Detección de anomalías mediante aprendizaje automático en tráfico de servidores web

Mitsiu Alejandro Carreño Sarabia
Maestría en Ciencia de Datos

Agenda

- Problemática
- Objetivos
- Metodología
- Resultados y conclusiones

Problemática

Problemática

Evitar analizar el tráfico de servidores puede impactar en múltiples contextos:

- Tener **infraestructura insuficiente**, afecta la calidad del servicio ofertado.
- Tener **infraestructura excedente**, tiene repercusiones monetarias la pagar por recursos no empleados.
- **No detectar cambios en el uso del servicio**, reduce la comprensión de uso y necesidades de los clientes.
- Sufrir **ataques informáticos**, pone en riesgo la integridad y seguridad del sistema así como la información almacenada
- Formar parte de **botnets**, implica costos de ancho de banda, así como estresar redes y recursos.

Objetivos

Objetivo general

El objetivo de este proyecto es explorar la **implementación de técnicas heurísticas**, así como de **aprendizaje automático** para **determinar si la actividad y tráfico de un servidor web es anómala**, generando un sistema integral de monitoreo y detección de tráfico anómalo que sea **capaz de analizar grandes cantidades de datos** de manera automática, y a la vez **permitir la constante actualización de patrones**, ajustando el concepto de comportamiento normal y detectando nuevas anomalías.

Objetivo específicos

- Desarrollar o implementar un **algoritmo que permita la detección de anomalías** que sea tolerante a grandes cantidades de datos y ofrezca **resultados de calidad en un tiempo manejable.**
- Desarrollar una **infraestructura que permita el entrenamiento y alojamiento de múltiples modelos**, dando flexibilidad a la temporalidad del análisis.
- Desarrollar una infraestructura que permita **alojar múltiples clientes**, posibilitando la **escalabilidad horizontal.**

Metodología

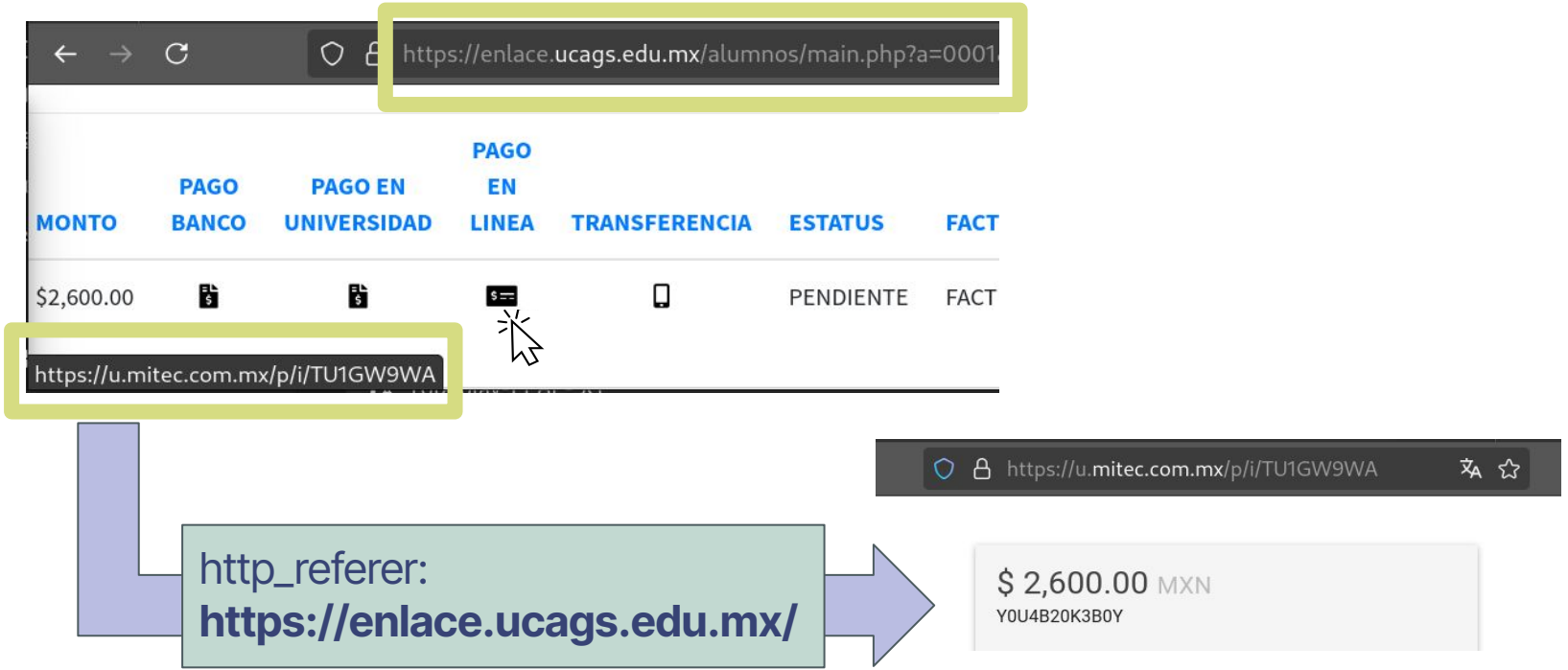
Variables

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

Formato del contenido en archivo access.log generado por NGINX.
Fuente: NGINX, 2024

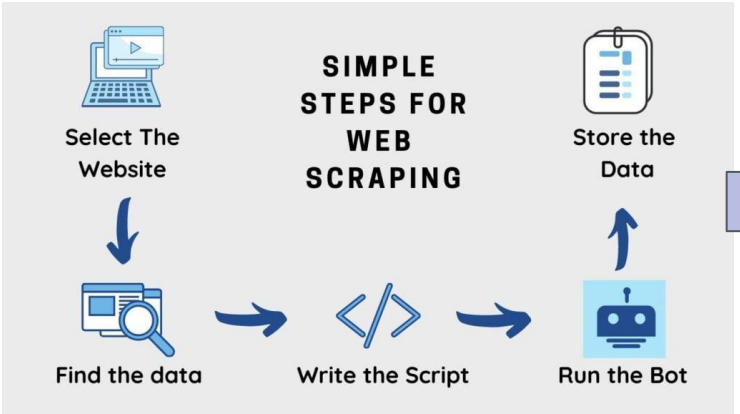
```
45.166.93.223 - - [23/Aug/2024:00:00:20 +0000] "GET
/api/manual/find/?category=De%20todo%20un%20poco&searchIn=category&page=1&limit=12&sear
ch=%7B%22searchAllStatuses%22%3Atrue%2C%22searchParam%22%3A%22De%20todo%20un%
20poco%22%7D HTTP/1.1" 304 0 "https://a.com/manual/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
```

Http_referer – Enlace a otro dominio



Ejemplo de http_referer entre dominios
Fuente propia

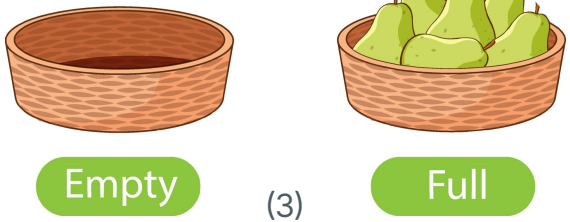
Http_referer – Vacío



(1)

(2)

http_referer:
(vacío)



(3)

Ejemplo de http_referer vacío
Fuente propia

1. analyticslearn.com
2. nzteco.co.nz
3. static.vecteezy.com

Arquitectura

Se cuenta con varios servidores web manejando tráfico a través de NGINX.

Cada servidor maneja **múltiples dominios**.

Pero **NGINX no registra el dominio**.

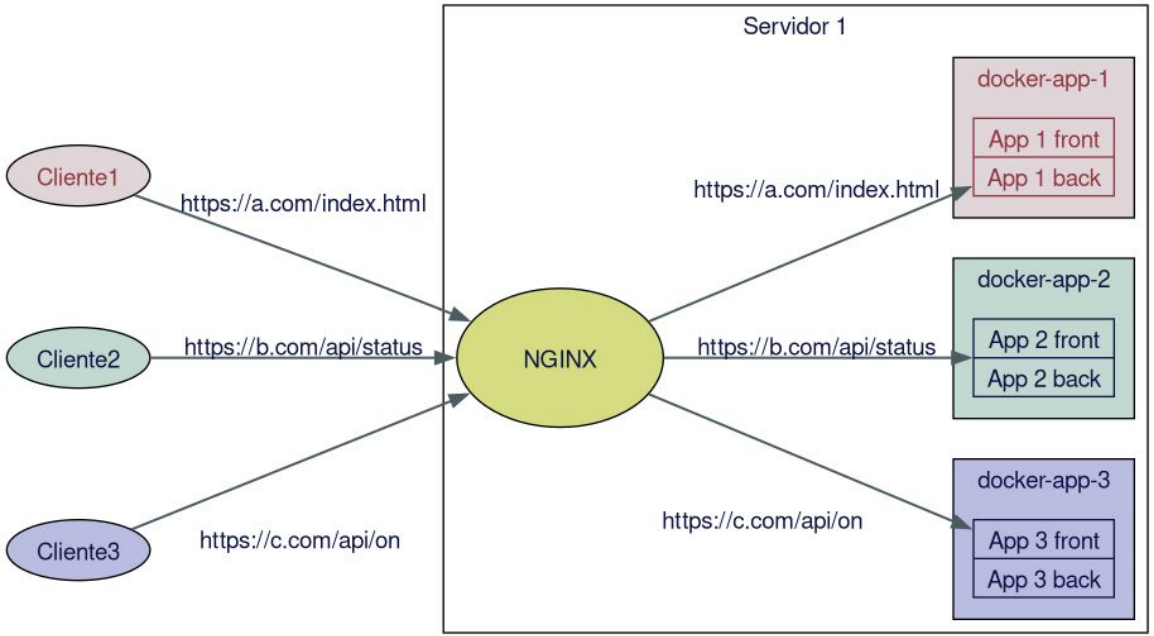


Diagrama de componentes en servidor.
Fuente propia

Metadatos

El tráfico puede llegar de cuatro fuentes (referers):

- **Mismo dominio**

domain_category = http_referer

- **Otro dominio**

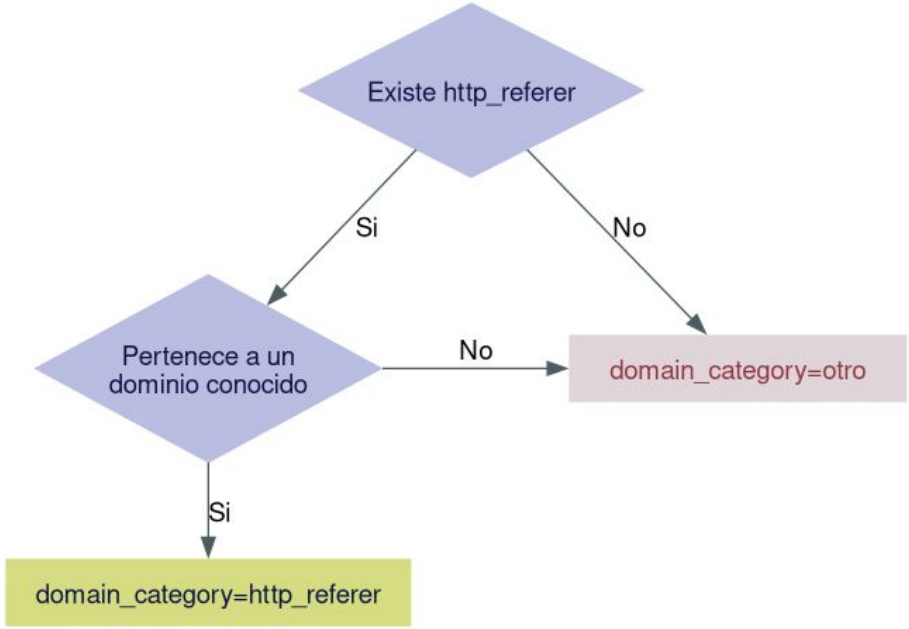
domain_category = otro

- **Sin referer**

domain_category = otro

- ***Otros dominio alojados**

domain_category = falso positivo



Flujo para asignación de domain_category
Fuente propia

Metadata

Variable	Valor	Variable	Valor	Variable	Valor
remote_addr	45.166.93.223	http_ver	HTTP/1.1	body_bytes_sent	0
remote_usr	(Vacío)	status	304	http_referer	https://youtube.com.com/
fdate_time	23/Aug/2024:00:00:20	method	GET	domain (de http_referer)	youtube.com
clean_path	/api/manual/find/	day_week	4	domain_category	otro
req_uri	/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				
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"}				
clean_query_list	[“category=De todo un poco”, “searchIn=category”, “page=1”, “limit=12”, `search={"searchAllStatuses":true,"searchParam":"De todo un poco"}`]				

Información de archivo de registros estructurada y enriquecida
Fuente propia

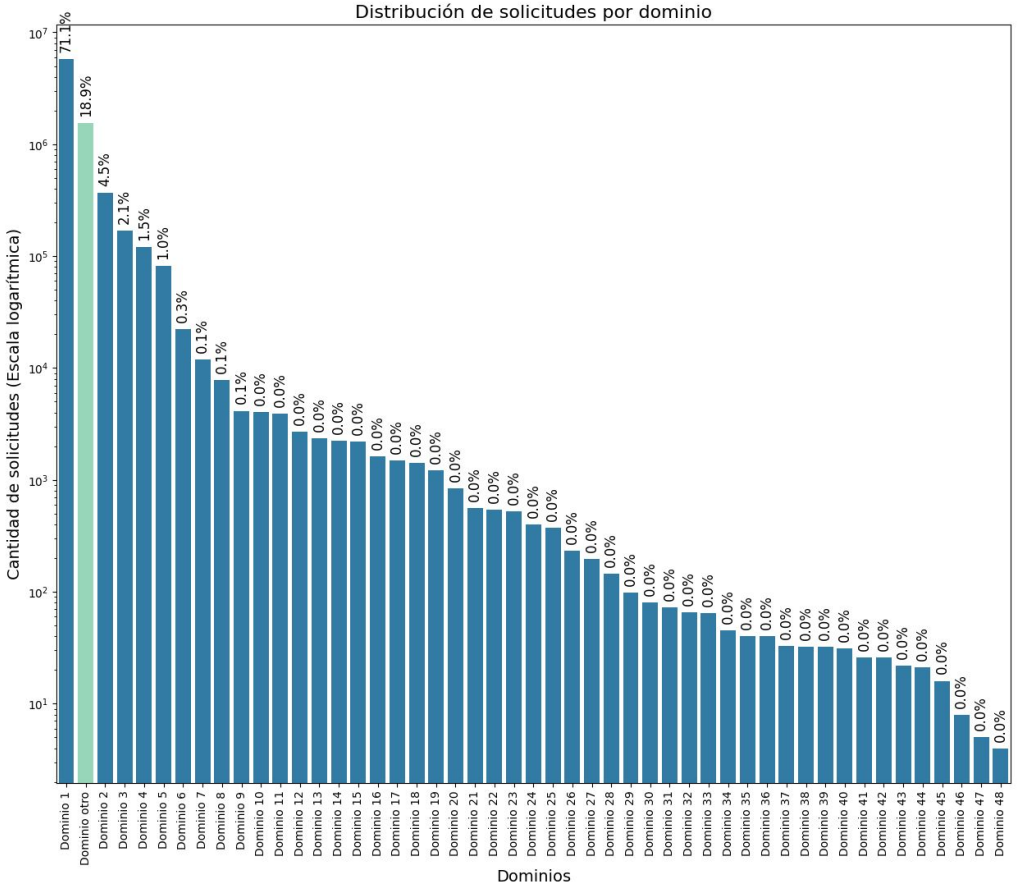
Contexto estadístico

El servidor estudiado aloja 48 dominios.

Se dió seguimiento por 72 días.

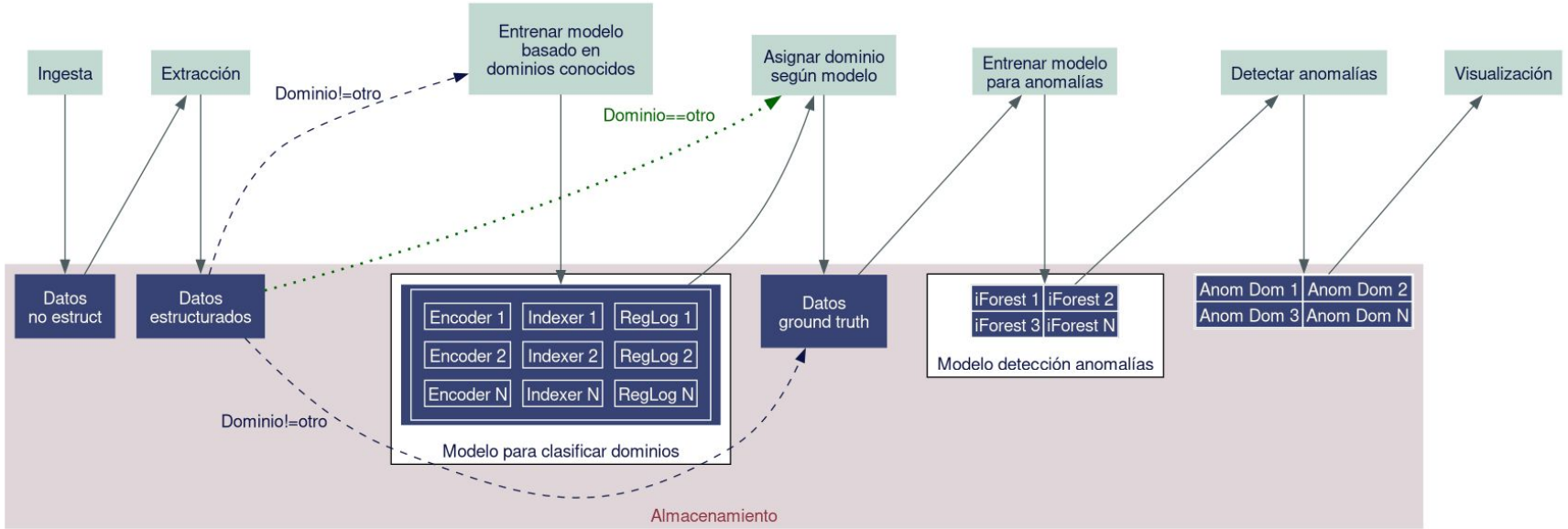
Se captaron 8,170,910 de conexiones totales.

Aproximadamente 20% se clasificó como dominio= otro



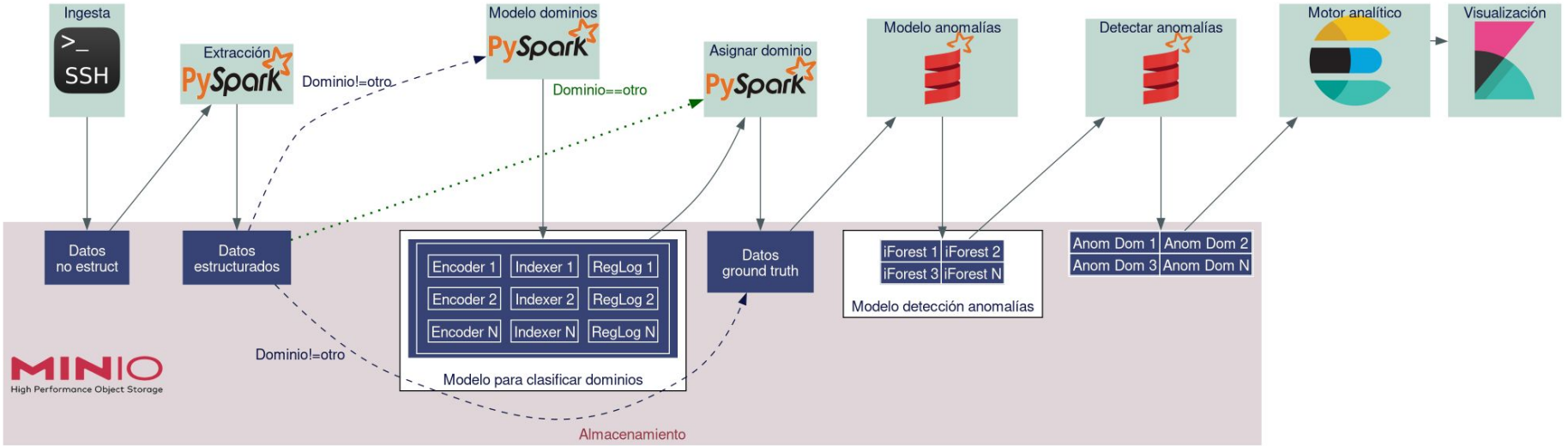
Distribución de solicitudes por dominio
Fuente propia

Flujo de trabajo



Flujo de procesamiento para entrenamiento de modelos.
Fuente propia

Solución tecnológica – Implementación



Implementación tecnológica para entrenamiento de modelos.
Fuente propia

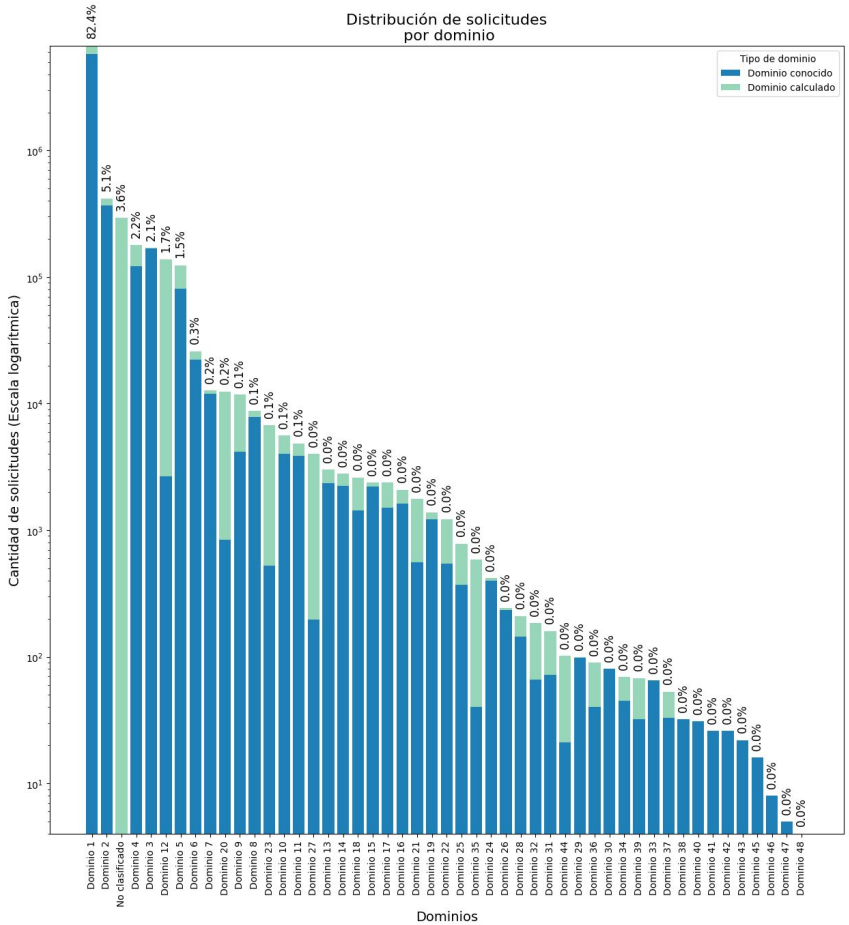
Resultados y conclusiones

Regresión logística – Resultados

Se asignaron 1,249,397 peticiones a sus dominios, **disminuyendo 17%** la cantidad de peticiones sin dominio.

Prueba	Resultado
Precisión	98.82%
Precisión ponderada (Weighted precision)	98.91%
Exhaustividad ponderada (Weighted recall)	98.97%
Puntaje F1	98.82%

Métricas para evaluar el desempeño del modelo de clasificación de dominios.
Fuente propia



Distribución de solicitudes por dominio
Fuente propia

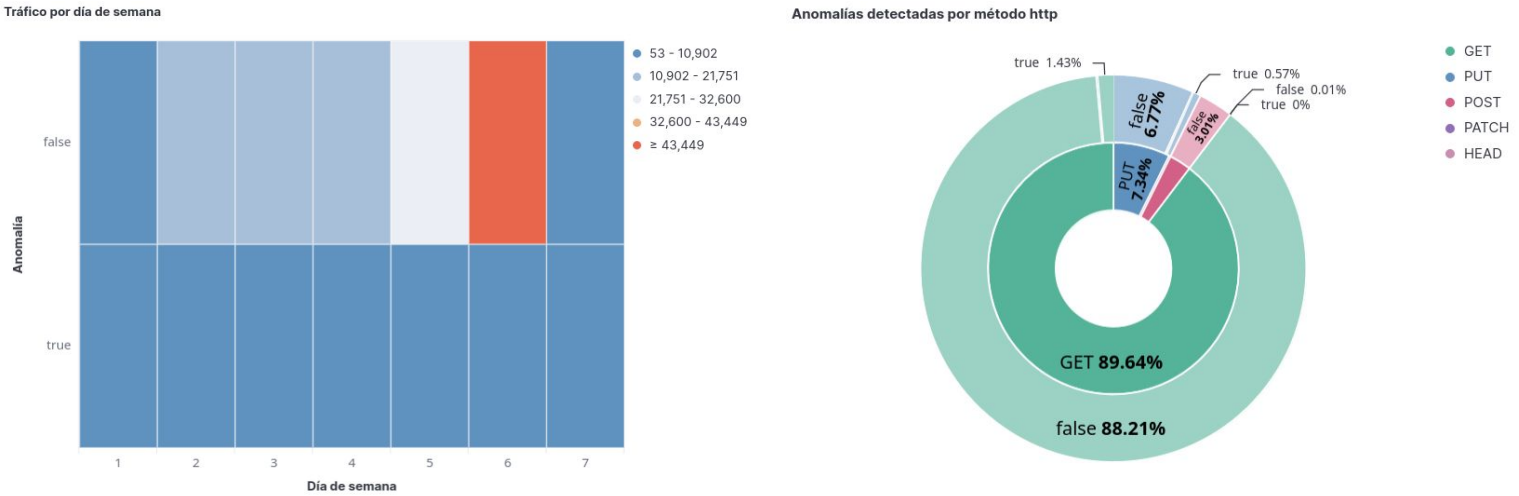
Anomalías

```
/__debugging_center_utils___.php?log=;echo ljyabmwesqxkknnjecooewtpopjvuxsk|id
/__debugging_center_utils___.php?log=;echo ljyabmwesqxkknnjecooewtpopjvuxsk|ipconfig
/services/auth/config/aws_credentials.json
/plus/recommend.php?action=&aid=1&_FILES[type][tmp_name]=\x5C' or mid=@`\x5C'`
/*!50000union*//*!50000select*/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
```

Peticiones anómalas maliciosas detectadas
Fuente propia

Visualizaciones

Mediante Kibana se realizaron tableros de visualización que ofrecen información sobre tendencias de uso del servidor, así como métricas para dar contexto a las peticiones marcadas como anómalas.



Tablero de visualizaciones desarrollado en kibana
Fuente propia

Apache Spark – Cómputo Distribuido

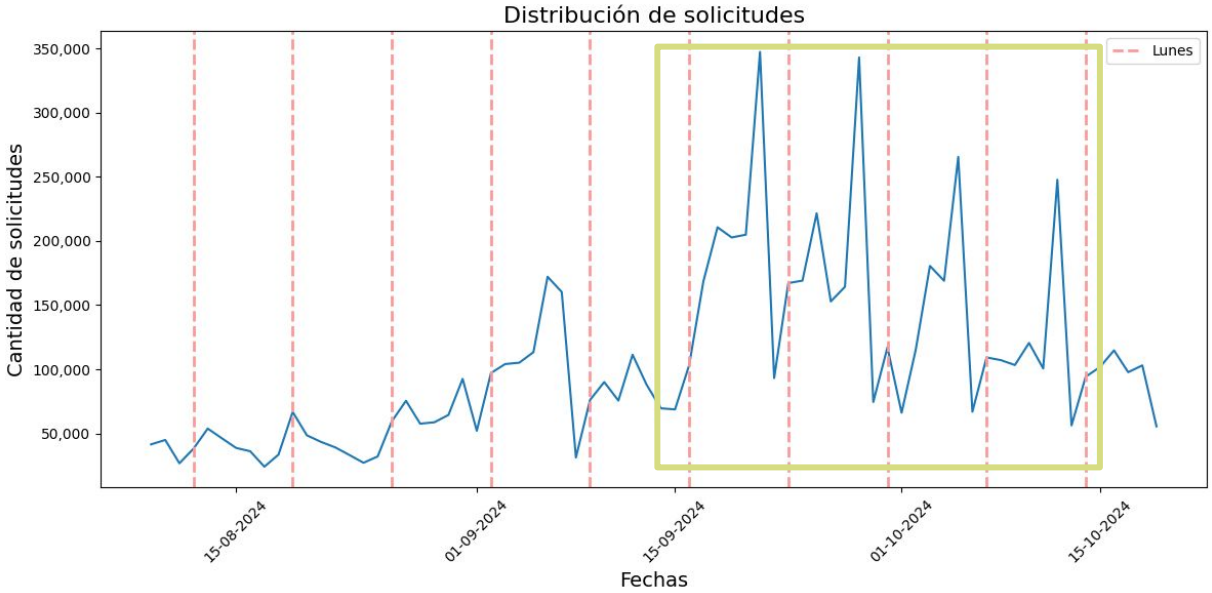
Con la arquitectura y configuración actual se consiguió ofrecer una plataforma y código **escalable, tolerante a grandes volúmenes de información**.

Se considera que se lograron establecer las **bases tecnológicas y algorítmicas** para generar un sistema que exitosamente pueda **analizar y obtener conocimiento** de grandes volúmenes de información.

Tarea	Tiempo de procesamiento	Volumen de información
Extracción	35 minutos	8,170,910 registros
Entrenamiento de clasificador	3 hora, 6 minutos en entrenar el modelo basado en	6,636,438 registros
Predicción de modelo	1 minuto 30 seg	1,543,472 de registros
Generar ground truth	1 minuto	8,169,584 registros
Detección de anomalías	Aproximadamente 20 minutos por dominio	

Tabla de rendimiento (tiempo/volumen) de sistema
Fuente propia

Trabajo futuro – Series de tiempo



Línea de tiempo con total de solicitudes al servidor
Fuente propia

Se detecta un punto de mejora en aplicar **análisis de serie de tiempo,** ya que la solución actual analiza el tráfico **a nivel de petición unitaria**, ignorando tendencias de volumen y tiempo.

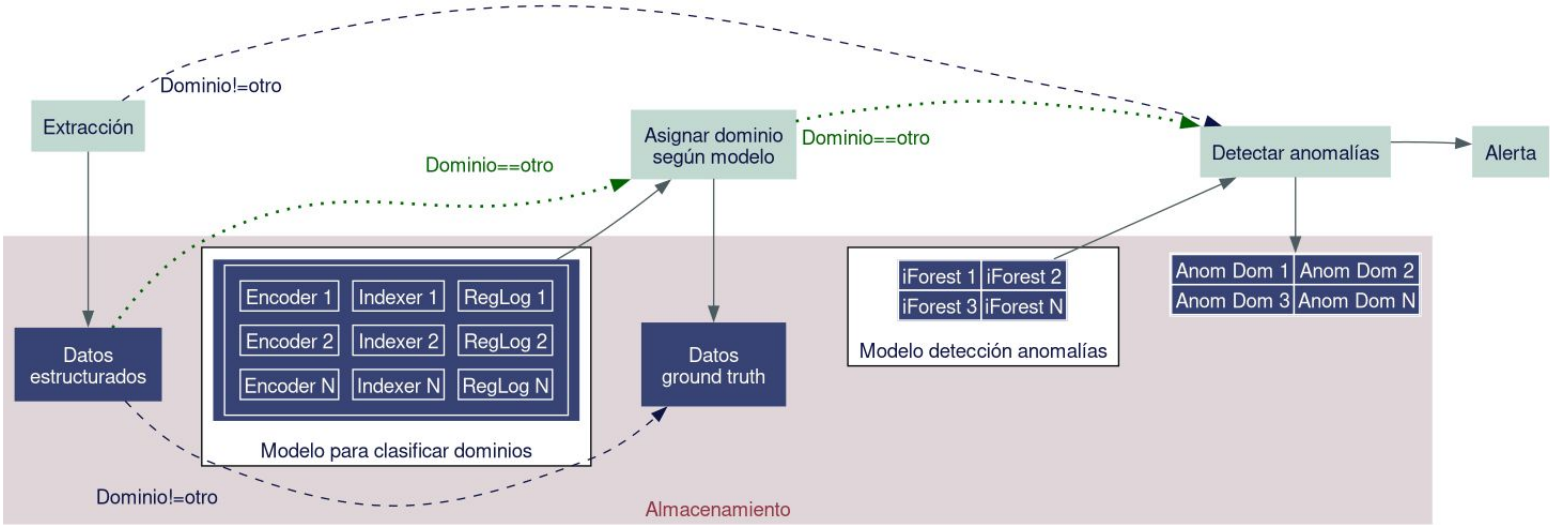
Trabajo futuro – Clusterización de dominios

A pesar de que el servidor maneja **48 dominios distintos, no significa que sean 48 proyectos distintos**, es común ofrecer el **mismo software a múltiples clientes,** aplicar técnicas de clusterización puede ayudar a expandir el ground truth de los datos.

Esto puede **ayudar a capturar el comportamiento normal** de un proyecto **repartido entre múltiples clientes.**

Trabajo futuro – Implementación comercial

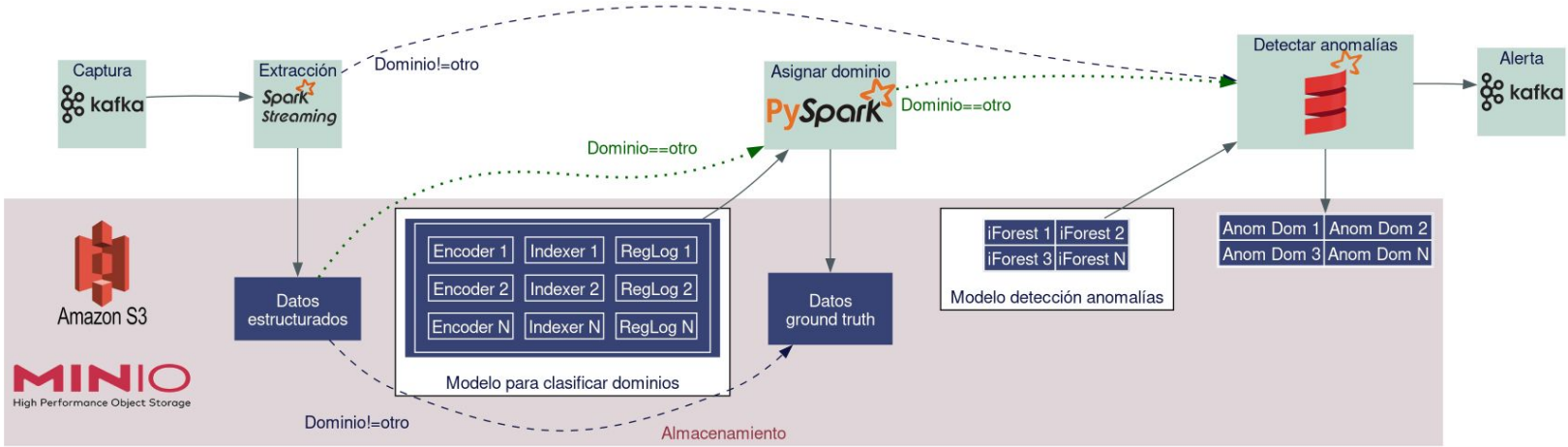
Con los modelos entrenados, se propone el siguiente flujo para la detección automática de anomalías



Propuesta de flujo para implementación productiva
Fuente propia

Trabajo futuro – Implementación comercial

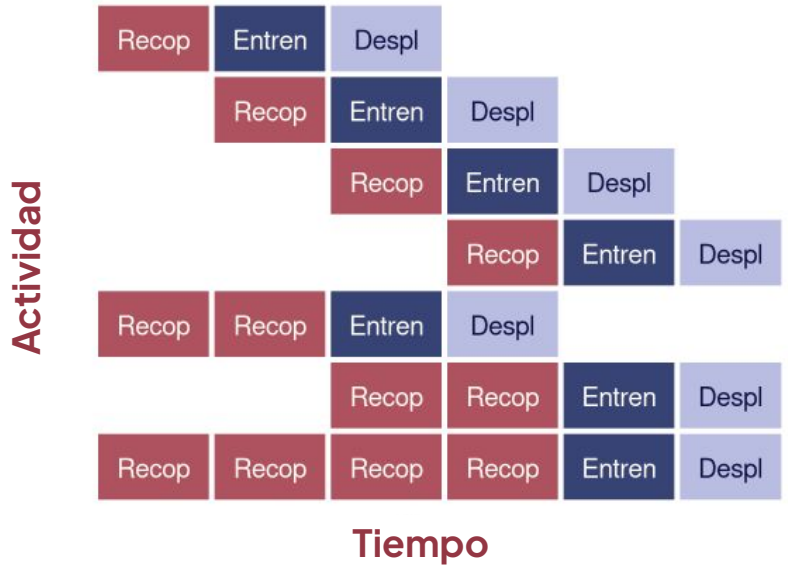
Aprovechando **tecnologías de flujos de datos** como Apache kafka y Spark Streaming



Propuesta de flujo para implementación productiva
Fuente propia

Trabajo futuro – Implementación comercial

Se propone un **flujo paralelizado de entrenamiento**, permitiendo captar y entrenar múltiples temporalidades y ofreciendo **diversos contextos históricos**.



Propuesta de flujo de entrenamiento paralelizado
Fuente propia

Fuentes y referencias

- 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
- Nginx, (2024), Configuring Logging, https://docs.nginx.com/nginx/admin-guide/monitoring/logging/
- 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://journals.sagepub.com/doi/full/10.1177/2374289519873088