

UNIVERSIDAD DE LA CIUDAD DE AGUASCALIENTES

MAESTRÍA EN CIENCIA DE DATOS



SEMINARIO DE TESIS

**“Detección de anomalías mediante aprendizaje automático en
tráfico de servidores web”**

Presenta:

MITSIU ALEJANDRO CARREÑO SARABIA

Periodo Agosto 2024 - Diciembre 2024, Aguascalientes, Ags

Variable independiente: Cada petición procesada por el servidor, la cuál agrupa distintas características como, dirección IP del cliente, fecha, hora, cantidad de bytes enviados, recursos solicitados, entre otros.

Variable dependiente: Grado de anormalidad en la actividad.

Glosario

API - Pieza de código que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades.

Backend - También conocido como server-side, Es la capa de acceso a la información en una solución de software, es responsable de implementar la lógica de negocio y hacerla accesible a través de una API.

Bot - Un programa que ejecuta tareas automatizadas en internet, usualmente con la intención de imitar actividad humana a gran escala.

Distancia Euclídea - Distancia ordinaria entre dos puntos en un espacio euclídeo (basado en coordenadas cartesianas).

Endpoint - Identifica los recursos y forma de acceso en una API web del lado del servidor. Define y permite la comunicación con software de terceros.

Framework - Término genérico empleado para describir una estructura base sobre la cual se construyen nuevas aplicaciones.

Gzip - Es un formato de archivos que permite la compresión y descompresión de información.

Host - También conocido como anfitrión, es un ente que provee los recursos y espacio necesarios para un evento; en el contexto de este documento se refiere a un equipo de computo.

HTTP/HTTPS - Son protocolos de internet para el modelo de sistemas de información hipermedia (gráficos, audio, video, texto e hipervínculos) distribuidos y colaborativos

Input - Proceso de proveer o ingresar información a la computadora.

Internet de las cosas - Describe dispositivos con sensores, capacidad de procesamiento, software y otras tecnologías que permiten conectarse e intercambiar información con otros dispositivos y sistemas en internet u otras redes de comunicación.

Logs - Grabación secuencial en un archivo o base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular; de esta forma se construye una evidencia del comportamiento del sistema.

Parámetro - En computación se refiere a una variable especial usada en una subrutina (función) para referirse a información provista como input.

Payload - Parte de la información transmitida que comprende el mensaje en sí que se quiere enviar, dejando fuera cabeceras y metadatos.

Protocolo de aplicación - Se refiere a una capa de abstracción que especifica los protocolos de comunicación (TCP/IP), interfaces y métodos que usa un anfitrión para comunicarse en una red.

Servidor web - Un software de computadora que acepta peticiones vía HTTP/HTTPS (protocolo de red creado para distribuir contenido web). Para interactuar con él un agente de usuario (comúnmente un navegador web o un bot) inicia la comunicación al hacer una petición de una página web o recurso, el servidor contesta con el contenido del recurso o un mensaje de error.

SSH - Protocolo de shell segura, es un protocolo criptográfico de red que permite interactuar con equipos remotos en un canal de comunicación seguro.

Tensor - Un objeto algebraico que describe una relación multilineal entre conjuntos de objetos algebraicos relacionados con un espacio vectorial.

Sobreajuste - Se refieren a modelos que son demasiado apegados a los datos con los que fue entrenado y su desempeño predictivo es pobre en datos nuevos.

Subajuste - Se refiere a modelos que se desempeñan pobremente tanto en fase de entrenamiento como con datos nuevos, volviendo el modelo demasiado simplista y sin capacidad de capturar o aprender los patrones de los datos.

Agradecimientos

Cuando se agradece a todos, ¿es igual a no agradecer a nadie? Estoy profundamente agradecido con mis buenos profesores, uno de ellos fue quién me dió la oportunidad y respaldo para estudiar esta maestría, con mi familia por enseñarme a creer en mí mismo, a mi pareja por respaldar las decisiones que tomé o me ayudó a tomar.

Agradezco a las matemáticas y a la computación, por darme retos y verdades por comprender y a los amigos por volver el viaje llevadero.

Agradezco a todas las personas que he conocido porque todos me han influenciado a lo que se culmina hoy, todo me formó y deformó, me hicieron y me volvieron yo y por eso les estoy profundamente agradecido.

Resumen

El tráfico en servidores web incluso para aplicaciones y servicios medianamente exitosos es inmenso por lo que un análisis manual no es viable, ya que los servidores al estar accesible desde internet queda expuesto a múltiples usuarios y bots con diversos objetivos, por ello es necesario desarrollar soluciones tecnológicas que empleen técnicas de aprendizaje automático así como de cómputo distribuido que puedan procesar grandes volúmenes de información ofreciendo una detección rápida y confiable del grado de anomalía. Para cubrir estos requerimientos se empleó Apache Spark que no solo ofrece grandes capacidades de procesamiento distribuido sino también flexibilidad para escalar horizontalmente las capacidades y recursos disponibles para procesar la información. Además se optó por implementar el algoritmo de bosque de aislamiento debido a su rendimiento y adaptabilidad a entornos distribuidos. Con esta implementación se logró establecer una métrica de anomalía en el tráfico, permitiendo categorizar la actividad inusual para su posterior evaluación. Si bien el trabajo realizado aún requiere evaluación humana, reduce significativamente el volumen a analizar aumentando la probabilidad de exitosamente detectar peticiones maliciosas por lo que se considera que se logró desarrollar un sistema con bases sólidas de ingeniería de software y ciencia de datos, ofreciendo una plataforma para continuar refinando y expandiendo sus capacidades en estudios e implementaciones futuras.

Palabras clave: Detección de anomalías, cómputo distribuido, aprendizaje no supervisado, bosque de aislamiento.

Glosario.....	3
Agradecimientos.....	5
Resumen.....	6
Capítulo 1 - Antecedentes contextuales.....	9
1.1 Arquitectura web cliente-servidor.....	9
1.1.1 Comunicación Petición/Respuesta (Request/Response).....	9
1.2 Modelo OSI.....	11
1.3 Protocolo HTTP.....	11
1.3.1 Métodos de peticiones.....	13
1.3.2 Código de estado.....	13
1.4 Servidor proxy y proxy inverso.....	14
1.5 NGINX.....	16
1.5.1 Archivos de registros.....	16
1.6 Antecedentes.....	17
1.6.1 Contexto actual.....	18
Capítulo 2 - Problemática.....	20
2.1 Planteamiento del problema.....	20
2.1.1 Pregunta central de investigación.....	21
2.1.2 Preguntas secundarias de investigación.....	21
2.1.3 Objetivos de investigación.....	21
2.2 Justificación.....	21
2.3 Hipótesis.....	22
Capítulo 3 - Marco teórico.....	23
3.1 Aprendizaje no supervisado.....	23
3.1.1 Redes neuronales.....	23
3.1.1.1 Autoencoders.....	25
3.1.1.2 Mapa Autoorganizado.....	26
3.1.2 Clusterización.....	28
3.1.2.1 Observaciones anómalas no pertenecen a ningún cluster.....	28
3.1.2.2 Observaciones normales se sitúan cerca del centroide del cluster, mientras las anomalías están lejos.....	29
3.1.2.3 Las observaciones normales forman clusters grandes y densos, mientras que las anomalías forman clusters pequeños o dispersos.....	30
3.1.3 Bosque de aislamiento.....	30
3.2 Regresión Logística.....	32
3.3 Procesamiento del lenguaje natural.....	34

3.3.1 Atributos hash.....	35
3.4 Cómputo distribuído.....	36
3.4.1 Apache Spark.....	38
3.5 Estudios relacionados.....	41
3.5.1 Arquitectura Distribuida para la Predicción de DDoS y Detección de Bots / A Distributed Architecture for DDoS Prediction and Bot Detection.....	41
3.5.2 Análisis Empírico de Métodos para Detección de Anomalías en Series Temporales Multivariadas / An Empirical Analysis of Anomaly Detection Methods for Multivariate Time Series.....	45
3.6 Marco jurídico.....	48
Capítulo 4 - Metodología.....	51
4.1 Introducción.....	51
4.1.1 Contexto estadístico y estructural de los datos.....	51
4.1.2 Arquitectura tecnológica empleada.....	56
4.1.3 Descripción general del procesamiento desarrollado.....	59
4.2 Clasificador de dominio.....	60
4.2.1 Ingesta.....	60
4.2.2 Extracción y estructuración.....	60
4.2.3 Entrenamiento de modelo para determinar dominio.....	71
4.2.3.1 Codificación One-hot.....	71
4.2.3.2 Procesamiento de lenguaje natural.....	73
4.2.3.3 Modelo de regresión logística.....	77
4.2.4 Aplicar modelo para determinar dominio.....	77
4.3 Detección de anomalías.....	79
Capítulo 5 - Resultados y conclusiones.....	82
5.1 Resultados.....	82
5.1.1 Extracción.....	82
5.1.2 Entrenamiento de modelo.....	82
5.1.3 Aplicar modelo para determinar dominio.....	82
5.1.4 Detección de anomalías.....	85
5.1.5 Visualización.....	86
5.2 Limitaciones.....	87
5.3 Conclusiones.....	87
5.4 Trabajos futuros.....	88
Capítulo 6 - Bibliografía.....	90

Capítulo 1 - Antecedentes contextuales

A través del siguiente documento se realiza un estudio sobre cómo, porqué y para qué es útil el análisis de tráfico web. Para ello es necesario conocer los aspectos y terminologías fundamentales que componen el internet moderno así como las tecnologías involucradas en la generación de los datos estudiados.

1.1 Arquitectura web cliente-servidor

Se refiere a un “modelo de computación para el desarrollo de sistemas computarizados distribuidos” (Yādava, 2009) en el que existen dos entes “Clientes” y “Servidores” donde los clientes se refiere a dispositivos u otros programas de software los cuáles solicitan recursos y/o servicios, mientras que el “servidor” es un dispositivo con gran capacidad de cómputo que provee dichos recursos o servicios.

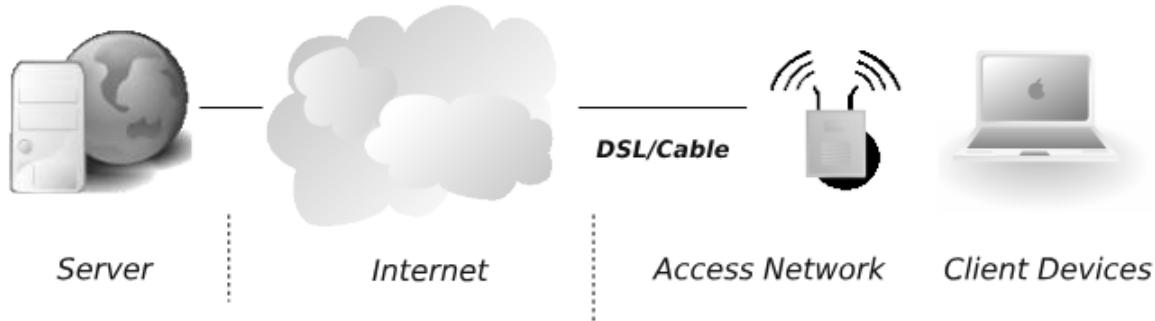


Figura 1: Representación de arquitectura cliente-servidor

Fuente: Kuschnig, 2011

Este modelo es un concepto fundamental en el desarrollo de páginas web y representa uno de los pilares fundamentales en la construcción del internet ya que prioriza el manejo eficiente de recursos al centralizar funciones críticas en el servidor mientras que el cliente maneja interacciones con el usuario.

1.1.1 Comunicación Petición/Respuesta (Request/Response)

La comunicación entre “clientes” y “servidores” se lleva a cabo bajo el patrón de intercambio de mensajes de transporte en la modalidad de petición/respuesta única en la que “se define un patrón para el intercambio de dos mensajes entre dos nodos, un mensaje proveniente del nodo solicitante al nodo respondiente, si el nodo respondiente realiza un procesamiento exitoso, el nodo respondiente envía un mensaje de respuesta al nodo solicitante” (W3C, 2001)

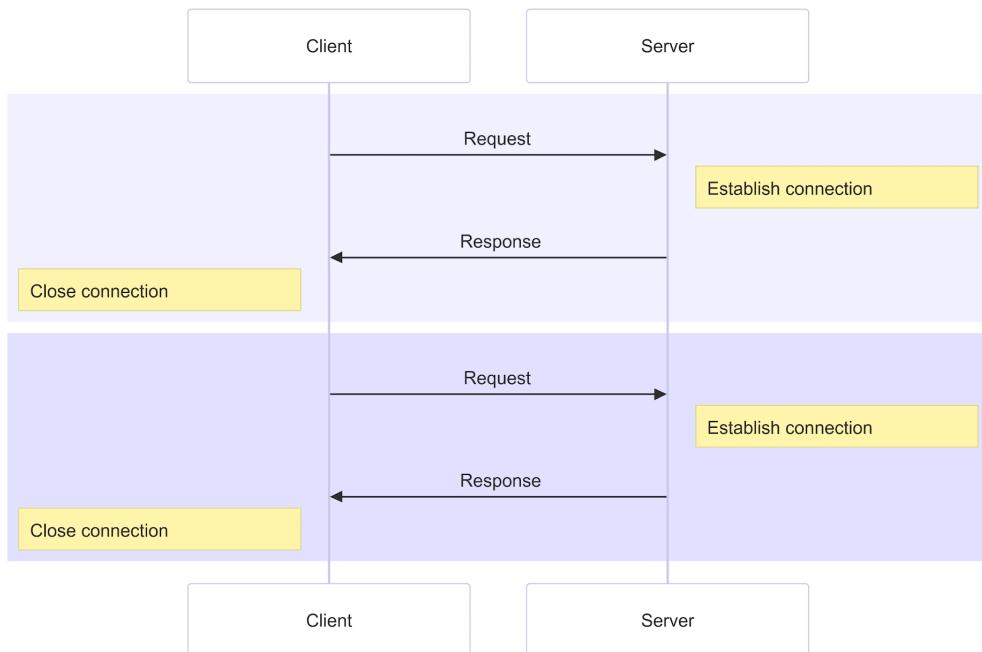


Figura 2: Representación de comunicación petición/respuesta

Fuente: Mozilla Foundation (2024a)

Es posible que surjan errores durante la comunicación tanto por parte del solicitante como del respondiente, por lo que se establecen los siguientes estados para cada una de las entidades:

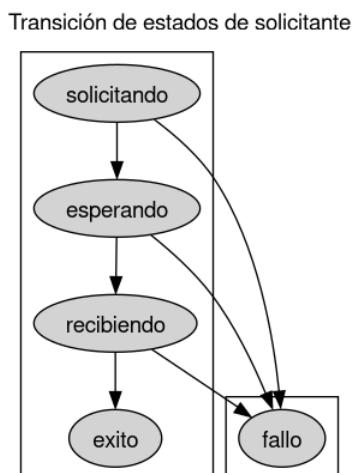


Figura 3: Flujo de estados del solicitante

Fuente: Adaptación de W3C, 2001

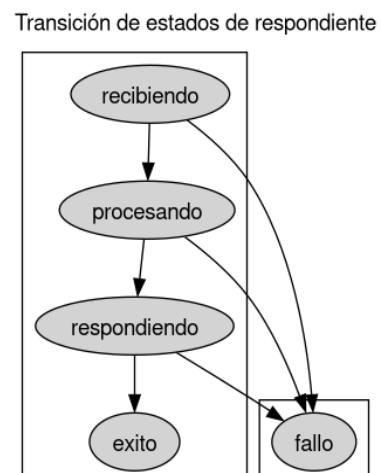


Figura 4: Flujo de estados de respondiente

Fuente: Adaptación de W3C, 2001

1.2 Modelo OSI

“El modelo de referencia OSI provee un marco de trabajo que define las convenciones y tareas requeridas para que sistemas de red se comuniquen uno con otro” (Kumar, 2014)

El modelo de interconexión de sistemas abiertos (OSI por sus siglas en inglés) establece siete capas, donde “una capa es una colección de funciones conceptualmente similares que provee servicios a la capa encima de ella y a su vez solicita servicios de la capa debajo. Por ejemplo una capa que provee comunicación libre de errores a través de la red, proporcionando el camino necesario para que una aplicación lo use.” (Kumar, 2014)

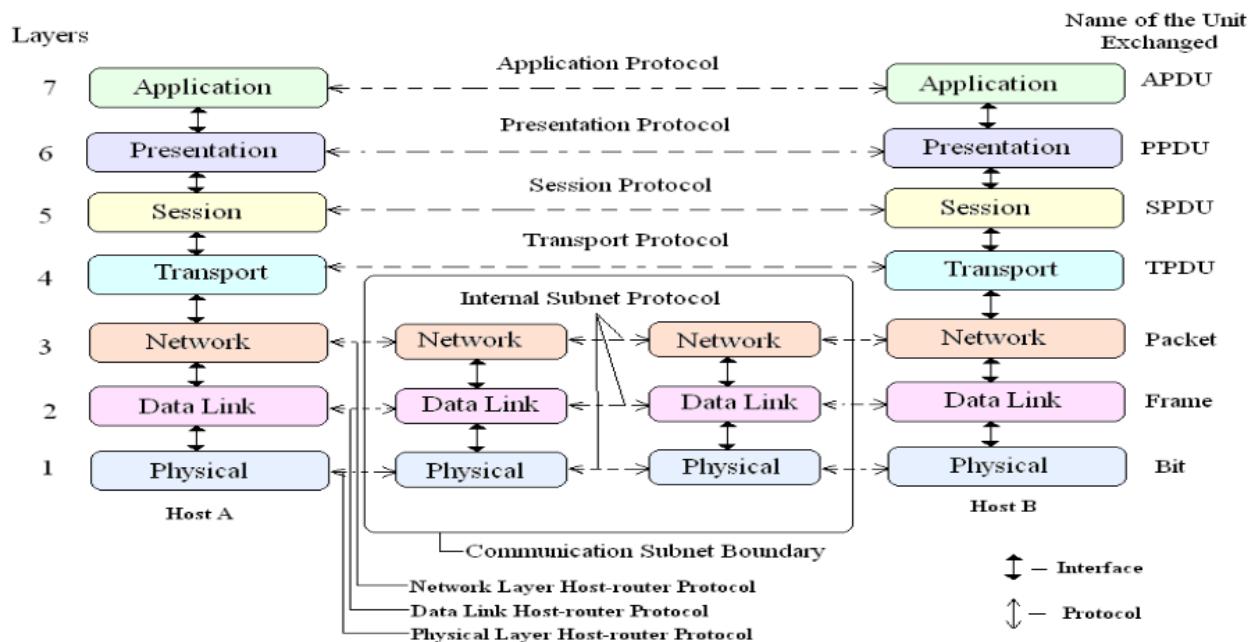


Figura 5: Descripción de capas en el modelo de red OSI

Fuente: Saxena, 2014

1.3 Protocolo HTTP

Del acrónimo Protocolo de Transferencia de Hipertexto, es el protocolo de facto para la comunicación del internet moderno, “los mensajes HTTP son bloques de datos enviados entre aplicaciones HTTP, estos bloques comienzan con una sección de ‘meta-information’ que sirve para describir el contenido y significado del mensaje, seguido de información opcional” (Gourley, 2002)

HTTP permite transferir cualquier tipo de información de manera confiable ya que opera implementando protocolos de transmisión de datos segura, lo que permite asegurar que la información no sea corrompida, dañada, perdida o desordenada.

Un mensaje HTTP está compuesto de las siguientes capas (David Gourley et al, 2002):

Solicitud HTTP	
GET /index.html HTTP/1.1	Línea de solicitud
Date: Thu, 5 Oct 2024 21:34:12 GMT Connection: close	Encabezados generales
Host: www.enlace.ucags.edu.mx From: JoeDoe@moodle.ucags.edu.mx Accept: text/html, text/plain User-Agent: Mozilla/4.0 (compatible: MSIE 6.0; Windows NT 5.1)	Encabezados de solicitud
	Cabezas de entrada
	Cuerpo de mensaje

Figura 6: Capas en una petición HTTP

Fuente: Adaptación de <https://developer.mozilla.org/en-US/docs/Web/HTTP/Session>

Respuesta HTTP	
HTTP/1.1 200 OK	Línea de solicitud
Date: Thu, 5 Oct 2024 21:34:13 GMT Connection: close	Encabezados generales
Server: Apache/1.3.27 Accept-Ranges: bytes	Encabezados de respuesta
Content-Type: text/html Content-Length: 140 Last-Modified: Tue, 18 Oct 10:14:49 GMT	Cabezas de entrada
<html> <body> <h1>This site is poorly build</h1> </body> </html>	Cuerpo de mensaje

Figura 7: Capas en una respuesta HTTP

Fuente: Adaptación de <https://developer.mozilla.org/en-US/docs/Web/HTTP/Session>

1.3.1 Métodos de peticiones

HTTP soporta varios comandos en la petición, a los cuales el protocolo se refiere como “Métodos de petición”. Cada petición HTTP debe tenerlo y sirve para informar al servidor la acción que se quiere realizar sobre el recurso.

Método HTTP	Descripción
Get	Solicita la representación del recurso especificado
Head	Solicita una respuesta idéntica a Get, pero sin el cuerpo de mensaje en la respuesta
Post	Envía una entidad nueva a un recurso en específico, usualmente causa un cambio en el estado o efectos secundarios en el servidor
Put	Reemplaza las representaciones actuales del recurso de destino con las presentes en el payload
Delete	Borra el recurso especificado
Connect	Establece un túnel hacia el servidor identificado por el recurso
Options	Sirve para que el servidor describa las opciones de comunicación a el recurso especificado
Trace	Realiza una prueba conexión y retorno de mensaje a lo largo de la ruta al recurso de destino
Patch	Sirve para aplicar modificaciones parciales a un recurso

Tabla 1: Métodos HTTP disponibles

Fuente: Mozilla Foundation (2024b)

1.3.2 Código de estado

HTTP emplea códigos de estado (status codes) representados de manera numérica para indicar si una petición HTTP ha sido completada con éxito. Estos códigos se agrupan en 5 clases:

Categoría	Rango de código de estado	Ejemplos
Respuestas informativas	100-199	100 Continue = Respuesta provisional, todo hasta ahora está bien
Respuestas satisfactorias	200-299	200 OK = Petición ha tenido éxito 201 Created = Se ha creado un nuevo recurso
Redirecciones	300-399	301 Moved Permanently = La URI del recurso ha sido cambiada, probablemente se regrese una nueva URI
Errores de los clientes	400-499	400 Bad Request = El servidor no pudo interpretar la solicitud dada una sintaxis inválida
Errores de los servidores	500-599	500 Internal Server Error = El servidor ha encontrado una situación que no sabe cómo resolver

Tabla 2: Clases de códigos de estado

Fuente: Mozilla Foundation (2024c)

1.4 Servidor proxy y proxy inverso

“A nivel básico un servidor proxy es una capa intermedia de software o hardware que recibe peticiones de clientes y las transmite a los servidores del backend” (Matsudaira, 2012).

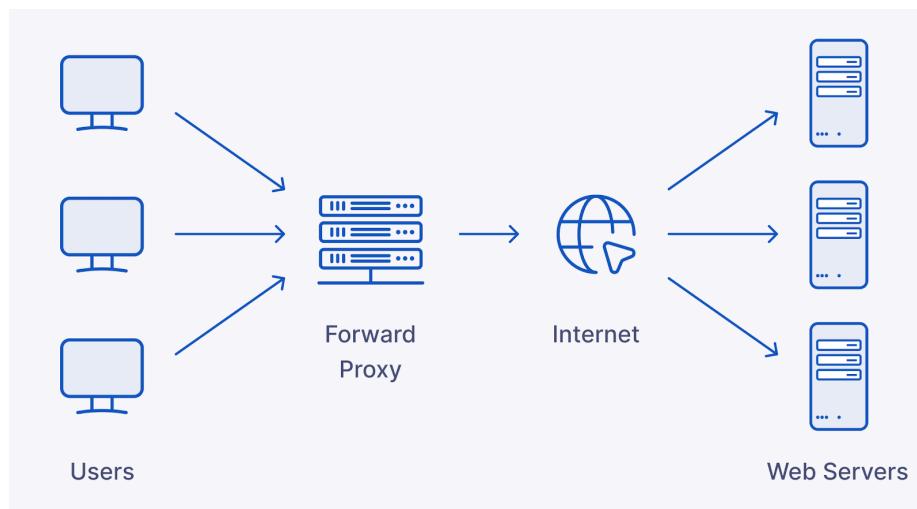


Figura 8: Arquitectura de servidor proxy

Fuente: UpGuard, 2024

Un servidor proxy se sitúa entre los clientes de una red específica (una red empresarial, de institución académica, etc) y la salida a internet, esto permite que el administrador de red, aplique filtros en las peticiones que pueden acceder a internet, monitorear peticiones, interceptar y transformar peticiones, además ofrece una capa de anonimidad a los usuarios, ya que el tráfico en internet aparece como generado por el servidor proxy en lugar de el usuario real.

Además existen los servidores proxy inverso, en el que de igual manera que un servidor proxy, reciben peticiones de clientes y las redirigen a servidores de backend, la diferencia radica en que estas peticiones vienen de internet, sin conocer realmente al cliente responsable de iniciar la comunicación.

“En sistemas distribuidos los proxies inversos suelen encontrarse en la parte más frontal del sistema, de modo que todas las peticiones entrantes sean interceptadas y redirigidas” (Matsudaira, 2012).

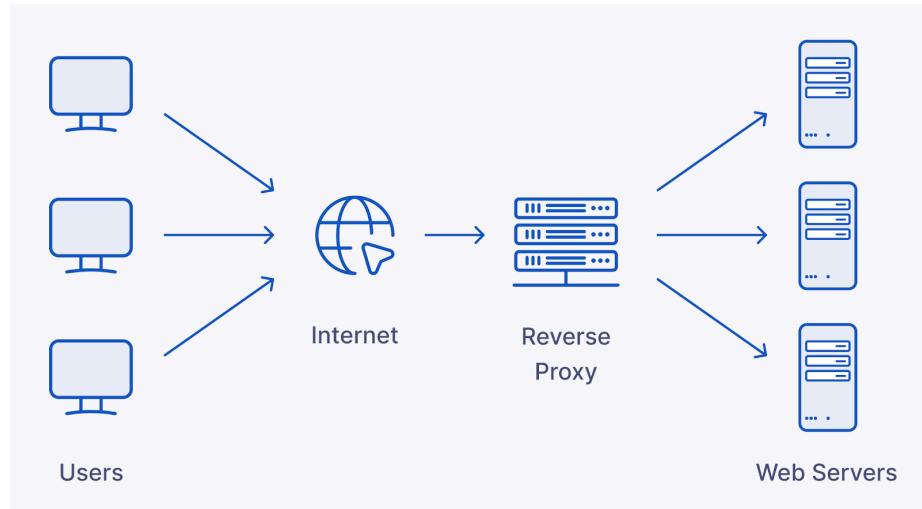


Figura 9: Arquitectura de servidor proxy inverso

Fuente: UpGuard, 2024

Entre las ventajas de un servidor de proxy inverso, destaca que permite mejorar la seguridad del sistema ya que “poner un servidor web o servidor de aplicación directamente accesible desde internet otorga a atacantes, contacto directo para encontrar cualquier vulnerabilidad de la plataforma subyacente. Sin embargo, para proveer un servicio útil a usuarios de internet, es necesario ofrecer acceso al servidor [...] un proxy inverso protege el servidor al nivel de protocolo de aplicación.” (Sommerlad, 2003).

1.5 NGINX

Es un software que permite levantar un servidor web y proxy inverso, ligero, de alto rendimiento, basado en los principios de software libre de código abierto, es uno de los servidores web más populares mundialmente. (Netcraft, 2024)

“Desde su lanzamiento en 2004, Nginx se ha enfocado en ofrecer alto rendimiento, alta concurrencia y bajo consumo de memoria. Funcionalidades adicionales [...] incluyen balance de carga, almacenamiento en caché, control de ancho de banda así como la habilidad para integrar eficientemente una variedad de aplicaciones, lo cuál ha hecho a Nginx una buena elección para la arquitectura de sitios web modernos” (Alexeev, 2012)

1.5.1 Archivos de registros

Un archivo de registros es un mecanismo común mediante el cual se realiza una evaluación de un programa durante tiempo de ejecución, esto permite minimizar el impacto en el software observado.

Estos archivos tienden a ser largos ya que “proveen almacenamiento y recuperación eficiente de información, la cuál es escrita de manera secuencial (solo-anexar) y que por tanto no sufre alteraciones una vez escrita. Programas y subsistemas usan servicios de registro para tareas de recuperación, de auditoría, o para monitoreo del desempeño” (Finlayson, 1987)

En el caso específico de NGINX la documentación indica que “NGINX escribe información acerca de las peticiones del cliente en el archivo access justo después de procesar la petición [...] la información escrita está determinada por un formato combinado [...] compuesto de variables” (Nginx, 2024)

Las variables y formato que componen una solicitud son las siguientes:

```
'$remote_addr - $remote_user - [$date_time] "$request" $status $body_bytes_sent  
"$http_referer" "$user_agent" "$gzip_ratio"
```

Código 1: Variables y formato de solicitud en NGINX

Fuente: NGINX, 2024

Donde cada variable representa:

Variable	Descripción
remote_addr	Dirección IPv4 o IPv6 del cliente que inició la conexión.
remote_usr	Nombre de usuario (solo aplica si el usuario está autenticado).
date_time	Fecha y hora en tiempo local del servidor en formato “día/mes/año:horas:minutos:segundos +0000”
request	Concatenación de método de la petición (véase Capítulo 1.3.1 para referencia), ruta del recurso solicitado por el cliente y versión del protocolo HTTP.
status	Código de estatus HTTP que resolvió el servidor (véase Capítulo 1.3.2 para referencia).
body_bytes_sent	Cantidad de bytes enviados en la respuesta.
http_referer	Valor de la cabecera http_referer con el que se conectó el cliente.
user_agent	Valor de la cabecera user_agent con el que se conectó el cliente.
gzip_ratio	Relación de compresión gzip en la respuesta.

Tabla 3: Descripción de variables en archivos de registro

Fuente elaboración propia

1.6 Antecedentes

El incremento en el uso de internet se debe gracias a la creciente disponibilidad de dispositivos de distintas categorías para conectarse a la red, el abaratamiento de servicios, la expansión de catálogo de contenido “esto incluye pasos como reducir los costos de acceso a las telecomunicaciones, mejorar la eficiencia de la red, expandir la infraestructura digital, fortalecer la alfabetización digital, proveer contenido diverso [...], promover servicios alcanzables e incrementar la competencia digital. Cada una de estas acciones ayuda a reducir la brecha entre usuarios y no usuarios (de internet)” (West, 2015).

Se ha avanzado significativamente en desarrollar soluciones y tecnología que permita escalar la infraestructura así como adopción de soluciones distribuidas para dar servicio a la ascendente demanda, desde la producción en masa de dispositivos celulares y móviles, hasta la progresiva adopción de tecnologías como el internet de las cosas que permite que dispositivos que tradicionalmente e históricamente no se conectaban a

internet como electrodomésticos, cuenten con funcionalidades dependientes una conexión a internet, pero derivado de dicha disponibilidad, se genera una cantidad inmensa de tráfico que cualquier servidor web disponible desde internet debe dar seguimiento, procesar, contestar. (Evans, 2011)

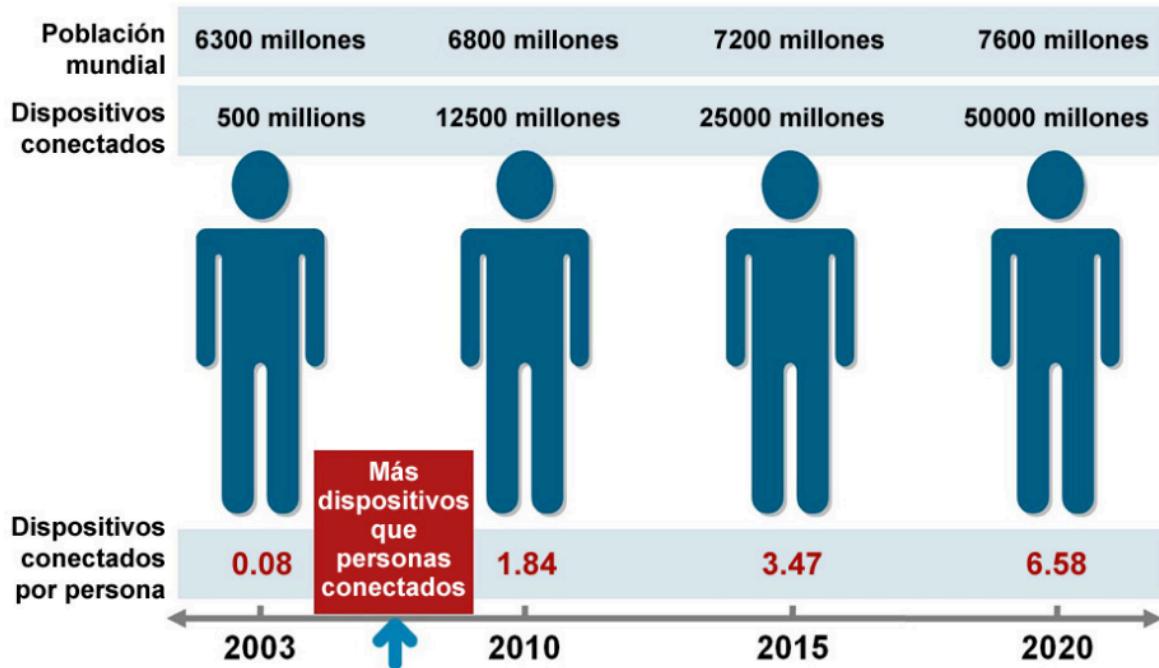


Figura 10: Crecimiento entre personas y dispositivos conectados

Fuente: Evans, 2011

1.6.1 Contexto actual

“En años recientes, el incremento en el tamaño y complejidad del software ha llevado a un rápido crecimiento en el volumen de logs. Para manejar estos volúmenes de logs de manera eficiente y efectivamente, una línea de investigación se centra en desarrollar técnicas inteligentes y automatizadas de análisis de estos datos” (Zhu, 2023)

El origen de dicho tráfico puede ser generado por peticiones de usuarios reales, peticiones de bots, peticiones automatizadas y peticiones de usuarios con intenciones maliciosas cada uno de estos grupos de clientes se comportan de maneras muy específicas.

“El objetivo de predecir recae en ‘anticipar la curva’ permitiendo a individuos y sistemas de salud reaccionar acorde a los eventos observados. Lo mismo ocurre con el tráfico de red, donde las tendencias se observan y predicen. Cuando se trata de detectar

anomalías, los algoritmos pueden agrupar datos normales e inferir comportamientos anómalos.” (Rahal et al, 2020)

El tráfico de servidores web tiene claras tendencias como recursos solicitados, región geográfica de donde se solicita, hora en que se solicitó, cantidad de bytes enviados, por lo que identificar las tendencias y detectar las anomalías es un trabajo que puede ser automatizado y al que se le pueden aplicar distintas técnicas de aprendizaje automático que permitan analizar la información desde múltiples dimensiones y perspectivas.

“La recolección de información de sensores y sistemas puede usarse para determinar tendencias en el comportamiento del consumidor [...] Esta información puede usarse después para desarrollar productos, medir la eficiencia de sistemas, identificar debilidades en el sistema y para cambiar estrategias de negocio” (Sardjono et al, 2021)

Como consecuencia del acceso generalizado a servicios en internet, es común confiar en las protecciones que ofrece el proveedor del servicio y permitirle alojar datos personales y sensibles en sus servidores, lo que aumenta la relevancia de evaluar qué y cómo se están accediendo a los recursos solicitados. “Ciertos riesgos han crecido con la adopción acelerada del internet. Preocupaciones sobre privacidad, fraude en línea, robo de identidad, hackeo de materiales y bases de datos sensibles han atraído atención significativa [...] pero esta preocupación por legislación necesitan ser comparadas contra la oportunidad y crecimiento potencial que ofrece el internet para enriquecer vidas, construir negocios y ofrecer a consumidores elecciones mejoradas en los años por venir.” (Manyika, 2011)

Capítulo 2 - Problemática

2.1 Planteamiento del problema

En los últimos años hemos experimentado una impresionante expansión de servicios a través de internet, por ejemplo compras en línea, redes sociales, plataformas de entretenimiento entre otras, “el tráfico total de internet ha experimentado un crecimiento dramático, en las dos últimas décadas. [...] en 1992, las redes de internet global transmitían aproximadamente 100 GB de tráfico al día. [...] En 2017, el tráfico global de internet alcanzó más de 45,000 GB por segundo” (Cisco, 2017).

Un producto de este aumento en el tráfico de internet es que cada servidor web debe manejar un mayor número de conexiones e intercambios de información con una mayor cantidad de clientes.

Los términos Cliente/servidor se usan para describir un modelo de computación para el desarrollo de sistemas computarizados. Este modelo se basa en la distribución de funciones entre dos procesos independientes y autónomos. Un cliente es cualquier proceso que solicita servicios específicos del proceso servidor. Cuando el cliente y servidor residen en dos o más computadoras independientes en una red, el servidor puede proveer servicio a más de un cliente sin importar la ubicación o características físicas del computador donde reside el proceso servidor. (Yādava, 2009)

El tráfico de un servidor web provee datos confiables sobre accesos, solicitudes y procesamiento de peticiones y el contexto bajo el que se usan sus recursos, pero el volumen de información generada es tan grande que un análisis manual no es viable. Entender los usos típicos y diferenciarlos de los atípicos es una herramienta poderosa que aplicada en tiempo real permitirá mejorar la calidad, y resguardo de la información contenida.

Al incrementar el número de servicios y usuarios en internet, realizar un análisis exploratorio profundo y de calidad resulta una tarea compleja pero de gran valor, ya que permite conocer rasgos y características puros sobre la interacción entre usuarios y el servicio proporcionado. Por ello es que se considera necesario desarrollar soluciones tecnológicas que permitan procesar de manera automatizada la cantidad de información generada en los archivos de registros (véase Capítulo 1.5.1 para referencia) de los servidores web.

“El viejo sistema de “almacenar y procesar” obstaculizará en gran medida el sistema de cómputo [...] porque le toma tiempo procesar, por ello es necesario mirar a tecnologías que hagan todo el procesamiento al mismo tiempo” (Sardjono, 2021)

2.1.1 Pregunta central de investigación

Este estudio se centra en la relación entre tipo de usuario y patrones de uso e interacciones con los servidores por lo que se plantea la siguiente pregunta de investigación.

¿Es confiable la estimación del grado de anomalía en el tráfico de servidores web mediante técnicas heurísticas y de aprendizaje automático para detectar cambios en los patrones de uso de los recursos disponibles en dicho servidor web, y puede esta medida ser útil para detectar exitosamente actividad maliciosa?

2.1.2 Preguntas secundarias de investigación

Se generaron las siguientes preguntas de investigación las cuales permiten complementar aspectos importantes a considerar durante el estudio.

- ¿De qué manera se analiza el tráfico de servidores web actualmente?
- ¿Qué elementos debe tener un sistema de detección de anomalías para ser útil (falsos negativos/falsos positivos, canales de comunicación, protocolos de contingencia y respuesta)?
- ¿Actualmente cómo se ha implementado el aprendizaje automático en análisis de tráfico de servidores web?

2.1.3 Objetivos de investigación

El objetivo de este estudio es explorar la implementación de técnicas heurísticas, así como de aprendizaje automático en un sistema integral y automatizado que permita determinar si la actividad y tráfico de un servidor web es anómala, permitiendo:

- a) Analizar grandes volúmenes de datos provenientes de múltiples clientes y dominios.
- b) Integrar en los flujos de trabajo mecanismos que permitan la constante actualización de patrones, ajustando la definición de comportamiento normal para múltiples temporalidades y detectando nuevas anomalías a lo largo del tiempo.

2.2 Justificación

Este trabajo se realiza principalmente para cubrir la necesidad de las corporaciones e instituciones de entender el uso y aprovechamiento de sus recursos tecnológicos alojados y accesibles en internet. Analizar los registros de tráfico web permite no solo entender la manera en que se consume la información que contiene un servidor, sino también detectar si el uso generalizado se transforma, o si existen anomalías e incluso calcular un parámetro de probabilidad de ser malintencionadas. Dado el volumen de información que se genera, y la creciente sensibilidad de los datos alojados, aplicar

herramientas de aprendizaje automático permitirá agilizar y perfeccionar cualquier proceso manual.

Además realizar este análisis de interacción usuario/servicio tiene impacto en múltiples contextos ya que al permitir conocer la cantidad de recursos necesarios para cubrir la demanda de servicio se puede estimar si la infraestructura provista es excedente o insuficiente, en ambos casos generando consecuencias negativas; en el caso de tener una infraestructura insuficiente, se afecta directamente la calidad del servicio, resultando en tiempos de espera largos, caídas de servicio o incluso pérdida de información. Por otra parte si la infraestructura es excedente significa que se están asignando recursos que realmente no se aprovechan, con repercusiones monetarias directas.

Finalmente no dar seguimiento a los archivos de registro, expone los servidores a sufrir ataques informáticos, lo cuál puede comprometer la capacidad del sistema para proteger la información personal o confidencial alojada, y vulnerar el control de contenido ejecutado en el servidor potencialmente volviéndolo parte de un botnet (véase Capítulo 3.4.1 para referencia)

2.3 Hipótesis.

Se plantea la hipótesis de que el tráfico presente patrones claros y definidos, lo cuál permitiría comparar el tráfico actual con solicitudes históricas, y calcular su grado de diferencia.

Estos patrones se espera que evolucionen según factores cíclicos externos como por ejemplo temporada de vacaciones, fines de semana, temporadas festivas; evolución del producto como nuevas implementaciones y endpoints, los cuales permiten a clientes comunicarse e interactuar con el servidor o cambios sociales (surgimiento y decadencia de tendencias).

Dada la hipótesis de que existen dichos patrones y que evolucionan con el tiempo, se estima que las variables independientes de una, así como un grupo de conexiones al servidor, serán capaz de predecir el patrón correspondiente esperado. De esta manera se podrá cuantificar el grado de diferencia entre el comportamiento del patrón esperado y el comportamiento real.

Cuantificar esta diferencia es fundamental para determinar el grado de anomalía del tráfico, finalmente una vez que se obtiene el grado de anormalidad se puede determinar un límite (threshold) que clasifique y determine si el tráfico es o no anómalo.

Capítulo 3 - Marco teórico

Para lograr un análisis de anomalías relevante y adaptable, es necesario aplicar técnicas más allá de la estadística tradicional, ya que el volumen es un factor determinante así como la constante modificación y actualización en los patrones de uso e interacción con los servidores.

“Las dificultades enfrentadas por sistemas dependientes de conocimiento codificado estáticamente sugiere que los sistemas de inteligencia artificial tienen la necesidad de conseguir su propio conocimiento a través de extraer patrones de los datos puros. A esta capacidad se le conoce como aprendizaje máquina.” (Goodfellow, 2016)

3.1 Aprendizaje no supervisado.

El aprendizaje no supervisado hace referencia a técnicas y modelos que se aplican a datos en los que no se posee una etiqueta o un valor esperado explícito, en el caso de este estudio, los datos representan observaciones del archivo de registros de NGINX (véase Capítulo 1.5 para referencia), por lo que no cuentan con una etiqueta explícita que identifique si cada solicitud se considera tráfico anómalo o normal.

“Los modelos no supervisados se refieren a aquellos en los que la máquina o computadora debe aprender patrones a partir de información sin referirse a una respuesta dada. Es decir implica el uso de conjuntos de datos sin una clara percepción de la variable dependiente (grado de anomalía en cada respuesta). El aprendizaje no supervisado tiene como objetivo explorar la estructura de los datos y generar una hipótesis en lugar de probar cualquier hipótesis mediante métodos estadísticos o construir modelos de predicción o clasificación basado en un conjunto de condiciones. Los algoritmos para el aprendizaje no supervisado se pueden subdividir en dos categorías: algoritmos de agrupamiento o transformaciones de datos informativos.” (Valkenborg, 2023)

3.1.1 Redes neuronales

Uno de los métodos más comunes para realizar aprendizaje no supervisado consiste en aplicar redes neuronales las cuales son programas o modelos de aprendizaje máquina inspiradas en el cerebro humano ya que usan flujos de información similares a las neuronas biológicas trabajando juntas para identificar fenómenos, evaluar opciones y llegar a conclusiones.

“Las redes neuronales agrupan sus neuronas artificiales en capas, siempre contando con una capa de entrada, cierto número de capas ocultas (internas) y una capa de salida. Cada nodo o neurona se conecta a los demás, y tiene su propia ponderación y umbral asociado, cada neurona sólo se activará y enviará datos a la siguiente capa de la red si la salida está por encima de un umbral especificado.” (IBM, 2024a)

Una red neuronal entrenada y perfeccionada ofrece una herramienta poderosa para clasificar y agrupar datos a gran velocidad.

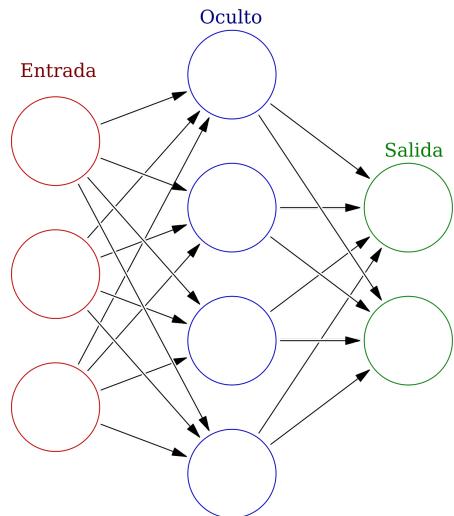


Figura 11: Arquitectura de red neuronal con una capa oculta

Fuente: Cburnett, 2019

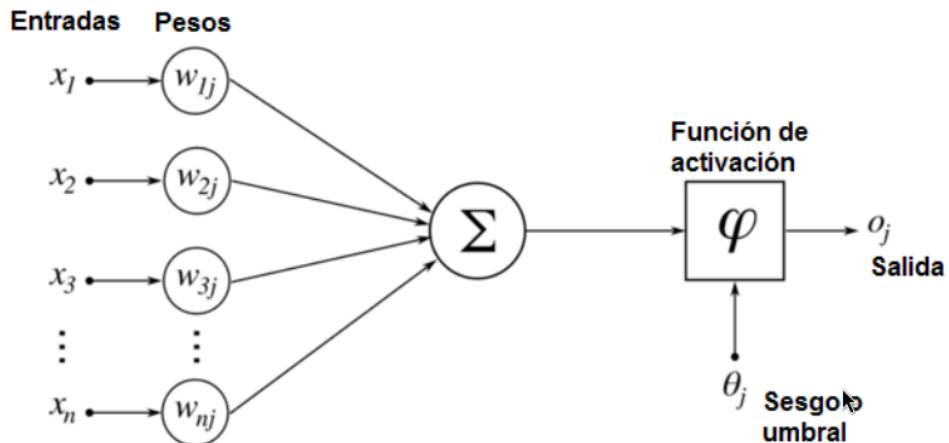


Figura 12: Esquema de una neurona en una red neuronal

Fuente: Molero, 2015

Las redes neuronales se han aplicado exitosamente en tareas de detección de anomalías en arquitecturas multi clase así como de clase única, las clases se refieren a la cantidad de categorías o posibles resultados para este estudio se consideran las clases dicotómicas tráfico normal o tráfico anómalo.

“Una técnica básica de detección de anomalías en arquitectura multi clase opera en dos pasos. Primero una red neuronal se entrena en la información normal de entrenamiento

para aprender las diferentes clases en condiciones normales. En segundo lugar, cada instancia de prueba (un registro en el archivo de registros de Nginx) se proporciona como una entrada a la red neuronal. Si la red acepta la entrada de prueba, es normal, y si la red rechaza la entrada es una anomalía.” (Stefano, 2000)

3.1.1.1 Autoencoders

Uno de las arquitecturas de red neuronal más empleadas para la identificación de anomalías es el autoencoder el cuál “es un tipo de algoritmo con el objetivo principal de aprender una representación ‘informativa’ de la información que pueda emplearse en otra aplicación a través de aprender a reconstruir los datos de entrada lo suficientemente bien” (Bank, 2021)

Los componentes principales de un autoencoder son; un codificador, la representación de característica latente, y un decodificador. El codificador y decodificador son simplemente funciones, mientras que la representación de característica latente es usualmente un tensor (una matriz numérica) de números reales.

La representación de característica latente, a pesar del nombre, se refiere a una transformación de los datos de entrada en la que se captura únicamente lo esencial y abstracto, por ejemplo cuando aprendemos el concepto de caballo, información como el color de pelaje, raza, postura, edad son irrelevantes, y solo recordamos características principales como la silueta, la forma de la cabeza, etc. De igual manera la representación latente, captura en forma de tensor que desecha las dimensiones y características no esenciales.

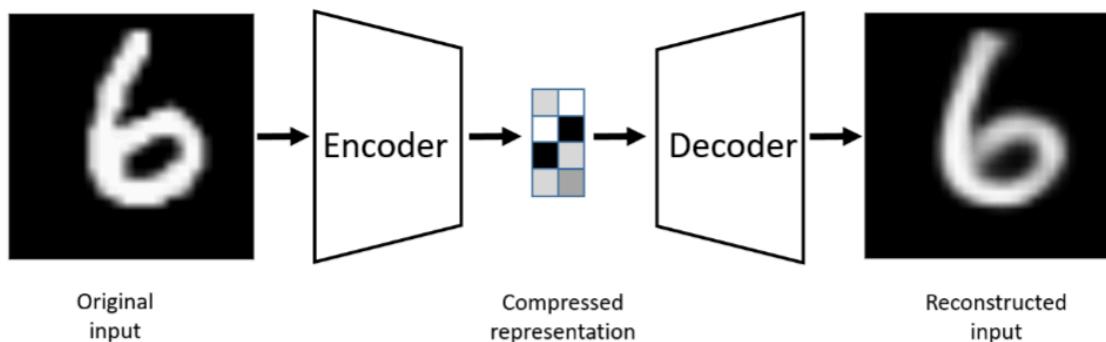


Figura 13: Autoencoder, mostrando observación, capa latente y resultado

Fuente: Bank, 2021

Los autoencoders pueden usarse para detectar anomalías ya que “su objetivo es aprender un perfil normal y después puede identificar las observaciones que no encajen con dicho perfil normal [...] esto sigue la suposición que el autoencoder entrenado aprenda del subespacio latente de observaciones normales. Una vez entrenado, tendrá

una baja cantidad de errores al reconstruir observaciones normales y una alta cantidad de errores al reconstruir anomalías" (Bank, 2021)

Una vez que el autoencoder ha finalizado su entrenamiento se debe definir el grado de tolerancia de errores en la reconstrucción para poder categorizar si el tráfico es anómalo o no.

3.1.1.2 Mapa Autoorganizado

Los mapas autoorganizados (también conocidos como mapas de Kohonen o SOM por sus siglas en inglés) se refieren a un tipo de red neuronal entrenada a través de algoritmos competitivos. "La propiedad central de los mapas autoorganizados es que crean proyecciones no lineales de un conjunto de datos de alta dimensión en un plano regular de baja dimensión (usualmente 2d). En esta representación los clusters de datos así como las relaciones métrico-topológicas de los datos son claramente visibles" (Kohonen, 2001)

Este tipo de red se usa para clusterizar y convertir datos multidimensionales en representaciones de baja dimensionalidad lo cuál permite reducir la complejidad de problemas y facilitar su interpretación.

"Las observaciones se agrupan de manera que tiendan a ser similares dentro del grupo" (IBM, 2024b)

La arquitectura de un mapa autoorganizado se conforma de dos capas, la capa de entrada y la capa de salida, como otras redes neuronales cada capa está compuesta de neuronas, todas las neuronas de la capa de entrada están conectadas con todas las neuronas de la capa de salida.

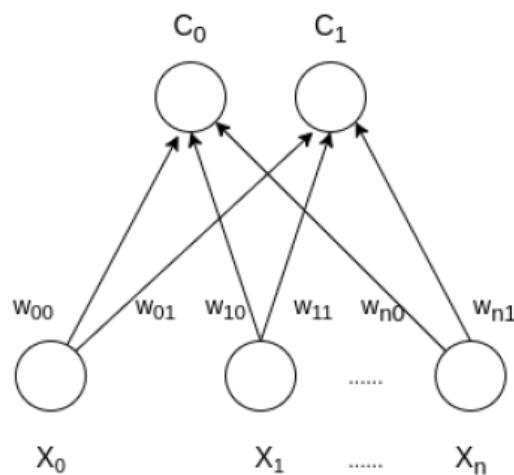


Figura 14: Arquitectura de mapa autoorganizado (abajo entradas, arriba salidas)

Fuente: Baliyan, 2022

"Durante el entrenamiento se distinguen dos etapas: la etapa competitiva y la etapa cooperativa. Durante la etapa competitiva se elige la neurona del mapa mejor adaptada a un dato de entrada y se asigna como 'ganadora' en la segunda etapa, los pesos de la neurona 'ganadora' se adaptan a las neuronas vecinas" (Van Hulle, 2012)

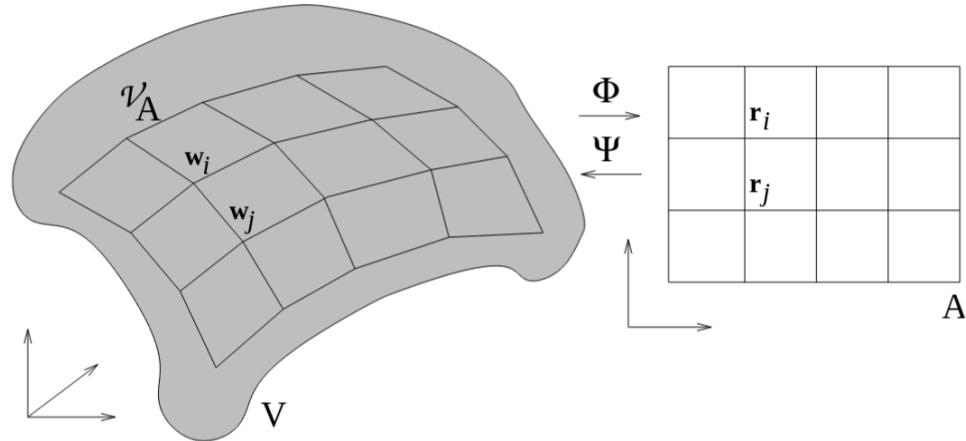


Figura 15: Izquierda representación multidimensional de datos, derecha representación en mapa kohonen

Fuente: Van Hulle, 2012

El resultado de este proceso es una representación de baja dimensión (usualmente un plano de dos dimensiones) en el que los datos similares (en este contexto correspondientes a tráfico normal) se agrupan y los datos anómalos se encuentran a distancias notables.

3.1.2 Clusterización

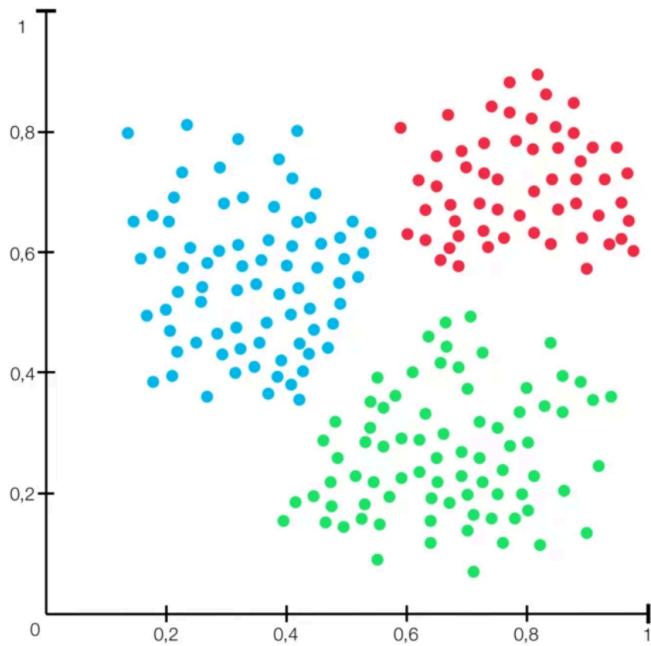


Figura 16: Ejemplo de clusterización de datos

Fuente: Marzell, 2021

La clusterización se emplea para agrupar observaciones similares en grupos. La clusterización es principalmente una técnica no supervisada en la que “el concepto de análisis basado en cercanía de vecinos ha sido empleado en varias técnicas de detección de anomalías, estas técnicas se basan en la suposición que las observaciones normales ocurren en grupos densos mientras que las anomalías ocurren lejos de estos grupos.” (Chandola, 2009)

La clusterización como técnica de detección de anomalías puede ser agrupada en tres categorías, las cuales se describen a continuación.

3.1.2.1 Observaciones anómalas no pertenecen a ningún cluster

Existen las técnicas basadas en la premisa de que las observaciones anómalas no pertenecen a ningún cluster, por lo que aplican algoritmos de clusterización caracterizados por qué no fuerzan a cada observación a pertenecer a un cluster por ejemplo DBSCAN (Ester, 1996).

Una desventaja de estas técnicas es que no están optimizadas para encontrar anomalías, ya que su objetivo principal es encontrar clusters” (Chandola, 2009)

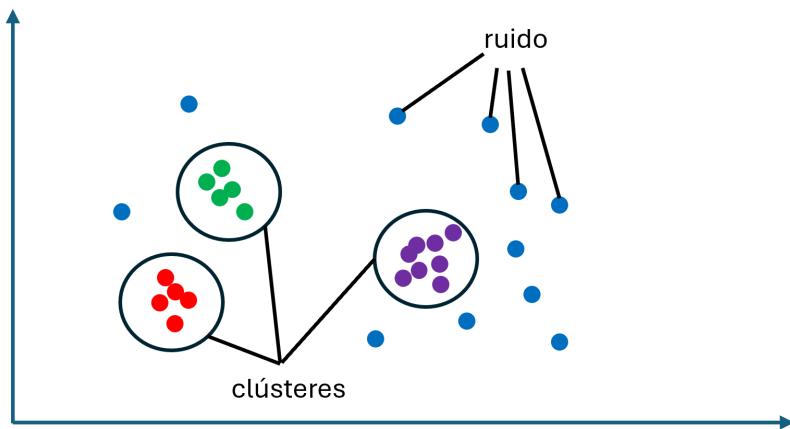


Figura 17: Ejemplo de clusterización usando DBSCAN

Fuente: Rueda, 2024

3.1.2.2 Observaciones normales se sitúan cerca del centroide del cluster, mientras las anomalías están lejos.

“Esta técnica consiste en dos pasos, el primero se encarga de clusterizar las observaciones [...] en el segundo paso, para cada observación se calcula su distancia con el centroide más cercano, esta medida se usa como escala para medir su anormalidad.” (Chandola, 2009)

Una desventaja de esta técnica es que si las anomalías son lo suficientemente numerosas y similares entre sí para formar un cluster, estas anomalías no serán detectadas como anomalías sino como otro grupo más de tráfico normal.

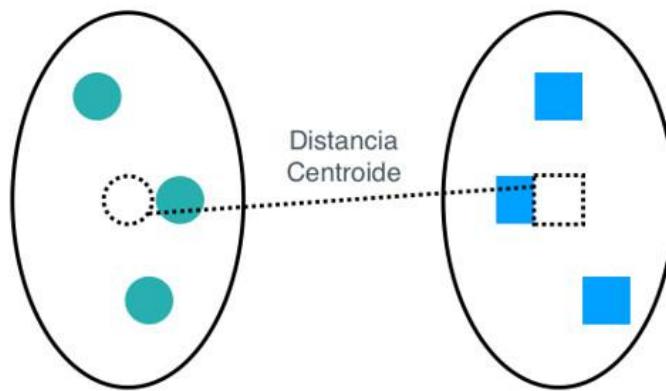


Figura 18: Ejemplo de clusterización resaltando centroides

Fuente: Gonzalez, 2020

3.1.2.3 Las observaciones normales forman clusters grandes y densos, mientras que las anomalías forman clusters pequeños o dispersos

Derivado de las limitaciones del grupo anterior, se crearon técnicas basadas en la densidad y tamaño del cluster.

“Técnicas basadas en este grupo declaran instancias pertenecientes al cluster basado en límites de tamaño o densidad.” (Chandola, 2009)

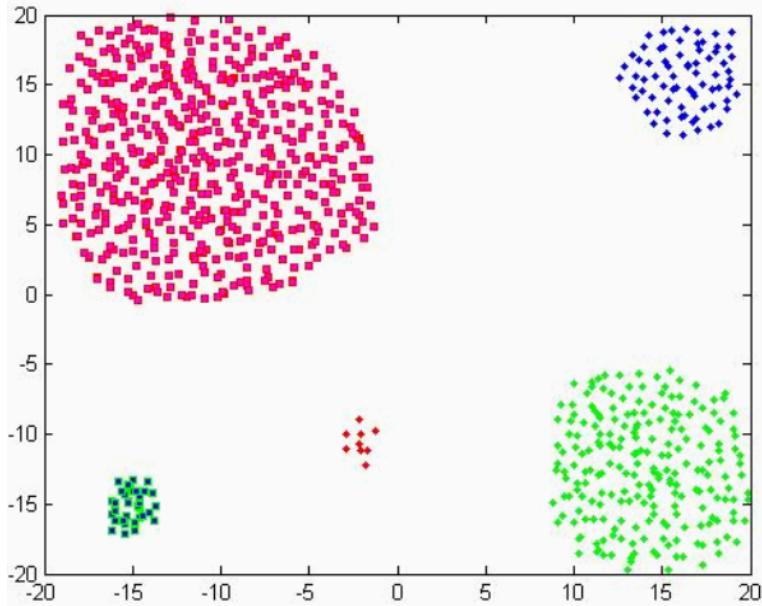


Figura 19: Ejemplo de clusterización resaltando clústeres dispersos

Fuente: Reddy, 2014

Si bien esta técnica puede ser mucho más precisa, computacionalmente es más demandante lo que puede afectar el tiempo y rendimiento para obtener resultados.

3.1.3 Bosque de aislamiento

El bosque de aislamiento es uno de los modelos que mejor se ajusta a la detección de anomalías en contextos donde se requiere rápida detección a través de una ejecución rápida. “La mayoría de los modelos existentes de detección de anomalías construyen un perfil de las instancias normales y después, identifican instancias que no encajan en este perfil como anomalías” (Liu, 2008)

“En este contexto el término ‘aislamiento’ se refiere a ‘separar una instancia del resto de las instancias’. Tomando en cuenta que las anomalías son ‘pocas y diferentes’ y por lo tanto más susceptibles a ser aisladas” (Liu, 2008)

La construcción “árbol de aislamiento” corresponde a un árbol binario que representa una secuencia de divisiones anidadas en un espacio de características, donde el nodo raíz corresponde al espacio completo (los datos en su totalidad) y cada nodo hijo corresponde a un subconjunto de los datos totales al cuál se le ha aplicado 1 división en alguna de sus características, por lo tanto la profundidad del árbol corresponde con la cantidad de cortes realizados. (Staerman, 2019)

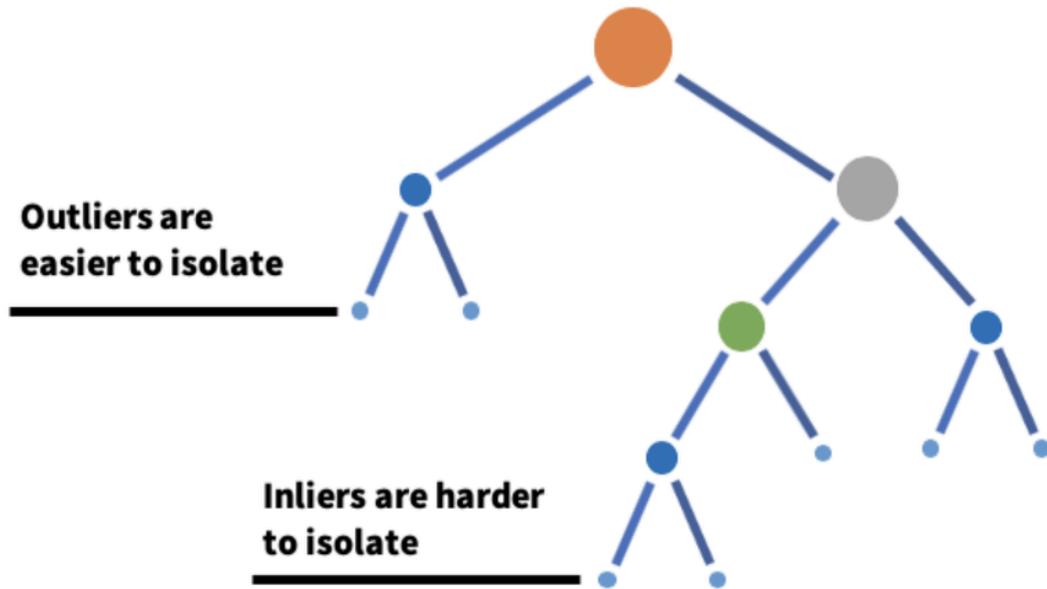


Figura 20: Ejemplo de árbol de aislamiento

Fuente: LinkedIn, 2019

“Las divisiones en el grupo de datos se realizan de manera aleatoria, y este proceso iterativo de división se realiza hasta que todas las instancias (datos) sean aisladas. Esta división aleatoria produce caminos (ramas) notablemente más cortas en las anomalías” (Liu, 2008)

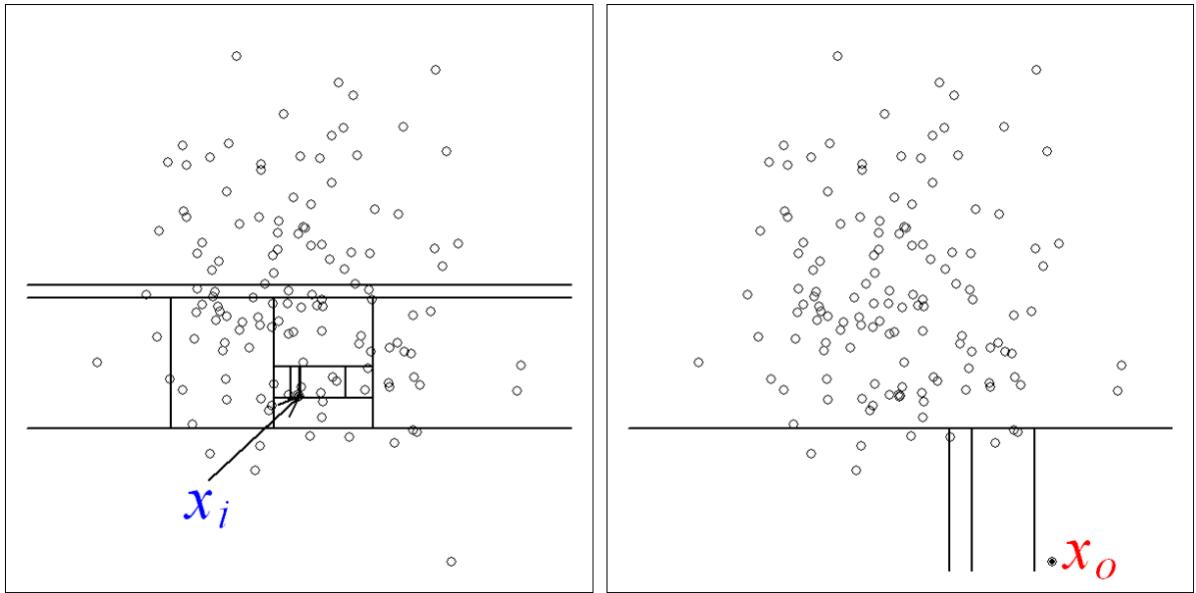


Figura 21: Comparativa de divisiones necesarias para aislar datos normales (izquierda) y datos anómalos (derecha)

Fuente: Liu, 2008

Para construir un bosque de aislamiento, se generan n cantidad de árboles de aislamiento para reducir el posible sobreajuste o subajuste de árboles específicos.

Cabe destacar la escalabilidad y performance del modelo ya que al no requerir medición de distancia o densidad, comparado con algunas de las técnicas de clusterización, para detectar anomalías, se reducen costos computacionales, además de tener una complejidad lineal de tiempo baja así como bajos requisitos de memoria. (Liu, 2008)

3.2 Regresión Logística

La regresión logística es un modelo estadístico perteneciente al grupo del aprendizaje supervisado, ya que contiene datos etiquetados (es decir una porción de los datos están clasificados) los cuales se usan para entrenar el modelo, una vez que el modelo abstrae las relaciones entre variables independientes, es capaz de clasificar datos no etiquetados.

“La regresión logística puede ser expresada como un método de mínimos cuadrados reponderados y regularizados iterativamente” (Karsmakers, 2007)

En el campo del aprendizaje automático “se usa para identificar la probabilidad de que ocurra un proceso o resultado en base a una serie de variables explicativas.” (Silva, 2023)

Una distinción importante entre la regresión lineal y la regresión logística es “que la salida o producto de la regresión logística es binaria o dicotómica” (Hosmer, 2013)

Dada esta distinción, se puede interpretar el resultado de la regresión logística como la probabilidad de que un evento pertenezca a cierto grupo, en la figura 22 en la gráfica referente a la regresión logística (derecha) se puede dividir los resultados en dos grupos (rojo y negro), entonces, la regresión logística nos daría por cada punto en el eje x (variable independiente) una probabilidad (valor del eje y) de pertenecer al grupo negro.

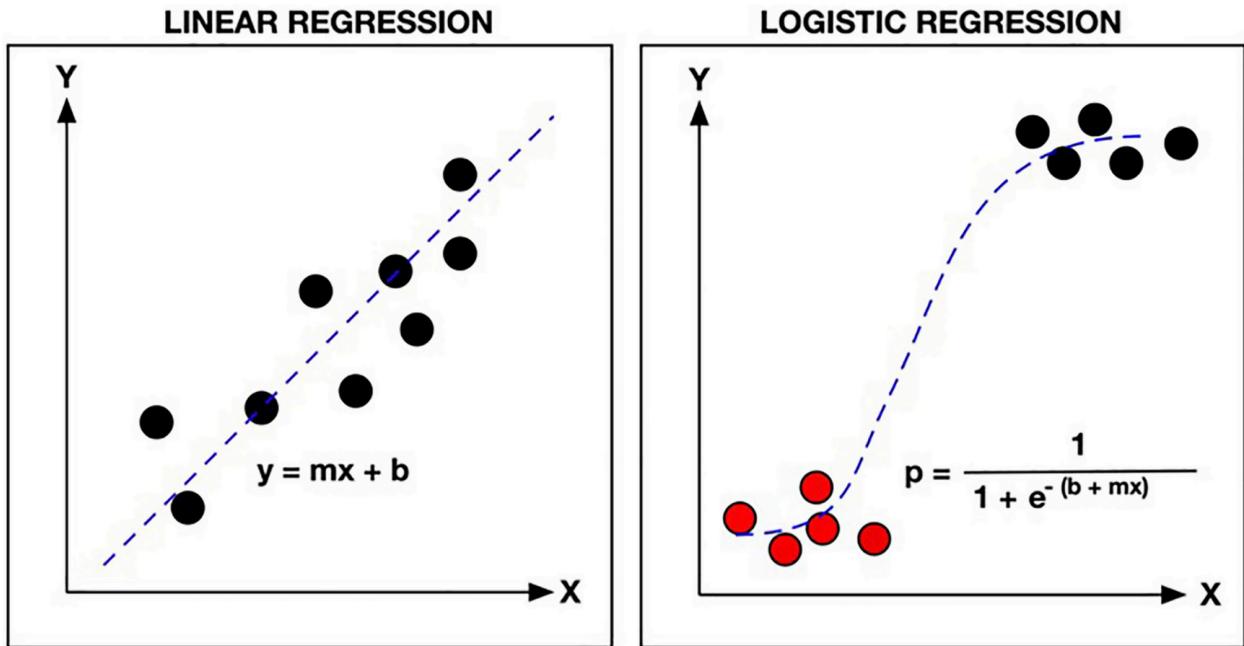


Figura 22: Gráfica y fórmula de la regresión lineal y logística

Fuente: Rashidi, 2019

Como ya se mencionó la regresión logística permite realizar tareas de clasificación (por ejemplo la fig. 22 clasifica los puntos en las categorías rojo y negro), pero también existen la variante de clasificación multinomial, en el que existen más de dos posibles categorías. Donde se realizan evaluaciones para cada categoría contra el resto de categorías (uno contra el resto).

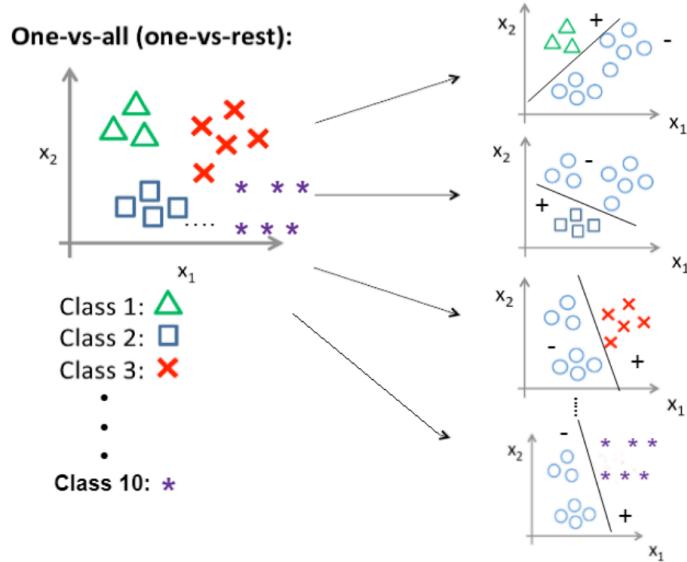


Figura 23: Ejemplo de regresión multinomial

Fuente: Flohil (2018)

3.3 Procesamiento del lenguaje natural

“Las técnicas de procesamiento de lenguaje natural (NLP por sus siglas en inglés) comenzaron en 1950 con la intersección entre la inteligencia artificial y la lingüística. Originalmente el procesamiento de lenguaje natural era distinto de la recuperación de información en textos (IR por sus siglas en inglés) el cuál emplea técnicas estadísticas altamente escalables para indizar y buscar grandes volúmenes de información en textos de manera eficiente. [...] Con el tiempo el lenguaje de procesamiento natural y la recuperación de información en textos han convergido.” (Nadkarni, 2011)

Las técnicas de procesamiento de lenguaje natural permiten extraer información de textos a través de modelos y métodos computacionales, “el ‘lenguaje natural’ se refiere a cualquier lenguaje, modo, género, etc. [...] El único requisito es que sea lenguaje empleado por humanos para comunicarse entre ellos. Además, el texto analizado no debe ser construido con el propósito específico de ser analizado, sino de un texto recolectado con fines de comunicación real.” (Liddy, 2001)

“Los recientes avances en inteligencia artificial han demostrado que los enfoques eficientes emplean las fortalezas de los circuitos electrónicos - alta velocidad y gran capacidad de memoria y almacenamiento, compresión de datos, búsqueda altamente eficiente en lugar de intentar imitar la función neuronal humana [...] Ha habido un incremento de herramientas de recuperación de información en textos a motores de bases de datos relacionales, paquetería estadística, herramientas de minería de datos,

los cuales tienen características como, disponibilidad, facilidad de uso, y gran valor respecto a costos" (Nadkarni, 2011)

De estas técnicas una relevante para esta investigación es la tokenización, "Después de importar un texto, usualmente el siguiente paso es convertir un texto entendible para humanos en tokens entendibles para máquinas. Los tokens se definen como segmentos de un texto en unidades identificables y con significado. Los tokens pueden consistir en palabras individuales o segmentos más grandes o pequeños, como fragmentos de palabras, secuencias de palabras, párrafos, oraciones o líneas" (Mullen, 2018)

Por ejemplo se puede tokenizar la siguiente oración por palabra, por dos palabras o por cuatro caracteres:

Input:

Lorem ipsum dolor sit amet

Output tokenizado por palabra:

Lorem	ipsum	dolor	sit	amet
-------	-------	-------	-----	------

Output tokenizado por dos palabras:

Lorem ipsum	ipsum dolor	dolor sit	sit amet
-------------	-------------	-----------	----------

Output tokenizado por cuatro caracteres:

Lore	orem	rem	em i	m ip	ips	ipsu	psum	sum	um d	...
------	------	-----	------	------	-----	------	------	-----	------	-----

"Un token es una instancia de una secuencia de caracteres en un documento en particular el cuál es agrupado para formar una unidad semántica útil para procesar." (Manning, 2009)

Una vez que se ha completado el proceso de tokenización usualmente estos tokens se someten a procesos de normalización como quitar palabras repetidas o palabras comunes de poco valor (stop words), o se realiza algún tipo de conversión de tokens a representaciones vectoriales numéricas como el atributo hash explicado a continuación.

3.3.1 Atributos hash

Los atributos hash (también conocidos como método hash) es una técnica para vectorizar atributos de manera eficiente y rápida, "de manera general el método hash usa una matriz de proyección dispersa aleatoria, convirtiendo de un plano de "n" dimensiones a otro plano de "m" dimensiones donde "m" es menor a "n", con el objetivo

de reducir la dimensionalidad de los datos intentando mantener la norma Euclíadiana” (Benjamin, 2018)

Es decir se refiere a una técnica de transformación de datos en cuyo principal objetivos son, con gran probabilidad, preservar las estructuras (distancias Euclidianas), y ser eficaz en la cantidad de memoria necesaria para dicha reducción de dimensionalidad.

Para ello emplea una función hash donde “cada término del vocabulario (llave o token) se transforma (hashea) en un número en un espacio lo suficientemente grande para que las colisiones sean poco probables” (Manning, 2009)

En este contexto una colisión se refiere a dos mensajes distintos que generen el mismo resultado (message digest) después de aplicarles la función hash

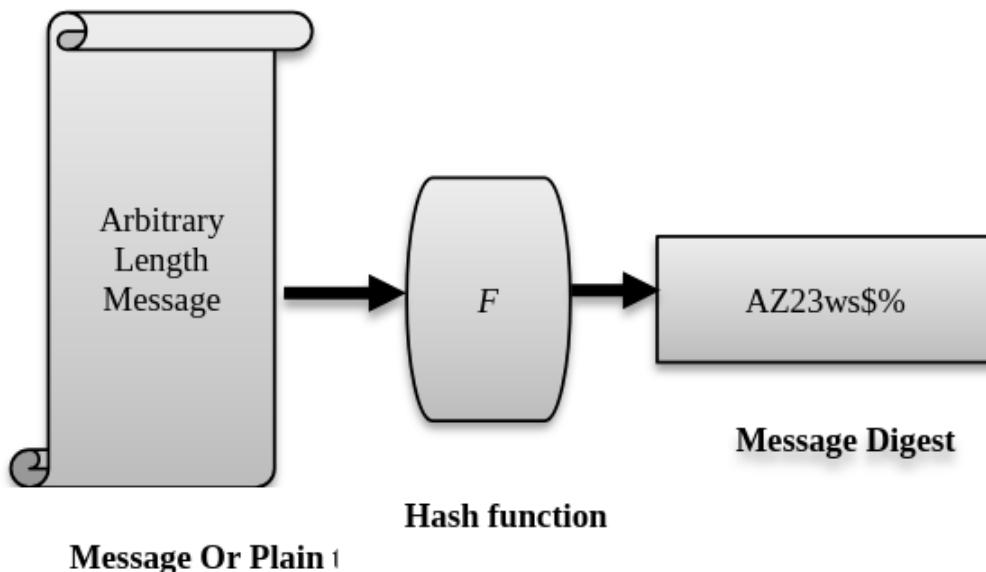


Figura 24: Funcionamiento general de una función hash

Fuente: Maetouq, 2018

Una vez que la información ha sido transformada a representaciones numéricas es fácil de integrar a modelos como redes neuronales o algoritmos de aprendizaje automático que analicen y aprendan las características y relaciones de estas representaciones numéricas.

3.4 Cómputo distribuído

El cómputo distribuido se refiere a un modelo computacional en el que las tareas se dividen entre múltiples agentes computacionales (nodos9 de procesamiento, mediante

este modelo, los requisitos de capacidad computacional de cada nodo se reduce entre mayor sea la cantidad de agentes computacionales disponibles para realizar tareas.

Un sistema distribuido es una colección de computadoras independientes que se presentan ante el usuario como una sola computadora y proveen una vista de sistema único. La coordinación entre estos equipos es lo que permite el acceso un gran poder computacional

A través del cómputo distribuido se intenta mejorar el desempeño de problemas de cómputo de gran escala mediante el intercambio de recursos. Además, el aumento de potencia computacional a bajo costo, aunado a los avances en telecomunicaciones y redes y la asequibilidad del big data, permiten nuevos paradigmas de computación distribuida. (Hajibaba, 2014)

“Se puede entender el cómputo distribuido en términos de dividir una aplicación en agentes computacionales individuales que se pueden distribuir a través de una red de computadoras, las cuales trabajan juntas para completar tareas cooperativas” (Farley, 1998)

El cómputo distribuido permite resolver problemas que involucran grandes volúmenes de información u operaciones recurrentes ya que si bien cada agente computacional debe procesar solo una fracción del problema/datos el intercambio de información entre agentes ofrece una estrategia “dividir y vencer”.

“Una aplicación distribuida se construye sobre diversas capas. A bajo nivel, una red conecta un grupo de computadoras para que puedan comunicarse entre sí. Protocolos de red como TCP/IP permiten a las computadoras enviar información a través de la red, permitiendo la habilidad para empaquetar y direccionar datos para ser enviados a otra computadora. Las capas de alto nivel se definen aprovechando los protocolos de red, tales como directorios de servicios y protocolos de seguridad. Finalmente las aplicaciones distribuidas corren por encima de todas estas capas, empleando servicios intermedios y protocolos de red así como sistemas operativos para llevar a cabo tareas coordinadas a través de la red.” (Farley, 1998)

Una aplicación distribuida a su vez cuenta con diversos elementos, Farley destaca los siguientes:

Procesos: Un proceso describe la secuencia de pasos a ejecutar, además de contar con acceso a distintos recursos del computador (CPU, dispositivos entrada/salida, memoria, almacenamiento, etc) a través del sistema operativo.

Hilos: Cada proceso debe contar al menos con un hilo, pero un proceso puede generar tantos hilos como sea necesario. Cada hilo es independiente aunque existen mecanismos de sincronización entre ellos, lo que les permite coordinarse e intercambiar información.

Objetos: Un objeto es un grupo de datos o información, con métodos que permiten invocar acciones ya sea para modificar los datos existentes o para realizar una tarea basada en dichos datos. Un proceso puede contener muchos objetos, y estos objetos pueden ser accedidos por los hilos dentro del proceso.

Agentes: Un agente es una manera general de referirse a elementos funcionales dentro de una aplicación distribuida, representa un componente abstracto a alto nivel el cuál se define alrededor de una funcionalidad particular, una utilidad, o un rol en el sistema general. Por ejemplo, un sistema bancario puede tener el agente cliente, el agente transaccional, etc. Los agentes pueden abarcar múltiples procesos, objetos e hilos.

Entre las ventajas que ofrecen los sistemas distribuidos destaca Garg los siguientes:

- Escalabilidad: Un sistema distribuido es inherentemente más escalable que un sistema paralelo, ya que la memoria compartida se convierte en un cuello de botella a medida que aumenta el número de procesadores.
- Modularidad y heterogeneidad: Un sistema distribuido ofrece mayor flexibilidad al permitir agregar o quitar procesadores de manera sencilla. Incluso, dicho procesador puede ser de un tipo completamente distinto al resto sin presentar mayor problemática.
- Intercambio de datos: Los sistemas distribuidos ofrecen intercambio de datos, lo que permite a múltiples organizaciones compartir información.
- Intercambio de recursos: Un mismo recurso (por ejemplo un procesador especializado) puede compartirse entre múltiples organizaciones y emplearse en distintas aplicaciones.
- Estructura geográfica: La estructura de una aplicación puede ser inherentemente distribuida, más se debe considerar factores como ancho de banda lo cuál puede favorecer procesamiento local.
- Confiabilidad: Los sistemas distribuidos cuentan con mejor confiabilidad ya que el fallo de una computadora no afecta la disponibilidad de otras.
- Bajo costo: El uso de equipos de cómputo baratos, así como la disponibilidad de alto ancho de banda favorece a los sistemas distribuidos en razones económicas.

3.4.1 Apache Spark

“Apache Spark es actualmente uno de los sistemas más populares para el procesamiento de datos a gran escala [...] Apache Spark es un motor de cómputo unificado y un conjunto de librerías para procesamiento de datos en paralelo en clusters de computadoras. Spark sport una amplia variedad de lenguajes de programación (Python, Java, Scala, y R), además incluye librerías para diversas tareas que van desde SQL (Lenguaje de Consulta Estructurada), hasta streaming y aprendizaje automático, y corre en cualquier entorno, desde una laptop hasta un cluster de miles de servidores.

“Esto convierte Apache Spark en un sistema sencillo de aprender y escalar” (Chambers, 2018)

“Vivimos en un mundo donde recolectar información es extremadamente barato -al grado de que muchas organizaciones consideran una negligencia no almacenar bitácoras de registros con posible relevancia para el negocio- pero procesarlo requiere gran poder computacional paralelo, usualmente realizado en clusters de máquinas” (Chambers, 2018)

Spark ofrece un catálogo de librerías que extienden la capacidad del framework, actualmente (versión 3.5.3) existen cuatro grandes librerías; Spark SQL para procesar datos estructurados mediante queries, Spark Streaming que permite procesamiento de datos en stream permitiendo gran escalabilidad, alto rendimiento y tolerancia a fallos, permitiendo procesamiento de fuentes como Kafka, Kinesis o sockets TCP (Figura 25), Spark MLlib que ofrece algoritmos de aprendizaje automático escalables y de implementación sencilla, y finalmente GraphX la más reciente librería de Spark para soportar cómputo de gráficos en paralelo.



Figura 25: Arquitectura de Apache Spark en procesamiento de streams.

Fuente: Apache Spark, 2024a

“Las aplicaciones de spark se ejecutan como un conjunto de procesos independientes en un cluster, coordinados a través del objeto ‘SparkContext’ en el programa principal (denominado el programa ‘driver’).

Para correr un cluster, el ‘SparkContext’ se conecta con un administrador de cluster (cluster manager) el cuál puede ser el nativo de Spark, Mesos, YARN o Kubernetes, el cuál provee recursos para la aplicación. Una vez que el programa ‘driver’ se conecta con el administrador del cluster, Spark adquiere ejecutores (executors) en cada nodo

del cluster, los cuales son procesos que corren las operaciones y almacenan información y resultados de la aplicación. Finalmente ‘SparkContext’ envía tareas a los ejecutores para correr.” (Apache Spark, 2024b)

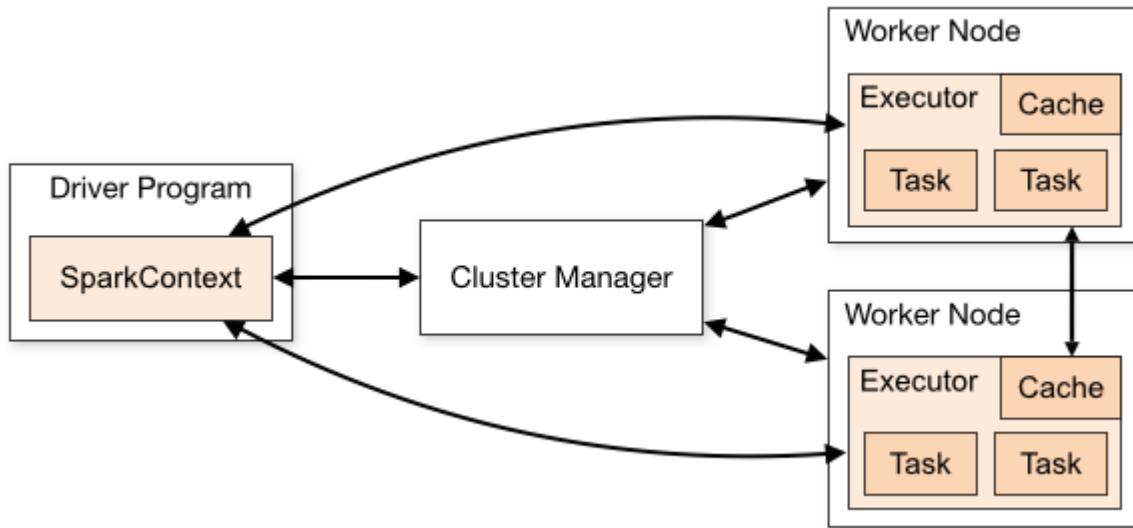


Figura 26: Arquitectura de Apache Spark en modo cluster

Fuente: Apache Spark, 2024b

Es importante notar algunas especificaciones de esta arquitectura.

Cada aplicación tiene su propio ejecutor, el cual existe durante la ejecución completa de la aplicación y a su vez levanta múltiples hilos. Esto es beneficioso para aislar aplicaciones entre sí, pero a la vez tiene el efecto de que los datos no pueden ser compartidos entre múltiples aplicaciones.

Spark es independiente de cualquier administrador de cluster, y solo lo requiere para obtener procesos del ejecutor y comunicarse.

El programa ‘driver’ debe ser capaz de recibir y aceptar conexiones de ejecutores por lo que el programa ‘driver’ debe ser alcanzable desde la perspectiva de redes por los nodos ‘worker’.

Debido a que el programa ‘driver’ orquesta tareas en el cluster, este debe correr cerca de los nodos ‘worker’, preferentemente en la misma red de área local. (Apache Spark, 2024b)

3.5 Estudios relacionados

La expansión de servicios de internet su rápida adopción global, así como el incremento de dispositivos con capacidad de interconexión, ha permitido que más personas se conecten, esto aunado al progresivo aumento de asequibilidad a equipos y servicios de cómputo ha desembocado en diversos estudios centrados exclusivamente en monitorear y detectar el uso y abuso de servicios en internet.

Para este trabajo se tomaron como referencia técnicas, métodos y procesos provenientes de dos investigaciones las cuales se describen a continuación.

3.5.1 Arquitectura Distribuida para la Predicción de DDoS y Detección de Bots / A Distributed Architecture for DDoS Prediction and Bot Detection

Este estudio se centra específicamente en la detección pronta de ataques distribuidos de denegación de servicio (DDoS por sus siglas en inglés) el cuál "software malicioso infecta diversos dispositivos, transformándolos en robots (bots) que operan en red (botnet), atacando usuarios, servicios y dispositivos" (Rahal, 2020)

Una vez que un dispositivo forma parte de una botnet, se encuentra a la espera de instrucciones de un atacante remoto. "Las botnets son capaces de llevar a cabo ataques mucho más potentes que el que haría una máquina sola. El objetivo de estos actores maliciosos es vulnerar la confidencialidad e integridad de datos y/o comprometer la disponibilidad del servicio atacado." (Rahal, 2020)

El objetivo de este estudio es la detección temprana de un ataque distribuido de denegación de servicio, incluso antes de que este inicie, enfocado en la detección de etapas tempranas del ataque.

"Emplean una arquitectura jerárquica compuesta de dos niveles: local e internet. [...] En el nivel local, se identifican señales de un inminente ataque antes de que alcance etapas avanzadas. Estas señales se basan en indicadores como tasa de retorno, autocorrelación, coeficiente de varianza y asimetría [...] En el nivel de internet se emplea un sistema híbrido de detección de bots. El cuál emplea técnicas de clustering acompañado de procesamiento de señales." (Rahal, 2020)

"Para ambos niveles local e internet se captura el tráfico enviado y recibido por los dispositivos conectados y este se emplea para construir y analizar una serie temporal" (Rahal, 2020)

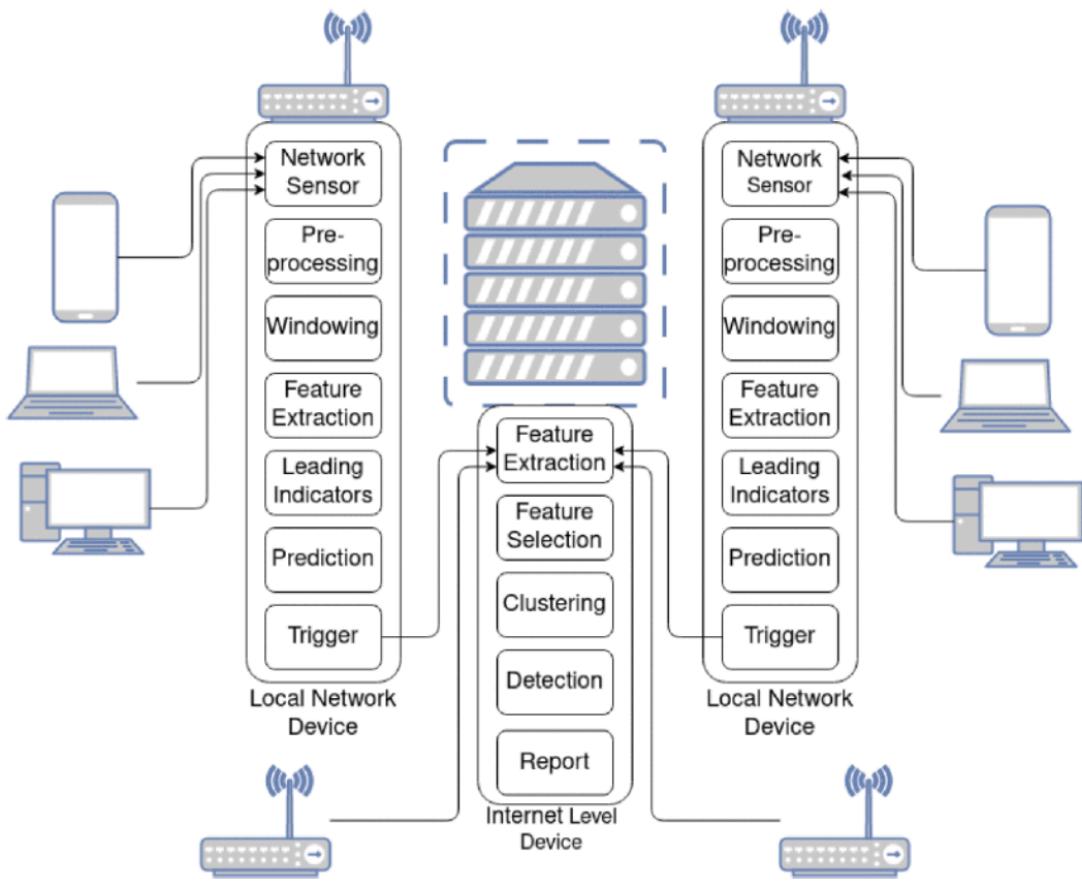


Figura 27: Arquitectura para predicción de ataques DDoS

Fuente: Rahal, 2020

A nivel conceptual, la solución presenta distintos módulos, los cuales se describen a continuación: “El módulo de ‘recolección de datos’ encargado de integrar sensores de red para capturar el tráfico y realizar preprocesamiento. [...] El segundo módulo ‘extracción de características’ lee los datos preprocesados y los divide en ventanas (ya sean ventanas por tiempo o por cantidad de paquetes). Este módulo realiza la extracción de características mediante agregaciones y conteos lo que permite obtener las características esenciales que representen el tráfico. El módulo de ‘procesamiento’ que realiza selección de características, clusterización, evaluación de tráfico. [...] Identifica las características más relevantes en el tráfico y genera conjuntos de datos listos para realizar detección de bots. El módulo de ‘análisis’ [...] es responsable de activar el análisis de detección cuando se predice un ataque, además permite afinar parámetros como el tamaño de ventana del módulo de extracción. Finalmente el módulo de ‘notificación’ realiza reportes con los resultados a administradores de sistema u otros sistemas automatizados en la red.” (Rahal, 2020)

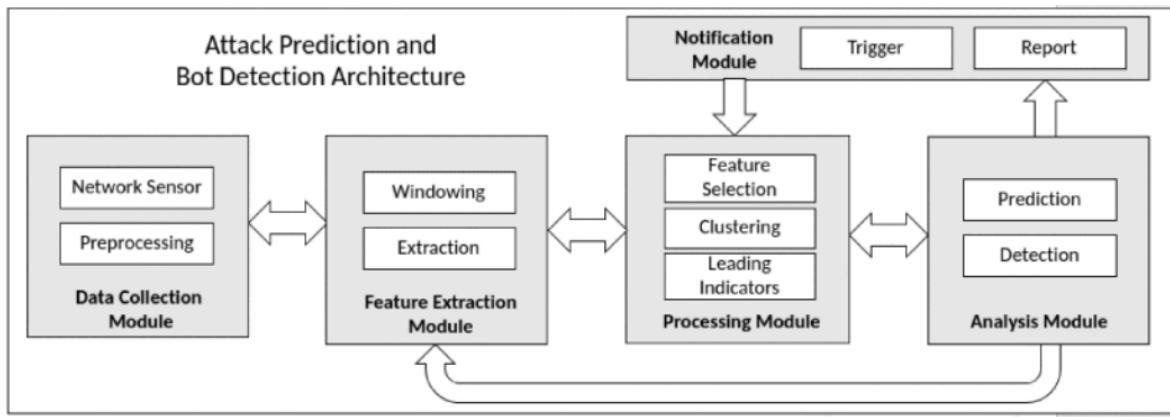


Figura 28: Interacción entre módulos para predicción de ataques DDoS

Fuente: Rahal, 2020

Cabe mencionar que existen discrepancias importantes entre este estudio y el llevado a cabo por Rahal et al 2020, ya que se operan con datos en distintos niveles del modelo OSI, el desarrollo y estudio de Rahal se encuentra en la tercera capa del medio, conocida como paquete a nivel de red, por ello tiene acceso a metadatos como:

Variable	Descripción
Cantidad de protocolos	Cantidad de protocolos que usa cada nodo.
TTL Promedio	Tiempo de vida promedio de los paquetes enviados.
Tamaño de ventana TCP	Tamaño de ventana para el protocolo TCP.
Porcentaje TCP	Porcentaje de paquetes que usan el protocolo TCP.
Porcentaje UDP	Porcentaje de paquetes que usan el protocolo UDP.
Porcentaje DNS	Porcentaje de paquetes que usan el protocolo DNS.
Porcentaje ICMP	Porcentaje de paquetes que usan el protocolo ICMP.
Porcentaje otros	Porcentaje de paquetes que usan otros protocolos.
Origen puertos privilegiados	Cantidad de puertos en la fuente que son privilegiados (< 1024).
Origen puertos no privilegiados	Cantidad de puertos en la fuente que no son privilegiados (>1024).
Cantidad puertos destino	Cantidad de puertos en el destino conectados por el nodo.
Tamaño de trama	Tamaño de trama en el paquete (se refiere a la capa 2 del modelo OSI).
Grado de entrada	Grafo referente al número de nodos que enviaron al menos un paquete a otros nodos.
Grado de salida	Grafo referente al número de nodos que han recibido al menos un paquete de un nodo dado.
Peso de grado de entrada	Similar al grado de entrada pero ponderado a la cantidad de paquetes enviados a otros nodos.
Peso de grado de salida	Similar al grado de salida pero ponderado a la cantidad de paquetes recibidos de un nodo dado.
Centralidad entre nodos	Grafo para cuantificar el número de veces que un nodo ha actuado como medio en la comunicación de dos nodos.
Coeficiente de clusterización local	Grafo para indicar el grado de concentración en un vecindario de nodos.
Centralidad de autovector	Grafo para identificar el grado de influencia de un nodo sobre otros, proporcionando un peso normalizado.

Tabla 4: Variables para predictor de ataques DDoS

Fuente: Adaptación de Rahal, 2020

Sin embargo a pesar de las discrepancias en los datos de origen este estudio se considera provechoso dadas las distintas técnicas empleadas en su arquitectura y conceptualización de módulos, incluyendo el diseño de una arquitectura multi-nivel, generar y usar grafos como variables, a pesar de que, como se comentó, opera sobre la capa tres del modelo osi, mientras que los datos de este estudio se encuentran en la quinta capa, correspondiente al manejo de sesión.

3.5.2 Análisis Empírico de Métodos para Detección de Anomalías en Series Temporales Multivariadas / An Empirical Analysis of Anomaly Detection Methods for Multivariate Time Series

Este estudio se centra en comparar de manera empírica distintos métodos de análisis de detección de anomalías en series de tiempo, aplicado a tráfico web similar al de este estudio, “Investigadores recientemente han propuesto buenos algoritmos para datos de series temporales multivariadas (MTS por sus siglas en inglés) para la detección de anomalías bajo distintas perspectivas. Cuándo son aplicados a escenarios reales, observamos que ningún algoritmo es adaptable a todos los escenarios debido a la complejidad de los datos y las características de las anomalías” (Shenglin, 2023)

Cómo precisión las series temporales multivariadas se refieren al estudio de fenómenos que ocurren en una temporalidad variable y que requieren de múltiples mediciones para captar todos sus atributos, el análisis de series temporales se refiere a un grupo de métodos que permiten extraer estadísticas significativas así como características de los datos.

“Con el desarrollo de internet, hemos enfrentado una explosión de información. Por ejemplo, un estudio de tasas de cáncer mensual en los Estados Unidos durante los últimos 10 años puede involucrar cientos o miles de series de tiempo, dependiendo de si investigamos por estados, ciudades o condados. Los métodos de análisis de series temporales multivariadas son necesarios para analizar de manera adecuada la información de este estudio, debido a que su análisis es diferente de la teoría estándar de estadística o los métodos basados en muestreo aleatorio que asume independencia. La dependencia es la naturaleza fundamental de las series temporales.” (William, 2019)

“En el estudio de procesos multivariados, se necesita un marco de trabajo para describir no únicamente las propiedades individuales de la serie sino también las posibles relaciones entre las mismas. El propósito de analizar y modelar las series en conjunto es para entender las relaciones y dinámicas a través del tiempo entre las series para mejorar la precisión en las predicciones.” (Reinsel, 2003)

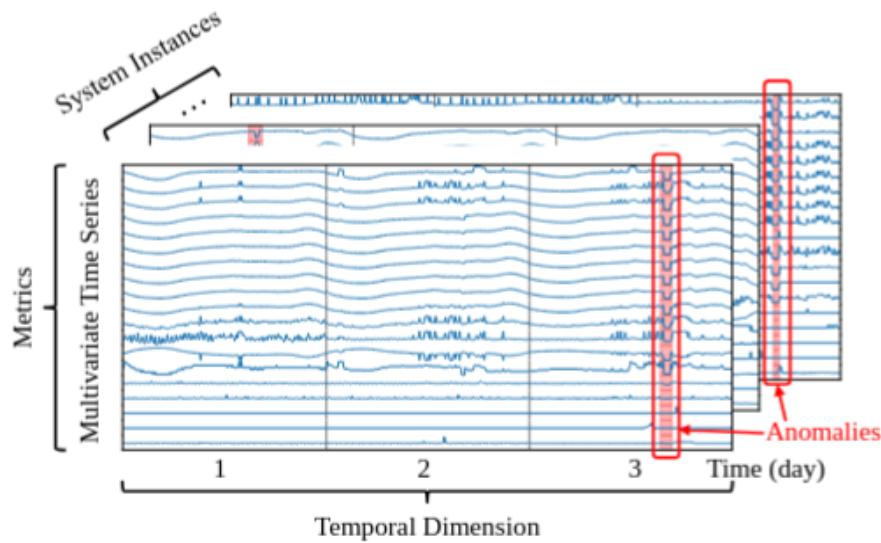


Figura 29: Ejemplo de serie temporal multivariada

Fuente:Shenglin, 2023

Para la selección de fuentes de datos decidieron tomar seis conjuntos de datos públicos y dos conjuntos (D1 y D2) de datos recabados de empresas asociadas. Los conjuntos de datos se muestran a continuación:

Conjunto de datos	Fuente	Escenario	Núm entidades	Núm métricas	Intervalo temporal	Porcentaje anomalías
D1	Un proveedor de contenido global	Servicios web	26	49	30 seg	0.05
D2	Un proveedor de servicio de internet	Operaciones de servicio de internet	107	22	15 min	0.02
SMD	Una empresa de internet	/	28	38	1 min	0.04
ASD	Una empresa de internet	/	12	19	5 min	0.05
SMAP	NASA	Mediciones de humedad en suelo y estado de congelación- descongelación	54	25	1 min	0.13
MSL	NASA	Operaciones de Curiosity, explorador de marte	27	55	1 min	0.11
SWaT	Una planta tratadora de agua	Operación y estatus de planta tratadora real	1	51	1 sec	0.12
WADI	Un banco de pruebas	Operación y estatus de una planta industrial	1	123	1 sec	0.06

Tabla 5: Información detallada sobre los conjuntos de datos empleados en el estudio

Fuente: Adaptación de Shenglin, 2023

Además los modelos empleados para el estudio fueron los siguientes:

Modelo	Ventajas	Método de procesamiento de información	Núm entidades	Núm métricas	Intervalo temporal	Porcentaje anomalías
DAGMM	- Basado en un punto de tiempo - Preserva las características de baja dimensionalidad y reconstrucción de errores para detección de anomalías	Usa estandarización	26	49	30 seg	0.05
USAD	- Toma las ventajas de un Autoencoder y entrenamiento adversarial - Emplea una estructura y modelo directo y un número limitado de parámetros	Usa estandarización	107	22	15 min	0.02
OmniAnomaly	- Modela la dependencia temporal explícita - Emplea Autoencoders variacionales para convertir observaciones de entrada en variables estocásticas	Usa ceros para completar valores faltantes y usa normalización	28	38	1 min	0.04
DOMI	- Simultáneamente extrae variables categóricas y características de baja dimensionalidad - Funciona mejor con series temporales multivariadas que presentan múltiples patrones normales	Usa ceros para completar valores faltantes y usa estandarización	12	19	5 min	0.05
SDFVAE	- Es capaz de explicitamente aprender representaciones variantes e invariantes en el tiempo	Usa normalización	54	25	1 min	0.13
InterFusion	-Emplea un Autoencoder variacional jerárquico para aprender diferentes características independientemente	Usa ceros para completar valores faltantes y usa estandarización	27	55	1 min	0.11
JumpStarter	- Clusteriza series temporales univariadas en series temporales multivariadas - Efectivamente reduce el tiempo de inicialización	Usa normalización	1	51	1 sec	0.12
GDN	- Emplea el modelo de mezcla gaussiana para aprender dependencias entre mediciones	Usa valores promedio o ceros para completar valores faltantes y usa normalización	1	123	1 sec	0.06

Tabla 6:Tabla comparativa de modelos no supervisados para series temporales multivariadas

Fuente: Adaptación de Shenglin, 2023

A continuación se presentan los resultados de cada modelo evaluando cada conjunto de datos, estos resultados corresponden únicamente a la evaluación **F1 score** (medida común para evaluar la precisión de un modelo de predicción basado en la relación entre la precisión y la exhaustividad), marcando en verde los mejores resultados y en rojo los peores.

	SMD	ASD	SMAP	MSL	SWaT	WADI
DAGMM	0.9492	0.8613	0.9098	0.9433	0.8663	0.6952
USAD	0.9038	0.9125	0.9812	0.9471	0.8336	0.4129
OmniAnomaly	0.9748	0.8751	0.9402	0.9202	0.6147	0.7101
DOMI	0.9141	0.5350	0.9299	0.9550	0.9422	0.1993
SDFVAE	0.9365	0.9087	0.9016	0.9365	0.9052	0.8851
InterFusion	0.9601	0.9101	0.9580	0.9611	0.8949	0.8999
JumpStarter	0.9233	0.7001	0.7540	0.8451	0.8694	0.8012
GDN	0.9494	0.8968	0.9380	0.9093	0.8463	0.9258

Tabla 7: Tabla comparativa de modelos y conjuntos de datos según la métrica F1 score

Fuente: Shenglin, 2023

“Nuestros hallazgos indican que los algoritmos de detección de anomalías usualmente enfrentan tres retos notables: 1) Ineficiencia para manejar series temporales multivariadas a gran escala. 2) Poca adaptabilidad para manejar series temporales multivariadas diversas. 3) Poca habilidad para detectar varios tipos de anomalías.” (Shenglin, 2023)

Este estudio es importante porque sienta un precedente comparativo sobre distintos algoritmos y modelos que es posible emplear para la detección de anomalías en series temporales multivariadas, análisis que se ajusta a las características de los datos con los que se cuenta. Además provee un resumen de contextos en los que ciertos modelos pueden ser preferidos sobre otros. Finalmente está implícito un listado de posibles modelos a implementar y probar.

3.6 Marco jurídico

Si bien existen distintas regulaciones con distintos niveles de castigo respecto a ataques ciberneticos, de manera generalizada se considera como un crimen “156 países (80%) ha promulgado legislación respecto al cibercrimen, el patrón varía por región: Europa tiene el nivel de adopción más alto (91%) y África el más bajo (72%)” (UNCTAD, 2024)

Incluso en los Estados Unidos, “servicios ofrecidos por actores maliciosos para atacar cualquier objetivo conectado a internet [...] no son de uso legítimo; son diseñados para saturar recursos de un servidor o exceder las capacidades de ancho de banda. [...]

Independientemente de si alguien lanza un ataques distribuidos de denegación de servicio usando su propia infraestructura o contrata un servicio de estres para conducir un ataque, la transmisión del programa, información, código o comando hacia una computadora protegida es ilegal y puede resultar en cargos criminales” (FBI, 2024)

Esto bajo la Ley de Fraude y Abuso Informático (18 U.S.C. § 1030) según la que puede haber una o múltiples de las siguientes consecuencias (USC, 2010):

- Incautación de ordenadores y otros dispositivos electrónicos
- Arresto y procesamiento penal
- Pena de prisión significativa
- Pena o multa

La situación de México en el panorama de ciberseguridad es preocupante, “el país se posiciona entre las naciones que más registra pérdidas provocadas por el cibercrimen, junto con Estados Unidos, Brasil y China. Además México fue uno de los cinco países de América Latina que, en 2018, recibió más ataques ransomware (infección de sistemas informáticos mediante virus que bloquean una computadora)” (Dominguez, 2023)

México ha realizado esfuerzos por regular y definir políticas y organismos enfocados en cuidar el ciberespacio “México cuenta con la Policía Cibernética, adscrita a la Dirección General Científica de la Guardia Nacional [...] además al Centro Nacional de Respuesta a Incidentes Cibernéticos (CERT-MX), encargado de vigilar virtualmente la infraestructura tecnológica de la nación” (Dominguez, 2023)

Sin embargo estos esfuerzos aún son insuficientes ya que según el Banco Interamericano de Desarrollo, “Centroamérica y México deberían centrarse en mejorar el despliegue de estándares de seguridad cibernética y controles técnicos, así como fomentar el desarrollo de un mercado de ciberseguridad” (BID, 2020)

“México no cuenta con una ley dedicada de delito cibernético, pero el artículo nº 211 del Código Penal prevé el delito informático. Sin embargo, estas disposiciones son limitadas y dejan varias lagunas, lo que dificulta la lucha contra el cibercrimen.” (BID, 2020)

“México detenta un claro rezago respecto a sus socios en la región [...] eso implica que el país tiene un menor nivel de compromiso con documentos como son la Agenda Global de Ciberseguridad, de la ITU, creada en 2004, que tiene como fin promover el compromiso de los países con la cooperación y el multilateralismo para la gobernanza global del ciberespacio” (Aguilar, 2024)

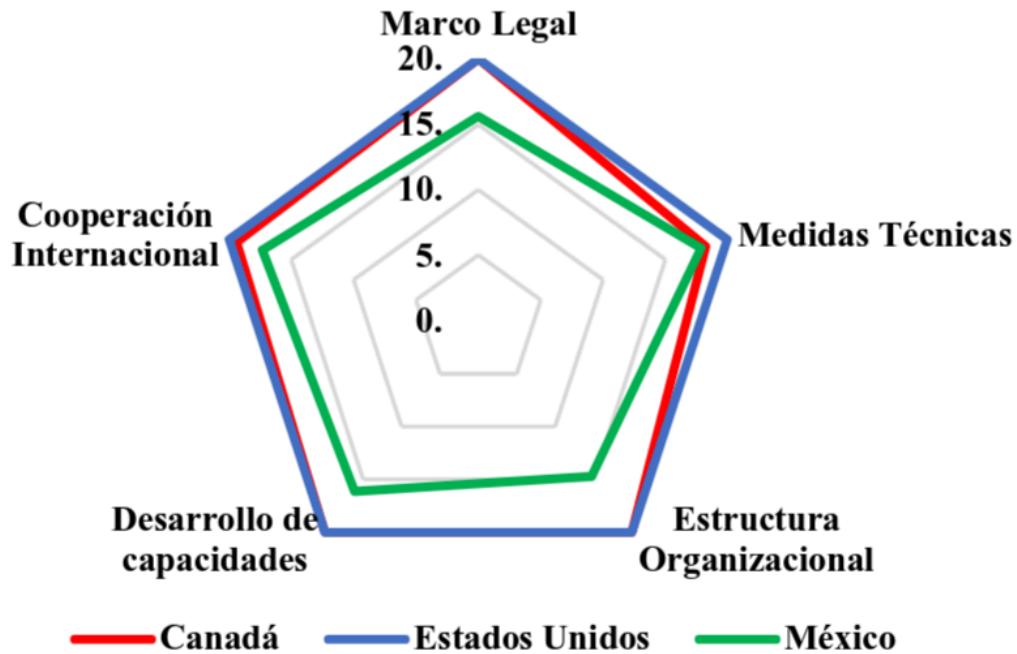


Figura 30: Comparativo de naciones de América del norte en el desarrollo de capacidades ciberneticas

Fuente: GCI, 2021

Bajo este contexto, resulta aún más relevante desarrollar herramientas como las que propone este estudio, que permitan proteger los activos e información sensible tanto de potenciales ciberataques como de cambios orgánicos en la demanda o uso de los servicios.

Capítulo 4 - Metodología

4.1 Introducción

Para la realización de este trabajo, se consideró la arquitectura, tecnología y especificaciones de la empresa **Designa**, la cuál ofrece desarrollo de software a la medida, alojamiento y administración de cómputo en la nube e implementación de tecnología eficiente e innovadora, derivado de ello se producen y alojan múltiples plataformas y sitios web.

4.1.1 Contexto estadístico y estructural de los datos

Para el contexto de este trabajo, se cuenta con acceso a un servidor web el cuál implementa un proxy inverso (véase Capítulo 1.4 para referencia) en NGINX el cuál recibe y redirige todo el tráfico que llega al servidor. A la vez llena una serie de archivos de registro, tanto para tráfico correctamente manejado (access.log) como tráfico que no se puede manejar (error.log), para este estudio se consideró únicamente los archivos access.log ya que representan una superficie más amplia para encontrar vulnerabilidades y ataques esto debido a que en el tráfico correctamente manejado es donde existe un intercambio de información y el servidor ofrece respuestas (que corresponden a distintos estados del servidor; véase Capítulo 1.3.2 para referencia).

Un elemento a considerar es el volumen de información, para este estudio se recolectó la información de tráfico de 72 días, en los cuales se captaron 8,170,910 de conexiones totales.

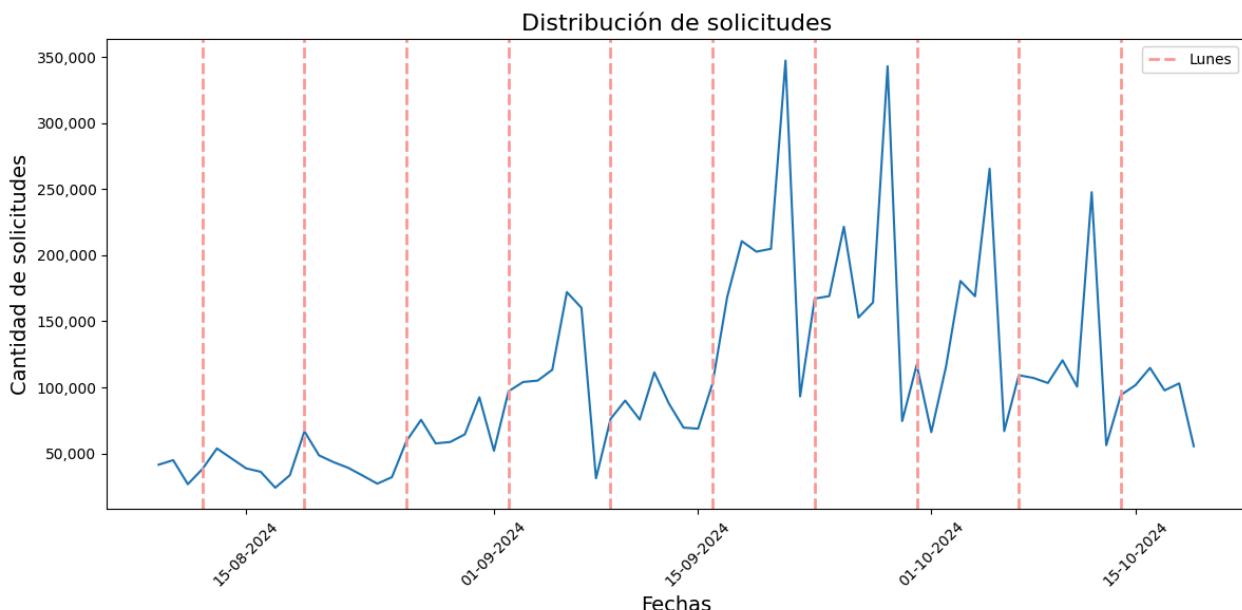


Figura 31: Línea de tiempo con total de solicitudes al servidor

Fuente elaboración propia

Cabe destacar que en la arquitectura estudiada se procesan múltiples dominios (sitios) desde un único servidor, nótese que NGINX se emplea a manera de reverse proxy (véase Capítulo 1.4 para referencia) con la distinción de que el proxy se encuentra dentro del mismo servidor, esta arquitectura permite maximizar el uso de recursos del servidor, minimizando costos de infraestructura.

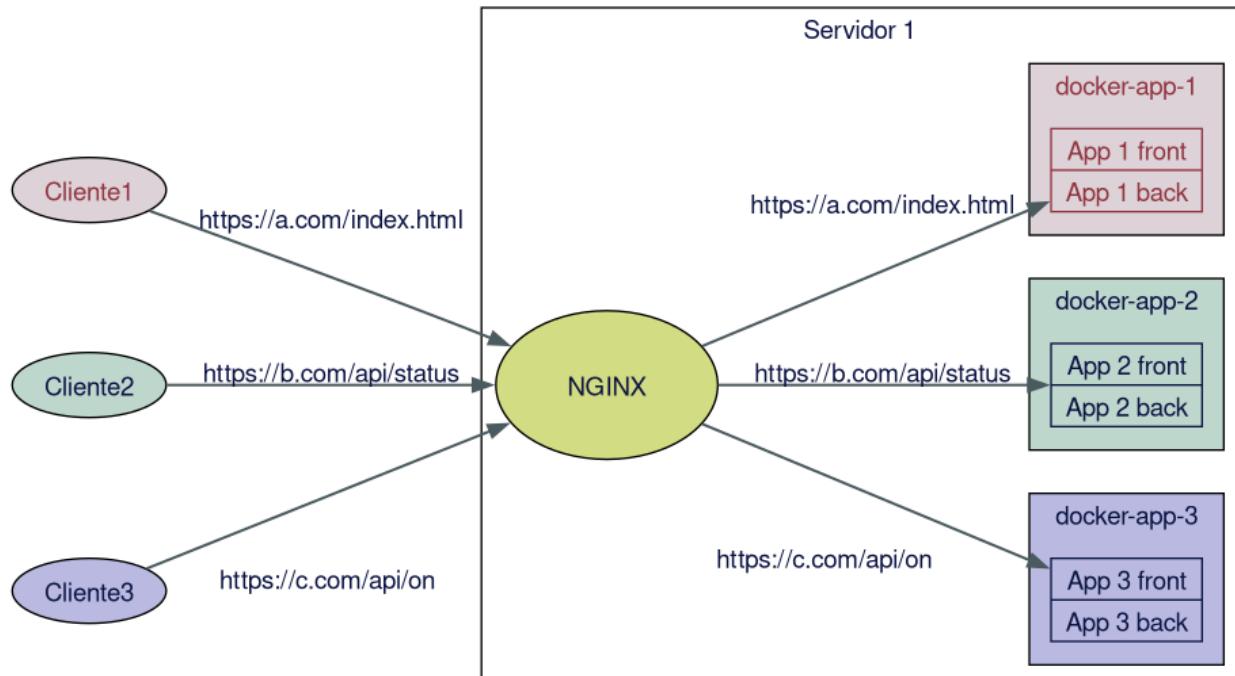


Figura 32: Arquitectura de nginx

Fuente elaboración propia

Una consecuencia importante de que nginx esté dentro del servidor significa que los archivos de registro contienen tráfico de todas las aplicaciones que el servidor aloja, usando la nomenclatura de la figura 32, los archivos de registro del servidor 1 contiene tráfico del sitio 1 (a.com), sitio 2 (b.com) y sitio 3 (c.com) combinado.

```

45.166.93.223 - - [23/Aug/2024:00:00:20 +0000] "GET /api/manual/find/?category=De%20todo%20un%20poco&searchIn=category&page=1&limit=1 2&search=%7B%22searchAllStatuses%22%3Atrue%2C%22searchParam%22%3A%22De%20todo%20un%20poco%22%7D" HTTP/1.1" 304 0 "https://a.com/manual/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"

201.141.19.215 - - [23/Aug/2024:00:01:05 +0000] "POST /api/v1/courses/11083/quizzes/331373/submissions/94734/events HTTP/1.1" 204 0 "https://b.com/courses/11083/quizzes/331373/take" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36"

177.225.161.41 - - [23/Aug/2024:00:01:10 +0000] "GET /profile HTTP/1.1" 302 105 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 17_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.5 Mobile/15E148 Safari/604.1"

18.212.63.99 - - [23/Aug/2024:00:27:05 +0000] "GET /vendor/phpunit/phpunit/LICENSE/eval-stdin.php HTTP/1.1" 502 173 "-" "Custom-AsyncHttpClient"

```

Código 2: Fragmento enmascarado de archivo de registro generado por NGINX

Fuente elaboración propia

Si analizamos el fragmento anterior es posible notar que existen el primer registro pertenece al dominio a.com, el segundo registro pertenece al dominio b.com en cambio los últimos dos no especifican a cuál dominio pertenecen, esto se debe a que el campo en cuestión es `http_referer`, (véase Capítulo 1.5.1 para referencia) este campo usualmente lo llena el navegador indicando de cuál url se generó la nueva petición, aterrizando la frase en acciones cotidianas, supongamos que se está navegando el sitio “a.com/inicio” y tiene un botón que nos lleva a “a.com/perfil”, cuando el navegador solicite “a.com/perfil” va a llenar automáticamente `http_referer` con el la url de la cuál venimos, en este caso “a.com/inicio”, en otras palabras “a.com/inicio” nos refirió a “a.com/perfil”. Pero `http_referer` puede no llenarse, como es el caso de los registros tres y cuatro del fragmento anterior, esto sucede por ejemplo si directamente en la barra de navegación escribimos “a.com/perfil” nadie nos refiere, o si la petición viene de un dispositivo que ya sabe la url (por ejemplo un reloj checador, un refrigerador inteligente o un bot).

Poniendo en contexto con los datos recabados, el servidor analizado durante el tiempo recolectado proceso solicitudes para 48 dominios distintos (identificados por las configuraciones del servidor), de los cuales 1,543,472 solicitudes no tenían `http_referer` válido y a los cuales me referiré como dominio “otro” durante este trabajo, a

continuación se muestra la distribución de peticiones por dominio.

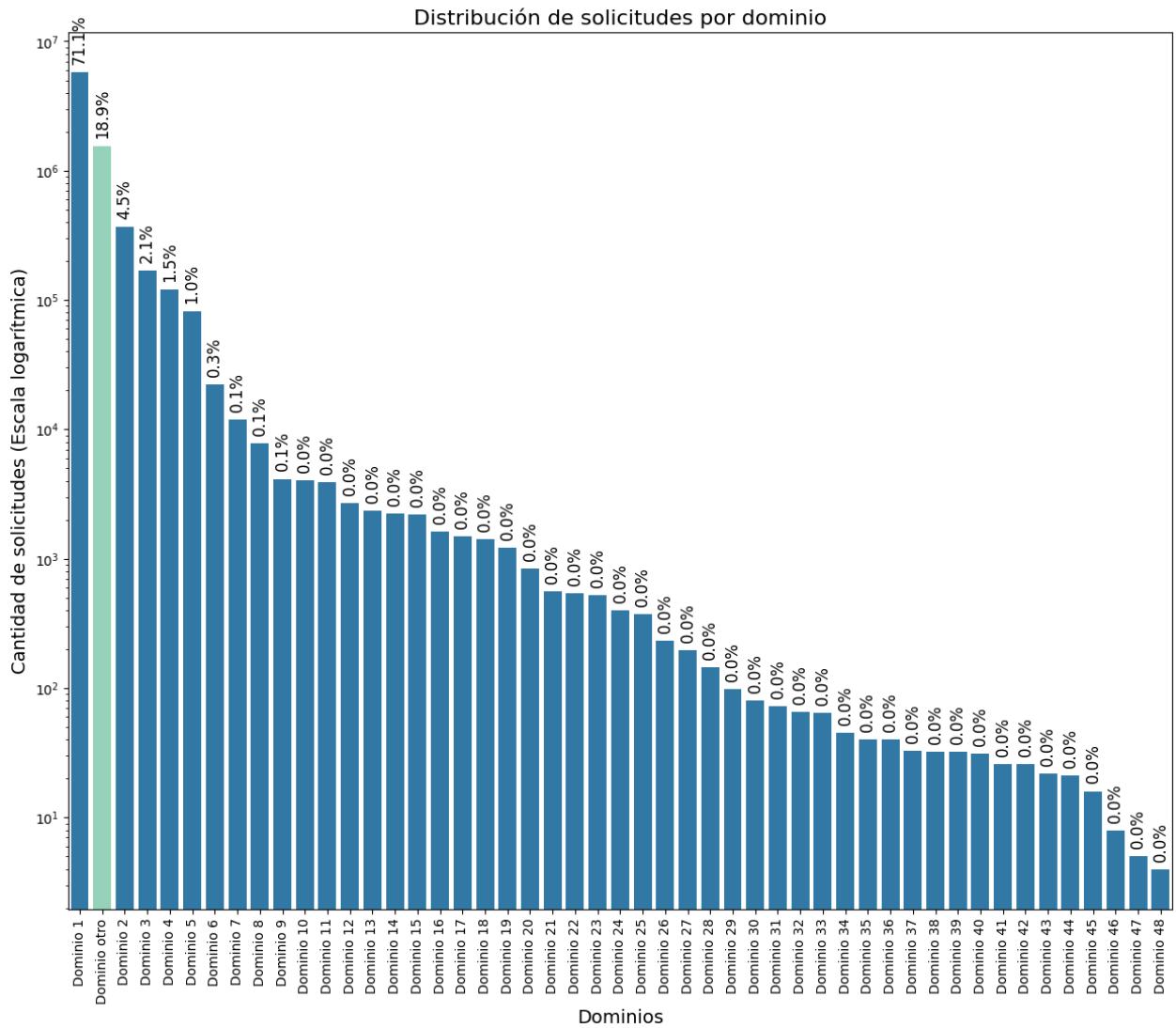


Figura 33: Distribución de solicitudes por dominio

Fuente elaboración propia

Nótese que en la figura 33, la cantidad de solicitudes (eje y) está en una escala logarítmica dada la significativa diferencia entre los valores máximos y mínimos, esto mismo explica el rápido descenso en valor porcentual de varios dominios. Además resaltado en verde se muestra el denominado “Dominio otro” (solicitudes sin http_referer válido) que corresponden a casi el 20% del tráfico capturado.

A continuación se desglosa el conteo y porcentaje detallado para cada dominio:

Dominio	Cantidad de solicitudes	Porcentaje	Dominio	Cantidad de solicitudes	Porcentaje
Dominio 1	5,812,121	71.13187	Dominio 25	372	0.00455
Dominio otro	1,543,472	18.88984	Dominio 26	234	0.00286
Dominio 2	370,235	4.53114	Dominio 27	196	0.00240
Dominio 3	168,335	2.06017	Dominio 28	145	0.00178
Dominio 4	121,492	1.48689	Dominio 29	98	0.00120
Dominio 5	81,461	0.99696	Dominio 30	80	0.00098
Dominio 6	22,140	0.27096	Dominio 31	72	0.00088
Dominio 7	11,953	0.14629	Dominio 32	66	0.00081
Dominio 8	7,803	0.09550	Dominio 33	65	0.00080
Dominio 9	4,148	0.05077	Dominio 34	45	0.00055
Dominio 10	4,038	0.04942	Dominio 35	40	0.00049
Dominio 11	3,891	0.04762	Dominio 36	40	0.00049
Dominio 12	2,681	0.03281	Dominio 37	33	0.00040
Dominio 13	2,348	0.02874	Dominio 38	32	0.00039
Dominio 14	2,235	0.02735	Dominio 39	32	0.00039
Dominio 15	2,214	0.02710	Dominio 40	31	0.00038
Dominio 16	1,615	0.01977	Dominio 41	26	0.00032
Dominio 17	1,500	0.01836	Dominio 42	26	0.00032
Dominio 18	1,430	0.01750	Dominio 43	22	0.00027
Dominio 19	1,215	0.01487	Dominio 44	21	0.00026
Dominio 20	842	0.01031	Dominio 45	16	0.00020
Dominio 21	561	0.00687	Dominio 46	8	0.00010
Dominio 22	545	0.00667	Dominio 47	5	0.00006
Dominio 23	526	0.00644	Dominio 48	4	0.00005
Dominio 24	400	0.00490			

Tabla 8: Conteo y porcentaje de tráfico por dominio según http_referer

Fuente elaboración propia

4.1.2 Arquitectura tecnológica empleada

La selección de tecnologías para llevar a cabo las operaciones de ciencia de datos (extracción, preprocesamiento, integración, modelado, analítica y visualización) se realizó considerando cuatro características fundamentales; escalabilidad, adaptabilidad de volumen de información, rendimiento y portabilidad, las cuales se buscó maximizar para aprovechar de manera óptima los recursos computacionales disponibles.

La escalabilidad, terminó que la Real Academia de la Lengua Española no incorpora en su diccionario, que ante una consulta específica formulada respondió de la siguiente manera sobre la posible utilización del término escalabilidad en español:

‘Término empleado en informática con el sentido de ‘capacidad de un sistema informático de adaptarse a un número de usuarios cada vez mayor, sin perder calidad en los servicios’’ (Buenadicha, 2013)

La adaptabilidad de volumen de información está relacionado con la escalabilidad, en el sentido de que ambos tienen como objetivo mantener un nivel de calidad, ya sea en la cantidad de procesos paralelos a realizar (escalabilidad) o la cantidad de información que se debe procesar (adaptabilidad de volumen).

El rendimiento se refiere al “grado en que un sistema o componente de software cumple sus objetivos de puntualidad.” (Smith, 2008)

Smith menciona la responsividad como una de las dimensiones en el rendimiento de software, donde “la responsividad es la capacidad de un sistema de alcanzar objetivos para tiempo de respuesta o de rendimiento. En los sistemas en tiempo real, la responsividad es la medida de que tan rápido responde el sistema a un evento, o el número de eventos que se pueden procesar en un momento determinado.” (Smith, 2008)

Principalmente se buscó poder paralelizar el cómputo, distribuyendo los datos entre múltiples instancias.

Bajo estas definiciones, se eligió el framework Apache Spark (véase Capítulo 3.4.1 para referencia) como base para las operaciones de procesamiento y transformación de la información, dada su naturaleza de cómputo distribuido (véase Capítulo 3.4 para referencia) la cuál permite incrementar el poder computacional agregando múltiples recursos relativamente baratos (escalamiento horizontal), lo cuál tiene un impacto directo en términos de escalabilidad (es fácil incrementar el poder de cómputo) y rendimiento (se paraleliza la carga de trabajo al agregar más procesadores). Respecto a la característica de adaptabilidad de volumen, Spark ofrece recomendaciones sobre buenas prácticas recayendo la responsabilidad mayoritariamente en el programador.

Finalmente para asegurar la portabilidad, se optó por emplear docker la cuál es “una plataforma abierta de desarrollo, despliegue y ejecución. Docker permite segregar las

aplicaciones de la infraestructura, lo que permite entregar software con mayor rapidez.” (Docker, 2024a)

Docker introdujo el término de “contenedor” el cuál se refiere a “ambientes aislados en donde se ejecutan procesos para cada uno de los componentes de una aplicación” (Docker, 2024b)

Se generó una arquitectura en que un host, tiene un contenedor de almacenamiento (MinIO) que permite acceso a datos para lectura y escritura distribuido, un contenedor de Apache Spark master (cumpliendo el rol de driver, véase Capítulo 3.4.1) el cuál orquesta y distribuye las tareas entre los contenedores Apache Spark workers como se ejemplifica en la Fig. 34.

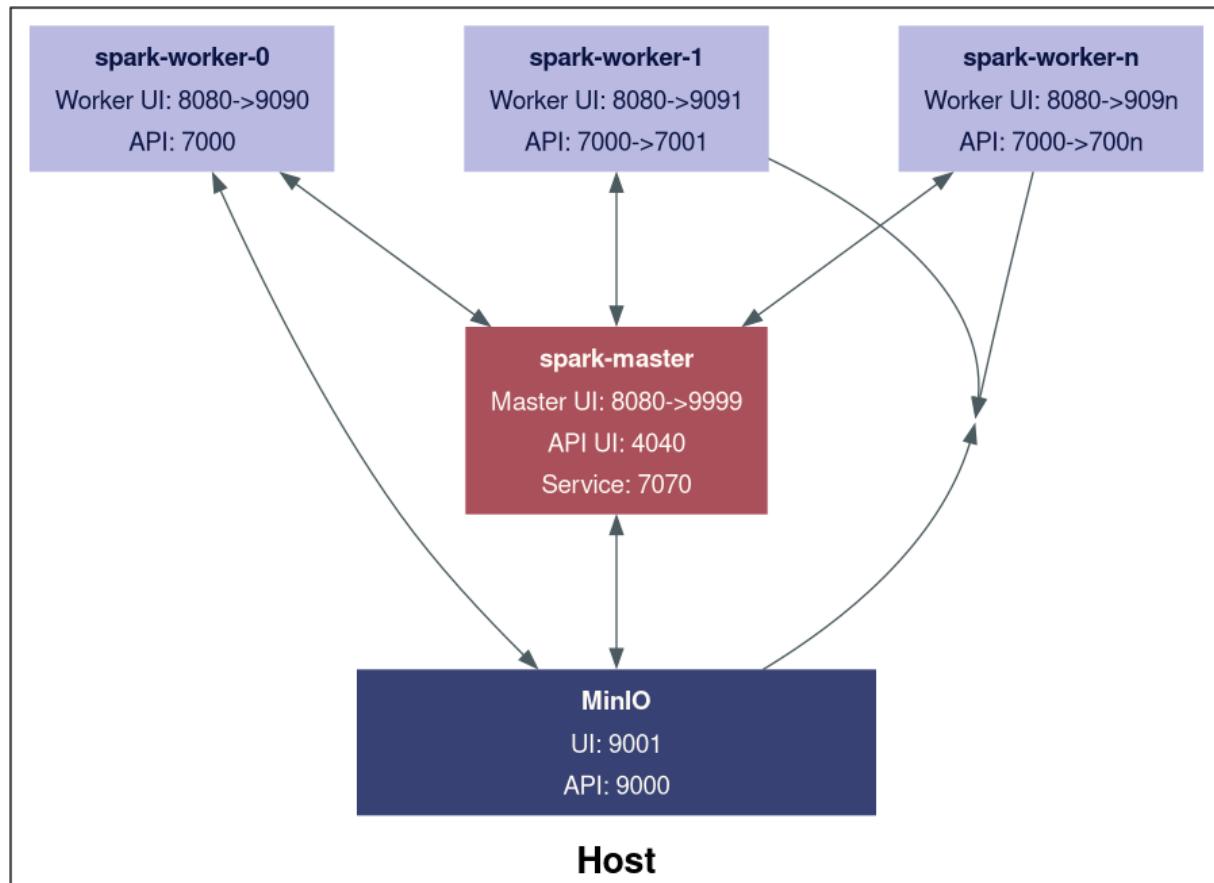


Figura 34: Componentes e interacciones entre contenedores

Fuente elaboración propia

Finalmente como precisiones técnicas, se crearon 7 instancias de spark workers, cada una con dos cores y 100 Gb de memoria RAM

APACHE Spark 3.5.3 Spark Master at spark://spark-master:7077

URL: spark://spark-master:7077

Alive Workers: 7

Cores in use: 14 Total, 0 Used

Memory in use: 700.0 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

▼ Workers (7)

Worker Id	Address	State	Cores	Memory	Resources
worker-20241128001807-10.89.3.10-7000	10.89.3.10:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	
worker-20241128001807-10.89.3.4-7000	10.89.3.4:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	
worker-20241128001807-10.89.3.5-7000	10.89.3.5:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	
worker-20241128001807-10.89.3.6-7000	10.89.3.6:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	
worker-20241128001807-10.89.3.7-7000	10.89.3.7:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	
worker-20241128001807-10.89.3.8-7000	10.89.3.8:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	
worker-20241128001807-10.89.3.9-7000	10.89.3.9:7000	ALIVE	2 (0 Used)	100.0 GiB (0.0 B Used)	

Figura 35: Despliegue de cluster de Apache Spark

Fuente elaboración propia

4.1.3 Descripción general del procesamiento desarrollado

En términos generales el flujo de procesamiento de sistema se puede resumir en las siguientes etapas y acciones:

1. Trasladar los archivos de registro del servidor web al cluster de procesamiento.
2. Estructurar los datos de texto semi-estructurados obtenidos de access.log
3. Entrenar y aplicar un modelo de regresión logística para clasificar el dominio del tráfico catalogado como “otro”.
4. Entrenar y aplicar un modelo de bosque de aislamiento para detectar anomalías en el tráfico de un dominio en específico.
5. Aplicar herramientas de visualización para analizar los resultados.

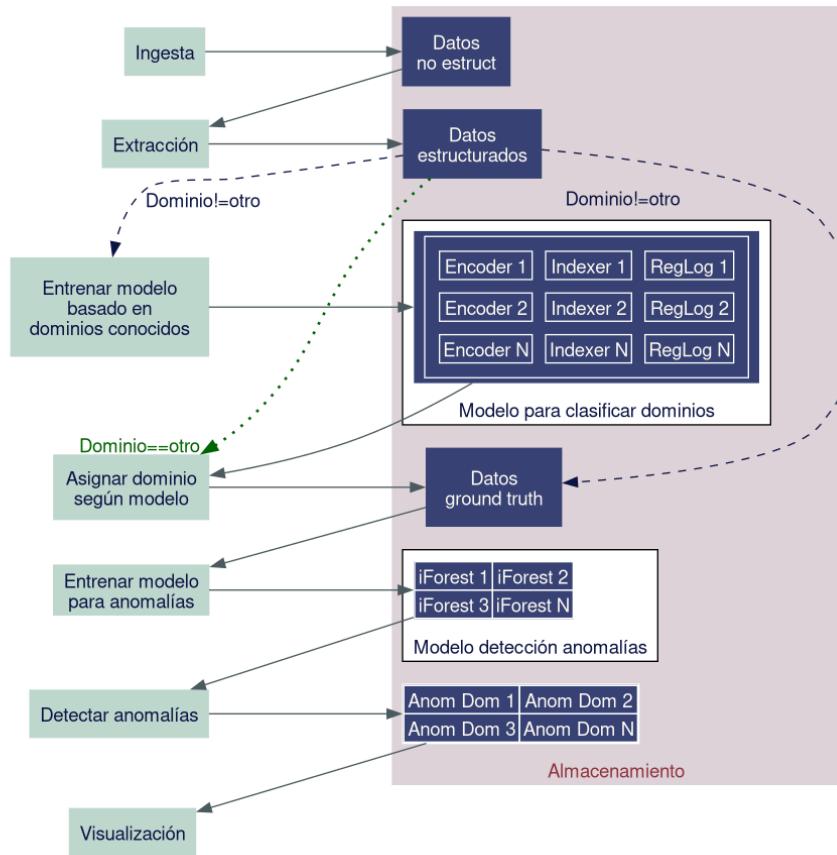


Figura 36: Flujo de procesamiento para el proceso de entrenamiento del sistema

Fuente elaboración propia

4.2 Clasificador de dominio

Como se mencionó en el Capítulo 4.1.1 el tráfico capturado pierde la información sobre el dominio al que corresponde, pero a pesar de ello sigue siendo parte orgánica del uso de la aplicación que contiene información valiosa, por ello se optó por desarrollar una solución que aplica técnicas de aprendizaje automático con apoyo de procesamiento de lenguaje natural con el objetivo de disminuir la cantidad de tráfico catalogado como “Dominio otro” esto permitiría, ampliar el tráfico sobre el cuál se entrena la detección de anomalías para obtener mejores resultados. Además esta categorización basada en dominios es importante ya que cada dominio tiene su propia actividad y construye su propia definición de normalidad. Para ello se desarrolló el siguiente flujo de procesamiento.

4.2.1 Ingesta

El proceso de ingestión para el estado actual del proyecto consiste en acceder al servidor vía SSH y copiar los archivos de registro a la solución de almacenamiento MinIO en el host de procesamiento.

4.2.2 Extracción y estructuración

Como se mencionó anteriormente, en el contexto de los datos (Capítulo 4.1.1) se cuenta inicialmente con una cadena de texto similar a la siguiente:

```
45.166.93.223 - - [23/Aug/2024:00:00:20 +0000] "GET /api/manual/find/?category=De%20todo%20un%20poco&searchIn=category&page=1&limit=1 2&search=%7B%22searchAllStatuses%22%3Atrue%2C%22searchParam%22%3A%22De%20todo%20un%20poco%22%7D HTTP/1.1" 304 0 "https://a.com/manual/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
```

Código 3: Ejemplo de registro de conexión en archivo de registro generado por NGINX

Fuente elaboración propia

Podemos seleccionar este registro de conexión retomando la estructura base de los archivos de registro (véase Capítulo 1.5.1)

Variable	Valor
remote_addr	45.166.93.223
remote_usr	(Vacío)
date_time	23/Aug/2024:00:00:20 +0000
request	GET /api/manual/find/?category=De%20todo%20un%20poco&searchIn=category&page=1&limit=12&search=%7B%22searchAllStatuses%22%3Atrue%2C%22searchParam%22%3A%22De%20todo%20un%20poco%22%7D HTTP/1.1
status	304
body_bytes_sent	0
http_referer	https://a.com/manual/
user_agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36
gzip_ratio	(Vacío)

Tabla 9: Descripción de variables en archivos de registro

Fuente elaboración propia

Como se puede observar existen algunas operaciones y transformaciones que se pueden realizar para obtener mayor información o información mejor estructurada:

El campo **date_time** se puede dividir para obtener fecha, hora, hora absoluta, unix timestamp y día de semana, la hora absoluta convierten los minutos en números decimales, por ejemplo 8:30 se representa como 8.5, el día de semana convierte la fecha en un número del 1 al 7 correspondiente del lunes al domingo y unix timestamp que es una representación del tiempo basada en “el número de segundos que han transcurrido desde el primero de enero de 1970 a las 00:00:00 UTC [...] de manera que el unix timestamp es la misma en todas partes” (Louis, 2020)

El campo request se puede dividir según sus tres componentes, método, request uri y versión de http; más aún, si analizamos más a detalle el request uri del ejemplo (“`/api/manual/find/?category=De%20todo%20un%20poco&searchIn=category&page=1&limit=12&search=%7B%22searchAllStatuses%22%3Atrue%2C%22searchParam%22%3A%22De%20todo%20un%20poco%22%7D`”) podemos notar que existen palabras conocidas como “category”, “De”, “todo”, “un”, “poco” separadas por caracteres desconocidos “%20”, “%7B”, “%3A” por nombrar algunos, estos caracteres siempre comienzan con “%” ya que son parte de una codificación estandar para uso en redes computacionales, algunos caracteres pueden ser inseguros “porque puertas de enlace y otros agentes de transmisión son conocidos por modificarlos, estos caracteres son ‘{’, ‘}’, ‘|’, ‘\’, ‘^’, ‘~’ [...] por lo que se codifican transformandolos a una representación hexadecimal (‘0123456789ABCDEF’) inciendo con un %” (Berners-Lee, 1994)

Si decodificamos la uri obtenemos el texto `'/api/manual/find/?category=De todo un poco&searchIn=category&page=1&limit=12&search={"searchAllStatuses":true,"searchParam":"De todo un poco"}'` y podemos subdividir esta cadena nuevamente, ya que “el componente query contiene información no jerárquica, que con el request uri permiten identificar el recurso solicitado” (Berners-Lee, 2005)

Las transformaciones descritas y ejemplificadas se codificaron de la siguiente manera.

```

def decode_uri(uri):
    return urllib.parse.unquote(uri) if uri is not None else None

def get_path(uri):
    return urllib.parse.urlparse(uri).path if uri is not None else None

def get_query_list(uri):
    if uri is None:
        return []
    query = urllib.parse.urlparse(uri).query
    return [f"{k}={v}" for k, v in urllib.parse.parse_qsl(query)]

def get_domain(referer):
    netloc = urllib.parse.urlparse(referer).netloc
    return netloc if netloc not in (None, "", "-") else "Unknown"

def parse_date(date_str):
    return (
        datetime.strptime(date_str, "%d/%b/%Y:%H:%M:%S +0000")
        if date_str is not None
        else None
    )

def to_unix_timestamp(dt):
    return time.mktime(dt.timetuple()) if dt is not None else None

```

Código 4: Fragmento de código para ampliar y transformar datos iniciales

Fuente elaboración propia

También se agrega una columna extra llamada “domain_category” donde se determina si el valor de http_referer es uno de los dominios que el servidor procesa (es decir, que el tráfico probablemente fue generado visitando el mismo sitio) o fue generado por un sitio distinto (por ejemplo google.com, lo que indicaría que el usuario, llegó al sitio del

servidor por una búsqueda en google.com) o si http_referer está vacío, en este caso se denomina el “domain_category” como “otro”. Esta toma de decisiones se visualiza mejor con el diagrama de la figura 37.

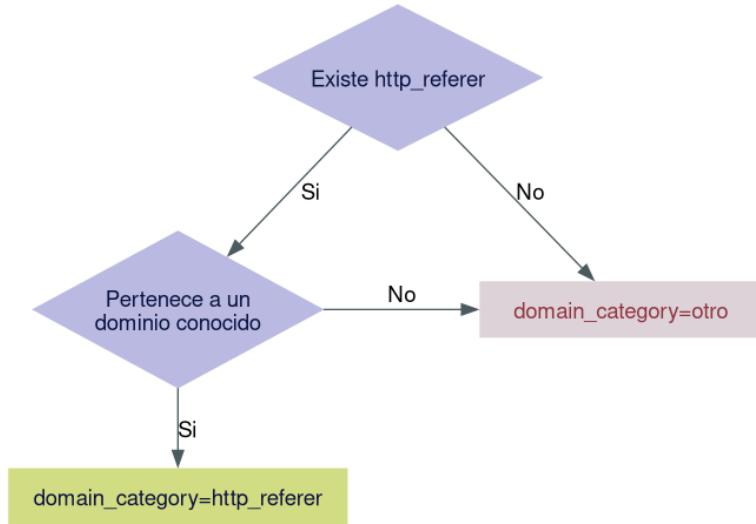


Figura 37: Flujo para asignación de domain_category

Fuente elaboración propia

Ahondando esta transformación se llevó a cabo con el siguiente código.

```
domains_str = os.getenv("DOMAINS")
registered_domains = ast.literal_eval(domains_str)

df = df.withColumn(
    "domain_category",
    when(col("domain").isin(*registered_domains), col("domain")).otherwise("otro"),
)
```

Código 5: Fragmento de código para establecer si el dominio en http_referer es útil

Fuente elaboración propia

Dadas las transformaciones descritas se obtienen las siguientes variables

Variable	Valor
remote_addr	45.166.93.23
remote_user	(Vacio)
date_time (tipo string)	23/Aug/2024:00:00:20 +0000
date	23-Aug-2024
time	0:00:20
method	GET
req_uri	/api/manual/find?category=De%20todo%20un%20poco&searchIn=category&page=1&limit=12&search=%7B%22searchAllStatuses%20%3Attrue%2C%22searchParam%22%3A%22De%20todo%20un%20poco%22%7D
http_ver	HTTP/1.1
status	304
body_bytes_sent	0
http_referer	https://a.com/manual/
user_agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36
dec_req_uri	/api/manual/find?category=De todo un poco&searchIn=category&page=1&limit=12&search={"searchAllStatuses":true,"searchParam":"De todo un poco"} /api/manual/find/
clean_path	
clean_query_list	[{"category": "De todo un poco", "searchIn": "category", "page": "1", "limit": "12", "search": {"searchAllStatuses": true, "searchParam": "De todo un poco"}}]
domain (obtenido de http_referer)	a.com
fdate_time (tipo fecha)	23/Aug/2024:00:00:20
dateunixtimest	1724392820
tabstime	0
day_of_week	4
domain_category	otro

Tabla 10: Información de archivo de registros estructurada y enriquecida

Fuente elaboración propia

Una vez que se lanza la aplicación a Apache Spark para su ejecución en el cluster, el driver de Apache Spark comienza a evaluar los recursos disponibles, las tareas a realizar y la distribución de tareas entre los workers disponibles, esto se puede apreciar en la interfaz gráfica de la aplicación.

Spark Jobs (?)

User: root

Total Uptime: 1.8 min

Scheduling Mode: FIFO

Active Jobs: 1

Completed Jobs: 1

▼ Event Timeline

Enable zooming

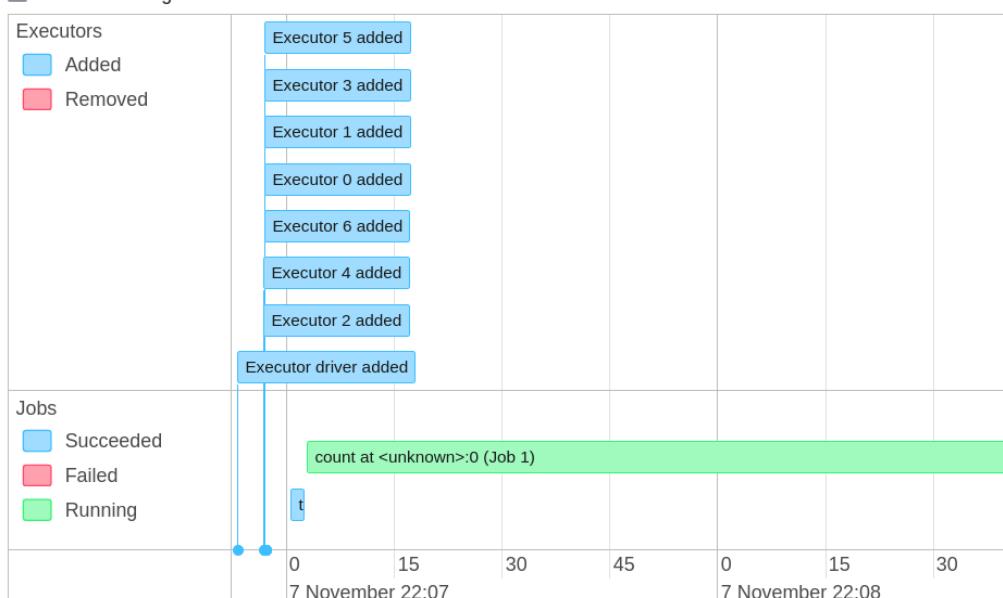


Figura 37: Arranque de aplicación en spark, asignación de workers

Fuente elaboración propia

De manera interna, Apache Spark trabaja con Datasets Resilientes Distribuidos (RDD por sus siglas en inglés) “los cuales son colecciones de datos tolerantes a fallos que pueden ser procesados en paralelo. Spark usa Grafos Dirigidos Acíclicos (DAG por sus siglas en inglés) para mostrar las transformaciones y acciones que se aplican a los RDD. Las transformaciones son operaciones que crean nuevos RDD a partir de RDD existentes mientras que las acciones se usan para regresar un resultado o escribir información en sistemas de almacenamiento externo.” (Mukesh, 2023)

A continuación en la figura 38 se muestra el diagrama DAG de las acciones y transformaciones descritas, se puede observar el escaneo (lectura) de los archivos de registro, la operación filter donde se extraen cada una de las columnas basadas en un regex, la operación HashAggregate donde se obtienen los distintos valores de domain (http_referer) por nodo, el intercambio de Aggregates para eliminar HashAggregates duplicados.

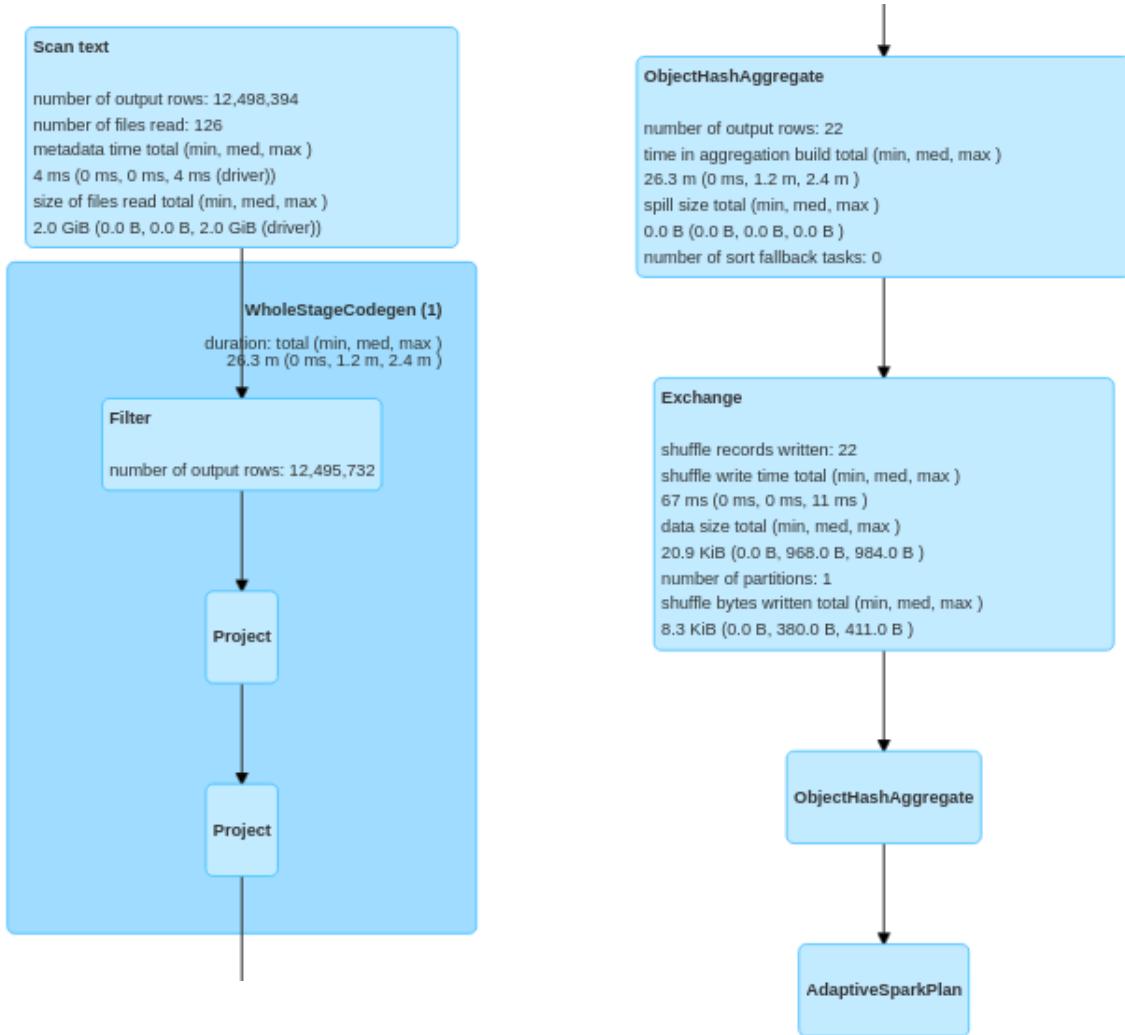


Figura 38: Ejemplo de DAG en Apache Spark

Fuente elaboración propia

Asimismo Spark permite conocer las tareas y trabajos completados y en progreso, mostrando el tiempo total o transcurrido, el nivel de progreso y la cantidad de núcleos (cores) realizando la tarea en paralelo.

▼ Active Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	count at <unknown>:0 count at <unknown>:0 (kill)	2024/11/07 22:07:02	2.3 min	0/1	21/26 (5 running)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	take at /opt/spark-apps/extract.py:150 take at /opt/spark-apps/extract.py:150	2024/11/07 22:07:00	2 s	1/1	1/1

▼ Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	take at /opt/spark-apps/extract.py:150 take at /opt/spark-apps/extract.py:150	2024/11/07 22:07:00	2 s	1/1	1/1

Page:

Figura 39: Trabajos en progreso y completados en Apache Spark

Fuente elaboración propia

Con la utilidad “htop” es posible analizar el uso de recursos en el equipo host, por ejemplo en la figura 40 podemos observar el uso de los 16 cores disponibles, una necesidad moderada de memoria RAM y los primeros 36 hilos (véase Capítulo 3.4) de los cuales, al momento de la captura, todos son de Apache Spark.

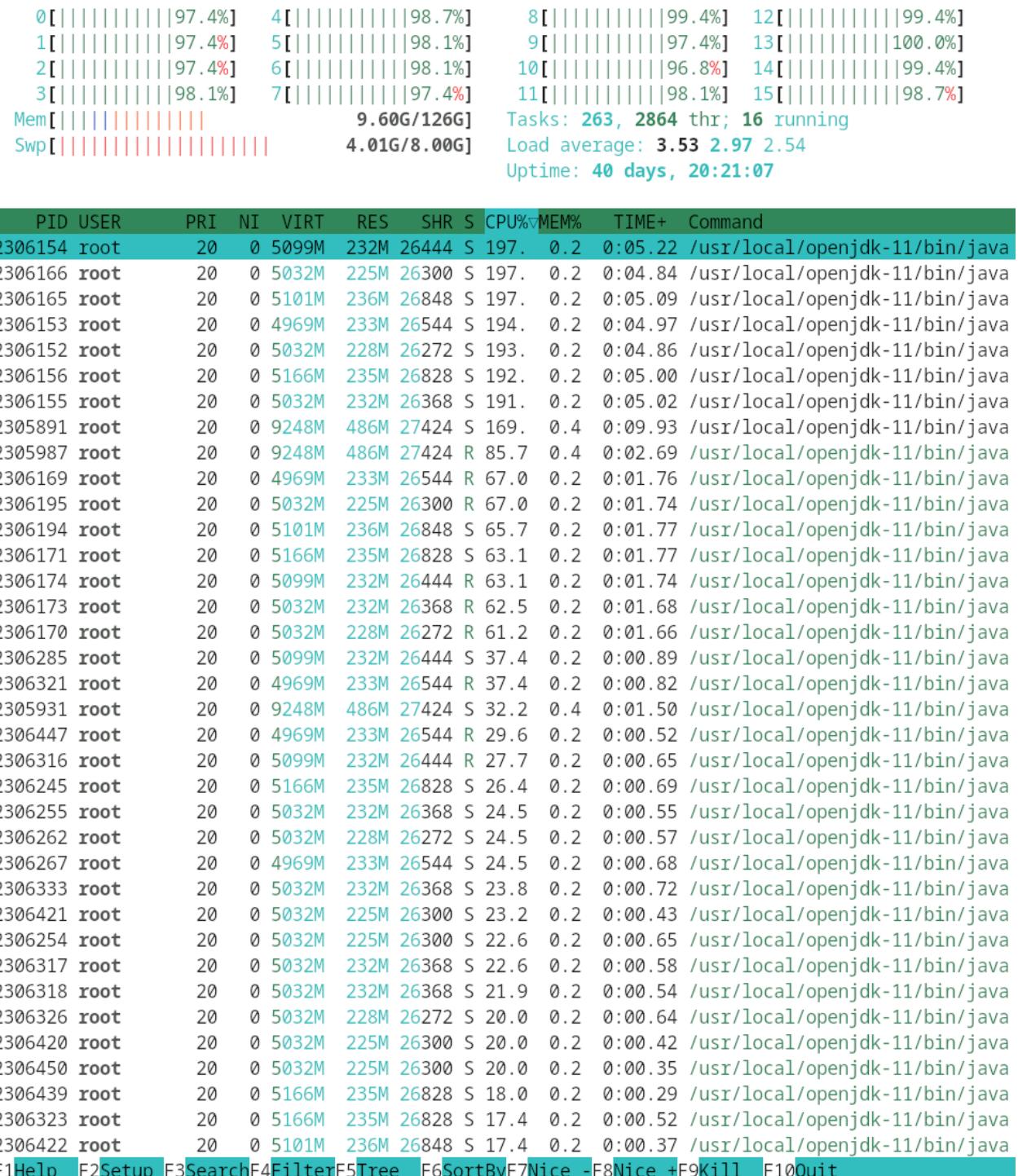


Figura 40: Monitoreo de recursos mediante htop

Fuente elaboración propia

El conjunto de datos resultante se almacenó en el contenedor MinIO, Apache Spark permite subdividir por “domain_category” generando una carpeta por cada valor.

The screenshot shows the MinIO web interface. On the left is a dark sidebar with various icons for file operations. The main area is titled 'logs' and shows a list of objects. At the top, it says 'Created on: Fri, Nov 08 2024 06:05:04 (CST) Access: PRIVATE 715.6 MiB - 1460 Objects'. Below this are three buttons: a refresh icon, a refresh/circular arrow icon, and an upload icon. The object list is titled 'logs / output / known' and includes a header row with columns: 'Name', 'Last Modified', and 'Size'. The data rows show the following entries:

Name	Last Modified	Size
_SUCCESS	Fri, Nov 08 2024 06:07 (CST)	-
domain=202		-
domain=a28		-
domain=acu		-
domain=alu		-
domain=alu		-
domain=alu		-
domain=aut		-
domain=aut		-
domain=avi		-
domain=caj		-
domain=caj		-

Figura 41: Producto de la etapa de extracción - dominios ocultos por cuestiones de privacidad

Fuente elaboración propia

4.2.3 Entrenamiento de modelo para determinar dominio

Para realizar la asignación de dominio al tráfico sin http_referer (url directa, bots, etc) o con http_referer de otros dominios (google.com, youtube.com, gmail.com) (véase Figura 37 para referencia) se entrenó un algoritmo de clasificación basado en los datos cuyo dominio (http_referer) es válido.

De los datos estructurados anteriores (véase Tabla 10 para referencia) se consideraron únicamente los campos **fabstime**, **day_of_week**, **body_bytes_sent**, **clean_path**, **clean_query_list**, **domain_category**, **method** y **domain**, se eligieron estos campos debido a que se considera son los que mejor representan de manera individual e única cada posible endpoint. Para emplearlos en un algoritmo de clasificación es necesario transformar estos campos a valores únicamente numéricos.

4.2.3.1 Codificación One-hot

Para los campos dominio y método que pertenecen a datos de tipo categórico, (es decir los posibles valores de dominio y método son grupos bien definidos como a.com, b.com y GET, POST, DELETE) lo cuál permite aplicar una transformación one-hot encoding “la cuál es una técnica de codificación común para manejo de datos categóricos. Consiste en un proceso de transformación de variables categóricas a variables en un formato que permite a los algoritmos de aprendizaje automático realizar un mejor trabajo. [...] Simplemente destaca la presencia de características variables para prevenir una interpretación de correlación entre variables independientes” (Al-Shehri, 2021)

La transformación **one-hot** consiste en tomar todas las distintas categorías y convertirlas en una representación vectorial. En la figura 42 por ejemplo se simuló tener tres registros de conexión, la primera conexión fue de método GET al dominio a.com, la segunda de método POST al dominio b.com y la última de método GET al dominio c.com. Después de realizar la codificación one-hot, se transforman las categorías en vectores, el tamaño del vector corresponde con la cantidad de valores dentro de la categoría (en el ejemplo hay dos métodos, “GET” y “POST” por lo que el vector resultante es de dos posiciones, en cambio hay tres dominios por lo que el vector de dominio es de tres posiciones). Finalmente Apache Spark genera un vector auxiliar el cuál permite decodificar el vector en sus categorías originales.

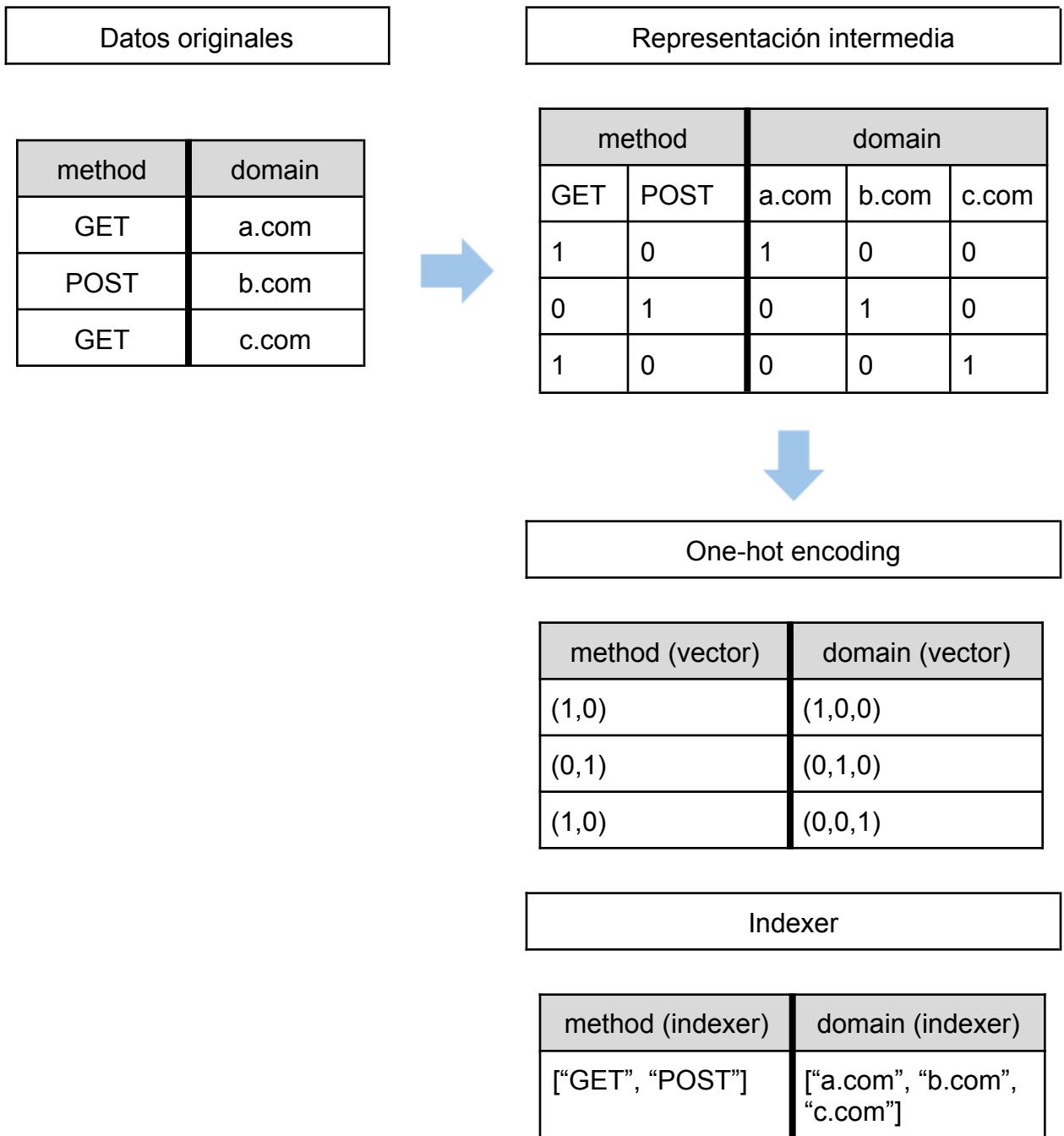


Figura 42: Ejemplo de codificación one-hot e indexer

Fuente elaboración propia

4.2.3.2 Procesamiento de lenguaje natural

Se aplicó la técnica de tokenización a los campos “clean_path” y “clean_query_list” (véase Tabla 10 para referencia) ya que se considera que estos campos son mejores identificadores semí-únicos para determinar un dominio.

Tomemos como ejemplo el siguiente valor.

Path original
<dominio>/api/v1/planner/items?end_date=2024-10-12T05:00:00.000Z&order=desc&per_page=1

Del cuál se obtienen los siguientes valores para “clean_path” y “clean_query_list”

clean_path	clean_query_list
/api/v1/planner/items	[end_date=2024-10-12T05:00:00.000Z, order=desc, per_page=1]

Para cada uno de estos campos se desean extraer distintos patrones, en el caso de “clean_path” cada “palabra” (conjunto de caracteres separados por “/”) es relevante (“api”, “v1”, “planner”, “items”) y además el orden de estos elementos es específico, es decir, un recurso alojado en /api/v1/planner/items es completamente distinto a /v1/items/api/planner por lo que el tratamiento de “clean_path” debe preservar el orden, para lograr esta preservación en el orden de las palabras, se decidió tokenizar basado en caracteres, en lugar de “palabras” (véase Capítulo 3.3 para referencia). Para determinar la cantidad óptima de caracteres se analizaron las palabras en todos los valores de “clean_path”, se obtuvieron las siguientes estadísticas.

Variable	Resultado
Cantidad de palabras únicas en clean_path	172,133
Cantidad de letras mínimas por palabra	0
Cantidad de letras máximas por palabra	305
Cantidad de letras promedio por palabra	6.81
Desviación estándar en la cantidad de letras por palabra	7.54

Tabla 11: Estadísticos de campo “clean_path”

Fuente elaboración propia

Dado los resultados estadísticos obtenidos, se determinó que un tokenizado por 9 caracteres (9-grams) permite cubrir al menos media palabra antes del separador “/“ y media palabra después.

```
df_known = df_known.withColumn("path_characters", split(col("clean_path"), ""))

ngram = NGram(n=9, inputCol="path_characters", outputCol="path_ngrams")

df_known = ngram.transform(df_known)
```

Código 6: Fragmento de código para obtener 9-grams del campo “clean_path”

Fuente elaboración propia

Al aplicar la tokenización de 9 caracteres al “clean_path” de ejemplo se obtiene el siguiente resultado

```
[  
/a/p/i/v1/p,  
a/p/i/v1/pl,  
pi/v1/pla,  
i/v1/pla/n,  
/v1/pla/n/n,  
v1/pla/n/n/e,  
1/pla/n/n/r,  
/pla/n/n/r/,  
pla/n/n/r/i,  
lanner/it,  
anner/ite,  
nner/ite/m,  
ner/ite/m/s  
]
```

Código 7: Depuración de variable “path_ngrams”

Fuente elaboración propia

En cambio para “clean_query_list” el orden de cada dupla llave-valor es irrelevante, en formato de url
`/api/v1/planner/items?end_date=2024-10-12T05:00:00.000Z&order=desc&per_page=1` es exactamente igual a

/api/v1/planner/items?order=desc&end_date=2024-10-12T05:00:00.000Z&per_page=1, ambas solicitan el mismo recurso con los mismos parámetros. Incluso para “clean_query_list”, podemos desechar los valores y mantener únicamente las llaves (**end_date, order y per_page**) ya que estás son constantes y ofrecen mayor estabilidad en el reconocimiento de patrones.

```
transform(col("clean_query_list"), lambda x: split(x, "=")[0]),
```

Código 8: Fragmento de código para obtener únicamente las llaves del campo “clean_query_list”

Fuente elaboración propia

Una vez transformados los campos “clean_path” y “clean_query_list” se unen ambos vectores para generar una representación completa del path solicitado

```
[  
/api/v1/p,  
api/v1/pl,  
pi/v1/pla,  
i/v1/plan,  
/v1/plan,  
v1/planne,  
1/planner,  
/planner/,  
planner/i,  
anner/it,  
anner/ite,  
nner/item,  
ner/items,  
end_date,  
order,  
per_page  
]
```

Código 9: Depuración de variable “url_features”

Fuente elaboración propia

Finalmente con el campo url_features (que une la estructura de “clean_path” con las llaves de “clean_query_list”) es necesario transformarlo a una representación numérica, para ello se eligió HashingTF que es la implementación de Apache Spark para atributos hash (véase Capítulo 3.3.1 para referencia) en el cuál solo es necesario determinar el tamaño del vector resultante que minimice la probabilidad de colisiones. Al respecto la documentación nos indica “es recomendable usar una potencia de dos como parámetro numFeatures (tamaño del vector); de otra manera los atributos no serán transformados de manera igualitaria”. (Apache Spark, 2024c)

Para este trabajo se eligió 2^{14} (16,384) para generar un espacio amplio que minimice colisiones.

Al aplicar HashingTF se logra eficientar la transformación de “url_features” a valores numéricos que se puedan ingresar al modelo, definir el tamaño del vector de atributos (similar al resultado de la codificación one-hot, véase Capítulo 4.2.3.1 para referencia), más uno de los costos es que el método hash es un proceso no reversible, es decir, es imposible regenerar “url_features” basado en el resultado de HashingTF, en este caso, esta pérdida de reversibilidad no es un problema, ya que este proceso solo se usa para entrenar el modelo.

Retomando el ejemplo la aplicación de HashingTF da un resultado similar a

```
(16384,[207,252,282,310,339,659,938,9  
81,995,1307,1891,2112,3393,3619,3700,  
3857],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,  
1.0,1.0,1.0,1.0,1.0,1.0,1.0])
```

Código 10: Depuración de variable “hash_url_features”

Fuente elaboración propia

Donde la respuesta sigue la sintaxis

```
(tamaño del vector, [índices de vector  
con valor], [valor del vector])
```

Código 11: Sintaxis del tipo de dato Sparse Vector en Apache Spark

Fuente: Apache Spark, 2024d

Por lo que este “hash_url_features” se debe interpretar como un vector de 16,384 posiciones el cuál está inicializado en ceros en todas sus posiciones excepto para las posiciones 207, 252, 282, 310, 339, 659, 938, 981, 995, 1307, 1891, 2112, 3393, 3619, 3700, 3857 que tienen un valor de 1.

4.2.3.3 *Modelo de regresión logística*

Una vez que los datos de la estructura de “clean_path” con las llaves de “clean_query_list” han sido capturados en hash_url_features se procede a preparar y entrenar el modelo que a partir de datos con un dominio conocido (datos etiquetados) pueda asignar un dominio al tráfico previamente catalogado como “dominio otro” (datos no etiquetados).

Se crea un nuevo vector que contenga los siguientes campos, **fabstime**, **day_of_week**, **method_onehot**, **body_bytes_sent** y **hash_url_features**, estos son los campos que se consideran más representativos de cada solicitud, y por ello se seleccionaron para el entrenamiento del modelo.

Con los datos agrupados y codificados en un vector, se inicializó el modelo de regresión logística (véase Capítulo 3.2 para referencia) ya que este es un problema de clasificación.

```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(
    featuresCol="features", labelCol="domain_index", family="multinomial", maxIter=100
)
train_data, test_data = df.randomSplit([0.8, 0.2], seed=123)
lr_model = lr.fit(train_data)
```

Código 12: Creación y entrenamiento de modelo de regresión logística

Fuente elaboración propia

Finalmente en la implementación del modelo de regresión logístico se definió de tipo “multinomial” ya que no solo se cuenta con dos categorías, sino con 48 categorías (cada posible dominio).

4.2.4 *Aplicar modelo para determinar dominio*

Una vez que se tiene el modelo para asignar dominios entrenado, se realizan las mismas transformaciones y preparación de datos que en el entrenamiento (véase 4.2.3 para referencia) en los datos hasta ahora catalogados como dominio “otro”.

El modelo regresa un vector de dimensión igual a las posibles categorías (en este contexto los posibles dominios) similar al siguiente ejemplo:

```
[  
 2.31E-10, 4.60E-13,  
 8.00E-11, 2.04E-10,  
 6.57E-10, 5.05E-09,  
 6.26E-11, 2.34E-10,  
 1.67E-10, 2.51E-09,  
 1.86E-10, 9.69E-10,  
 0.9999999887,  
 1.30E-10, 6.79E-11,  
 7.32E-11, 1.02E-10,  
 3.77E-11, 1.89E-10,  
 1.14E-10, 6.71E-11,  
 3.55E-11, 2.79E-11,  
 3.41E-11, 5.39E-11,  
 1.87E-11, 1.27E-11,  
 5.05E-12, 3.19E-12,  
 4.24E-12, 2.48E-12,  
 2.78E-12, 1.49E-12,  
 1.14E-12, 1.15E-12,  
 1.12E-12, 1.22E-12,  
 8.74E-13, 9.74E-13,  
 8.13E-13, 8.42E-13,  
 8.75E-13, 6.50E-13,  
 7.17E-13, 4.72E-13,  
 6.42E-13, 4.88E-13,  
 1.69E-13  
 ]
```

Código 13: Depuración de variable “probability”

Fuente elaboración propia

Este vector representa la probabilidad que tiene un ejemplo de pertenecer a cada uno de los dominios, en su mayoría la probabilidad es baja, excepto en uno, en el cuál el modelo calcula que tiene 99% de probabilidad de pertenecer, para asignar el nombre del dominio se recurre al indexador (véase Figura 42 para referencia).

Si bien en este ejemplo el modelo cataloga fuertemente hacia un dominio, pueden existir casos menos definidos, por ejemplo un vector donde ningún dominio supere el 50%, para prevenir falsos positivos, se optó por aumentar la barrera de categorización de dominio, es decir, el modelo debe indicar más del 70% de probabilidad de otra

manera se mantiene categorizado como “otro”, esta decisión recordando que existe tráfico que nunca tiene un dominio explícito en el campo http_referer (véase Capítulo 4.1.1 para referencia).

4.3 Detección de anomalías

Para la detección de anomalías se consideraron distintas implementaciones como autoencoders (véase Capítulo 3.1.1.1 para referencia), y mapas autoorganizados (véase Capítulo 3.1.1.2 para referencia) pero estos algoritmos tienen una pobre implementación en el entorno de cómputo distribuido generado en Apache Spark debido a que su operación se basa en operaciones secuenciales las cuales son difíciles de paralelizar y tienen un costo de recursos elevados ya que requieren una alta tasa de intercambio de información entre todos los nodos del cluster lo cuál impacta la capacidad de procesamiento individual de cada nodo, por ello se decidió realizar la implementación de bosques de aislamiento (véase Capítulo 3.1.3) pero antes de implementarlo se aplicaron nuevamente algunas técnicas de procesamiento de lenguaje natural al campo **clean_path** y **clean_query_list** (véase Capítulo 4.2.3.2 para referencia de procesamiento previo).

Se analizó el tráfico “ground truth” de un dominio dado y se detectó que múltiples peticiones usaban el envío de identificadores en la uri y por consecuencia se reflejaban en los campos “clean_path” y “clean_query_list”, los identificadores detectados con mayor frecuencia fueron identificadores de mongo los cuales tienen una estructura compuesta de 24 caracteres alfanuméricos en el mismo rango que el sistema hexadecimal, (por ejemplo 66c37af771dce6a2b2afbfad, o 66c38d0071dce6a2b2b09c2d) además otro patrón reconocible es una sección únicamente compuesta de dígitos, en el Código 14 se refiere a “stage”

```
/api/user/updateStageByAdmin/student/66c37af771dce6a2b2afbfad/admission/668c7729c8d3  
546189f626f6/stage/3  
/api/user/updateStageByAdmin/student/66ecb3aa53e24e7301fb4e7/admission/668c7754c8d  
3546189f62724/stage/3  
/api/user/updateStageByAdmin/student/66c3988371dce6a2b2b0f53b/admission/668c7729c8d  
3546189f626f6/stage/3  
/api/user/updateStageByAdmin/student/66ee0b2953e24e73013727f6/admission/668c7754c8d  
3546189f62724/stage/3  
/api/user/updateStageByAdmin/student/66bfdd5471dce6a2b2af40ac/admission/668c7754c8d  
3546189f62724/stage/3  
/api/user/updateStageByAdmin/student/66ec9b7153e24e7301f2162d/admission/668c7754c8d  
3546189f62724/stage/3
```

Código 14: Ejemplos de uri's distintas que representan el mismo endpoint

Fuente elaboración propia

Aplicando un Regex fue posible remover las secciones variables estandarizando la uri en sus componentes estáticos y únicamente representar las variables. Resultando en una reducción de volumen y variabilidad en los datos.

```
val df_prep = df
  .withColumn(
    "url_features_regex",
    expr("""
      transform(
        url_features,
        x -> regexp_replace(
          x,
          '(?:^|[a-fA-F0-9]{24})|(?:^\\d+$)',
          CASE
            WHEN x RLIKE '^|[a-fA-F0-9]{24}$' THEN 'MONGOID'
            WHEN x RLIKE '^\\d+' THEN 'DIGIT'
            ELSE X
          END
        )
      )
    """)
  )
```

Código 15: Aplicación de regex para reducir variabilidad

Fuente elaboración propia

Resultando en la siguiente uri
"/api/user/updateStageByAdmin/student/MONGOID/admission/MONGOID/stage/DIGIT" esta transformación ayuda los algoritmos posteriores a enfrentar menor variabilidad y ruido que puede afectar su capacidad de ofrecer resultados de calidad.

Una vez que se completó el preprocesamiento de la información, se empleó el algoritmo de bosque de aislamiento (véase Capítulo 3.1.3) por dos razones, fue el que mejor se adaptó al ambiente distribuido explotando las capacidades de los múltiples nodos spark worker del cluster así como su complejidad lineal de tiempo de ejecución, lo que lo vuelve más óptimo y rápido que el resto de los algoritmos probados.

Para la implementación del bosque de aislamiento se consideraron los siguientes campos “**“fabstime”**”, “**“status”**”, “**“body_bytes_sent”**” y “**“url_features_regex”**” además

se integró el paquete desarrollado por Linkedin el cuál está enfocado y optimizado para correr en el ambiente de spark.

```
import com.linkedin.relevance.isolationforest._

val isolationForest = new IsolationForest()
  .setNumEstimators(100)          // Cantidad de árboles
  .setBootstrap(false)            // Remuestreo por método de bootstrap
  .setMaxSamples(256)            // Muestras por árbol
  .setMaxFeatures(1.0)           // Porcentaje de características en entrenamiento
  .setFeaturesCol("features")    // Vector de características
  .setPredictionCol("anomaly")   // Columna de resultados
  .setScoreCol("outlierScore")    // Columna de puntaje de anomalía
  .setContamination(0.05)         // Porcentaje de anomalías en entrenamiento
  .setContaminationError(0.02)    // Porcentaje de error permitido
```

Código 16: Implementación del bosque de aislamiento

Fuente elaboración propia

Una vez que el modelo fue entrenado y aplicado a los datos se obtiene un resultado similar al siguiente

anomaly	outlierScore
1.0	0.766
1.0	0.636
1.0	0.684
0.0	0.437
0.0	0.553

Tabla 12: Depuración de variables “anomaly” y “outlierScore”

Fuente elaboración propia

Donde la columna **anomaly** define en categorías si el tráfico es anómalo o no y **outlierScore** presenta un puntaje de aislamiento, esté último se puede usar para definir un nuevo límite respecto a la categorización de normal o anormal.

Capítulo 5 - Resultados y conclusiones

5.1 Resultados

A través del estudio presentado se obtuvieron los siguientes resultados para cada una de sus etapas.

5.1.1 Extracción

Durante el proceso de extracción se tomaron datos textuales “crudos” no estructurados correspondientes a un periodo de 72 días consecutivos y se convirtieron en datos estructurados, ampliando de 9 variables iniciales (véase Tabla 9 para referencia) a 21 variables (véase Tabla 10 para referencia), sin embargo a pesar de este enriquecimiento de datos con metadatos se logró una reducción del almacenamiento de 2.0 Gb iniciales a 597 Mb esto supone una reducción del 70% en almacenamiento. Además subdividir por “domain_category” tiene un impacto considerable en etapas posteriores al permitir acceder a información segregada y evitar realizar lectura de datos irrelevantes.

5.1.2 Entrenamiento de modelo

Después de realizar el entrenamiento del clasificador logístico se realizaron diversas pruebas para evaluar su precisión, estos fueron los resultados obtenidos

Prueba	Resultado
Precisión	98.82%
Precisión ponderada	98.91%
Exhaustividad ponderada	98.97%
Puntaje F1	98.82%

Tabla 13: Resultados de distintas métricas para evaluar el desempeño del modelo entrenado

Fuente elaboración propia

5.1.3 Aplicar modelo para determinar dominio

Una vez aplicado el modelo de clasificación a los datos con dominio “otro” se logró reducir de 18.9% (1,543,472 registros) a 3.6% (294,075 registros) quedando distribuidos como lo indica la Figura 43

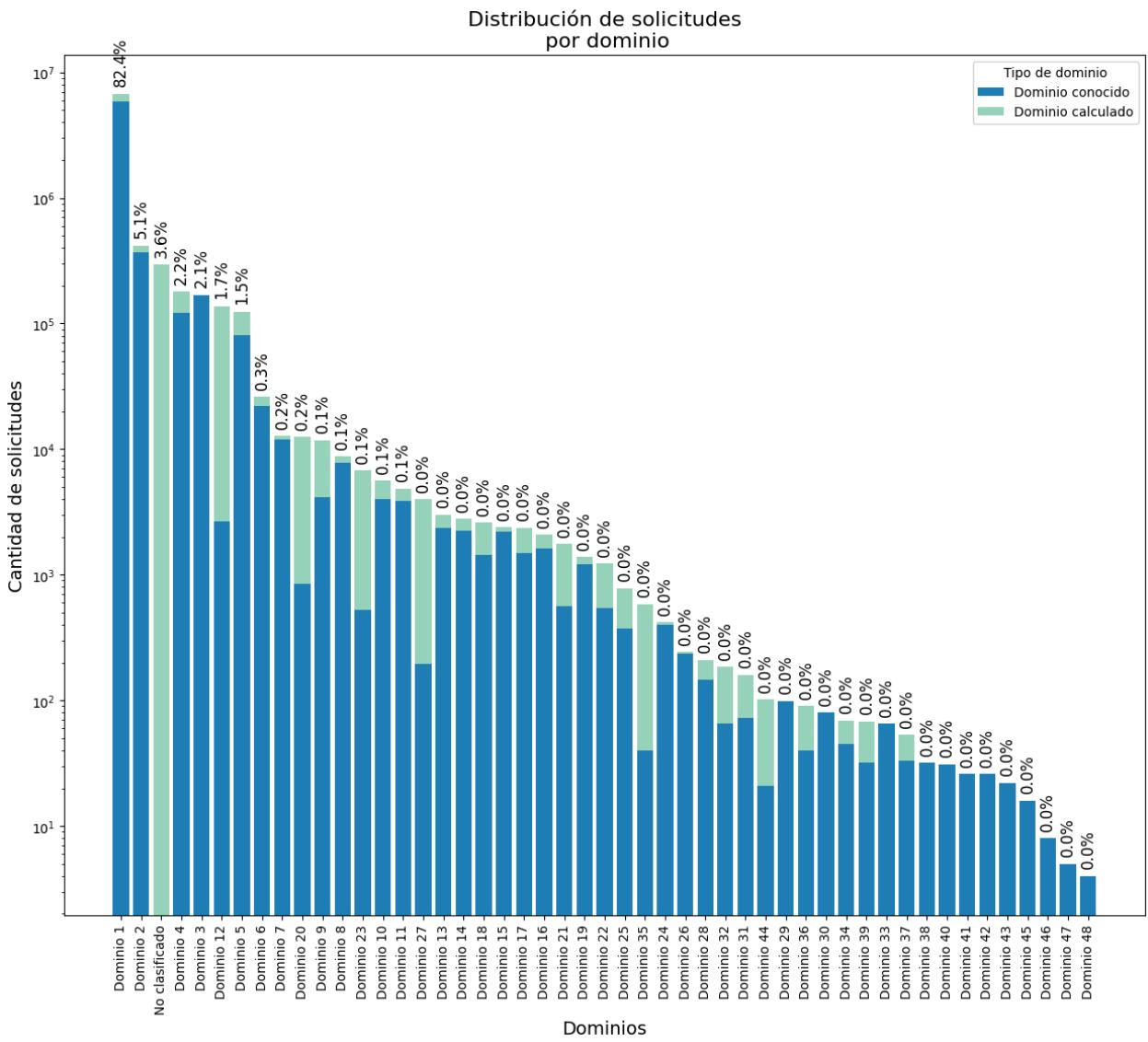


Figura 43: Distribución de solicitudes por dominio después de reclasificar el dominio “otro”

Fuente elaboración propia

A continuación se desglosa el conteo de solicitudes catalogadas usando `http_referer` (conocidas) y asignadas con el modelo de regresión logística (asignadas):

Dominio	Solicitudes con dominio conocidas	Solicitudes con dominio asignado	Dominio	Solicitudes con dominio conocidas	Solicitudes con dominio asignado
Dominio 1	5,812,121	919,503	Dominio 25	372	412
Dominio 2	370,235	47,251	Dominio 35	40	545
No clasificado	0	294,075	Dominio 24	400	19
Dominio 4	121,492	57,182	Dominio 26	234	9
Dominio 3	168,335	1,129	Dominio 28	145	65
Dominio 12	2,681	135,582	Dominio 32	66	119
Dominio 5	81,461	42,483	Dominio 31	72	88
Dominio 6	22,140	3,842	Dominio 44	21	81
Dominio 7	11,953	765	Dominio 29	98	1
Dominio 20	842	11,660	Dominio 36	40	50
Dominio 9	4,148	7,672	Dominio 30	80	0
Dominio 8	7,803	980	Dominio 34	45	24
Dominio 23	526	6,233	Dominio 39	32	36
Dominio 10	4,038	1,568	Dominio 33	65	0
Dominio 11	3,891	961	Dominio 37	33	20
Dominio 27	196	3,801	Dominio 38	32	0
Dominio 13	2,348	674	Dominio 40	31	0
Dominio 14	2,235	560	Dominio 41	26	0
Dominio 18	1,430	1,184	Dominio 42	26	0
Dominio 15	2,214	167	Dominio 43	22	0
Dominio 17	1,500	879	Dominio 45	16	0
Dominio 16	1,615	471	Dominio 46	8	0
Dominio 21	561	1,200	Dominio 47	5	0
Dominio 19	1,215	173	Dominio 48	4	0
Dominio 22	545	682			

Tabla 14: Conteo de tráfico conocido y calculado
Fuente elaboración propia

A pesar de que los datos indican que no fue posible categorizar todos los registros de dominio “otro”, se debe tomar en cuenta que nunca se fijó como objetivo reducir a cero esta categoría, ya que esto es virtualmente imposible, como se mencionó en el Capítulo 4.1.1, hay sistemas o partes de sistemas que por naturaleza siempre tendrán http_referer vacío (un sistema de relojes checadores, o un sistema de scraping o una API pública por nombrar algunos), ya que estos sistemas manejan tráfico que no es generado por humanos o no son generados desde un navegador web y por ello pueden prescindir del campo http_referer, bajo esta aclaración se considera que las técnicas empleadas fueron apropiadas y correctas para reducir la cantidad de registros con dominio “otro”, donde se prefirió reducir los falsos positivos (catalogar un registro en el dominio incorrecto) a costa de aumentar los verdaderos positivos.

5.1.4 Detección de anomalías

El entrenamiento del bosque de aislamiento presentó un desempeño notable, en el que a pesar de contar con miles o millones de registros y cerca de dos mil características por registro, el tiempo de procesamiento es reducido.

Cantidad de registros por dominio	Tiempo de entrenamiento
6, 731, 624 registros	50 minutos
178, 674 registros	2 minutos
169, 464 registros	11 minutos
138, 263 registros	3 minutos

Tabla 14: Métricas para evaluar el desempeño del modelo de clasificación de anomalías

Fuente elaboración propia

Una vez entrenado y aplicado el modelo bajo una inspección manual se detectaron los siguientes casos notables de actividad anómala.

```
/__debugging_center_utils__.php?log=;echo Ijyabmwesqxxknnjecoowtppopjvuxsk | id  
/__debugging_center_utils__.php?log=;echo Ijyabmwesqxxknnjecoowtppopjvuxsk | ipconfig  
/services/auth/config/aws_credentials.json  
/plus/recommend.php?action=&aid=1&_FILES[type][tmp_name]=\x5C' or mid=@`\x5C'  
/*!5000union/*!!5000select*/1,2,3,md5(871702),5,6,7,8,9#@`\x5C'+&_FILES[type][name]=  
1.jpg&_FILES[type][type]=application/octet-stream&_FILES[type][size]=4294
```

Código 16: Implementación del bosque de aislamiento

Fuente elaboración propia

Los primeros dos renglones son similares e intentan invocar el endpoint `_debugging_center_utils__.php` (el cuál no existe) e intenta ejecutar código (payload) vía query list params echo para imprimir un mensaje en salida estándar y los comandos `id` e `ipconfig` los cuales intentan obtener información sobre el entorno de ejecución del servidor, esté es un intento de aprovechar la vulnerabilidad CVE-2016-5674 que afecta a equipo de videovigilancia (NVD, 2024). El tercer renglón intenta aprovechar una posible configuración incorrecta del servidor en la que expone datos de autenticación para servicios de Amazon Web Services. El cuarto renglón es un intento de inyección de código sql y sobrepasar un posible web application firewall.

Como se puede observar, los ataques son diversos y emplean técnicas y frameworks o lenguajes distintos.

5.1.5 Visualización

Actualmente el sistema desarrollado procesa y detecta anomalías en las peticiones a nivel unitario, es decir, una petición a la vez, lo cuál permite el análisis detallado de peticiones pero pierde el contexto de porque la petición es anómala, por ello se integraron herramientas de visualización que permitan recontextualizar los datos.

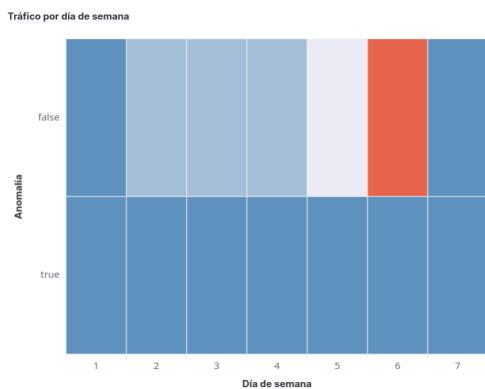


Figura 44. Cantidad de anomalías por día de semana

Fuente elaboración propia

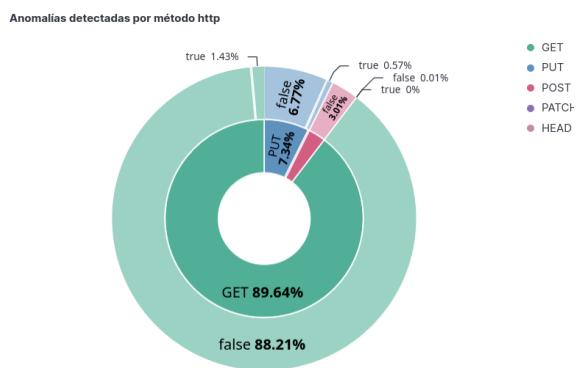


Figura 45. Distribución de métodos y anomalías

Fuente elaboración propia

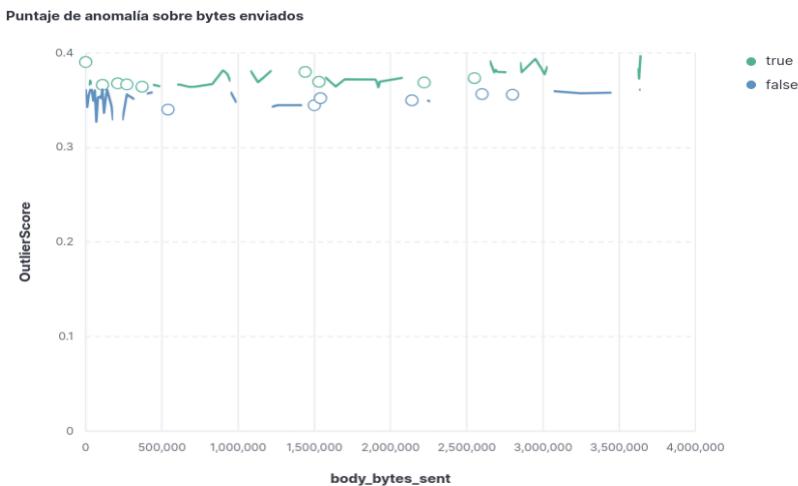


Figura 46. Grado de anormalidad según bytes contestados

Fuente elaboración propia

5.2 Limitaciones

Como se mencionó actualmente la implementación evalúa la anormalidad basado en tráfico puro, si bien esto es bastante útil para detectar intentos de ataque directo, se pierde el contexto temporal, por lo que tendencias de volumen, patrones temporales o cíclicos quedan fuera de los alcances de este trabajo.

Otra factor limitante es que el trabajo actual si bien logra realizar un filtrado y reducción de volumen de interés, aún requiere un analista humano que filtre las anomalías falsas positivas, si bien es más sencillo detectar anomalías e intentos de ataque en miles de peticiones que cientos de mil o millones, el análisis manual debe ser llevado a cabo por personal capacitado que comprenda los contextos y vectores de ataque por lo que la solución desarrollada no logró ser completamente automatizada.

5.3 Conclusiones

Mediante el trabajo realizado se logró una implementación que confirma la hipótesis planteada de que existe una correlación entre datos contextuales del tráfico como hora de la petición, cantidad de bytes contestados, recursos solicitado y parámetros enviados para establecer una estimación del grado de anormalidad, bajo la cuál es posible categorizar y detectar tráfico anómalo e incluso malicioso.

Se considera que el avance logrado establece las bases tecnológicas y algorítmicas fundamentales para producir un sistema productivo y comercial debido a sus capacidades técnicas y madurez en prácticas de ingeniería de software y ciencia de datos. Es necesario recordar el contexto que estos ataques, corresponden a una centena o menos de peticiones maliciosas entre miles o millones que procesa el

servidor, por ello se resalta la importancia y necesidad de aplicar técnicas de aprendizaje automático.

Con la arquitectura y configuración actual se consiguió ofrecer una plataforma y código escalable, tolerante a grandes volúmenes de información.

Tarea	Tiempo de ejecución y volumen de datos
Extracción	35 minutos en procesar y expandir 8,170,910 registros
Entrenamiento	3 hora, 6 minutos en entrenar el modelo basado en 6,636,438 registros
Predicciones de modelo	1 minuto y medio en aplicar modelo entrenado a 1,543,472 de registros
Generar ground truth	1 minuto en combinar 8,169,584 registros de dominios conocidos con dominios asignados
Detección de anomalías	Aproximadamente 5 minutos por dominio

Tabla 15: Tiempos y volumen de datos procesados

Fuente elaboración propia

Desde la perspectiva de procesos, el usar la capa de almacenamiento como medio distribuido del cluster permitió generar estructuras y jerarquías en los datos y modelos que flexibilizaron la trazabilidad y linaje de datos, bajo estas características es posible evaluar de manera independiente cada uno de las etapas del flujo, corroborando que las transformaciones en cada paso se realicen de manera correcta.

Finalmente la arquitectura desarrollada fue diseñada para permitir el entrenamiento y almacenamiento de múltiples modelos tanto en la etapa de clasificación de dominio como en la etapa de detección de anomalías, esta decisión posibilita expandir las capacidades actuales del sistema creando modelos con valores históricos variables (por ejemplo un modelo entrenado con datos del último mes, otro modelo con datos del último bimestre, del último semestre) esto posibilita analizar el tráfico bajo diferentes contextos históricos, y captura de mejor manera las distintas representaciones del tráfico “normal”.

5.4 Trabajos futuros

Existe margen de mejora y de expansión de capacidades del sistema actual, uno de los cambios más importantes será implementar tecnologías de flujo de información, Apache

Spark nativamente soporta información en streaming, por lo que el sistema actual se considera escalable, pero es necesario realizar algunos ajustes para implementarlo satisfactoriamente. Además se puede evaluar tecnologías adicionales como Apache Kafka con lo cuál se soportan “topics” y se pueden realizar flujos de trabajo flexibles.

A pesar de que el estudio consideró 48 dominios distintos, no significa que correspondan a 48 proyectos distintos, es una práctica común ofrecer el mismo software a múltiples clientes, por lo que se considera que se pueden aplicar técnicas de clusterización en la actual representación de “ground truth” para detectar proyectos únicos, esto tendría un efecto importante en expandir aún más la definición de ground truth, reduciendo la repetición de procesamiento y mejorando el aprendizaje de patrones.

Otro punto de mejora es la implementación de análisis de tipo series de tiempo, esto mejoraría la inclusión de factores grupales así como tendencias contextuales, por ejemplo fluctuaciones en el tráfico por temporadas o un aumento paulatino pero constante, la implementación de múltiples modelos pretende cubrir esta omisión, pero se considera que aplicar un tratamiento de serie de tiempo puede ofrecer ventajas notables aunado a la arquitectura de múltiples modelos propuesta.

Capítulo 6 - Bibliografía

Aguilar, (2024) Rezago y asimetrías de las política nacional e internacional de ciberseguridad de México frente a Estados Unidos y Canadá: retos de cooperación para Norteamérica, Norteamérica, Revista Académica del CISAN-UNAM, año 19, número 1, enero-junio de 2024, DOI: <https://doi.org/10.22201/cisan.24487228e.2024.1.663> <https://www.revistanorteamerica.unam.mx/index.php/nam/article/view/663/898>

Al-Shehari et al, (2021) An Insider Data Leakage Detection Using One-Hot Encoding, Synthetic Minority Oversampling and Machine Learning Techniques, <https://doi.org/10.3390/e23101258>

Alexeev, (2012), Nginx, The Architecture of Open Source Applications, Volume II https://raw.githubusercontent.com/kimth/aosa_pdf/master/The%20Architecture%20of%20Open%20Source%20Applications%202.pdf#page=152

Apache Spark, (2024a), Spark Streaming Programming Guide <https://spark.apache.org/docs/latest/streaming-programming-guide.html>

Apache Spark, (2024b), Cluster Mode Overview, <https://spark.apache.org/docs/latest/cluster-overview.html>

Apache Spark, (2024c), HashingTF, <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.HashingTF.html>

Apache Spark, (2024d), Vectors, <https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.mllib.linalg.Vectors.html>

Baliyan et al, (2022) Mapas de auto-organização - Mapas de Kohonen, <https://acervolima.com/mapas-de-auto-organizacao-mapas-de-kohonen/>

Bank, (2021), Autoencoders, <https://doi.org/10.48550/arXiv.2003.05991>, <https://arxiv.org/pdf/2003.05991>

Benjamin et al, (2018) Fully Understanding the Hashing Trick, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada, <https://dl.acm.org/doi/pdf/10.5555/3327345.3327444>

Berners-Lee, (1994) Uniform Resource Locators (URL), RFC 1738, <https://www.rfc-editor.org/rfc/rfc1738>

Berners-Lee, (2005) Uniform Resource Identifier (URI): Generic Syntax, RFC 3986, <https://www.rfc-editor.org/rfc/rfc3986#section-3.4>

BID, (2020) Ciberseguridad Riesgos, avances y el camino a seguir en América Latina y el Caribe, Reporte Ciberseguridad
<https://publications.iadb.org/es/publications/spanish/viewer/Reporte-Ciberseguridad-2020-riesgos-avances-y-el-camino-a-seguir-en-America-Latina-y-el-Caribe.pdf>

Buenadicha, (2013) El análisis de escalabilidad en la identificación y el diseño de los proyectos de desarrollo, Universidad Nacional de Educación A Distancia,
<https://apidspace.linhd.uned.es/server/api/core/bitstreams/0b2ac56f-a2f7-4cd6-9704-5288c5465f7a/content>

CBurnett, (2019) Colored neural network
https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Colored_neural_network_es.svg/1200px-Colored_neural_network_es.svg.png

Chambers et al, (2018), Spark: The Definitive Guide, Big Data Processing Made Simple, (1st Ed), O'Reilly Media Inc.

Chandola et al, (2009) Anomaly detection: A Survey, ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages. DOI = 10.1145/1541880.1541882
https://vs.inf.ethz.ch/edu/HS2011/CPS/papers/chandola09_anomaly-detection-survey.pdf

Cisco, (2017), Cisco Visual Networking Index: Forecast and Trends, 2017–2022.
<https://futuretimeline.net/data-trends/pdfs/cisco-2017-2022.pdf>

David Gourley et al, (2002), HTTP: The Definitive Guide (3rd Ed.) O'Reilly Media, Inc
https://books.google.com.mx/books?hl=en&lr=&id=3EybAgAAQBAJ&oi=fnd&pg=PR5&dq=http&ots=X70YShhYUm&sig=tSX8D9VtsqYux1ULhfUqVSt-bM&redir_esc=y#v=onepage&q=http&f=false

Docker, (2024a) What is Docker?, <https://docs.docker.com/get-started/docker-overview/>

Docker, (2024b) What is a container?,
<https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-a-container/>

Dominguez, (2023), Políticas de información en ciberseguridad en México: atención y tratamiento a conductas disvalorativas en red, Políticas de información: de lo instrumental a lo informacional. Instituto de Investigaciones Bibliotecológicas y de la Información / unam
https://ru.iibi.unam.mx/jspui/bitstream/IIBI_UNAM/805/1/01_politicas_informacion_rosa_dominguez%20.pdf

Ester et al, (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Institute for ComputerScience, University of Munich,
<https://cdn.aaai.org/KDD/1996/KDD96-037.pdf>

Evans (2011), Internet of Things. La próxima evolución de Internet lo está cambiando todo,
Cisco
https://media.telefonicatech.com/telefonicatech/uploads/2021/1/126528_Internet_of_Things_IoT_IBSG_0411FINAL.pdf

Farley, (1998) Java Distributed Computing, (1st Ed.) O'Reilly Media Inc.
https://books.google.com.mx/books?hl=es&lr=&id=sa06y0EzFx0C&oi=fnd&pg=PR9&dq=distributed+computing+&ots=BKwTr_LTf3&sig=9WrqUFNuodMjW8WM9gdd9ly1dwA&edir_esc=y#v=onepage&q=distributed%20computing&f=false

FBI, (2024) The FBI and International Law Enforcement Partners Intensify Efforts to Combat Illegal DDoS Attacks,
<https://www.fbi.gov/contact-us/field-offices/anchorage/fbi-intensify-efforts-to-combat-illegal-ddos-attacks>

Finlayson et al, (1987), Log files: an extended file service exploiting write-once storage, ACM SIGOPS Operating Systems Review, Volume 21, Issue 5, Pages 139 - 1
<https://doi.org/10.1145/37499.3751>

Flohil, (2018), Classification of Motion Behaviour of Animals using Supervised Learning Algorithms, University of Groningen, Faculty of Science and Engineering,
https://fse.studenttheses.ub.rug.nl/18142/1/bAI_2018_FlohilRT.pdf

Garg, (2002), Elements of Distributed Computing, (1st Ed.) Wiley-IEEE Press,
https://books.google.com.mx/books?hl=es&lr=&id=NIVBtVPeR0QC&oi=fnd&pg=PR17&dq=distributed+computing+&ots=6jW2y_MdAt&sig=RTT8CR6i37bAGK1nLsPh38_zXxY&edir_esc=y#v=onepage&q=distributed%20computing&f=false

GCI, (2021) Global Cybersecurity Index, 5th Edition,
<https://www.itu.int/en/ITU-D/Cybersecurity/Pages/global-cybersecurity-index.aspx>

Gonzalez (2020), Algoritmo Agrupamiento Jerárquico – Teoría,
<https://aprendeia.com/algoritmo-agrupamiento-jerarquico-teoria/>

Goodfellow, et al, (2016) Deep Learning, MIT Press, <https://www.deeplearningbook.org/>

Hajibaba et al, (2014) A Review on Modern Distributed Computing Paradigms: Cloud Computing, Jungle Computing and Fog Computing, Journal of Computing and Information Technology, doi:10.2498 /cit.1002381, <https://hrcak.srce.hr/file/185684>

Hosmer et al, (2013). *Applied logistic regression*. John Wiley & Sons. 2nd Edition,
https://www.researchgate.net/profile/Andrew-Cucchiara/publication/261659875_Applied_Logistic_Regression/links/542c7eff0cf277d58e8c811e/Applied-Logistic-Regression.pdf

IBM (2024a) ¿Qué son las redes neuronales?,
<https://www.ibm.com/es-es/topics/neural-networks>

IBM (2024b) Kohonen node,
https://dataplatform.cloud.ibm.com/docs/content/wsd/nodes/kohonen.html?context=cpda_as

Jieming Zhu et al, (2023), Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics, IEEE 34th International Symposium on Software Reliability Engineering (ISSRE) <https://ieeexplore.ieee.org/abstract/document/10301257>

Karsmakers et al, (2007, August). Multi-class kernel logistic regression: a fixed-size implementation. In 2007 International Joint Conference on Neural Networks (pp. 1756-1761). IEEE.
https://www.researchgate.net/publication/221534146_Multi-class_kernel_logistic_regression_A_fixed_size_implementation

Kohonen, (2001) Self-Organizing Maps, Springer Series in Information Sciences, Vol. 30, 3rd edition, <http://cis.legacy.ics.tkk.fi/research/reports/biennial02-03/cis-biennial-report-2002-2003-8.pdf>

Kumar, (2014), THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS, International Journal of Computer Science and Information Technology Research, <https://www.researchpublish.com/upload/book/THE%20OSI%20MODEL%20OVERVIEW%20ON%20THE%20SEVEN%20LAYERS-607.pdf>

Kuschnig et al, (2011) Evaluation of HTTP-based Request-Response Streams for Internet Video Streaming, Institute of Information Technology (ITEC), https://www.itec.aau.at/bib/files/mmsys11_kuschnig_preprint.pdf

Liddy, (2001) Natural Language Processing, Encyclopedia of Library and Information Science, 2nd Ed. Syracuse University, <https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub>

Linkedin, (2019) Detecting and preventing abuse on LinkedIn using isolation forests, Engineering Blog, Data Management, <https://www.linkedin.com/blog/engineering/data-management/isolation-forest>

Liu et al, (2008). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413-422). IEEE.
<https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf?q=isolation-forest>

Lous, (2020), Time on Unix, https://venam.net/blog/pdf/time_on_unix/time.pdf

Maetouq et al, (2018) Comparison of Hash Function Algorithms Against Attacks: A Review, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 8, 2018,

https://thesai.org/Downloads/Volume9No8/Paper_13-Comparison_of_Hash_Function_Algorithms.pdf

Manning et al, (2009) An Introduction to Information Retrieval, Cambridge University Press, Online Edition <https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf>

Manyika et al, (2011), The great transformer: The impact of the Internet on economic growth and prosperity. McKinsey Global Institute <http://ict-industry-reports.com.au/wp-content/uploads/sites/4/2011/11/2011-the-great-transformer-McKinsey-Oct2011.pdf>

Marzell, (2021) Clustering with Machine Learning — A Comprehensive Guide, <https://rocketloop.de/en/blog/clustering-machine-learning-comprehensive-guide/>

Matsudaira (2012), Scalable Web Architecture and Distributed Systems, The Architecture of Open Source Applications, Volume II https://raw.githubusercontent.com/kimth/aosa_pdf/master/The%20Architecture%20of%20Open%20Source%20Applications%202.pdf#page=152

Molero, (2015), Desarrollo de un sensor virtual de flujo usando Redes Neuronales Artificiales para Bombas de Cavidades Progresivas en el campo San Diego de Cabrutica,

https://www.researchgate.net/publication/314151933_Desarrollo_de_un_sensor_virtual_de_flujo_usando_Redes_Neuronales_Artificiales_para_Bombas_de_Cavidades_Progresivas_en_el_campo_San_Diego_de_Cabrutica

Mozilla Foundation (2024a), HTTP messages, <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

Mozilla Foundation (2024b), HTTP request methods, <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

Mozilla Foundation (2024c), Códigos de estado de respuesta HTTP, <https://developer.mozilla.org/es/docs/Web/HTTP>Status>

Mukesh et al, (2023) A Review on Apache Spark, Proceedings of the KILBY 100 7th International Conference on Computing Sciences 2023 (ICCS 2023), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4492445

Mullen et al, (2018) Fast, Consistent Tokenization of Natural Language Text, Journal of Open Source Software, 3(23), 655. <https://doi.org/10.21105/joss.00655>, <https://www.theoj.org/joss-papers/joss.00655/10.21105.joss.00655.pdf>

Nadkarni et al, (2011) Natural language processing: an introduction, J Am Med Inform Assoc 2011;18:544e551. doi:10.1136/ajmijnl-2011-000464, <https://academic.oup.com/jamia/article-pdf/18/5/544/5962687/18-5-544.pdf>

- Netcraft, (2024), Resources Web Server Survey
<https://www.netcraft.com/resources/?topic=web-server-survey>
- Nginx, (2024), Configuring Logging,
<https://docs.nginx.com/nginx/admin-guide/monitoring/logging/>
- NVD, (2024), CVE-2016-5674 Detail <https://nvd.nist.gov/vuln/detail/CVE-2016-5674>
- Rahal, (2020) A Distributed Architecture for DDoS Prediction and Bot Detection, IEEE Access, vol. 8, pp. 159756-159772, 2020, doi: 10.1109/ACCESS.2020.3020507
<https://ieeexplore.ieee.org/abstract/document/9180362>
- Rashidi et al, (2019) Artificial Intelligence and Machine Learning in Pathology: The Present Landscape of Supervised Methods. *Academic Pathology*. 2019;6. doi:[10.1177/2374289519873088](https://doi.org/10.1177/2374289519873088)
<https://journals.sagepub.com/doi/full/10.1177/2374289519873088>
- Reddy (2014), A new clustering algorithm based on Voronoi diagram, International Journal of Data Mining Modelling and Management, DOI:[10.1504/IJDMMM.2014.059977](https://doi.org/10.1504/IJDMMM.2014.059977)
https://www.researchgate.net/publication/264837631_A_new_clustering_algorithm_base_d_on_Voronoi_diagram
- Reinsel, (2003) Elements of Multivariate Time Series Analysis (Springer Series in Statistics), (2nd Ed.), Springer,
https://books.google.com.mx/books?hl=en&lr=&id=dDBmP1V5664C&oi=fnd&pg=PR7&dq=multivariate+time+series&ots=gp5ugnTwSG&sig=IJP0TVCrLdxSsOTPgcu8S5rsqsQ&redir_esc=y#v=onepage&q=multivariate%20time%20series&f=false
- Rueda (2024), Clusterización espacial: K-means vs DBSCAN. Descripción y ejemplo práctico con QGIS – COLUMNA DE INVESTIGACIÓN DATLAS,
<https://blogdatlas.wordpress.com/2024/04/06/clusterizacion-espacial-k-means-vs-dbscan-descripcion-y-ejemplo-practico-con-qgis-columna-de-investigacion-datlas/>
- Sardjono et al, (2021), The relationship between internet growth and implementation of the internet of things, Journal of Physics: Conference Ser. 1836 012030
<https://iopscience.iop.org/article/10.1088/1742-6596/1836/1/012030/pdf>
- Saxena, (2014), OSI Reference Model – A Seven Layered Architecture of OSI Model, International Journal of Research (IJR) Vol-1, Issue-10 November 2014 ISSN 2348-6848,
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fde021097ce7440a44ecf61470778d72f99e307f>
- Shenglin et al, (2023) An Empirical Analysis of Anomaly Detection Methods for Multivariate Time Series, Haihe Laboratory of Information Technology Application

Innovation, DOI: 10.1109/ISSRE59848.2023.00014,
https://nkcs.iops.ai/wp-content/uploads/2023/10/Empirical_Analysis.pdf

Silva et al (2023). Logistic regression: an example for the prediction of heart attacks. *LACCEI*, 1(8).
<https://proceedings.laccei.org/index.php/laccei/article/download/3049/3044>

Smith et al, (2008) Software Performance Engineering,
[https://www.academia.edu/79832121/Chapter_16_SOFTWARE_PERFORMANCE_ENG
INEERING](https://www.academia.edu/79832121/Chapter_16_SOFTWARE_PERFORMANCE_ENGINEERING)

Sommerlad, (2003), Reverse Proxy Patterns,
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fd604b5fb45cfdf841f97f268a5e125dc47d8c7>

Staerman, et al, (2019). Functional isolation forest. In *Asian Conference on Machine Learning* (pp. 332-347). PMLR.
<https://proceedings.mlr.press/v101/staerman19a/staerman19a.pdf>

Stefano et al 2000. To reject or not to reject: that is the question: An answer in the case of neural classifiers. *IEEE Trans. Syst. Manag. Cyber.* 30, 1, 84–94.
<https://dl.acm.org/doi/10.1109/5326.827457>

UNCTAD, (2024) Cybercrime Legislation Worldwide, United Nations Trade and Development <https://unctad.org/page/cybercrime-legislation-worldwide>

UpGuard (2024), What is a Reverse Proxy Server? Learn How they Protect You,
<https://www.upguard.com/blog/what-is-a-reverse-proxy>

USC, (2010) United States Constitution, Título 18, Capítulo 4,
<https://www.govinfo.gov/content/pkg/USCODE-2010-title18/html/USCODE-2010-title18-partI-chap47-sec1030.htm>

Valkenborg, et al, (2023). Unsupervised learning. *American Journal of Orthodontics and Dentofacial Orthopedics*, 163(6), 877-882.
<https://www.ajodo.org/action/showPdf?pii=S0889-5406%2823%2900193-2>

Van Hulle et al (2012) Self-organizing Maps. *Handbook of natural computing*, 1, 585-622. http://www.pspc.unige.it/~drivsco/Papers/VanHulle_Springer.pdf

W3C, (2001), Transport Message Exchange Pattern: Single-Request-Response
<https://www.w3.org/2000/xp/Group/1/10/11/2001-10-11-SRR-Transport MEP>

West (2015), Digital divide: Improving Internet access in the developing world through affordable services and diverse content, Center for Technology Innovation at Brookings
https://www.brookings.edu/wp-content/uploads/2016/06/West_Internet-Access.pdf

William, (2019) Multivariate Time Series Analysis and Applications (Wiley Series in Probability and Statistics), (1st Ed.) Wiley,
https://books.google.com.mx/books?hl=en&lr=&id=9naCDwAAQBAJ&oi=fnd&pg=PP13&dq=multivariate+time+series&ots=6AgR_0toPY&sig=6TwdllobKjlBbND5vzrD3VFkQCR_E&redir_esc=y#v=onepage&q=multivariate%20time%20series&f=false

Yādava (2009), Introduction To Client Server Computing (1st Ed.) New Age International Publisher
https://www.google.com.mx/books/edition/Introduction_To_Client_Sever_Computing/AD0TjdCRVEkC?hl=en&gbpv=1