address para contabilizar direcciones IPv4 como por ejemplo 192.168.0.1 Direcciones IPv4 en formato hexadecimal como 0xC0.0xA8.0x00.0x01 y Direcciones IPv6 como por ejemplo 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Dicha función retorna un -1 si encuentra una IP en lugar de dominio y devuelve un 1 si no hay IP. Esta función es útil para detectar si los sitios son legítimos ya que es muy raro que una página Benign use una dirección Ip directamente y podría considerarse como “Malicious”, otra razón es porque muchos sitios maliciosos intentan evadir la detección al implementar IPs directas en lugar de nombres de dominio de acuerdo con la Figura 31.

```
#Uso de IP en Dominio
def having_ip_address(url):
    match = re.search(
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4
        '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)' # IPv4 in hexadecimal
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)  # Ipv6
    if match:
        # print match.group()
        return -1
    else:
        # print 'No matching pattern found'
        return 1
dataset_url['use_of_ip'] = dataset_url['url'].apply(lambda i: having_ip_address(i))
```

Figura 31. Función para detectar IPv4 e IPv6. Fuente: este trabajo.

Por último, se agrega la función shortening_service para identificar URLs sospechosas las cuales intentan ocultar el verdadero destino, dicha función busca si la URL contiene servicios de acortamiento de enlaces, los más comunes son bit.ly, tinyurl.com, ow.ly, goo.gl entre otros. Si la URL está usando un acortador devuelve un -1, de lo contrario devuelve un 1. Además, crea una nueva columna llamada “short_url” donde va a alojar dicho valor. Esta función es importante para poder identificar si las URLs a analizar presentan acortamiento y así denominarlas como “Malicious” ya que ocultan un destino real y suelen ser usadas por enlaces phishing, así evaden la detección y pueden engañar a los usuarios.

```
def shortening_service(url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                      'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
                      'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                      'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                      'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
                      'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
                      'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                      'tr\.im|link\.zip\.net',
                      url)
    if match:
        return -1
    else:
        return 1
dataset_url['short_url'] = dataset_url['url'].apply(lambda i: shortening_service(i))
```

Figura 32. Identificación acortamiento URLs. Fuente: este trabajo.

AL finalizar de implementar las funciones necesarias para tener un completo cubrimiento de detección de URLs, cada función crea una columna en donde inserta el dato como resultado explicado anteriormente, se puede observar que ahora tiene 22 columnas las cuales contienen 19 features en la siguiente Figura 33.

```
dataset_url.head()
```

	url	type	result	url_length	hostname_length	path_length	fd_length	tld_length	count-	count@	...	count.	count=	count-http	count-https	count-www	count-digits	count-letters	count_dir	use_of_ip
0	https://www.eltiempo.com	benign	0	24	16	0	0	3	0	0	...	2	0	1	1	1	0	19	0	1
1	https://www.elespectador.com	benign	0	28	20	0	0	3	0	0	...	2	0	1	1	1	0	23	0	1
2	https://www.semana.com	benign	0	22	14	0	0	3	0	0	...	2	0	1	1	1	0	17	0	1
3	https://www.caracoltv.com	benign	0	25	17	0	0	3	0	0	...	2	0	1	1	1	0	20	0	1
4	https://www.rcnradio.com	benign	0	24	16	0	0	3	0	0	...	2	0	1	1	1	0	19	0	1

5 rows × 22 columns

Figura 33. dataset_url con 19 features. Fuente: este trabajo.

4.4 (K Nearest Neighbours)

Una vez analizado el dataset, se procede a emplear el primer método de ML de clasificación a través de vecinos más cercanos, a continuación se explica el código:

Se importó además de las librerías más usadas como Pandas y Numpy, las librerías como Counter, la cual nos permite contar objetos de una lista además de contar etiquetas, la métrica de evaluación como confusión_matrix, muestra el rendimiento del modelo a realizar, classification_report, que genera informe como precisión y puntaje F1. Importa KDTree de Scipy una estructura de datos para una búsqueda eficiente de vecinos más cercanos en espacios multidimensionales, Matplotlib librería la cual nos sirve para los gráficos, según la Figura 34.

```
#imports
import numpy as np
import pandas as pd
from collections import Counter
from sklearn.metrics import f1_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
from scipy.spatial import KDTree
from concurrent.futures import ThreadPoolExecutor
import matplotlib.pyplot as plt
import seaborn as sns
```

Figura 34. Importación librerías. Fuente este trabajo.

Se creó la clase KNNClassifier, diseñada para la implementación del modelo K Nearest Neighbors. __init__ que define el constructor de la clase, lo cual crea una nueva instancia y toma los parámetros como k=3 que es el número de vecino más cercanos y considera un punto por defecto que es el 3. Despues n_jobs=1 indica la cantidad de trabajos en paralelo que se puede ejecutar, el 1 significa que solo usa un subproceso y por último la variable self.k asigna valor K, que representa el num de vecino, self.data es el atributo que almacenará los datos,

después self.labels asignará las etiquetas, para pasar a self.tree que almacenará KDTree para la búsqueda eficiente de KNN.

Posteriormente se definen más métodos los cuales se usaron para representar en texto el objetivo, actualización de parámetros K y n_jobs l que es útil en caso de realizar alguna modificación de valores sin la necesidad de crear una nueva instancia. El método fit entrena el método KNNClassifier almacenando datos y etiquetas de entrenamiento, lo cual construye una estructura eficiente para la búsqueda de KNN y por último el método predict realiza predicciones de clase para la entrada X que usa el paralelismo para mejorar la velocidad de las predicciones si n_jobs es mayor a 1, tal como se observa en la Figura 35.

```
class KNNClassifier:
    def __init__(self, k=3,n_jobs=1):
        self.k = k
        self.data = None
        self.labels = None
        self.tree = None
        self.n_jobs = n_jobs

    def __repr__(self):
        return f"KNNClassifier(k={self.k}, n_jobs={self.n_jobs})"

    def set_params(self, k=None, n_jobs=None):
        if k is not None:
            self.k = k
        if n_jobs is not None:
            self.n_jobs = n_jobs

    def fit(self, x, y):
        x = np.array(x)
        y = np.array(y)
        if x.shape[0] != y.shape[0]:
            raise ValueError("Number of samples in x and y must be equal.")
        self.data = x
        self.labels = y
        self.tree = KDTree(self.data)

    def predict(self, X):
        X = np.array(X)
        if X.ndim == 1:
            X = X.reshape(1, -1)  # Ensure X is 2D for a single sample
        with ThreadPoolExecutor(max_workers=self.n_jobs) as executor:
            predictions = list(executor.map(self._predict, X))
        return np.array(predictions)
```

Figura 35. Se generan los métodos KNN. Fuente: este trabajo.

Una vez se haya definido los métodos KNN se procede a realizar la búsqueda de hiperparametros, k_values, se itera sobre cada valor de k que se quiere evaluar, para cada K

inicia una lista vacía llamada scores, la cual almacenará las puntuaciones de validación, posteriormente se realiza la validación cruzada entre x_train, y_train que genera índices para dividir los datos en pliegue de entrenamiento y validación, se realiza la creación de pliegues X_train_fold y X_val_fold que son los datos de características de entrenamiento y validación y tenemos y_train_fold y y_val_fold que son las etiquetas correspondientes a los datos de entrenamiento.

Entrenamiento del modelo: Se crea instancia del clasificador KNN con el valor actual k, que permite el uso de 4 trabajos en paralelo: está el método fit sirve para entrenar los datos de pliegue, después se usa el modelo entrenado para poder predecir las etiquetas de validación, se calcula la puntuación F1 usando f1_scores, la cual considera la precisión como el recall. Se calcula la puntuación promedio de validación cruzada para el valor actual de K y se almacena en cv_scores, se realiza la comparación de la puntuación promedio para el valor actual de K que es la mejor puntuación registrada y se actualiza best_score y best_k, para realizar la impresión de puntuaciones promedio de validación cruzada para los valores de K que se aprobaron. De acuerdo con la Figura 36.

```
# Entrenamiento y Evalkuación con KNN
best_knn = KNNClassifier(k=best_k, n_jobs=4)
best_knn.fit(x_train, y_train)
y_pred = best_knn.predict(x_test)
```

```
Average cross-validation score for k=1: 0.9926287344918933
Average cross-validation score for k=2: 0.9915964314478612
Average cross-validation score for k=3: 0.9906956394437803
Average cross-validation score for k=4: 0.990230391298461
Average cross-validation score for k=5: 0.9904892903096154
Average cross-validation score for k=6: 0.9899727414968288
Average cross-validation score for k=7: 0.9899219917607087
Average cross-validation score for k=8: 0.989586132164793
Average cross-validation score for k=9: 0.9895352047324868
Best k: 1
Best cross-validation score: 0.9926287344918933
```

Figura 36. Comparación de la puntuación promedio para el valor actual de K. Fuente: este trabajo.

Por último, se generó la matriz de confusión que favorece el entendimiento de la clasificación de las URLs, las cuales han sido etiquetadas como “Benign” y “Malicious”. Se entrena el 80% de URLs y se pone a prueba el porcentaje restante que es el 20% Según por el metodo KNN, en este caso se observó que 4.958 URLs “Benign” son verdaderos positivos, 38 URLs son falsos positivos (URLs “Benign” que KNN clasifico como “Malicious”), 4.687 URLs “Malicious” como verdadero negativo y 18 URLs como falso negativo (URLs “Malicious” que KNN clasificó como “Benign”). Según se ejemplifica en la Figura 37.

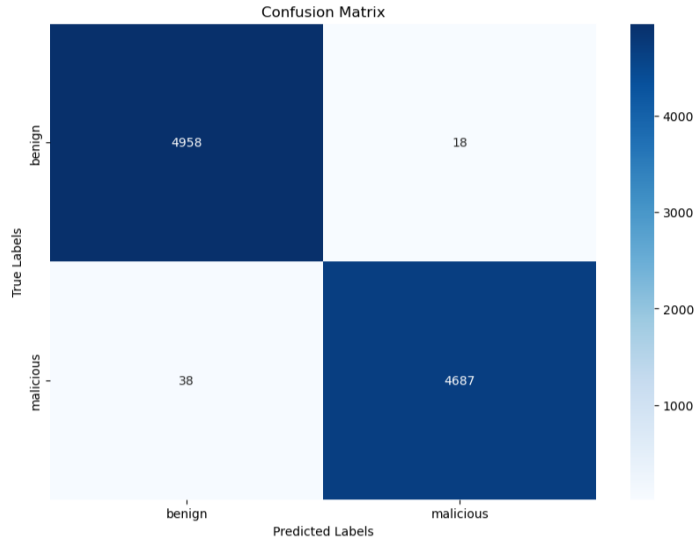


Figura 37. Matriz Confusión KNN. Fuente: este trabajo.

Validación Cruzada.

El método de validación cruzada sirve para dividir los datos en k partes(folds), se entrenan el modelo k veces, usa partes distintas como test y calcula las métricas de rendimiento para cada instancia, este método de validación es muy útil para la detección de overfitting ya que mitiga el riesgo de evaluar el modelo por una mala división de datos. El propósito de evaluar el overfitting del modelo K-Neighbors probando distintos valores de K vecinos por medio del método de validación cruzada estratificada (5Folds) para poder entrenar el modelo de forma robusta con cada K.Se realiza el calculo de la diferencia entre el F1-Score en entrenamiento y validación, para determinar si hay una brecha grande que es igual a overfitting, se asigna un nivel de riesgo basado en el tamaño de la brecha así poder determinar de manera automática que el valor de K genera un modelo mas generalizable sin sobreajuste.

```python
skf = StratifiedKFold(n_splits=cv, shuffle=True, random_state=42)

for k in k_values:
    print(f"🔍 Evaluando k={k}...")

    train_scores = []
    val_scores = []

    for fold, (train_index, val_index) in enumerate(skf.split(X_vectorized, y)):
        X_train, X_val = X_vectorized[train_index], X_vectorized[val_index]
        y_train, y_val = y[train_index], y[val_index]

        # Entrenar modelo
        knn = KNeighborsClassifier(n_neighbors=k, n_jobs=4)
        knn.fit(X_train, y_train)

        # Predecir en train y validation
        y_train_pred = knn.predict(X_train)
        y_val_pred = knn.predict(X_val)

        # Calcular F1 scores
        train_f1 = f1_score(y_train, y_train_pred, average='weighted')
        val_f1 = f1_score(y_val, y_val_pred, average='weighted')

        train_scores.append(train_f1)
        val_scores.append(val_f1)
```

Figura 38. Código validación cruzada para KNN. Fuente: este trabajo.

Los resultados del modelamiento KNN por validación cruzada muestra estabilidad, los valores de K del 1 al 14 muestran un riesgo mínimo, las diferencias entre entrenamiento y validación son del 1.12% lo cual se concluye que el modelo generaliza de forma eficiente. El Score de validación se mantiene alto en todos los casos lo cual significa que no sacrifica rendimiento por generalizar. Finalmente se selecciona K=2 porque tiene alto rendimiento de 0.9900, mínima diferencia del 0.69% entre train y validación y máxima generalización.

```
======================================================================
📈 RESULTADOS DEL ANÁLISIS DE OVERFITTING
======================================================================
 k  train_score  val_score  score_difference overfitting_risk
 1     1.000000   0.988791          0.011209           MÍNIMO
 2     0.996883   0.989963          0.006920           MÍNIMO
 3     0.994951   0.989203          0.005749           MÍNIMO
 4     0.994163   0.989572          0.004590           MÍNIMO
 5     0.992571   0.988811          0.003760           MÍNIMO
 6     0.992452   0.989284          0.003168           MÍNIMO
 7     0.991325   0.988152          0.003173           MÍNIMO
 8     0.991463   0.988769          0.002694           MÍNIMO
 9     0.990300   0.987719          0.002581           MÍNIMO
10     0.990238   0.987801          0.002437           MÍNIMO
11     0.989110   0.986339          0.002771           MÍNIMO
12     0.989032   0.986791          0.002241           MÍNIMO
13     0.987817   0.985865          0.001952           MÍNIMO
14     0.987935   0.985988          0.001947           MÍNIMO

🎯 MEJOR PARÁMETRO: k=2
   Validation Score: 0.9900
   Riesgo de overfitting: MÍNIMO
```

Figura 39. Resultados Overfitting KNN. Fuente: este trabajo.

Recall.

Para el modelamiento KNN se tiene un recall promedio alto de 0.9926, lo cual detecta mas del 99% de las URLs “Malicious” se tiene una desviación estándar mínima del 0.0013 y rango estrecho, 0.0032 de diferencia del menor al mayor resultado, lo que indica que el recall es consistente, sin fluctuaciones. El valor K=1 obtiene el mejor recall de 0.9936 lo que confirma que al usar el vecino mas cercano maximiza la detección de URLs “Malicious” aunque K=1 tiene el recall mas alto, análisis previos muestran que también tenia brecha grande del 1.12%. Sin embargo, esta diferencia es mínima.

```
============================================================
ESTADÍSTICAS FINALES DE RECALL MALICIOUS
============================================================
Recall Malicious promedio: 0.9926
Desviación estándar: 0.0013
Rango: [0.9906 - 0.9938]


============================================================
OPTIMIZACIÓN PARA RECALL MALICIOUS
============================================================
k=1: Recall Malicious = 0.9936 (±0.0014)
k=2: Recall Malicious = 0.9864 (±0.0023)
k=3: Recall Malicious = 0.9926 (±0.0013)
k=4: Recall Malicious = 0.9889 (±0.0019)
k=5: Recall Malicious = 0.9920 (±0.0017)
k=6: Recall Malicious = 0.9893 (±0.0012)
k=7: Recall Malicious = 0.9914 (±0.0013)
k=8: Recall Malicious = 0.9897 (±0.0012)
k=9: Recall Malicious = 0.9913 (±0.0014)
k=10: Recall Malicious = 0.9893 (±0.0013)
k=11: Recall Malicious = 0.9904 (±0.0014)
k=12: Recall Malicious = 0.9887 (±0.0011)
k=13: Recall Malicious = 0.9900 (±0.0015)
k=14: Recall Malicious = 0.9885 (±0.0012)

Mejor k para Recall Malicious: 1
Mejor Recall Malicious: 0.9936
```

Figura 40. Resultados Recall KNN. Fuente: este trabajo.

En el resultado para el modelo KNN con validación cruzada, se observó que 23.327 URLs “Benign” son verdaderos positivos, 349 URLs son falsos positivos (URLs “Benign” que RF clasifico como “Malicious”), 24.675 URLs “Malicious” como verdadero negativo y 175 URLs como falso negativo (URLs “Malicious” que RF clasificó como “Benign”). Según se ejemplifica en la Figura 41.

```
MATRIZ DE CONFUSIÓN:
[[24675   349]
 [  175 23327]]

Verdaderos Positivos (Malicious): 23327
Falsos Negativos (Malicious): 175
Verdaderos Negativos (Benign): 24675
Falsos Positivos (Benign): 349
```

Figura 41. Matriz Confusión KNN-Validación Cruzada Fuente: este trabajo.

4.5 Random Forest

Extracción y División de Datos para Entrenamiento.

Las siguientes líneas de código extrae las columnas de dataset_url a través de pandas se selecciona inicialmente con ["url"] la cual se está guardando en la variable x y ["result"] la cual contiene las etiquetas “Benign,0” y “Malicious,1” las cuales se guardan en la variable y. Después usamos .head() para mostrar las primeras 5 filas tanto de x como de y.

```
# Features
x = dataset_url["url"]
```

```
# Labels
y = dataset_url["result"]
```

```
x.head()
```

```
0        https://www.eltiempo.com
1    https://www.elespectador.com
2          https://www.semana.com
3       https://www.caracoltv.com
4        https://www.rcnradio.com
Name: url, dtype: object
```

```
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: result, dtype: int64
```

Figura 42. Extracción de features y etiquetas. Fuente: este trabajo.

Se procede a crear un vectorizador de texto a través de TfidfVectorizer() Esta herramienta convierte el texto en números, se crea una instancia del vectorizador y se guarda en vectorizer, se entrena el vectorizador con los datos de (x) la cual se transforma en una matriz dispersa de características numéricas. Se procede a dividir los datos de entrenamiento y de prueba a través de la función train_test_split de sklearn.model_selection, lo cual permite que las características del parámetro (x) y las etiquetas del parámetro (y) se dividan en un 20% para pruebas y el porcentaje restante que es el 80% para entrenamiento usando test_size=0.2. Después se usa unas verificaciones de dimensiones en los conjuntos de entrenamiento y prueba por lo cual debe coincidir en cada conjunto, para nuestro dataframe se puede observar en la Figura 39 que coinciden los datos numéricos `(38820, 38820, 9706, 9706).`

```
# Uso de vectorizador
vectorizer = TfidfVectorizer()
```

```
# Store vectors into X variable as Our XFeatures
X = vectorizer.fit_transform(x)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X_train.shape[0],len(y_train),X_test.shape[0],len(y_test)
```

```
(38820, 38820, 9706, 9706)
```

Figura 43. Vectorización y División de entrenamiento y Pruebas. Fuente: este trabajo.

Una vez se haya vectorizado y dividido los datos para iniciar el entrenamiento se procede a crear el modelo tipo Random Forest el cual viene de sklearn.ensemble, este modelo realiza ensamble de árboles de decisión el cual construye y toma una decisión basada en la mayoría por voto a clasificar, esto hace que se reduzca el sobreajuste (overfitting). Se entrena el modelo usando los datos de entrenamiento X_train el cual contiene las features que son las URLs vectorizadas y y_train el cual contiene las etiquetas Benign:0 – Malicious:1.

Se calcula la precisión (Accuracy) a través de random.score(X_train, y_train) para determinar cuántas predicciones se hizo de manera correcta en promedio, el cual muestra el resultado como porcentaje con 2 decimales a través de:.2%. Este mismo procedimiento de calcular la precisión ahora se va a mostrar sobre datos que no no vista antes el modelo los cuales con los de prueba (X_test, y_test)

```
random = RandomForestClassifier()
random.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
# Accuracy of Our Model Using Test Data
print(f"RandomForestClassifier Training Accuracy: {random.score(X_train, y_train):.2%}")
```

```
RandomForestClassifier Training Accuracy: 100.00%
```

```
# Accuracy of Our Model Using Test Data
print(f"RandomForestClassifier Accuracy: {random.score(X_test, y_test):.2%}")
```

```
RandomForestClassifier Accuracy: 99.69%
```

Figura 44. Cálculo de Accuracy en (X_train, y_train X_test, y_test). Fuente: este trabajo.

Para poder visualizar la matriz de confusión se va a usar el modelo ya entrenado (model) para poder hacer predicciones sobre los datos X_test, esta función devuelve un array con las líneas etiquetadas. Y_pred guarda las predicciones del modelo las cuales posteriormente se

comparan con y_test y así poder evaluar el rendimiento. El resultado para el modelo Random Forest se observó que 4.943 URLs "Benign" son verdaderos positivos, 24 URLs son falsos positivos (URLs "Benign" que RF clasifico como "Malicious"), 4.733 URLs "Malicious" como verdadero negativo y 6 URLs como falso negativo (URLs "Malicious" que RF clasificó como "Benign"). Según se ejemplifica en la Figura 45.

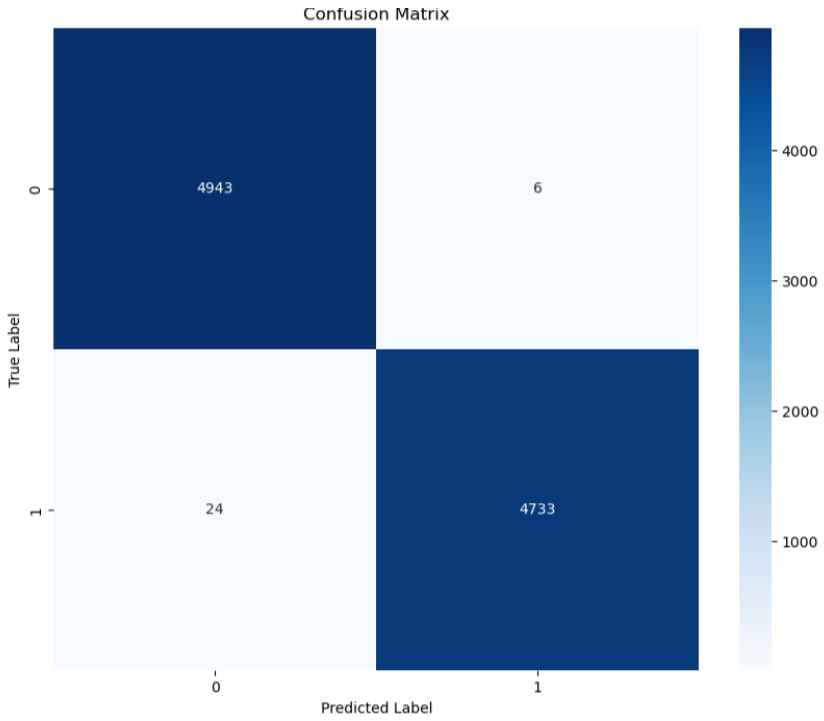


Figura 45. Matriz Confusión RF Fuente: este trabajo.

Validación Cruzada.

Para el caso de Random Forest se divide y se ejecuta 5 folds, los cuales usa 4 folds para entrenar y 1 fold para probar en cada vuelta. Cross_validate en estado true calcual y devuelve las puntuaciones del entrenamiento por cada fold, este punto es importante ya que permite detectar overfitting, si el score de train es mucho mayor que el test. Posteriormente guarda los modelos entrenados en cada fold lo cual queda todo este proceso guardado en la variable cv_results.

```
# Validación cruzada de 5 folds (es el valor por defecto)
#cv_results = cross_validate(model, X, y, cv=5, scoring=scoring)
cv_results = cross_validate(model, X, y, cv=5, scoring=scoring, return_train_score=True, return_estimator=True)
```

Figura 46. Instrucción Scikit-learn, Entrenamiento y Calculo Métrica. Fuente: este trabajo.

Una vez realizado el proceso de validación cruzada se obtiene que todas las métricas de validación son muy altas, alrededor del 99% lo cual significa que el modelo es muy preciso y generaliza de manera correcta los datos que no ha visto antes que son los de prueba. En cuanto al entrenamiento, el modelo aprendió perfectamente los datos train lo cual tiene un valor del 100%, para la validación, el modelo no memorizo de manera perfecta el entrenamiento, pero si aprendió patrones generales. Como resultado se obtiene una brecha mínima del 1%. Conforme a los resultados obtenidos se concluye que el modelo no tiene overfitting, esto hace que el modelo pueda hacer buenas predicciones sobre datos nuevos.

```
=== DETECCIÓN DE OVERFITTING ===
Precisión - Entrenamiento: 1.0000 | Validación: 0.9894
Recall - Entrenamiento: 1.0000 | Validación: 0.9858
F1-Score - Entrenamiento: 1.0000 | Validación: 0.9875
Accuracy - Entrenamiento: 1.0000 | Validación: 0.9879

Brechas (Train - Test):
Precisión: 0.0106
Recall: 0.0142
F1-Score: 0.0125

 El modelo OK
```

Figura 47. Resultado Modelo Detección Overfitting. Fuente: este trabajo.

Recall.

El modelo es capaz de identificar el 99% de todas las URLs etiquetadas como “Malicious” esto significa que el numero de falsos negativos es muy bajo (1%) Un recall alto y balanceado indica que el modelo no presenta un sesgo alarmante hacia cualquier clase. El recall se debe analizar con la precisión que igualmente es alta 99% ya que, no solo está encontrando casi todas las URLs “Malicious” (Recall Alto) si no que también alerta sobre cualquier amenaza y el 99% concuerda (Precisión Alta).

```
=== REPORTE DE CLASIFICACIÓN ===
              precision    recall  f1-score   support

      Benign       0.99      0.99      0.99     25024
   Malicious       0.99      0.99      0.99     23502

    accuracy                           0.99     48526
   macro avg       0.99      0.99      0.99     48526
weighted avg       0.99      0.99      0.99     48526
```

Figura 48. Resultados Precisión-Recall. Fuente: este trabajo.

En el resultado para el modelo Random Forest con validación cruzada, se observó que 24.770 URLs "Benign" son verdaderos positivos, 334 URLs son falsos positivos (URLs "Benign" que RF clasifico como "Malicious"), 23.168 URLs "Malicious" como verdadero negativo y 254 URLs como falso negativo (URLs "Malicious" que RF clasificó como "Benign"). Según se ejemplifica en la Figura.

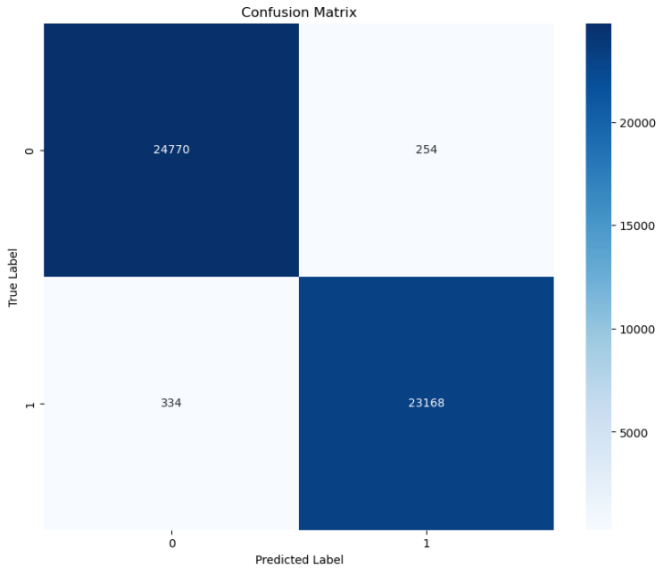


Figura 49. Matriz Confusión RF-Validación Cruzada Fuente: este trabajo.

4.6 Deep Neural Network

El último modelo propuesto se basó en redes neuronales profundas, el cual usa varias capas ocultas con miles de neuronas artificiales conectadas entre sí, este modelo puede manejar gran cantidad de datos con precisión y en menor tiempo que los demás modelos, para el caso, se emplea el modelo Sequential de Keras, útil para secuenciar una capa de la anterior, lo cual simplifica la construcción de dicha red, del mismo modo, se empleó las funciones Dense la cual sirve para conectar cada neurona de la capa actual con todas las neuronas de la siguiente. Por otro lado, se utilizó Dropout para evitar el sobreajuste en el entrenamiento, además, BatchNormalization para la salida y mejoramiento de una capa en cuanto a su estabilidad y velocidad, por consiguiente, se encuentra Earlystopping, una función que detiene el entrenamiento si no ve una mejora en el mismo durante cierto número de épocas. Por último, se utilizó la función Adam, la cual es un optimizador que mejora el entrenamiento de la red y ajusta la tasa de aprendizaje de cada parámetro de manera eficiente. Tal como se aprecia en la Figura 50.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
```

Figura 50. Librerías y módulos DNN. Fuente: este trabajo.

Se crea el siguiente diccionario denominado binary_class_dictionary, el cual se usó para asignar etiquetas “benign” y “malicious”. Dichas etiquetas asignaron un cierto valor que en este caso fue 0 y 1 respectivamente:

```
binary_class_dictionary = {
    'benign': 0,
    'malicious': 1,
}
```

Se procede a utilizar funciones como Apply que es útil para aplicar a cada elemento las funciones internas que se quiere aplicar, en este caso es Lambda, la cual transforma los valores de la columna type del dataframe dataset_url y verifica si x es igual a “Benign” el valor se va a mantener “Benign” de lo contrario se cambiará a “malicious” después se imprime el conteo de los valores type luego de que se haya aplicado la transformación la cual permite ver cuántos elementos estan categorizados como “Benign” y cuantos están etiquetados como “malicious”. Ver Figura 51.

```
# Modify the 'type' column to contain 'benign' or 'hostile'
dataset_url['type'] = dataset_url['type'].apply(lambda x: 'benign' if x == 'benign' else 'malicious')

# Verify the changes
print(dataset_url['type'].value_counts())
print(dataset_url.head())
```

```
type
benign       24999
malicious    23527
Name: count, dtype: int64
                          url    type  result  url_length  hostname_length  \
0      https://www.eltiempo.com  benign       0          24               16
1  https://www.elespectador.com  benign       0          28               20
2        https://www.semana.com  benign       0          22               14
3     https://www.caracoltv.com  benign       0          25               17
4      https://www.rcnradio.com  benign       0          24               16
```

Figura 51. Aplicación de funciones Apply y Lambda. Fuente: este trabajo.

A continuación, el siguiente código convierte y organiza los datos del dataframe dataset_url para que sean más fáciles de procesar en el modelamiento DNN. Para ello se selecciona la columna ”type” de dataset_url para que la función map que contiene el diccionario binary_class_dictionary donde “Benign” es 0 y “Malicious” es 1, pueda transformar cada valor de “type” en un valor numérico y este resultado se convierta por la función values, en un arreglo de valores en donde se va a almacenar en y. Después se crea la lista llamada “urls” la cual contiene todas las direcciones de dataset_url y se crea una lista separada de URLs que sirve de referencia para poder usarse en funciones posteriores. Se elimina las columnas type y url del dataframe porque después de mapear las etiquetas y extraer las URLs no es necesario las columnas restantes, solo se va a usar las características para el modelo. Por último, se crea la matriz “x” donde se va a almacenar las características sin las columnas “type” y “url”. Lo anterior se puede apreciar en la Figura 52.

```
y = dataset_url['type'].map(binary_class_dictionary).values
# Convert to a numpy array
#y = np.array(labels)
urls = [url for url in dataset_url['url']]

# Now i can drop url and type
dataset_url = dataset_url.drop(columns=['type', 'url'])

# Create x matrix
x = dataset_url.values
```

Figura 52. Arreglo de etiquetas, lista y matriz DNN. Fuente: este trabajo.

El siguiente código realiza la división de datos mediante 3 conjuntos que son: entrenamiento, validación y prueba, mediante la función train_test_split de Scikit-Learn. Primero se divide los datos en 2 subconjuntos, train servirá para el entrenamiento del modelo y Temp será subdividido con 2 subconjuntos “test” y “val”. Después se ingresa los datos de entrada que son las características de la matriz “x” el conjunto de etiquetas “y” y las URLs. Se define por medio de test_size=0.3 que el 30% de los datos será para el subconjunto “temp” y el 70% para el conjunto “train”. Finalmente, se divide el subconjunto “temp” de igual manera para los conjuntos por medio de la función test_size=0.5 “test” 50% y para “val” 50%. Por consiguiente, la división general queda distribuida de la siguiente manera: Ver también la Figura 53.

- **70%** en el conjunto de entrenamiento (x_train, y_train, url_train).
- **15%** en el conjunto de validación (x_val, y_val, url_val).
- **15%** en el conjunto de prueba (x_test, y_test, url_test).

```
# First, split the data into training and a combined test/validation set (temporary dataset)
x_train, x_temp, y_train, y_temp, url_train, url_temp = train_test_split(
    x, y, urls, test_size=0.3)

# Then, split the temporary set into separate test and validation sets
x_test, x_val, y_test, y_val, url_test, url_val = train_test_split(
    x_temp, y_temp, url_temp, test_size=0.5)
```

Figura 53. División de conjuntos de entrenamiento, prueba y validación. Fuente este trabajo.

En el siguiente fragmento de código se usa la función “to_categorical” la cual convierte las etiquetas de las clases y_train, y_val, y_test en el formato denominado one-hot encoding el cual convierte las clases en vector binario por medio de “num_classes=2” encargado de especificar que hay 2 clases “Benign” y “Hostile” por último transforma una matriz donde cada etiqueta se representa por medio de un vector binario. Por ejemplo, si la etiqueta original es “0 Benign” se convierte en [1, 0] Y si la etiqueta original es “1 Hostile” se convierte en [0, 1]. De la siguiente manera:

y_train = to_categorical(y_train, num_classes=2)
y_val = to_categorical(y_val, num_classes=2)
y_test = to_categorical(y_test, num_classes=2)

A continuación, se explica el proceso del modelamiento DNN que ejecuta la API Keras a través del modelo Sequential, este se compone de capas densas, normalización, regularización y optimización. “Sequential” es el modelo de keras en donde su función es apilar las capas una tras otra, conectando así la capa de la lista con la siguiente capa. La primera capa Dense está compuesta por 256 neuronas que toman como entrada, el número de características de “x_train”, después se emplea la activación relu para que el modelo pueda aprender representaciones no lineales. Luego, se regulariza por medio de la función “tf.keras.regularizers.l2” para reducir el sobreajuste, se normaliza las activaciones de capa anterior por la función “BatchNormalization” lo cual ayuda a acelerar el aprendizaje. Por último, se apaga aleatoriamente el 50% de las neuronas para evitar el sobreajuste a través de la función “Dropout(0.5)” este mismo procedimiento de aprendizaje profundo se repite para las siguientes 3 capas ocultas.

Se genera la última capa la cual contiene 2 neuronas, una para cada clase “Benign” y “Hostile” Donde se usa la activación Softmax que convierte las salidas en probabilidades, en este caso

es útil para tareas de clasificación binaria la cual se ejecuta en formato “one-hot-encoding”. Para finalizar, el modelo usa el optimizador Adam para una tasa de aprendizaje de 0.001 que ajusta el aprendizaje para cada parámetro en función de sus gradientes, como se puede apreciar en la Figura 54.

```
model = Sequential([
    Dense(256, input_shape=(x_train.shape[1],), activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    BatchNormalization(),
    Dropout(0.5),
    Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    BatchNormalization(),
    Dropout(0.5),
    Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    BatchNormalization(),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    BatchNormalization(),
    Dropout(0.5),
    Dense(2, activation='softmax')
])

optimizer = Adam(learning_rate=0.001)
```

Figura 54. DNN Proceso Normalización, regulación y optimización. Fuente este trabajo.

Se procede a compilar modelo Keras especificando el optimizador que usa el método Adam, la función de perdida a través de la función categorical_crossentropy mide la diferencia entre las distribuciones de probabilidad predicha y la función de probabilidad real, y las métricas Accuracy la cual mide el porcentaje de predicciones correctas del modelo, informa cuantas veces se predice el modelo de manera correcta a través de las épocas que se implementen en él. También se usan callbacks de keras “EarlyStopping” que evitan que el modelo entrene demasiado tiempo y sobreajuste los datos de entrenamiento y “ReduceLROnPlateau” que ayuda a mejorar el entrenamiento a pesar de que el modelo parezca estancarse. Reducir la tasa de aprendizaje ayuda al modelo a largo plazo lo cual alcanza más precisión en los resultados. Ver Figura *55*.

```
# Compile the model
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Define callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)
```

Figura 55. Uso de Callbacks Keras. Fuente: este trabajo.

Se procede a entrenar x_train, y_train para el aprendizaje de patrones, se especifica el número de épocas que es de 100 donde cada época es un ciclo completo de todos los datos de

entrenamiento, se implementa “EarlyStopping” en caso de que el modelo deba detenerse si en el rendimiento no se ve mejoría. Después se usa el parámetro “validation_data=(x_val, y_val)” que el modelo va a usar los datos de validación luego de cada época, con este parámetro se evita el sobreajuste ya que monitorea el rendimiento del modelo. Como se puede apreciar una vez entrenado el modelo propuesto se obtiene un accuracy de 99%. Según la *Figura 56*.

```
# Train the model
history = model.fit(x_train, y_train, epochs=100, batch_size=64, validation_data=(x_val, y_val), callbacks=[early_stopping, reduce_lr])

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_accuracy}')

model.summary()
```

```
Epoch 1/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.8854 - loss: 0.6609 - val_accuracy: 0.9816 - val_loss: 0.3518 - learning_rate: 0.0010
Epoch 2/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9710 - loss: 0.3586 - val_accuracy: 0.9861 - val_loss: 0.2497 - learning_rate: 0.0010
Epoch 3/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9790 - loss: 0.2551 - val_accuracy: 0.9850 - val_loss: 0.1853 - learning_rate: 0.0010
Epoch 4/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9790 - loss: 0.1975 - val_accuracy: 0.9929 - val_loss: 0.1240 - learning_rate: 0.0010
Epoch 5/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9841 - loss: 0.1435 - val_accuracy: 0.9912 - val_loss: 0.0937 - learning_rate: 0.0010
Epoch 6/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9863 - loss: 0.1123 - val_accuracy: 0.9986 - val_loss: 0.0629 - learning_rate: 0.0010
Epoch 7/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9905 - loss: 0.0824 - val_accuracy: 0.9989 - val_loss: 0.0557 - learning_rate: 0.0010
Epoch 8/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9885 - loss: 0.0830 - val_accuracy: 0.9754 - val_loss: 0.0993 - learning_rate: 0.0010
Epoch 9/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9902 - loss: 0.0685 - val_accuracy: 0.9992 - val_loss: 0.0401 - learning_rate: 0.0010
Epoch 10/100
531/531 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9947 - loss: 0.0552 - val_accuracy: 0.9992 - val_loss: 0.0370 - learning_rate: 0.0010
```

Figura 56. Entrenamiento del modelo mediante 100 Épocas. Fuente: este trabajo.

Se presenta la matriz de confusión, Se entrena el 70% de URLs y se pone a prueba el porcentaje restante que es el 30% Según en entrenamiento por el método DNN se observa que 3698 URLs “Benign” son verdaderos positivos, 2 URLs son falsos positivos (URLs “Benign” que RF clasificó como “Hostile”), 3579 URLs “Hostile” como verdadero negativo y 0 URLs como falso negativo (URLs “Hostile” que DNN clasificó como “Benign”). Según la Figura *57*.

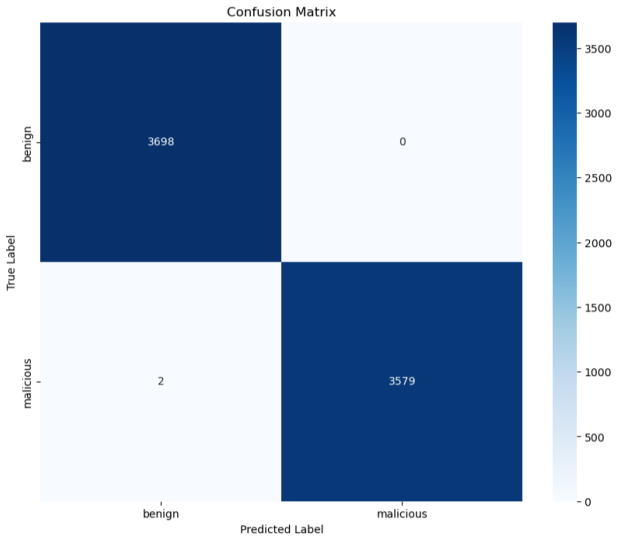


Figura 57. Matriz de Confusión DNN. Fuente: este trabajo.

Validación Cruzada.

Para el entrenamiento de validación cruzada con el modelo DNN se utiliza StratifiedKfold para mantener la proporción de clases en cada fold, así se asegura que cada subconjunto sea representativo del dataset. Se entrena un modelo desde 0 en cada Fold, así evalua el rendimiento real del modelo en diferentes particiones. Se utiliza las métricas de Loss, brecha entre train_loss y val_loss, si este es mayor hay overfitting. Se utiliza Precisión que es la diferencia entre train_acc y val_acc, si este ultimo es mayor, hay overfitting. Early Stopping detiene el entrenamiento si val_loss no mejora, lo cual podría evitar sobreajuste. ReduceLROnPlateau el cual reduce el learning rate si val:los se queda, lo cual mejora la convergencia. Por ultimo se saca un promedio entre todos los Folds para dar una medida robusta y generalizable.

```
# 📈 Capturar métricas para detección de overfitting
train_loss = history.history['loss'][-1]  # Última pérdida de entrenamiento
val_loss = history.history['val_loss'][-1]  # Última pérdida de validación
train_acc = history.history['accuracy'][-1] if 'accuracy' in history.history else history.history['acc'][-1]
val_acc = history.history['val_accuracy'][-1] if 'val_accuracy' in history.history else history.history['val_acc'][-1]

train_losses.append(train_loss)
val_losses.append(val_loss)
train_accuracies.append(train_acc)
val_accuracies.append(val_acc)

# Evaluar modelo
y_pred_probs = model.predict(X_val)
y_pred_classes = np.argmax(y_pred_probs, axis=1)  # Clase predicha
y_true_classes = np.argmax(y_val, axis=1)         # Clase verdadera

all_true.extend(y_true_classes)
all_pred.extend(y_pred_classes)

f1 = f1_score(y_true_classes, y_pred_classes, average='weighted')
f1_scores.append(f1)

print(f"F1-Score (Fold {fold_no}): {f1:.4f}")
print(f"Train Loss: {train_loss:.4f}, Val Loss: {val_loss:.4f}")
print(f"Train Acc: {train_acc:.4f}, Val Acc: {val_acc:.4f}")
```

Figura 58. Parte Codigo DNN-Validación Cruzada. Fuente: este trabajo.

Tras los resultados obtenidos mediante la validación cruzada en DNN, se tiene un rendimiento excelente con un F1-Score muy alto del 0.9981 en validación, un accuracy del 99.8% en validación. Presenta brechas mínimas con diferencia de perdida del 0.0099 la cual es muy pequeña y una diferencia de accuracy de 0.0015 lo cual es muy pequeña. En cuanto a la generalización se observa que es optima, el modelo funciona igual en entrenamiento y en validación, por tal motivo no se observa sobreajuste, aprende patrones generalizables.

```
F1-Score (Fold 5): 0.9981
Train Loss: 0.0147, Val Loss: 0.0199
Train Acc: 0.9988, Val Acc: 0.9981

============================================================
📊 ANÁLISIS DE OVERFITTING
============================================================
Pérdida promedio - Entrenamiento: 0.0115, Validación: 0.0205
Accuracy promedio - Entrenamiento: 0.9994, Validación: 0.9978

🔍 Brechas:
Pérdida (Val - Train): 0.0090
Accuracy (Train - Val): 0.0015
✅ El modelo generaliza bien
```

Figura 59. Resultado Overfitting DNN. Fuente: este trabajo.

Recall.

Para el modelo DNN con validación cruzada se presenta un recall perfecto del 100% El modelo identifico correctamente el 100% de las URLs maliciosas reales, 0 falsos negativos, capturo todas las URLs benignas reales 0 falsos positivos, lo cual garantiza que ningún contenido malicioso paso por alto, el modelo no tiene sesgos y generaliza de manera óptima, combina una precisión del 100% lo cual indica que el modelo es confiable. Este resultado hace que el modelo sea inusualmente alto, lo mas probable es que los patrones de URLs son muy distintos entre clases, como resultado el modelo aprendió características generalizables sin overfitting.

```
============================================================
📊 EVALUACIÓN GLOBAL - TODAS LAS 48,526 URLs
============================================================
              precision    recall  f1-score   support

      Benign       1.00      1.00      1.00     24999
   Malicious       1.00      1.00      1.00     23527

    accuracy                           1.00     48526
   macro avg       1.00      1.00      1.00     48526
weighted avg       1.00      1.00      1.00     48526

F1-Score Global: 0.9979
```

Figura 60. Tabla Clasificación DNN. Fuente: este trabajo.

La matriz de confusión para el modelo DNN con validación cruzada, se observó que 24.942 URLs “Benign” son verdaderos positivos, 44 URLs son falsos positivos (URLs “Benign” que RF clasifico como “Malicious”), 23.483 URLs “Malicious” como verdadero negativo y 57 URLs como falso negativo (URLs “Malicious” que RF clasificó como “Benign”). Según se ejemplifica en la Figura.

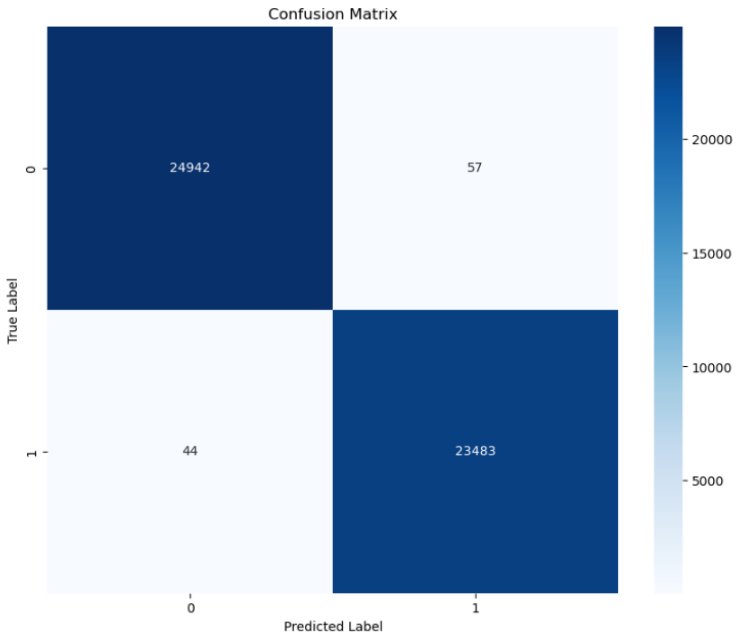


Figura 61. Matriz de Confusión Validación Cruzada DNN. Fuente: este trabajo.

Discrepancia en el reporte de Clasificación Vs Matriz de Confusión. Para DNN

El reporte de clasificación redondea las métricas a 2 decimales por defecto, por tal motivo, un recall de 1.00 no significa estrictamente 100%, pero si esta muy cerca, aprox 0.995. La matriz de confusión muestra los valores brutos y exactos, sin decimales. Por tal motivo para la clase "Malicious"se tiene:

$\text{Verdadero Negativo(TP)} = 23.483$

$Falso\ Negativo(FN) = 57$

$$Recall = \frac{TP}{(TP + FN)} = \frac{23483}{(23483 + 57)} = 0.9976(Redondeado\ a\ 1.00\ en\ el\ reporte)$$

Para la clase "Benign" se tiene:

$\text{Verdadero Positivo(TP)} = 24.942$

$Falso\ Positivo(FN) = 44$

$$Recall = \frac{24.942}{(24.942 + 44)} = 0.9982(Redondeado\ a\ 1.00\ en\ el\ reporte)$$

En conclusión, el recall no es del 100%, pero si es extremadamente alto, la matriz de confusión muestra los datos exactos, por lo cual, haya falsos negativos y falsos positivos en cantidades muy pequeñas comparadas con el dataset, por tal motivo, el modelo brinda excelentes resultados, pero no son perfectos.

Una vez realizado el entrenamiento en los 3 métodos ML como KNN, RF y Deep NN los cuales son los más usados en cuanto a clasificación, se debe seleccionar el modelo que más se ajuste al objetivo general el cual es bloquear las URLs de pornografía infantil etiquetadas como “Malicious”. Así poder implementar dicho código en el programa Visual Studio y ejecutarlo en un entorno web bajo el framework Django, estas 2 herramientas permitirán implementar la parte visible de la aplicación denominada Frontend, así poder interactuar con el usuario y realizar pruebas.

El modelo seleccionado para la implementación es el Random Forest ya que ofrece mayor ventaja de aplicación en este proyecto, el análisis de los resultados detallados de esta selección se explicará en el numeral 4.8.1 Resultados del Entrenamiento. Se guarda el modelo Random Forest y la Vectorización bajo la librería joblib usada para guardar y cargar los modelos ML, así poder desplegar el modelo en el Frontend los cuales se guardan bajo el formato. pkl. Una vez guardado el entrenamiento de RF se procede a realizar pruebas con diferentes URLs del dataframe lo cual predice de manera correcta:

```
model = joblib.load(r"./RandomForestModel2.pkl")
vec = joblib.load(r"./TfidfVectorizer2.pkl")

labels = ["BENIGN","MALICIOUS"]

test1 = model.predict(vec.transform(["https://www.eltiempo.com"]))

labels[test1[0]]

'BENIGN'

test2 = model.predict(vec.transform(["http://thumbnails110.imagebam.com/37243/192c45372425520.jpg"]))

labels[test2[0]]

'MALICIOUS'

test3 = model.predict(vec.transform(["edu.gov.on.ca/eng/sift/indexSec.asp"]))
labels[test3[0]]

'BENIGN'
```

Figura 62. Cargue y Pruebas de Predicción de URLs. Fuente este trabajo.

4.7 Despliegue Aplicación WEB con Visual Studio.

4.7.1 Index.html

Para este trabajo se crea el proyecto llamado “myproject” el cual se ejecuta el index.html. Se inicia con la etiqueta <body> en el cual va todo el contenido visible de la página, aqui se incluye textos, imágenes, formularios, botones, etc. Después viene el contenedor usado para agrupar elementos HTML como en este caso class=”container” el cual le da la apariencia visual CSS, se implementa la imagen de mintic.png con estilo centralizado. Después se usa encabezado nivel 1 mostrando el título “DETECTOR URL” seguido del código para la creación del formulario. En esta sección se asigna un único ID para dicho formulario, lo cual es más accesible a la hora de manipularlo con JavaScript, se define etiqueta asociada al campo de entrada que es el ID “url”. Se genera un input tipo texto para el ingreso de la URL la cual se va a validar por medio de id=”url” y name=”url”, el required hace que este campo sea obligatorio seguido de un etilo de línea como el ancho del contenedor, espacio del campo, el borde, etc. Button type entra a validar la URL agregada en el campo lo cual se enviará al formulario y por último el div id=”result” en donde se mostrara los resultados de manera dinámica.

```
</head>
<body>
    <div class="container">
        <div style="text-align: center;">
            <img src="{% static 'img/mintic1.png' %}" alt="Imagen Mintic">
        </div>
        <h1>DETECTOR URL</h1>
        <form id="urlForm">
            <label for="url">Ingrese la URL a validar:</label>
            <input type="text" id="url" name="url" placeholder="https://example.com" required style="width: 100%; padding: 10px; margin: 10px 0; border: 1px solid #c
            <button type="submit"  style="padding: 10px 20px; background-color: #007bff; color: white; border: none; border-radius: 5px; cursor: pointer;">
                Validar URL
            </button>
        </form>
        <div id="result" class="result"></div>
    </div>
```

Figura 63. Diseño encabezado y formulario en Visual Studio. Fuente este trabajo.

4.7.2 Blocked.html

Se crea una segunda página la cual realizará el bloqueo de las páginas catalogadas como maliciosas, la cual consiste en un contenedor <body> principal visible en el navegador, se maneja otro contenedor principal con base blocked-container el cual le da forma centralizada, colores de advertencia para establecer el diseño de la página, se adjunta ícono SVG(gráfico vectorial escalable) el cual representa un candado cerrado lo cual indica que el acceso está restringido. Se muestra un mensaje de advertencia <h1>Acceso Bloqueado<h1> Seguido de 2 párrafos explicando por qué no es posible el ingreso a esta página. Se inserta un botón (enlace) el cual redirige al usuario de regreso a la página principal y por último un mensaje de soporte indicando una vía de solución en caso de que el bloqueo haya sido incorrecto.

```
<body>
    <div class="blocked-container">
        <div class="icon">
            <svg xmlns="http://www.w3.org/2000/svg" width="64" height="64" fill="currentColor" viewBox="0 0 16 16">
                <path d="M8 1a2 2 0 0 1 2 2v4H6V3a2 2 0 0 1 2-2zm3 6V3a3 3 0 0 0-6 0v4a2 2 0 0 0-2 2v5a2 2 0 0 0 2 2h
            </svg>
        </div>
        <h1>Acceso Bloqueado</h1>
        <p>La página que intentas visitar ha sido identificada como <strong>MALICIOSA</strong> por nuestro sistema de
        <p>Por tu protección, el acceso ha sido restringido.</p>
        <a href="{% url 'home' %}" class="btn">Volver al Inicio</a>
        <div class="footer">
            <p>Si crees que esto es un error, por favor contacta al administrador.</p>
        </div>
    </div>
</body>
</html>
```

Figura 64. Diseño página de bloqueo URL Malicious en Visual Studio. Fuente este trabajo.

Una vez se tiene listo el diseño de la página principal se debe implementar varias extensiones para poder llamar el código escrito por ML a través del método RF como views.py, manage.py, urls.py, dichas extensiones se crean de forma automática una vez se instala el framework Django, para el proyecto propuesto se va a realizar una breve descripción de las funcionalidades de cada extensión.

4.7.3 Views.py

Inicialmente se define la función llamada get_prediction_from_url para poder analizar el parámetro test_url, se procede a generar un try el cual “intenta” ejecutar el código descrito en su campo, si por algún motivo esta falla, saltará a except para poder manejar errores. Se carga modelo ML previamente entrenado en backend via RF el cual se guardó por la extensión.pkl. Se procede a cargar el vectorizador TF-IDF del archivo TfidVectorizer el cual convierte el texto de las URLs en números. Se transforma la URL de entrada (test_url) usando el vectorizador que fue cargada previamente, transform necesita la lista para su evaluación y retorna un resultado (url_vector) el cual está listo para ser evaluado por el modelo. Después el modelo predice si la URL es Malicious o Benign usando el vector generado. Se interpreta la predicción en pred[0] donde 0 corresponde a “BENIGN” y 1 a “MALICIOUS” donde devuelve el resultado final. En caso de que ocurra un error lo reconoce except e imprime el mensaje de error y devuelve un none para informar que algo no salió bien en el proceso.

```
def get_prediction_from_url(test_url):
    try:
        model = joblib.load(str(settings.BASE_DIR)+'/static/RandomForestModel2.pkl')
        vectorizer = joblib.load(str(settings.BASE_DIR)+'/static/TfidfVectorizer2.pkl')  # Este es el HashingVectorizer

        #test_url = "corporationwiki.com/Ohio/Columbus/frank-s-benson-P3333917.aspx"
        # Transformar la URL usando el vectorizer
        url_vector = vectorizer.transform([test_url])  # Nota: transform espera una lista de URLs
        pred = model.predict(url_vector)
        labels = ["BENIGN", "MALICIOUS"]
        status = labels[pred[0]]
        return status
    except:
        print('Error get_prediction_from_url() ')
        return None
```

Figura 65. Función get_prediction_from_url en Visual Studio. Fuente este trabajo.

La segunda función llamada classifyurl define una función Django, la cual recibe una URL por medio de una solicitud HTTP obteniendo un parámetro url, se llama a la función get_prediction_from_url la cual clasifica como “BENIGN” o “MALICIOUS” Dependiendo de su resultado la bloquea y muestra la página blocked.html En caso contrario asegura que se tenga un esquema http:// ó https:// en caso de que no lo tenga, agrega el https:// y la redirige. En caso de que no sea válida la página, responde con un error 400 y un mensaje JSON.

```python
def classifyurl(request):
    try:
        url = request.GET.get('url'," ")
        #vectorizer = CountVectorizer()
        #url_vector = vectorizer.fit_transform([url])
        resultado=get_prediction_from_url(url)

        if resultado == "MALICIOUS":
            return render(request, 'pages/blocked.html', status=403)
        else:
            #Verificar si la URL es válida y agregar https:// si no tiene esquema
            if not url.startswith(('http://', 'https://')):
                url = 'https://' + url

            if is_valid_url(url):
                return redirect(url)  # Redirige a la URL benigna
            else:
                return JsonResponse({'error': 'URL no válida'}, status=400)
    except Exception as e:
        context= {
            'resultado': f'Error en classify_url(): {str(e)}',
        }
        return JsonResponse(context, status=500)
```

Figura 66. Función classifyurl en Visual Studio. Fuente este trabajo.

La tercera función en views.py se define como home la cual recibe una solicitud HTTP (request) Se verifica si la solicitud es de tipo GET en donde el usuario envía parámetros de la URL o entra a cierta página. Se obtiene el parámetro url el cual viene de la solicitud y en caso si no se envía se queda con la cadena vacía “ ”. Posteriormente se inserta una condicional IF en donde si no hay URL enviada o si esta vacía, se carga la página principal la cual es index.html, en caso que si haya una URL enviada llama a get_prediction:from_url(url) para poder obtener la URL ya sea BENIGN o MALICIOUS. Si es maliciosa muestra la página blocked.html, si es benigna se asegura que tenga el esquema http:// ó https://, en caso de que no la tenga, agrega https:// SI la página es invalida devuelve un JSON de error 400.

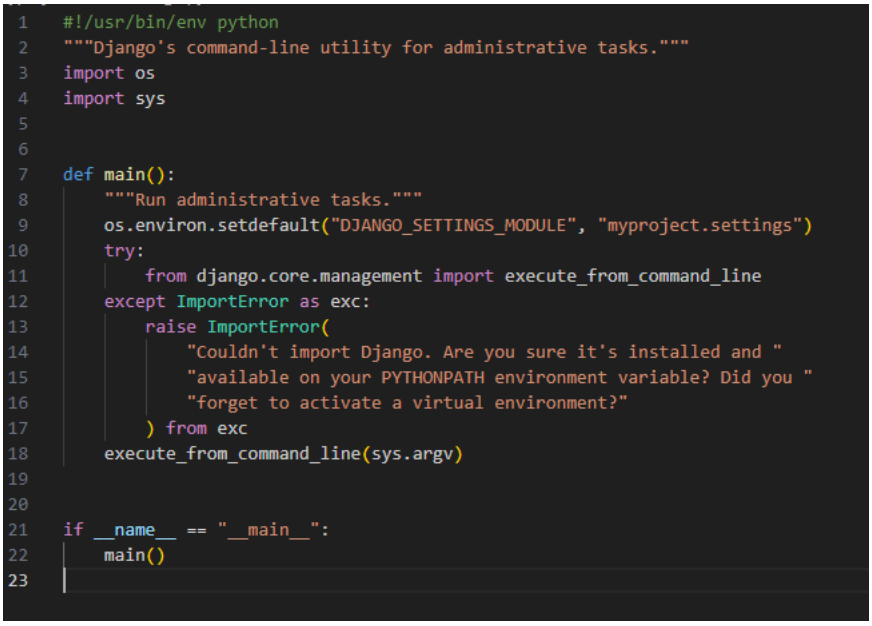
```python
def home(request):
    if(request.method=='GET'):
        url = request.GET.get('url',"")
        if url=="":
            return render(request, 'pages/index.html')
        resultado = get_prediction_from_url(url)

        if resultado == "MALICIOUS":
            return render(request, 'pages/blocked.html', status=403)
        elif resultado == "BENIGN":
            # Asegurar que la URL tenga http:// o https://
            if not url.startswith(('http://', 'https://')):
                url = 'https://' + url

            if is_valid_url(url):
                return redirect(url)  # Redirige a la página segura
            else:
                return JsonResponse({'error': 'URL no válida'}, status=400)
        else:
            return JsonResponse({'error': 'Error en la clasificación'}, status=500)
    return render(request, 'pages/index.html')
```

Figura 67. Función home en Visual Studio. Fuente este trabajo.

4.7.4 Manage.py

Extensión la cual juega un papel importante en la sincronización con el framework Django, ya que nos permite visualizar en un entorno web las funcionales de la página diseñada para tal fin, a continuación, se explica el código:

Se defina la función main la cual ejecutara las tareas administrativas, se configura la variable de entorno llamada DJANGO_SETTINGS_MODULE en donde le dice al framework en donde encontrar la configuración del proyecto que en este caso es (`myproject/settings.py`) Se intenta importar la función execute_from_command_line en donde interpreta los comando usando en consola, en este caso se usa Anaconda Prompt y reconocera ciertos tipos de comandos como runserver, migrate y createsuperuser. En caso de que Django no este instalado o no se encuentre su ubicación, esta captura el error y envía el mensaje. Después se ejecuta Djando por execute_from_command_line pasando todos los argumentos a la consola lo cual es útil para lanzar comandos como por ejemplo python manage.py runserver. Django captura la línea para su ejecución.



```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "myproject.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == "__main__":
    main()
```

Figura 68. Función main en manage.py en Visual Studio. Fuente este trabajo.

4.7.5 urls.py

Esta extensión define la configuración de las rutas (URLs) del framework Django en donde se importan los módulos admin lo cual permite usar la interfaz de administrador, se importa la

función patch para poder definir las rutas de las URLs e importa las funciones views desde la aplicación home en donde también está definida la función classifyurl. Se procede a crear la lista urlpatterns que es donde va a contener todas las rutas que Djando va a usar. En la lista están las rutas classify-url - admin – home.

Ruta Classify-url: El usuario accede a esta ruta y llama la función classify-url la cual se definió en views.py.

Ruta Admin: Abre el panel administrativo de Django donde requiere usuario y contraseña de superusuario.

Ruta Home: Es cuando el usuario accede a home y muestra o evalúa la URL (GET).

```python
from django.contrib import admin
from django.urls import path
from home import views

urlpatterns = [
    path('classify-url', views.classifyurl, name='classifyurl'),
    path("admin/", admin.site.urls),
    path("home/", views.home, name='home'),
]
```

Figura 69. Configuración de rutas urls.py en Visual Studio. Fuente este trabajo.

4.8 Resultados

4.8.1 Resultados del Entrenamiento.

Para poder seleccionar el método que más se ajusta al objetivo general debemos basarnos en 2 componentes los cuales son los resultados de precisión (Accuracy) y la matriz de confusión. Como se muestra a continuación en la siguiente tabla.

	K Nearest Neighbours	Random Forest	Deep NN
Accuracy	99.42%	99.69%	99.97%
Falsos Positivos/Falsos Negativos	38/12	24/6	2/0

Tabla 3. Resultados Accuracy/Falsos Positivos-Negativos. Fuente: este trabajo.

A simple vista se observa claramente que el mejor método a implementar en el modelamiento es Deep NN y tiene varias razones lo cual lo hace el mejor entre los demás métodos, entre las cuales está la alta capacidad de aprendizaje no lineal lo cual facilita modelar relaciones y sus características de entrada como determinar si es maliciosa o no, Deep NN maneja muchas capas y neuronas lo cual permite sostener mayor nivel de abstracción así aprender patrones

como palabras comunes de URLs maliciosas, dominios sospechosos, caracteres especiales, etc. Por ende, obtiene menos resultados de falsos negativos y falsos positivos.

Sin embargo, se debe tener en cuenta que estos valores resultantes podrían estar relacionados a un sobreajuste(Overfitting) ya que un Accuracy de 99.97% con 0 falsos negativos y 2 Falsos Positivos son valores excelentes pero no confiables porque Deep NN maneja un numero grande de parámetros, capas y capacidad de aprendizaje lo cual es usado para manejar una cantidad enorme de datos, en este caso estamos manejando un archivo Excel que contiene alrededor de 45000 URLs que podría considerarse que no es demasiado grande para la complejidad de modelo Deep NN, lo cual podría ser la causa del overfitting. Por esta razón se implementó el método con valores promedio y el que mejor se ajusta es el método Random Forest.

Al seleccionar el Método Random Forest estamos superando la expectativa del número de URLs bloqueadas que se había mencionado en el alcance de este trabajo, lo cual se pretendía bloquear el 30% de las URLs del segundo listado. La matriz de confusión del método Random Forest nos da como resultado 6 Falsos Negativos (URLs “Malicious” que RF clasificó como “Benign”) lo que significa que estamos bloqueando alrededor del 60% de las URLs del segundo listado que son aproximadamente 15 URLs, las cuales el sistema FW Allot no pudo bloquear.

4.8.2 Resultados del programa.

Una vez se definen las extensiones en Visual Studio y con sincronización previa del framework Django se procede a realizar pruebas en la web diseñada para tal fin. Inicialmente se corre a través de Anaconda Prompt la extensión manage.py para la visualización de la página que está contenida en el framework por medio del siguiente código: myproject>python manage.py runserver y muestra la página diseñada en index.html:

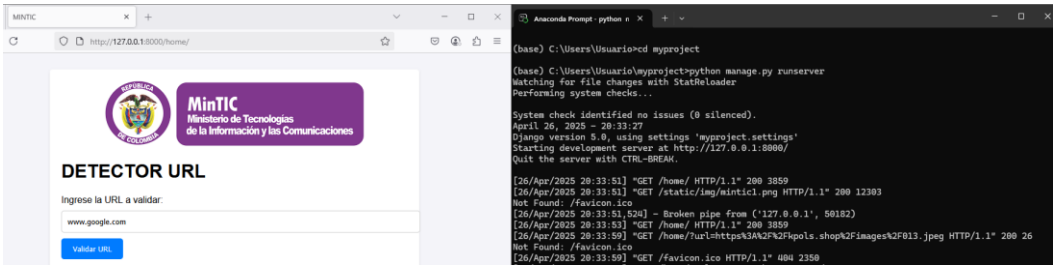


Figura 70. Página principal index.html con Django. Fuente este trabajo.

La primera prueba es con la página de Google.com lo cual retorna el resultado redirigiendo a la página. Se puede observar en framework Django (lado Derecho) la respectiva consulta a la pagina

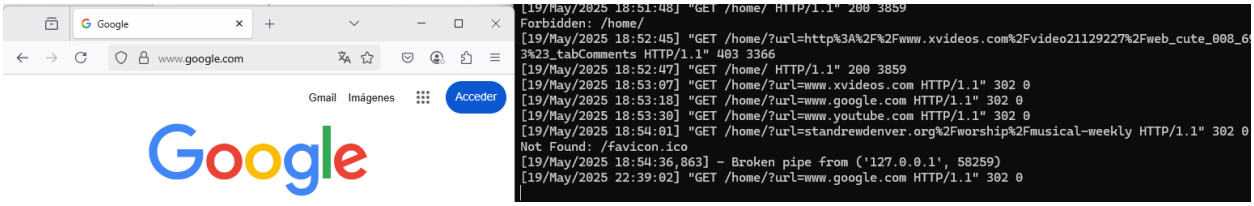


Figura 71. Prueba página Benign Google. Fuente este trabajo.

Se procede a realizar pruebas con la URL http://www.xvideos.com/xxxx/web_cute_xxx_xxx#_tabComments (por temas de seguridad se reemplaza algunos caracteres con x) catalogada como URL de pornografía infantil, se realiza la consulta y redirige a la página Blocked.html.

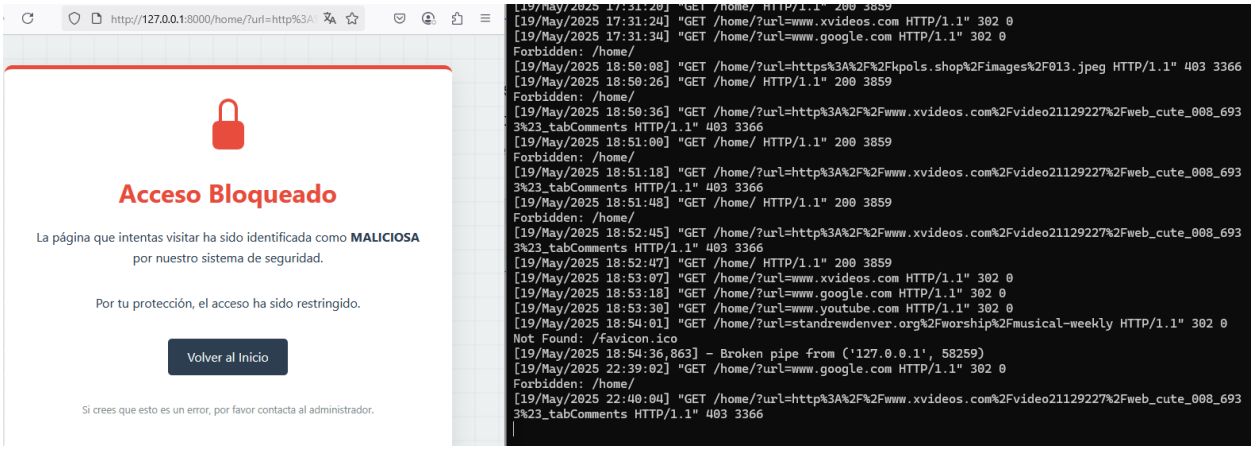


Figura 72. Prueba página Malicious. Fuente este trabajo.

4.9 Conclusiones.

1. Se desarrolló e implementó un sistema automatizado de bloqueo para las URLs con contenido de pornografía infantil apoyados con el lenguaje Python como entorno de desarrollo y técnicas de aprendizaje automático. Se implementa el método de Random Forest el cual demostró ser eficaz en la clasificación de URLs, permitiendo una detección precisa y robusta frente a distintos patrones de comportamiento de los datos.

2. En el desarrollo de este trabajo se implementaron diversas herramientas y tecnologías basadas en IA para la detección de contenido ilícito (CSAM). Se investigaron y analizaron soluciones como Thorn, NetClean y Safer Networking, las cuales usan enfoques avanzados para identificar y bloquear contenido ilegal, se explora diferentes tecnologías de IA, incluyendo sistemas expertos, redes neuronales artificiales, Deep learning, robótica y agentes inteligentes, teniendo presente su aplicación en sistemas automatizados de detección. En la implementación práctica, se emplean plataformas como Allot y entornos de desarrollo Python, Visual Studio y Django los cuales permiten construir soluciones escalables y funcionales. Además. El uso combinado de estas tecnologías aportó significativamente a la efectividad del sistema propuesto y al fortalecimiento de los mecanismos de prevención ante este tipo de delitos.

3. El desarrollo del sistema de bloqueo de URLs de pornografía infantil es presentado bajo un marco legal sólido, establecido por la Constitución Política de la República de Colombia. Se tiene en cuenta el artículo 44, el cual consagra los derechos fundamentales de los niños y exige su protección frente a todo acto de violencia, abuso y explotación. Asimismo, el articulo 13 obliga al estado a proteger a las personas en condición de vulnerabilidad, como es el caso de los menores de edad. El proyecto también se enfoca en la ley 1273 de 2009, que introduce la protección de información y de los sistemas informáticos como bien jurídico, junto a la ley 1336 de 2009, la cual aborda de manera drástica las penas relacionadas con actos de esta índole. La ley 679 de 2001 con el Decreto 1524 de 2002 refuerza el deber de denuncia ante la difusión de material ilegal relacionado con menores de edad. Bajo el contexto normativo, el modelo presentado en este trabajo no solo busca una solución tecnológica, sino también una respuesta alineada con el compromiso legal y ético de proteger los derechos de los niños frente a los delitos que se cometen a través de los medios digitales.

4. Para el sistema propuesto se realiza la implementación de distintos modelos de clasificación supervisada basados en técnicas de machine learning, el objetivo es analizar las URLs por categorías previamente definidas como “Benign” y “Malicious” Entre los modelos utilizados, el método K Vecinos más Cercanos (KNN) permite clasificar nuevas URLs en función de similitud con datos previamente etiquetados, utilizando la métrica de distancia euclidiana. El modelo Random Forest ofrece una clasificación robusta al combinar múltiples arboles de decisión, lo cual mejoró significativamente la precisión y se redujo el sobreajuste. El modelo de redes neuronales profundas, las cuales son inspiradas en la estructural neuronal cerebral, demostraron ser eficientes para reconocer patrones complejos en grandes volúmenes de datos. Para garantizar la efectividad del sistema, se ha dedicado un esfuerzo considerable a la medición de calidad de modelos, utilizando métricas específicas de clasificación que permitieron evaluar objetivamente el desempeño de cada enfoque implementado. Este proceso fue clave para la selección del modelo más adecuado, optimizando los resultados para la detección automatizada de contenido ilegal en línea.

5. Durante el desarrollo del sistema de clasificación y bloqueo de URLs, se utilizaron herramientas y plataformas que facilitaron el análisis de datos y la implementación de la solución final. Jupyter lab hace parte de las herramientas utilizadas para el procesamiento de datos, ya que permitió cargar, transformar, modelar y probar el código de forma eficiente en un entorno controlado, lo cual favorece la documentación que sea clara en el proceso y generación de reportes. En cuanto al dataset, se organizaron 2 conjuntos de URLs. El primer grupo esta etiquetado como “Malicious” las cuales se cargan en el archivo 23.620 URLs, las cuales provienen del listado oficial de MINTIC, de acceso únicamente para los ISP en territorio nacional, el segundo grupo etiquetado como “Benign” conforman 24.999 URLs las cuales son extraídas del dataset público “Malicious_URL’s_Dataset” de Kaggle. La cual se consolida en un archivo llamado “DatasetMintic.csv”, el cual fue útil para entrenar y validar los modelos de clasificación que se manejaron en este proyecto como K Vecinos más Cercanos (KNN), Random Forest y Redes Neuronales Profundas. Para la implementación practica del sistema, se utilizó Visual Studio como plataforma para el desarrollo del frontend el cual permite validar las URLs ingresadas por el usuario, además del framework Django que facilitó el despliegue de la aplicación en un entorno web funcional y accesible bajo un entorno controlado.

6. En la fase de evaluación de los 3 modelos mencionados, se realiza pruebas con KNN, RF y DNN utilizando matrices de confusión como herramienta para el análisis de rendimiento de cada modelo. Para KNN, se entrenó el 80% del conjunto de datos y se obtuvo un comportamiento aceptable, obteniendo 38 falsos positivos y 12 falsos negativos. Para RF se observa mejoras significativas, reduciendo errores de clasificación y presentando mayor precisión con un resultado de 24 falsos positivos y 6 falsos negativos. Finalmente, DNN arrojó un desempeño sobresaliente con un accuracy del 99.7%, con 2 falsos positivos y 0 falsos negativos, lo cual lo posiciona como el mejor modelo a implementar, sin embargo, se identifica que el modelo probablemente tenga sobreajuste (overfitting), porque se entrenó un modelo de alta complejidad sobre un conjunto de datos relativamente reducido (Aprox 45.000 URLs), lo cual podría comprometer su funcionamiento en escenarios de pruebas. Por tal motivo, se concluye que, aunque el modelo DNN muestra mejores métricas, el modelo más confiable y balanceado a implementar es Random Forest, ya que ofrece resultados solidos con mejor riesgo de sobreajuste, lo cual se adapta de manera eficiente al objetivo general del proyecto.

7. Esta investigación aporta al campo tecnológico una solución practica y automatizada para la detección y bloqueo de URLs con contenido de pornografía infantil, integrando herramientas modernas de Inteligencia artificial y machine learning en el ámbito regulatorio colombiano. A través del uso de modelos como KNN, Random Forest y redes neuronales profundas, se confirma que es posible clasificar con alta precisión URLs en grandes volúmenes de datos, lo cual facilitaría la labor de las entidades encargadas de proteger a los menores de edad a través del entorno digital. A diferencia de estudios previos, se enfocan en análisis de texto o filtrado manual, este trabajo propone una arquitectura completa que va desde la consolidación del dataset hasta el despliegue web funcional, utilizando herramientas como Python, JupyterLab, Visual Studio y Django. Se realiza evaluación comparativa de los modelos con métricas reales como la matriz de confusión, lo cual permite seleccionar el modelo mas apropiado por su precisión y eficiencia, en conclusión, este estudio demuestra que es viable manejar técnicas de inteligencia artificial para la protección infantil en línea y marca un precedente en la implementación de soluciones alineadas con la legislación nacional, lo cual es potencial de escalabilidad para uso de contextos institucionales y privados.

8. Para los modelos de clasificación KNN, RF y DNN propuestos en este trabajo de grado, se ha realizado el entrenamiento por medio de 2 métodos. El primero es el método de división de datos, 80% train, 20% test. El segundo es el método de validación cruzada el cual usa el 100%

de los datos para su respectivo entrenamiento y validación. Los resultados muestran que la mejor opción, si se aplica el método de división de datos es por Random Forest ya que tiene solo tiene 6 falsos negativos, según su matriz de confusión. En cambio, si se aplica el método de validación cruzada el mejor modelo es con DNN ya que arroja 57 falsos negativos. Ambos métodos muestran eficiencia, si embargo para poder cumplir con el objetivo de bloquear el 60% del segundo listado de las URLs las cuale seguían sin filtrar, se usa Random Forest.

REFERENCIAS

[1] National Center for Missing & Exploited Children, “Every child deserves a safe childhood.,” National Center for Missing & Exploited Children. Accessed: Sep. 09, 2023. [Online]. Available: https://www.missingkids.org/

[2] International Centre for Missing and Exploited Children, “Give now and help us build a safer world for children.,” International Centre for Missing and Exploited Children. Accessed: Sep. 09, 2023. [Online]. Available: https://www.icmec.org/

[3] Internet Watch Foundation, “We work to stop the repeated victimisation of people abused in childhood and make the internet a safer place, by identifying & removing global online child sexual abuse imagery.,” Internet Watch Foundation. Accessed: Sep. 09, 2023. [Online]. Available: https://www.iwf.org.uk/

[4] ECPAT Internationa, “Working Together to End the Sexual Exploitation of Children,” ECPAT Internationa. Accessed: Sep. 09, 2023. [Online]. Available: https://ecpat.org/

[5] THORN, “It is the greatest tool we have in the fight against human trafficking.,” THORN. Accessed: Sep. 09, 2023. [Online]. Available: https://www.thorn.org/spotlight/

[6] NetClean Technologies, “Protection for all devices,” NetClean Technologies. Accessed: Sep. 09, 2023. [Online]. Available: https://www.netclean.com/

[7] Safer-Networking Ltd., “Your privacy and the security of your computer is important to us, see how we can help you!,” Safer-Networking Ltd. Accessed: Sep. 09, 2023. [Online]. Available: https://www.safer-networking.org/

[8] Investigador: Ing. Carlos Roberto Monroy Alfaro Consejo de Arbitraje: Ing. Jorge Edwin Machado Leiva Ing. Rolando de Jesús Medrano Contreras Oponente: Lic. Francisco Rolando Torres Cárcamo, "'EL LENGUAJE PYTHON Y SU POTENCIAL EN EL DESARROLLO DE SOFTWARE DE INTELIGENCIA ARTIFICIAL,'" *Masferrer Investiga*.

[9] "Conoce 5 tipos de inteligencia artificial y para qué te servirán en 2023," blog.hubspot.es. Accessed: Oct. 07, 2023. [Online]. Available: https://blog.hubspot.es/marketing/tipos-inteligencia-artificial

[10] Ingecom, "ALLOT," *Ingecom*, 2023, Accessed: Sep. 09, 2023. [Online]. Available: https://www.ingecom.net/es/solucion/38/allot/

[11] AWS, "¿Qué es Python?," Amazon. Accessed: Sep. 09, 2023. [Online]. Available: https://aws.amazon.com/es/what-is/python/

[12] Python Software Foundation, "History and License." Accessed: Sep. 09, 2023. [Online]. Available: https://docs.python.org/3/license.html

[13] "Inteligencia Artificial: ¿Qué es?," SALESFORCE. Accessed: Oct. 07, 2023. [Online]. Available: https:// www.salesforce.com/mx/blog/2017/6/Que-es-la-inteligencia-artificial.html

[14] "Que es IA y porque Importa?," Goodnight, J. (s.f.). SAS. Accessed: Oct. 07, 2023. [Online]. Available: https://www.sas.com/es_ar/insights/analytics/what-is-artificial-intelligence.html

[15] "Microsoft Visual Studio - Danysoft." Accessed: Apr. 16, 2025. [Online]. Available: https://www.danysoft.com/visual-studio/

[16] "Framework Web Django (Python) - Aprende desarrollo web | MDN." Accessed: Apr. 19, 2025. [Online]. Available: https://developer.mozilla.org/es/docs/Learn_web_development/Extensions/Server-side/Django

[17] Fernando Miró Llinares, *El cibercrimen. Fenomenología y criminología de la delincuencia en el ciberespacio.* Madrid, 2012.

[18] Werle / F. Jessberger. En T. Weigend / G. Dannecker / G. Werle (Hrsgs.), "En este grupo de delitos lo único que se traslada o transmite es la información, los contenidos, pero no el delito," 2011, *De Gruyter, Berlin*.

[19] C.F. Müller; J. Kappel., "'Reflexiones sobre la ciberdelincuencia hoy (En torno a la ley penal en el espacio virtual),'" *Revista Electrónica del Departamento de Derecho de la Universidad de La Rioja (REDUR)*, Hamburgo, 2002.

[20] J. M. Sánchez-Torres, “Propuesta metodológica para evaluar las políticas públicas de promoción del egovernment como campo de aplicación de la Sociedad de la información. Conceptualización y aplicación empírica en el caso colombiano,” tesis doctoral, Universidad Autónoma de Madrid, Madrid, 2005.

[21] M. Velicogna, *In Search of Smartness: The EU e- Justice Challenge*, vol. 4. nformatics, 2017.

[22] M. P. Rodríguez, “Ciberseguridad en la justicia digital: recomendaciones para el caso colombiano,” *Revista UIS Ingenierías*, vol. 20, no. 3, p. 2, May 2021, doi: 10.18273/revuin.v20n3-2021002.

[23] R. N. Londoño-Sepulveda, “The use of ICT in judicial procedures: a proposal for online justice L‘usage des TIC dans les procédures judiciaires: une proposition de la justice en ligne,” *Rev. Fac. Derecho y Ciencias Política*s, 2010.

[24] Unión Internacional de Telecomunicaciones UIT, “UIT-T X.1205 Aspectos generales de la ciberseguridad,” vol. 1205, 2008.

[25] ISO/IEC, “Information Technology-Security Techniques- Guidelines for Cybersecurity,” 2012.

[26] Departamento Nacional de Planeación, “Política Nacional de Confianza y Seguridad Digital.,” 2020.

[27] Departamento Nacional de Planeación, “Política de Seguridad Digital,” Bogota, 2016.

[28] inhope.org, “Combatir el CSAM,” INHOPE Association. Accessed: Sep. 09, 2023. [Online]. Available: https://www.inhope.org/EN

[29] H.-E. Lee, T. Ermakova, V. Ververis, and B. Fabian, “Detecting child sexual abuse material: A comprehensive survey,” *Forensic Science International: Digital Investigation*, vol. 34, p. 301022, Sep. 2020, doi: 10.1016/j.fsidi.2020.301022.

[30] P. Eleuterio and M. Polastro, “An adaptive sampling strategy for automatic detection of child pornographic videos,” in *Proceedings of the Seventh International Conference on Forensic Computer Science*, Abeat, Sep. 2012, pp. 12–19. doi: 10.5769/C2012002.

[31] E. Howard†, “DON’T CACHE OUT YOUR CASE: PROSECUTING CHILD PORNOGRAPHY POSSESSION LAWS BASED ON IMAGES LOCATED IN TEMPORARY INTERNET FILES,” *Berkeley Tech. LJ*, 2004.

[32] B. Westlake, M. Bouchard, and R. Frank, “Assessing the Validity of Automated Webcrawlers as Data Collection Tools to Investigate Online Child Sexual Exploitation,” *Sexual Abuse*, vol. 29, no. 7, pp. 685–708, Oct. 2017, doi: 10.1177/1079063215616818.

[33] Economia 3, “Peer to peer: ¿En qué consiste este modelo de negocio y sus ventajas?” Accessed: Oct. 25, 2023. [Online]. Available: https://economia3.com/peer-to-peer/

[34] C. Peersman, C. Schulze, A. Rashid, M. Brennan, and C. Fischer, “iCOP: Automatically Identifying New Child Abuse Media in P2P Networks,” in *2014 IEEE Security and Privacy Workshops*, IEEE, May 2014, pp. 124–131. doi: 10.1109/SPW.2014.27.

[35] “Constitución Politica de Colombia 1991,” https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=4125.

[36] “LEY 1273 DE 2009 ,” Jan. 2009.

[37] P. Público and R. Legislativa, “Texto oficial sin modificaciones.” Accessed: Sep. 09, 2023. [Online]. Available: https://www.oas.org/dil/esp/codigo_penal_colombia.pdf

[38] C. Menores, “DEL USO DE REDES GLOBALES DE INFORMACIÓN EN RELACIÓN.” Accessed: Sep. 09, 2023. [Online]. Available: http://www.oas.org/juridico/spanish/cyb_col_ley_679_2001.pdf

[39] IBM, “¿Qué es el aprendizaje supervisado?,” https://www.ibm.com/es-es/topics/supervised-learning.

[40] J. Bodanilla Sancho, *Machine Learning y Deep Learning*, vol. 1. Madrid: Rama Editorial, 2020.

[41] J. M. Ortega Candel, *Big Data, Machine Learning y Data Science en Python*, vol. 1. Madrid: Rama Editorial, 2023.

[42] “StandardScaler — scikit-learn 1.6.1 documentation.” Accessed: Apr. 30, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[43] Jacar, “La Función Sigmoide: Una Herramienta Clave en Redes Neuronales.”

[44] Codificando Bits, “La Función de Activación,” La función tangente hiperbólica (tanh).

[45] Data Camp, “Introducción a las funciones de activación en las redes neuronales,” © 2024 DataCamp, Inc.

[46] J. Martel, “¿Conoces PyTorch? La herramienta de open source con la que puedes crear redes neuronales,” itelligent.es.

[47] S. E. GeeksforGeeks, “Adam Optimizer in Tensorflow,” GeeksforGeeks.

[48] https://docs.python.org/, “urllib.parse — Analiza URL en componentes,” https://docs.python.org/.