Trabajo de Fin de Máster

Máster Universitario en Ciberseguridad e Inteligencia de los Datos

Estudio Comparativo de algoritmos para la detección de ciber ataques SQL con IA

Comparative study of algorithms for detecting SQL cyber attacks with AI

Alba Maura Candelario Brito

La Laguna, 6 de julio de 2024

D. **Cándido Caballero Gil**, profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Jezabel Miriam Molina Gil**, profesora Contratado Doctor de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutora

C E R T I F I C A (N)

Que la presente memoria titulada:

”*Estudio Comparativo de algoritmos para la detección de ciber ataques SQL con IA*”

ha sido realizada bajo su dirección por D. **Alba Maura Candelario Brito**, con N.I.F. 42.255.273-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de julio de 2024

Agradecimientos

Quiero darles las gracias a mis tutores por ayudarme en el proceso de elaboración de este proyecto. Y, por supuesto, a mi familia y amigos, por estar siempre ahí.

Licencia

Resumen

Las inyecciones SQL representan una de las amenazas más comunes y peligrosas en la seguridad de las bases de datos, permitiendo a los atacantes manipular consultas SQL para acceder o modificar información sensible. Detectar y prevenir estas inyecciones es crucial para mantener la integridad y seguridad de los sistemas de información.

En este Trabajo de Fin de Máster, se abordará el problema de la clasificación de inyecciones SQL mediante la aplicación de varios algoritmos de aprendizaje automático: k-Nearest Neighbors (k-NN), Árboles de Decisión, Regresión Logística y Máquinas de Vectores de Soporte (SVM). Además de explicar matemáticamente en qué se fundamentan para llevar a cabo esta tarea de clasificación.

El estudio incluirá la limpieza, comprensión y transformación de las características; y el entrenamiento y prueba para evaluar los modelos. Además, se utilizarán técnicas de validación cruzada y búsqueda de hiperparámetros para optimizar el rendimiento de los modelos y se evaluarán utilizando métricas de rendimiento como precisión, recall y F1-score. Los mejores modelos serán validados adicionalmente con una base de datos diferente para confirmar su capacidad de generalización.

Este proyecto tiene como objetivo principal identificar el algoritmo más eficaz para la clasificación de inyecciones SQL y proporcionar una implementación robusta que pueda ser utilizada en entornos reales para mejorar la seguridad de las bases de datos.

Palabras clave: Inyecciones SQL, Ciberseguridad, Ciberataques, Clasificación, k-Nearest Neighbors, Árboles de Decisión, Regresión Logística, Máquinas de Vectores de Soporte, Machine Learning

Abstract

SQL injections represent one of the most common and dangerous threats to database security, allowing attackers to manipulate SQL queries to access or modify sensitive information. Detecting and preventing these injections is crucial for maintaining the integrity and security of information systems.

In this Master's Thesis, the problem of classifying SQL injections will be addressed through the application of various machine learning algorithms: k-Nearest Neighbors (k-NN), Decision Trees, Logistic Regression, and Support Vector Machines (SVM). Additionally, the mathematical foundations of these algorithms for classification tasks will be explained.

The study will include data cleaning, understanding, and transformation of features; and the training and testing phases to evaluate the models. Techniques such as cross-validation and hyperparameter tuning will be used to optimize model performance, and they will be evaluated using performance metrics such as accuracy, recall and F1-score. The best models will be further validated with a different database to confirm their generalization capability.

The primary objective of this project is to identify the most effective algorithm for SQL injection classification and to provide a robust implementation that can be used in real-world environments to improve database security.

Keywords: SQL Injections, Cybersecurity, Cyberattacks, Classification, k-Nearest Neighbors, Decision Trees, Logistic Regression, Support Vector Machines, Machine Learning

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Hoy en día, muchas organizaciones, tanto públicas como privadas han hecho la transición de sus actividades a términos digitales, permitiendo que su información esté disponible en bases de datos, más fácilmente accesibles a través de internet. Este cambio ha permitido una mayor eficiencia y accesibilidad en el manejo de datos, mejorando los procesos internos y la capacidad de respuesta hacia clientes y usuarios. Las bases de datos se han convertido en el corazón de las operaciones de las empresas, ya que almacenan información crucial como datos de clientes, registros financieros, inventarios, etc.

Sin embargo, con este aumento en la digitalización y la dependencia de las bases de datos, también han surgido nuevos riesgos. Las personas con intenciones maliciosas, que buscan acceder o apropiarse de esta información, explotan las vulnerabilidades en las aplicaciones y sistemas implementados por las organizaciones. Estos atacantes, motivados por diversos intereses que pueden ir desde el espionaje industrial hasta el robo de datos personales para actividades fraudulentas, utilizan diversas técnicas para infiltrarse en los sistemas de las empresas.

Una de las técnicas más comunes es la inyección de código SQL (Structured Query Language). Este tipo de ataque se aprovecha de las deficiencias en la seguridad de las aplicaciones web, insertando código malicioso en los campos de entrada de formularios o directamente en la URL del sitio web. Si una aplicación web no valida adecuadamente estas entradas, el código malicioso puede ser ejecutado por la base de datos, permitiendo al atacante acceder, modificar o eliminar información crítica.

Este tipo de ataque no solo compromete la integridad y la confidencialidad de los datos, sino que también puede llevar a pérdidas financieras significativas y daños a la reputación de la organización. En este trabajo de fin de máster se propone un análisis comparativo de los algoritmos de aprendizaje automático para mitigar los ataques de inyección SQL.

En un primer lugar, se introducen los conceptos básicos de ciberseguridad y los tipos principales de ataques, poniendo el foco en las inyecciones SQL. Posteriormente, se presenta una revisión de los algoritmos de machine learning que se van a utilizar: Regresión Logística, K-Nearest Neighborgs, Árboles de Decisión y Support Vector Machine. Estos cuatro modelos han sido trabajados en las asignaturas de Extracción de Conocimiento de Bases de Datos y Técnicas Avanzadas Para Análisis de Datos. En tercer lugar, se lleva a cabo la etapa de análisis de datos, comenzado con un preprocesado de la base de datos que se va a utilizar. Después, se hará una comparación del rendimiento en cuanto a distin-

tas métricas de los algoritmos de machine learning a la hora de detectar inyecciones SQL. La implementación de estos algoritmos se realizará en un Jupyter Notebook utilizando la librería Pandas y Scikit-Learn de Phyton. Luego, se discuten los resultados obtenidos. En último lugar, se presentan las conclusiones del estudio y recomendaciones para futuras investigaciones.

Capítulo 2

Contextualización

Esta sección proporciona una idea general del propósito de la ciberseguridad, de los principales tipos de ciberataques a los que se enfrenta y su impacto para las empresas.

2.1. Ciberataques

Los ciberataques se han convertido en una de las principales amenazas para la seguridad de las organizaciones, tanto públicas como privadas. Un ciberataque se define como cualquier esfuerzo intencional para robar, exponer, alterar, deshabilitar o destruir datos, aplicaciones u otros activos a través del acceso no autorizado a una red, sistema informático o dispositivo digital.

Estos ataques pueden tener consecuencias devastadoras, incluyendo la pérdida de información confidencial, interrupciones operativas y daños significativos a la reputación y las finanzas de una organización. Estos ataques suelen estar destinados a acceder, modificar o destruir información sensible, extorsionar a los usuarios o interrumpir los procesos normales del negocio.

La ciberseguridad se define como la práctica de proteger sistemas, redes y programas de estos ataques digitales. Entre sus principios fundamentales se incluyen (Figura 2.1):

Confidencialidad: Asegurar que la información no esté disponible o divulgada a individuos, entidades o procesos no autorizados.

Integridad: Garantizar la exactitud y completitud de la información y sus métodos de procesamiento.

Autenticidad: Controlar que la información de la que dispone la organización es legítima.

Disponibilidad: Asegurar que los usuarios autorizados tengan acceso a la información y a los activos asociados cuando lo requieran. (1)

Figura 2.1: Principios ciberseguridad

2.2. Tipos de ciberataques

2.2.1. Programa malicioso (malware)

El malware es un software malicioso que puede destruir datos, robar información o incluso borrar archivos críticos para la capacidad de ejecución del sistema operativo. El malware se presenta en varias formas (Figura 2.2):

- Los troyanos se disfrazan de programas útiles o se esconden dentro de software legítimo para engañar a los usuarios para que los instalen.
- El ransomware es un malware sofisticado que utiliza un cifrado sólido para mantener como rehenes los datos o los sistemas. Los ciberdelincuentes exigen un pago a cambio de liberar el sistema y restaurar la funcionalidad.
- El spyware es un tipo de malware que recopila secretamente información confidencial, como nombres de usuario, contraseñas y números de tarjetas de crédito. Luego envía esta información al atacante.
- Los rootkits son paquetes de malware que permiten a los piratas informáticos obtener acceso de nivel de administrador al sistema operativo de un equipo u otros activos.
- Los gusanos son códigos maliciosos autorreplicantes que pueden propagarse automáticamente entre aplicaciones y dispositivos. (1)

2.2.2. Ingeniería social

Los ataques de ingeniería social manipulan a las personas para que compartan información delicada, descarguen software malicioso o envíen dinero a los delincuentes.

El phishing es uno de los ataques de ingeniería social más generalizados. Las estafas más básicas de phishing utilizan correos electrónicos falsos o mensajes de texto para robar las credenciales de los usuarios, exfiltrar datos confidenciales o difundir malware.

Figura 2.2: Tipos de malware

Normalmente dirigen a la víctima a hacer clic en un hipervínculo que los lleva a un sitio web malicioso o abre un archivo adjunto de correo electrónico que resulta ser malware.

El spear phishing es un ataque muy específico que tiene como objetivo manipular a una persona específica y, a menudo, utiliza detalles de los perfiles públicos de las redes sociales de la víctima. (1)

2.2.3. Ataques web

Script entre sitios (XSS): Los ataques de secuencias de comandos entre sitios insertan código malicioso en una página web o aplicación web legítima. Cuando un usuario visita el sitio o la aplicación, el código se ejecuta automáticamente en el navegador web del usuario, generalmente robando información confidencial o redirigiendo al usuario a un sitio web falsificado. Los atacantes suelen utilizar JavaScript para ataques XSS.

Inyección SQL: Los ataques de inyección SQL usan lenguaje de consulta estructurado (SQL) para enviar comandos maliciosos a la base de datos back-end de un sitio web o aplicación. Los hackers ingresan los comandos a través de campos orientados al usuario como barras de búsqueda y ventanas de inicio de sesión. Los comandos se pasan a la base de datos, solicitándole que devuelva datos privados como contraseñas o detalles del cliente.

Ataques de denegación de servicio (Figura 2.3): Los ataques de denegación de servicio y denegación de servicio distribuido (DDoS) inundan los recursos de un sistema con tráfico fraudulento. Este tráfico abruma al sistema, evitando las respuestas a solicitudes legítimas y reduciendo la capacidad del sistema. Estos ataques pueden ser un fin en sí mismo o una configuración para otro ataque. Los ataques DDoS suelen llevarse a cabo con una botnet, una red de dispositivos conectados a Internet

e infectados con malware bajo el control de un hacker. Pueden incluir portátiles, smartphones y dispositivos de Internet de las cosas (IoT). (1)

Tunelización DNS: La finalidad de esta técnica es crear un túnel a través del firewall de un equipo o sistema de una organización, para tener acceso a toda su información de forma remota. (3)

Figura 2.3: Ataque DDoS

2.2.4. Otros ataques

Ataques de intermediario (MITM): Un hacker intercepta las comunicaciones entre dos personas o entre un usuario y un servidor. Se llevan a cabo comúnmente a través de redes wifi públicas no seguras, donde es relativamente fácil espiar el tráfico. Así, por ejemplo, pueden leer los correos electrónicos de un usuario o incluso alterarlos antes de que lleguen al destinatario.

Ataques a la cadena de suministro: Son ataques cibernéticos en los que los hackers se dirigen a los proveedores de software, proveedores de materiales y otros proveedores de servicios de las empresas. Dado que los proveedores suelen conectarse a las redes de sus clientes de alguna manera, los hackeres pueden utilizar la red del proveedor como un vector de ataque para acceder a varios destinos a la vez.

En 2020, los rusos piratearon al proveedor de software SolarWinds y distribuyeron programas maliciosos a sus clientes bajo la apariencia de una actualización de software. El malware permitió a los espías rusos acceder a los datos confidenciales de varias agencias gubernamentales de EEUU, incluidos los departamentos del Tesoro, Justicia y Estado. (2)

Exploits de día cero: Aprovechan las vulnerabilidades de día cero, que son vulnerabilidades desconocidas para la comunidad de seguridad o identificadas pero aún no parcheadas. Estas vulnerabilidades pueden existir durante días, meses o años antes de que los desarrolladores las conozcan, lo que las convierte en objetivos prioritarios para los piratas informáticos.

2.3. Impacto y motivación

En 2021, el 90 % de las empresas españolas sufrió un ciberataque, según señalan las consultoras IDC Research y Seidor, con el objetivo de extraer información. (4)

Los ciberataques recibidos en España y gestionados por el Centro Criptológico Nacional (se encarga de la defensa del ciberespacio en lo referente a las administraciones públicas) alcanzaron en 2023 una cifra récord de 107.777 incidentes registrados. Así se refleja en el Informe de Seguridad Nacional 2023, elaborado por el Departamento de Seguridad Nacional, que muestra un aumento del 94 % en los ciberataques respecto al año 2022 (Figura 2.4). (5)

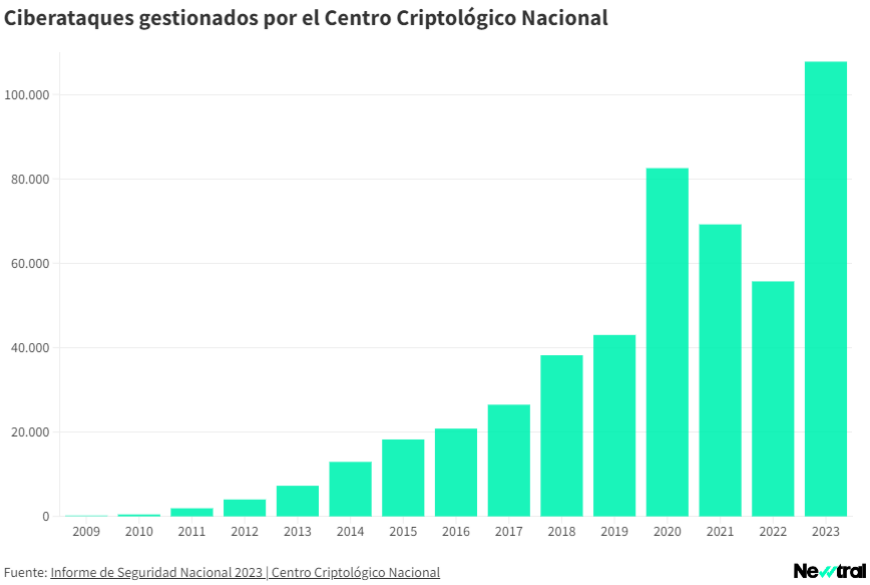


Figura 2.4: Ciberataques gestionados por el CNN

Las motivaciones detrás de los ciberataques pueden ser de tipo criminal, política o personal. Los atacantes con motivaciones delictivas buscan obtener ganancias financieras mediante el robo de dinero, el robo de datos o la interrupción del negocio. Pueden utilizar estafas de ingeniería social para engañar a las personas para que les envíen dinero o robar datos y utilizarlos para venderlos en la Dark Web. La extorsión es otra táctica popular, según el X-Force Threat Intelligence Index, el 27 % de los ciberataques tienen como objetivo extorsionar a sus víctimas.

Los ataques de ransomware han supuesto el pago de rescates de hasta 40 millones de dólares. Es el segundo tipo más común de ciberataque y representa el 17 % de los ataques. Las estafas de compromiso de correo electrónico empresarial han robado hasta 47 millones de dólares a las víctimas en un solo ataque. Los ciberataques que comprometen la información de identificación personal los clientes pueden provocar la pérdida de la confianza de los clientes, multas reglamentarias e incluso acciones legales. Según una estimación, la ciberdelincuencia costará a la economía mundial 10,5 billones de dólares al año de aquí a 2025. (1)

Los agresores con motivaciones personales, como los empleados actuales o antiguos descontentos, buscan principalmente la retribución por algún desaire percibido. Pueden tomar dinero, robar datos confidenciales o interrumpir los sistemas de una empresa.

Los atacantes con motivaciones políticas suelen asociarse con la guerra cibernética, el ciberterrorismo o el "hacktivismo". En la ciberguerra, los estados-nación suelen atacar las agencias gubernamentales o la infraestructura crítica de sus enemigos. Desde el inicio de la Guerra de Rusia y Ucrania, ambos países han experimentado una erupción de ciberataques contra instituciones vitales. (6)

Por ello, la gestión de seguridad en los sistemas de información se ha convertido en una ardua tarea. Además, la mayoría de los fallos de seguridad ocurren debido a violaciones de los controles por parte del personal que interactúa con los sistemas. Cada vez es más necesario que empresas y organizaciones sepan qué es la ciberseguridad e implicar a trabajadores y usuarios, para que formen parte de las políticas de seguridad.

Un alto porcentaje de las vulneraciones es producto de los ataques a los sistemas que presentan la información en línea. Los servicios que brinda la red, específicamente en servicios web, los cuales interactúan con base de datos para enviar y recibir información para los usuarios finales. Es allí donde aplican técnicas de ataques de inyección SQL, y es el tipo de ataque en el que nos centraremos en este trabajo.

El OWASP (Open Web Application Security Project) es una organización sin ánimo de lucro cuyo propósito fundamental mejorar la seguridad del software. Entre los numerosos proyectos de OWASP, destaca el OWASP TOP 10, una lista que recoge los problemas más frecuentes y críticos detectados en las aplicaciones web. Esta lista es altamente valorada en la comunidad de seguridad informática. El ataque de inyección ha sido calificado como el tercero en el OWASP TOP 10, 2021 (Figura 2.5). Este, y los otros motivos mencionados con anterioridad, llevan a desarrollar la siguiente investigación en la cual se pretende obtener el algoritmo de aprendizaje automático más eficiente para detectar ataques de inyección SQL a base de datos. (7)

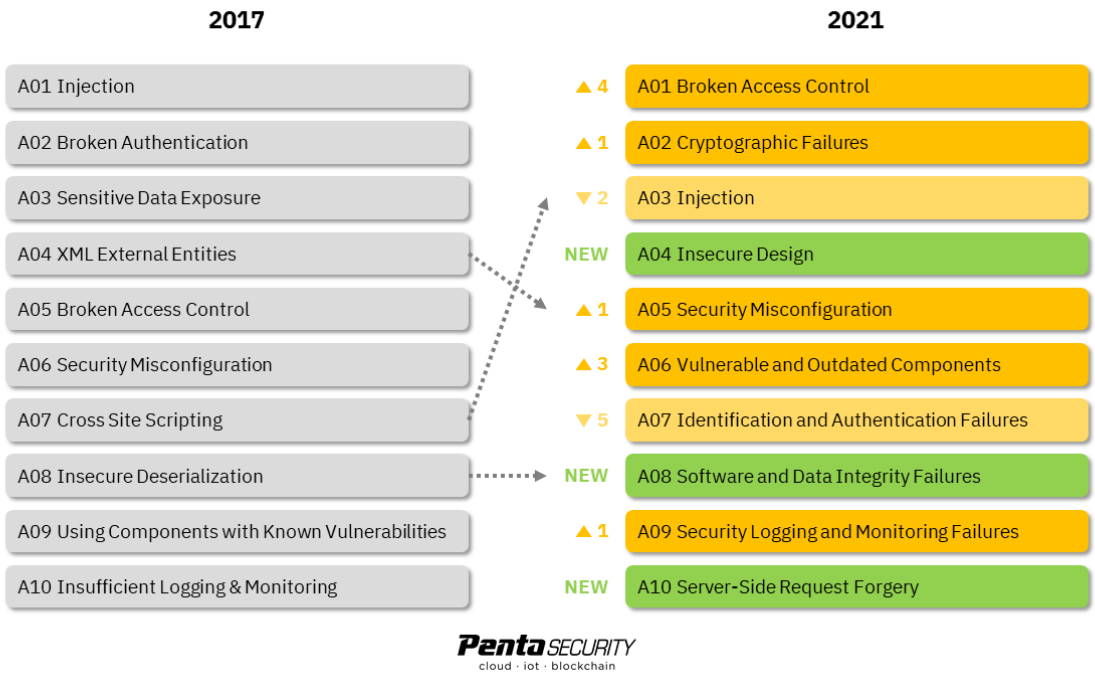


Figura 2.5: Comparativa OWASP TOP 10 2017 y 2021

Capítulo 3

Ataque de inyección SQL

SQL es un lenguaje de consulta que se utiliza en programación para modificar y eliminar los datos almacenados en bases de datos relacionales y acceder a ellos. La gran mayoría de los sitios y aplicaciones web utilizan bases de datos SQL.

La inyección SQL es una vulnerabilidad que ocurre en las capas de la base de datos de una aplicación. Este tipo de ataque explota vulnerabilidades en la forma en que una aplicación maneja las entradas del usuario para ejecutar comandos SQL no autorizados. Las consecuencias de una inyección SQL exitosa son muy graves, incluyendo el acceso no autorizado a datos sensibles, la modificación o eliminación de datos y la posible interrupción de los servicios. Esto sucede generalmente en formularios web, campos de búsqueda, URL o cualquier lugar donde la aplicación tome entrada del usuario y la utilice para construir consultas SQL sin una adecuada validación y limpieza de los datos. Por ejemplo, esta consulta utilizada para autenticar a un usuario:

```
SELECT * FROM users WHERE user = 'user' AND password = 'password';
```

Si la aplicación no valida adecuadamente las entradas, un atacante podría introducir una entrada maliciosa como:

```
SELECT * FROM users WHERE user = '' OR '1'='1' AND password = '' OR '1'='1';
```

La condición '1'='1' siempre es verdadera, lo que permitiría al atacante acceder sin proporcionar credenciales válidas.

Debido a la prevalencia de sitios web y servidores que utilizan bases de datos, las vulnerabilidades de inyección de SQL son uno de los tipos de ciberataques más antiguos y generalizados. Además, ya existen herramientas de código abierto que permiten a los cibercriminales realizar ataques automáticamente en tan solo minutos. (8)

3.1. Tipos de inyecciones SQL

Según la forma de acceso a los datos de backend, las inyecciones de SQL se pueden dividir en las siguientes categorías:

SQL en banda: Este tipo de ataque SQL es sencillo para los atacantes, porque usan el mismo canal de comunicación para lanzar ataques y obtener resultados. Tiene dos subvariantes:

SQL basado en errores: La base de datos genera un mensaje de error por las acciones del atacante. El atacante obtiene información sobre la infraestructura de la base de datos en función de los datos que generaron estos mensajes de error.

SQL basado en unión: El atacante usa el operador UNION SQL para obtener los datos deseados mediante la fusión de varias declaraciones *Select* en una única respuesta HTTP.

SQL ciega: Los atacantes usan patrones de respuesta y comportamiento del servidor después de enviar cargas útiles de datos para obtener más información sobre su estructura. Pero los datos no se transfieren de la base de datos del sitio web al atacante, así que no ve la información sobre el ataque. Sirve para inferir la construcción de la base de datos mediante la evaluación de expresiones que se combinan con declaraciones que siempre evalúan como verdaderas y declaraciones que siempre evalúan como falsas.

SQL fuera de banda: Este tipo de ataque SQL se puede llevar a cabo cuando los atacantes no pueden usar el mismo canal para lanzar el ataque y compartir información; o cuando un servidor es demasiado lento o inestable para realizar estas acciones. Este tipo de ataque se aprovecha de canales alternativos de comunicación para extraer datos o recibir información, como correos electrónicos, solicitudes HTTP, DNS... (8)

Sobre la base del ataque contra el sistema de administración de la base de datos, el ataque de inyección SQL se puede clasificar como:

Ataques SQL de Tautología: Los ataques de tautología se basan en la manipulación de una condición lógica que siempre resulta verdadera. Esto permite al atacante eludir la autenticación o extraer información sin restricciones. Por ejemplo, la sentencia que se vio anteriormente:

```
SELECT * FROM users WHERE user = '' OR '1'='1' AND password = '' OR '1'='1';
```

Consultas básicas de unión: Los ataques basados en la unión utilizan la cláusula 'UNION' para combinar el resultado de una consulta maliciosa con la consulta original, permitiendo al atacante recuperar datos adicionales. Supongamos una consulta SQL que muestra información de un producto:

```
SELECT name, prize FROM products WHERE id=1;
```

Un atacante podría obtener información de otra tabla de la siguiente forma:

```
SELECT name, prize FROM products WHERE id=1 UNION SELECT user, password FROM users;
```

Esto devolverá tanto la información del producto como los datos de usuario y contraseña.

Consultas complementarias (piggybacked queries): Ocurren cuando el atacante añade consultas adicionales a la consulta original. Esto puede permitir la ejecución de múltiples comandos SQL en una sola entrada. Por ejemplo, si la aplicación ejecuta una consulta para actualizar la información del usuario:

```
UPDATE users SET name = 'new_name' WHERE id = 1;
```

Un atacante podría convertir la consulta en:

```
UPDATE users SET name = 'new_name'; DROP TABLE users --' WHERE id = 1;
```

Esta inyección ejecutará dos consultas, la primera actualizará el nombre y la segunda eliminará la tabla users. El '- -' se utiliza para comentar el resto de la consulta original, asegurándose de que cualquier código SQL legítimo que debería seguir se ignore.

Mismatched Column Numbers: Este tipo de ataque ocurre cuando un atacante intenta utilizar la cláusula 'UNION' pero no coincide el número de columnas en las dos consultas combinadas. Se tiene la siguiente consulta:

```
SELECT name, prize FROM products WHERE id=1;
```

Un atacante puede intentar inyectar una consulta 'UNION'.

```
1 UNION SELECT user, password FROM users--
```

Pero si no coincide el número de columnas, obtendrá un error que puede usar para ajustar su ataque. Aquí, 'NULL' se utiliza para llenar la columna faltante en la consulta inyectada, haciendo que ambas tengan el mismo número de columnas.

```
SELECT name, prize FROM products WHERE id=1 UNION SELECT user, password,
NULL FROM usuarios--;
```

Insert Injection: Los ataques de inserción ocurren cuando el atacante manipula una consulta SQL para insertar datos no deseados en la base de datos. Por ejemplo, una consulta que inserta un nuevo usuario:

```
INSERT INTO users (user, password) VALUES ('new_user', 'new_password');
```

Un atacante puede inyectar una entrada para insertar datos adicionales, que eliminará la tabla users. Si introduce la siguiente entrada en los campos de un formulario:

- Usuario: new_user
- Contraseña: new_password'); DROP TABLE usuarios–

La consulta SQL resultante sería:

```
INSERT INTO users (user, password) VALUES ('new_user', 'new_password');
DROP TABLE users--');
```

Subquery Injection: Utilizan subconsultas dentro de la inyección para manipular la consulta principal y obtener o modificar datos no autorizados. Supongamos una consulta SQL que selecciona el precio de un producto específico:

```
SELECT prize FROM products WHERE id = 1;
```

Un atacante puede inyectar una subconsulta para obtener información adicional.

```
'1; (SELECT user FROM users WHERE id = 1)--'
```

La consulta ahora puede devolver información sensible.

```
SELECT prize FROM products WHERE id = 1; (SELECT user FROM users
WHERE id = 1)--;
```

(9)

Capítulo 4

Algoritmos de aprendizaje automático

En este estudio, se aplicarán cuatro algoritmos populares: k-Nearest Neighbors (k-NN), árboles de clasificación, regresión logística y máquinas de vectores de soporte (SVM).

4.1. k-Nearest Neighbors (k-NN)

El algoritmo de k-vecinos más cercanos es un algoritmo de aprendizaje supervisado que utiliza la proximidad para hacer clasificaciones de nuevas muestras. A una nueva observación se le asigna como etiqueta la que tiene la mayoría de las observaciones más cercanas (vecinos). Dependiendo de cuántos vecinos se consideren obtendremos un resultado u otro. Se recomienda que el número de vecinos (k) sea impar para evitar empates. Un valor muy bajo puede llevar al sobreajuste, y un valor muy alto, puede llevar a un ajuste insuficiente.

La parte determinante del algoritmo es cómo definir cercanía y cuántos vecinos se deben tomar. Si las variables se presentan en diferentes unidades físicas o vienen en escalas muy diferentes, es conveniente normalizar los datos de entrenamiento para mejorar la precisión. En este caso no será necesario.

Además, existen diferentes distancias que se pueden utilizar como: la euclídea, la Manhattan, similitud del coseno, la Jaccard, la Hamming... La distancia que se escoge depende del tipo de dato que se tenga, las dimensiones y el objetivo del estudio. En el caso de este estudio, se utilizará la distancia euclídea ya que convertiremos las sentencias SQL en vectores en el espacio en base a su forma.

Podemos resumir el proceso realizado al aplicar el algoritmo k-NN en los siguientes pasos (Figura 4.1):

1. Escoger la distancia más adecuada.
2. Decidir el valor de k.
3. Recibir un dato sin clasificar y medir su distancia a cada uno de los demás datos ya clasificados.
4. Seleccionar las k distancias más pequeñas.
5. Contar el número de veces que se eligió cada clase en esas k seleccionadas.
6. Clasificar el nuevo dato como perteneciente a la clase que más se repitió.

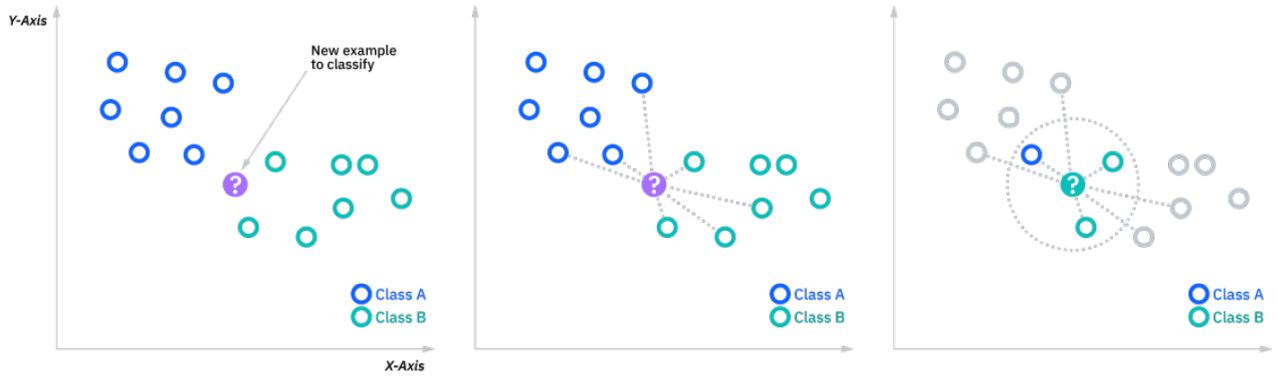


Figura 4.1: Algoritmo k-NN

El algoritmo k-NN fue escogido por ser fácil de implementar y de entender. Además, se ajusta para tener en cuenta cualquier dato nuevo, ya que todos los datos de entrenamiento se almacenan en la memoria, y solo requiere escoger el valor de k y la métrica a utilizar. Sin embargo, no funciona bien con entradas de datos grandes y de alta dimensión, ocupa más memoria y almacenamiento de datos en comparación con otros clasificadores, lo que aumenta el costo comercial y el tiempo de procesado. En este caso, aunque el volumen de datos que se van a tomar sea grande, la dimensionalidad no es tan alta. (10)

4.2. Árboles de decisión

Un árbol de decisión es otro algoritmo de aprendizaje supervisado. Tiene una estructura de árbol jerárquica, que consta de un nodo raíz, ramas, nodos internos y nodos hoja (Figura 4.2).

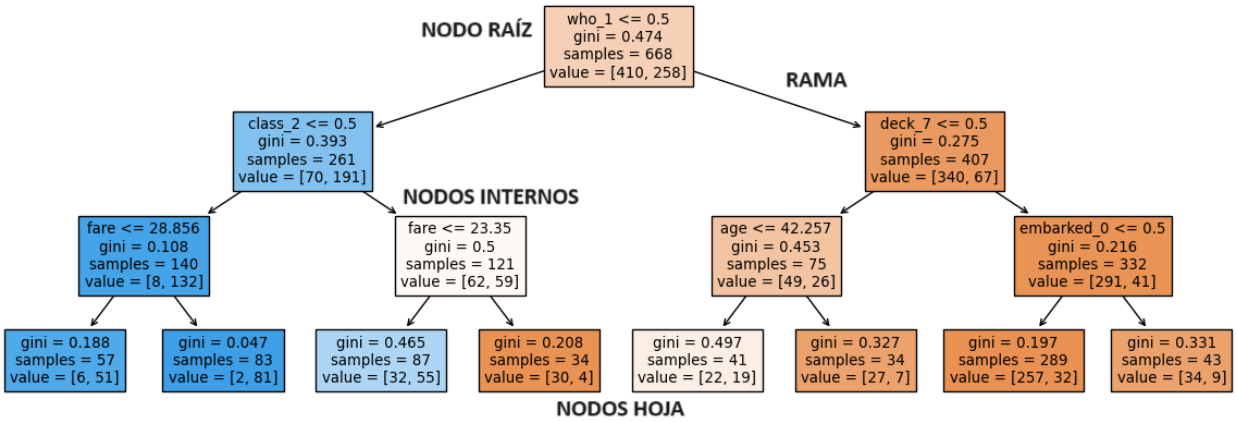


Figura 4.2: Ejemplo de árbol de decisión

Su aprendizaje emplea una estrategia de divide y vencerás mediante la realización de una búsqueda para identificar los puntos de división óptimos dentro de un árbol. Este proceso de división se repite de forma recursiva de arriba hacia abajo hasta que todos o la mayoría de los registros se hayan clasificado bajo etiquetas de clase específicas. Funciona de la siguiente forma:

1. Cada nodo del árbol se corresponde con un atributo y de él parten tantas ramas

como valores distintos tiene ese atributo.

2. En las hojas del árbol se encuentran todos o algunos de los valores de la variable clase.

3. Para clasificar una nueva instancia se inspecciona el mismo desde la raíz hasta llegar a un nodo hoja.

4. Cada nodo representa un test sobre un atributo y el valor correspondiente en la instancia indica la rama del árbol que debe recorrerse. El proceso se repite hasta alcanzar un nodo hoja. El valor de ese nodo suministra la clase a la que pertenece la instancia.

Si bien hay varias formas de seleccionar el mejor atributo en cada nodo, los criterios de división popular son: la ganancia de información (entropía) y la impureza de Gini. La entropía se define como:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{4.1}$$

donde,

- S es el conjunto de datos sobre el que se calcula la entropía.
- n es el número de clases.
- p_i es la proporción de ejemplos que pertenecen a la clase y_i en el conjunto S.
- Se asume que $\log_2(0) = 0$

Los valores de entropía pueden estar entre 0 y 1. Si todas las muestras en el conjunto de datos, S, pertenecen a una clase, entonces la entropía será igual a cero. Si la mitad de las muestras corresponde a cada clase, la entropía estará en 1.

Para seleccionar la mejor característica para dividir se debe usar el atributo con la menor cantidad de entropía. La ganancia de información representa la diferencia de entropía antes y después de una división en un atributo determinado. El atributo con la ganancia de información más alta producirá la mejor división, ya que hace el mejor trabajo al clasificar los datos de entrenamiento.

En este caso, se utilizará la impureza de Gini, que mide la probabilidad de que un elemento seleccionado al azar sea incorrectamente etiquetado si se asigna una etiqueta aleatoria según la distribución de etiquetas en un nodo. Se define con los mismos elementos de la Ecuación 4.1 como:

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{4.2}$$

Se ha escogido este algoritmo porque es sencillo de interpretar, las representaciones visuales los hacen más fáciles de entender. También facilita ver qué atributos son los más importantes, lo que no siempre es claro con otros algoritmos, como las redes neuronales. Son insensibles a las relaciones subyacentes entre los atributos, si dos variables están

altamente correlacionadas, el algoritmo solo elegirá una de las características para realizar la división.

Sin embargo, los árboles de decisión complejos tienden a sobreajustarse y no se generalizan bien a los nuevos datos. Se puede evitar mediante los procesos de poda, deteniendo el crecimiento del árbol cuando no hay datos suficientes. Pueden ser más costosos de entrenar en comparación con otros algoritmos. (11) (12)

4.3. Regresión logística

La regresión logística es un algoritmo de clasificación binaria que se utiliza para predecir la probabilidad de que una observación pertenezca a una de dos clases posibles. En lugar de predecir valores continuos, como en la regresión lineal, la regresión logística predice probabilidades que se encuentran entre 0 y 1. Es muy apropiada en este caso, puesto que la clasificación del problema al que nos enfrentamos es binaria.

La regresión logística comienza con un modelo lineal:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{4.3}$$

Para convertir la salida del modelo lineal 4.3 en una probabilidad, la regresión logística aplica la función sigmoide, también conocida como función logística (Figura 4.3):

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.4}$$

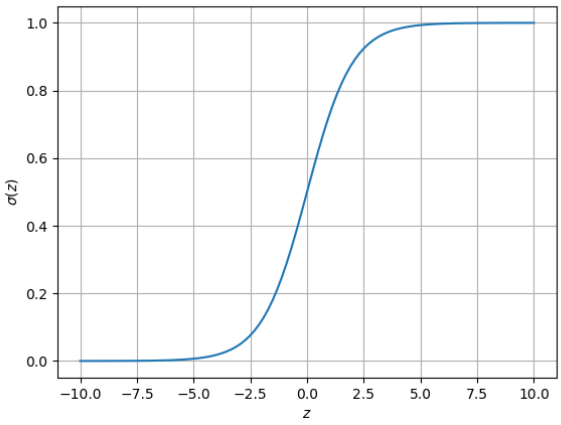


Figura 4.3: Función sigmoide

La función sigmoide toma cualquier valor real y lo transforma en un valor entre 0 y 1, lo cual puede interpretarse como una probabilidad. Para hacer la predicción binaria, se utiliza un umbral (generalmente 0.5):

$$\hat{y} = \begin{cases} 1 & \text{si } \sigma(z) \geq 0{,}5 \\ 0 & \text{si } \sigma(z) < 0{,}5 \end{cases}$$

donde $\hat{y}$ es la predicción.

Para evaluar la efectividad del modelo y minimizar los errores cometidos necesitamos una función de costo, que en el caso de la regresión logística es:

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right] + \frac{1}{C}\sum_{j=1}^{n}\beta_j^2 \tag{4.5}$$

Donde C es el parámetro de regularización. Se trata de minimizar 4.5 utilizando, normalmente, el descenso de gradiente. Puede verse con mayor detalle en (10).

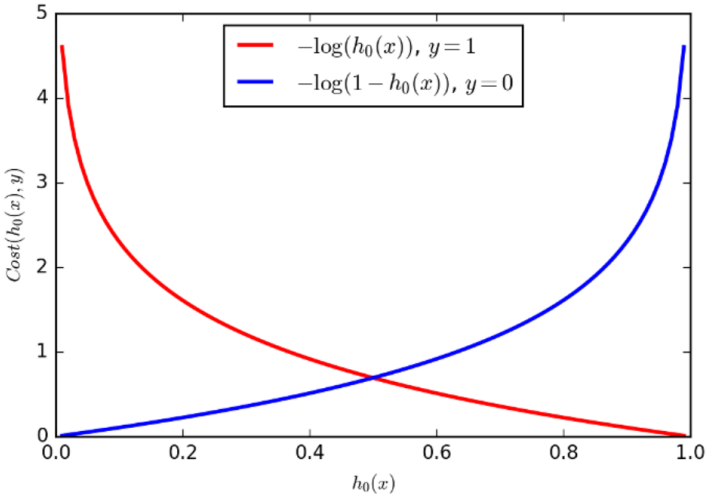


Figura 4.4: Función de penalización

La regularización en la regresión logística se usa para evitar el sobreajuste (overfitting) al penalizar los coeficientes grandes. Esto ayuda a mejorar la generalización del modelo a datos nuevos y desconocidos.

- Un valor pequeño de C significa una regularización fuerte, lo que hace que los coeficientes tiendan a cero, lo que puede llevar a un modelo más simple y menos propenso al sobreajuste.
- Un valor grande de C reduce el efecto de la regularización, permitiendo que el modelo se ajuste más a los datos de entrenamiento, lo que podría llevar a un posible sobreajuste si el conjunto de datos no es lo suficientemente grande o diverso.

Este algoritmo ha sido escogido porque es computacionalmente eficiente en comparación con modelos más complejos como las máquinas de vectores de soporte (SVM) o redes neuronales. Esto lo hace adecuado para grandes conjuntos de datos. Funciona bien cuando las clases son linealmente separables en el espacio de características. Puede proporcionar fronteras de decisión lineales que son efectivas y fáciles de interpretar. Incorporar la regularización ayuda a prevenir el sobreajuste y mejorar la generalización del modelo. Además, la interpretación de los coeficientes puede proporcionar conocimiento sobre qué características están más relacionadas con la variable objetivo, permitiendo interpretaciones más profundas y útiles. Es importante tener en cuenta que no exista un desequilibrio de las clases en el entrenamiento puesto que la clase más grande contribuirá demasiado a la función de costo en comparación con la clase más pequeña. (13)

4.4. Máquinas de vectores de soporte (SVM)

El objetivo de SVM es encontrar el hiperplano que mejor separa dos clases en el espacio de características. Busca el hiperplano que maximiza el margen entre las dos clases. Un hiperplano se define como:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{4.6}$$

donde,

- w es el vector de pesos.
- x es el vector de características.
- b es el sesgo.

Si se supone que tenemos una clase positiva (1) y una clase negativa (-1) separadas por el plano anterior, se puede predecir la etiqueta de cualquier ejemplo de la siguiente forma (Figura 4.5):

$$y = \begin{cases} 1 & \text{si } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1 & \text{si } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases} \tag{4.7}$$

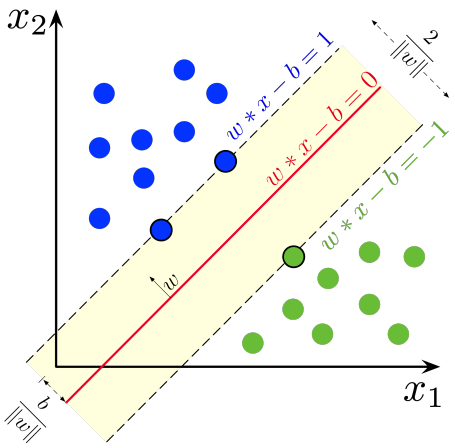


Figura 4.5: SVM

El margen es la distancia más cercana desde el hiperplano a cualquiera de los puntos de las dos clases. La distancia de un punto x al hiperplano se calcula como:

$$\text{Distancia} = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|} \tag{4.8}$$

Los vectores de soporte son los puntos más cercanos al hiperplano. Estos puntos son cruciales para definir la posición y orientación del hiperplano. Si se eliminan estos puntos, la frontera de decisión cambiará. Se basa en encontrar el hiperplano que maximiza el margen mientras minimiza el error de clasificación. Esto implica que sólo son importantes los vectores de soporte, se pueden ignorar otros ejemplos de entrenamiento. Por lo tanto, el problema de optimización consiste en:

$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|^2} \iff \min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2}, \text{ sujeto a } y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \geq 1, \forall i \tag{4.9}$$

Dando lugar a un problema de optimización convexa que se puede reescribir para tener en cuenta las restricciones como:

$$J = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{n} \lambda_i \left(y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) - 1 \right) \tag{4.10}$$

donde, los parámetros λ se denominan multiplicadores de Lagrange. Puede verse este desarrollo con mayor profundidad en (14).

Cuando las clases no son linealmente separables en el espacio original, SVM puede utilizar funciones de kernel para proyectar los datos a un espacio de mayor dimensión donde se vuelvan linealmente separables. Los kernels comunes incluyen el kernel lineal, el polinomial y el RBF (función de base radial). Para datos que no son linealmente separables, se introduce el término de regularización, como en el caso de la regresión logística.

En este estudio se utilizará el kernel RBF ya que puede capturar relaciones no lineales entre las características. Se define matemáticamente como:

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{z}\|^2\right) \tag{4.11}$$

donde,

- $\mathbf{x}$ y $\mathbf{z}$ son vectores de entrada.
- $\|\mathbf{x} - \mathbf{z}\|$ es la norma euclidiana (distancia) entre los vectores $\mathbf{x}$ y $\mathbf{z}$.
- γ es un parámetro que ajusta el alcance de la influencia de un solo punto de datos.

Además es más flexible y puede adaptarse a diferentes tipos de datos. A menudo ofrece un buen equilibrio entre precisión y complejidad del modelo. En comparación con otros kernels, tiende a ser menos propenso al overfitting en espacios de alta dimensión. Esto se debe a su capacidad para ajustar adecuadamente la influencia de cada punto de datos a través del parámetro γ. Y ha demostrado proporcionar un buen rendimiento, especialmente cuando no se tiene un conocimiento a priori sobre la distribución y la separabilidad de los datos.

Se ha escogido SVM como algoritmo para este estudio porque es muy eficaz en espacios de alta dimensión. Utiliza un subconjunto de los puntos de entrenamiento (vectores de soporte) en el proceso de decisión, lo que hace que el modelo sea eficiente en términos de memoria. El uso de kernels le permite a SVM adaptarse a una amplia variedad de problemas. Y la robutez ante el overfitting a través del parámetro de regularización. (14) (15)

Capítulo 5

Estudio de clasificación

5.1. Datos

Los datos que se van a utilizar en el estudio están disponibles en la plataforma Kaggle a través de este enlace `https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset`. En particular, se han utilizado las bases de datos sqliv2.csv, para entrenar y probar los algoritmos; y la base de datos sqli.csv para comprobar los resultados obtenidos con otra base de datos diferente.

La base de datos sqliv2.csv constaba de 33726 registros, que acabaron siendo 26924, para poder trabajar con ella en el Jupyter Notebook. La otra base de datos constaba de 4200 registros y no sufrió cambios. Cada una de ellas tenía solo dos variables:

- **Sentence:** La sentencia SQL.
- **Label:** Variable discreta binaria que indica si es una inyección SQL (1) o no (0).

Por este motivo, en la fase de preprocesado se van a generar variables artificiales que ayuden a los algoritmos a comprender las particularidades de las sentencias SQL.

Tanto las bases de datos modificadas como el cuaderno Jupyter pueden consultarse a través de este enlace:
`https://drive.google.com/drive/folders/1Se_V5ZPtlgwZlMbTkCT6NUFPFEnyHAPo?usp=drive_link`

5.2. Preprocesado

5.2.1. Variable artificiales

Se generaron las siguientes variables artificiales utilizando la biblioteca re de Python, que proporciona operaciones para trabajar con patrones de texto basados en expresiones regulares. Todas ellas son numéricas discretas:

- **Length:** La cantidad total de caracteres en la sentencia SQL.
- **Num_keywords:** Contar el número de palabras clave SQL comunes como SELECT, INSERT, UPDATE, DELETE, FROM, WHERE, DROP.
- **Num_logical_operators:** Contar operadores como AND, OR, NOT.

- **Num_comments:** Contar la presencia de comentarios – o /* */.
- **Num_single_quotes:** Contar la cantidad de comillas simples (’).
- **Num_double_quotes:** Contar la cantidad de comillas dobles dobles (").
- **Num_special_chars:** Verificar la presencia de caracteres especiales.

5.2.2. Descripción estadística

Como se puede apreciar en la Figura 5.1, no existen valores faltantes, y hay más valores 0 que 1, las clases están desbalanceadas. La media de la mayoría de las variables se encuentra cercana a 0, lo que significa que no hay caracteres especiales en la mayoría de sentencias. Tiene sentido ya que la mayor parte de las sentencias no son inyecciones. La media de la longitud de las sentencias está en torno a 44 con desviación típica alta, lo que indica diferencias de longitud grandes.

```
Estadísticas:
*************
             length  num_keywords  num_logical_operators  num_comments  \
count  26924.000000  26924.000000           26924.000000  26924.000000
mean      44.480426      0.541041               0.279862      0.134415
std       57.080068      1.040745               0.575672      1.426043
min        0.000000      0.000000               0.000000      0.000000
25%        9.000000      0.000000               0.000000      0.000000
50%       21.000000      0.000000               0.000000      0.000000
75%       60.000000      1.000000               0.000000      0.000000
max     1292.000000      7.000000               3.000000    227.000000

       num_single_quotes  num_double_quotes  num_special_chars
count       26924.000000       26924.000000       26924.000000
mean            0.382075           0.196442           3.640172
std             1.039440           0.714800           9.287666
min             0.000000           0.000000           0.000000
25%             0.000000           0.000000           0.000000
50%             0.000000           0.000000           0.000000
75%             0.000000           0.000000           3.000000
max             8.000000           5.000000         204.000000
```

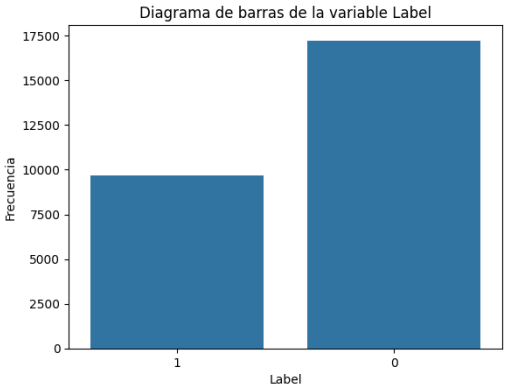
Figura 5.1: Descripción estadística de las variables

En cuanto a la distribución de las variables podemos observar los siguientes diagramas de barras (Figura 5.2), que se corresponde con lo comentado anteriormente.

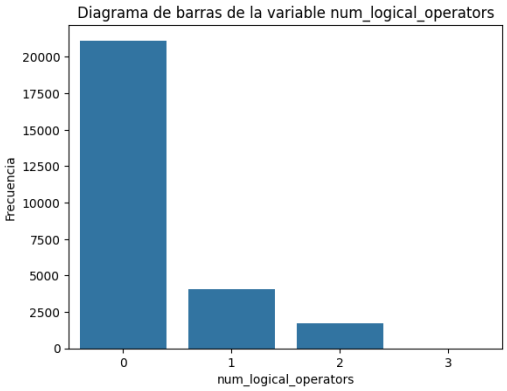
5.2.3. Análisis de correlaciones

Para el análisis de correlaciones podemos utilizar el coeficiente de correlación de Pearson y el coeficiente de correlación de Spearman. Aunque están diseñados para variables continuas, pueden usarse de manera aproximada para variables numéricas discretas, preferiblemente con muestras grandes.

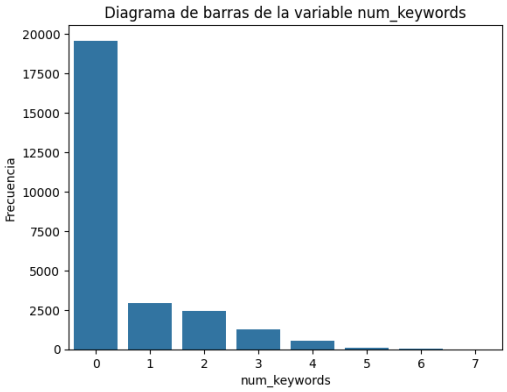
El coeficiente de correlación de Pearson mide la fuerza y la dirección de la relación lineal entre dos variables. El coeficiente de correlación de Spearman evalúa la relación



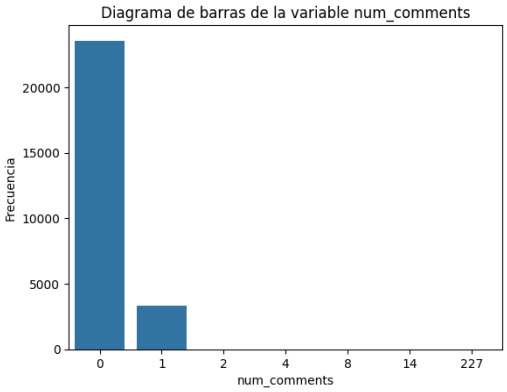
(a) Variable Label



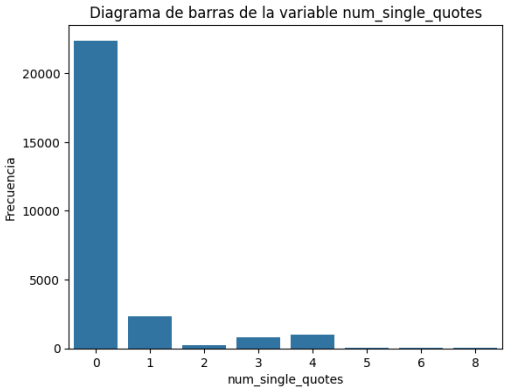
(b) Variable Num_logical_operators



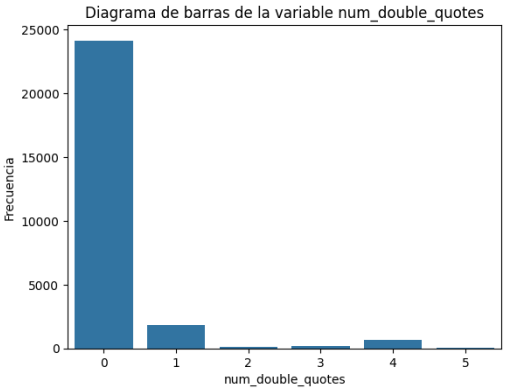
(c) Variable Num_keywords_operators



(d) Variable Num_comments



(e) Variable Num_single_quotes



(f) Variable Num_double_quotes

Figura 5.2: Distribución de las variables

monotónica (no necesariamente lineal) entre dos variables, utilizando los rangos de los datos en lugar de los valores exactos.

En la Figura 5.3 se puede observar correlaciones lineales directas entre la longitud y el número de caracteres especiales, el número de operadores lógicos y el número de palabras clave. Lo cual tiene bastante sentido. También entre el número de operadores lógicos y el número de comillas tanto simples como dobles. Estas dos últimas tienen correlación negativa, parece que no se contruyen sentencias con ambos tipos de comillas. Refleja como se contruyen este tipo de inyecciones SQL.

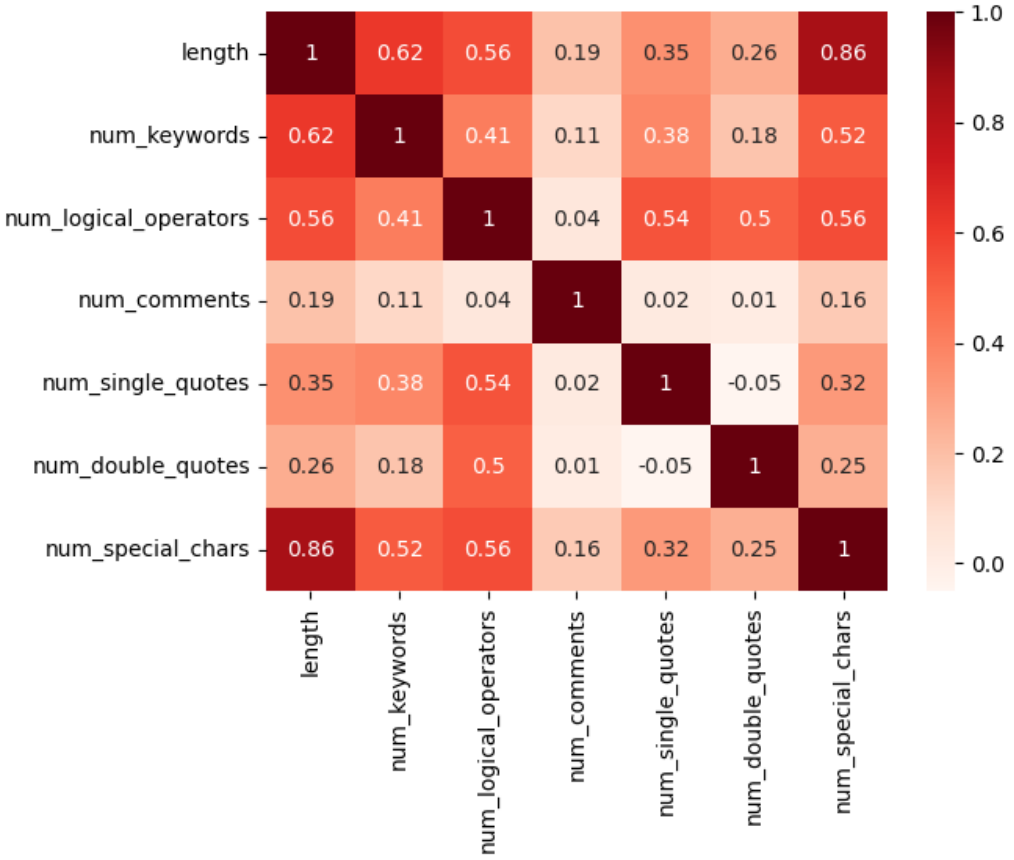


Figura 5.3: Correlación de Pearson

En la Figura 5.4 se puede observar las correlaciones anteriores y algunas nuevas directas no lineales, entre las variables número de caracteres especiales con el número de operadores lógicos y palabras clave.

Como el principal objetivo es realizar un análisis de clasificación para predecir una variable objetivo o etiqueta, y el número de variables es limitado, se va a considerar todas las variables en los modelos.

5.2.4. Valores atípicos

Para seleccionar los valores atípicos, una forma común es utilizar el rango intercuartil (IQR) o el método basado en la desviación estándar. Para ello, se calculan los límites superior e inferior del IQR y luego se definen los outliers como aquellos valores que caen

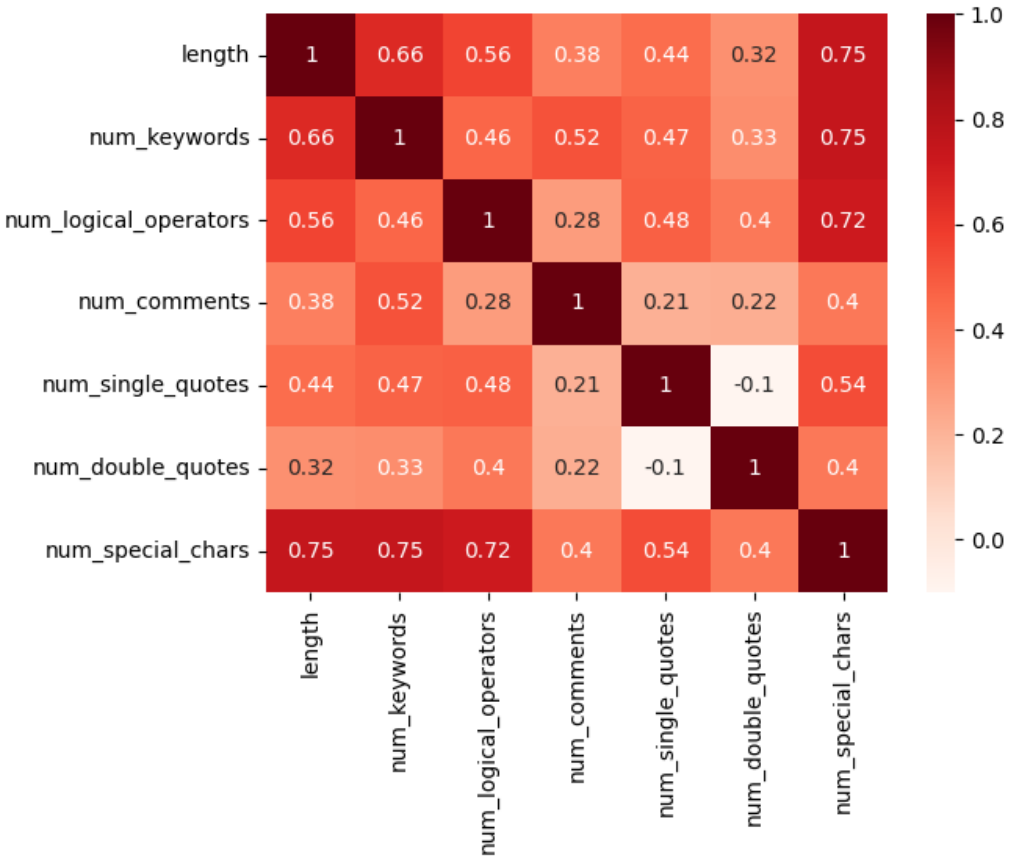


Figura 5.4: Correlación de Spearman

fuera de estos límites. El número de valores atípicos encontrados en las variables se reflejados en la Tabla 5.1.

Tabla 5.1: Número de valores atípicos

Variable	**Número de valores atípicos**
Length	1636
Num_keywords	1968
Num_logical_operators	5789
Num_comments	3358
Num_single_quotes	4540
Num_double_quotes	2779
Num_special_chars	3803

Se dejan estos outliers dentro de los datos, ya que no se deben a errores de la base de datos, sino que, como la mayoría de las sentencias no son inyecciones, aquellas que presentan particularidades son minoría.

5.3. Análisis de clasificación

Se ha dividido el nuevo conjunto de datos preprocesados en un conjunto de entrenamiento, con el 70 % de los datos; y un conjunto de prueba, con el 30 % de los datos, que nos servirá para validar la calidad del modelo estudiado.

5.3.1. Métodos de balanceo de clases

Como se vio en la sección de análisis de la distribución de los datos, la clase que se quiere predecir está muy desbalanceada lo que afecta a los algoritmos en su proceso de generalización de la información y perjudica a las clases minoritarias. Para solucionar este problema se recurre a los métodos de balanceo: oversampling, undersampling y smote.

- **UNDERSAMPLING:** Selecciona una muestra de la clase mayoritaria de igual tamaño que la clase minoritaria. Sin embargo, puede llevar a la pérdida de información y a la subrepresentación de la clase mayoritaria. Puede aumentar el riesgo de sesgo, ya que se eliminan ejemplos de la clase mayoritaria.
- **OVERSAMPLING:** Sobre muestrea la clase minoritaria. Sin embargo, este método, puede llevar al sobreajuste si se aumenta en exceso el tamaño de la clase minoritaria. Y no agrega nueva información, ya que simplemente duplica o triplica ejemplos existentes.
- **SMOTE:** Consiste en seleccionar un ejemplo de la clase minoritaria en el conjunto de datos original. Para el ejemplo seleccionado en el paso anterior, se eligen k (por defecto k=5) ejemplos cercanos de la misma clase. Estos vecinos se utilizan para generar ejemplos sintéticos. Los ejemplos sintéticos se incorporan al conjunto de datos original, lo que aumenta la representación de la clase minoritaria y equilibra las clases. Sin embargo, la generación de ejemplos sintéticos puede ser computacionalmente costosa. La calidad de los ejemplos sintéticos depende de la elección de vecinos cercanos y la distribución de los datos (Figura 5.5).

El balanceo se lleva a cabo en el conjunto de entrenamiento, nunca en el de prueba. Se van a aplicar los algoritmos mencionados en el capítulo anterior con los tres métodos para estudiar cuál se adapta mejor.

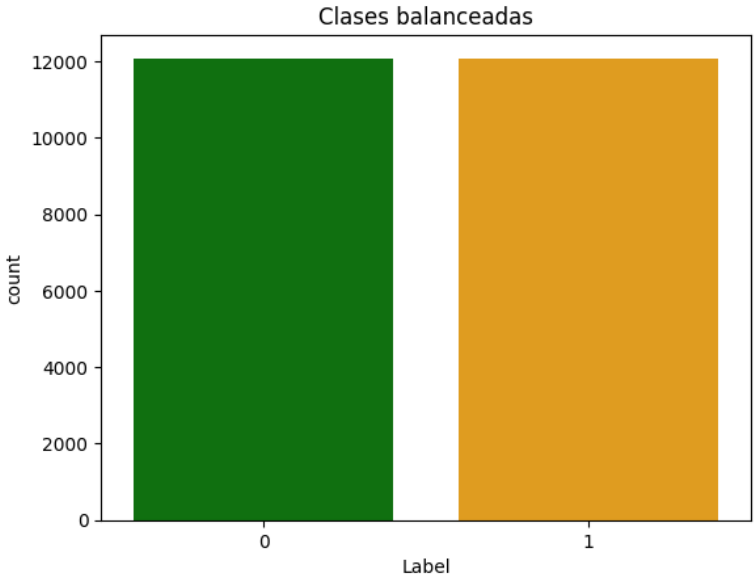


Figura 5.5: Clases balanceadas con el método smote

5.3.2. Métodos de evaluación

Para comprender los resultados es necesario conocer que:

- **Precisión (Precision):** Mide la proporción de predicciones positivas correctas en relación con todas las predicciones positivas. Indica qué tan preciso es el modelo al predecir la clase positiva.

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

- **Recall (Sensibilidad):** Mide la proporción de instancias positivas que fueron correctamente identificadas por el modelo en relación con todas las instancias positivas.

$$recall = \frac{TP}{TP + FN} \tag{5.2}$$

- **F1-Score:** Es una métrica que combina precisión y recall en un solo valor. Un F1-Score alto indica un buen equilibrio entre la capacidad del modelo para predecir correctamente la clase positiva y la capacidad para capturar todos los casos positivos.

$$F1 - Score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{5.3}$$

- **Exactitud (Accuracy):** Mide la proporción de predicciones correctas (tanto verdaderos positivos como verdaderos negativos) en relación con el total de predicciones. Indica con qué frecuencia el modelo hace predicciones correctas en general.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.4}$$

- **Curva de aprendizaje:** Es una herramienta útil para evaluar el rendimiento de un modelo a medida que varía el tamaño del conjunto de entrenamiento. Muestra cómo la precisión del modelo cambia en función del número de ejemplos de entrenamiento utilizados. Ayuda a evaluar si el modelo está sobreajustando o subajustando. La convergencia de las curvas sugiere que el modelo no está sobreajustando (alto en entrenamiento, pero bajo en validación) ni subajustando (bajo en entrenamiento y validación), lo que es deseable.

- **Validación cruzada:** Es una técnica para evaluar el rendimiento de un modelo utilizando diferentes particiones, k, de los datos de entrenamiento y validación (en este caso 5 particiones). Se entrena el modelo k veces, utilizando k-1 particiones como datos de entrenamiento y 1 partición como datos de validación. Su objetivo principal es obtener una estimación más precisa del rendimiento del modelo.

- **GridSearch:** Es una técnica para encontrar los mejores hiperparámetros para un modelo. Los hiperparámetros son configuraciones que no se aprenden durante el entrenamiento del modelo, como la profundidad máxima de un árbol de decisión o el valor de regularización en una regresión logística. Para ello, se evalúa el rendimiento del modelo (en este caso exactitud) para cada combinación de hiperparámetros utilizando validación cruzada.

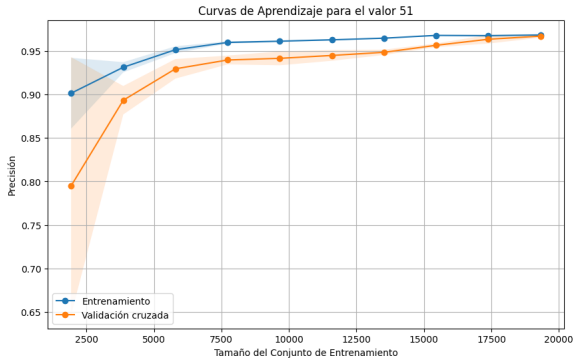
5.4. Resultados k-NN

Consideramos cuatro escenarios con el conjunto de test, aplicamos el modelo con las clases sin balancear y con los tres métodos de balanceo vistos anteriormente (Tabla 5.2). El valor óptimo de k se obtiene comparando mediante validación cruzada.

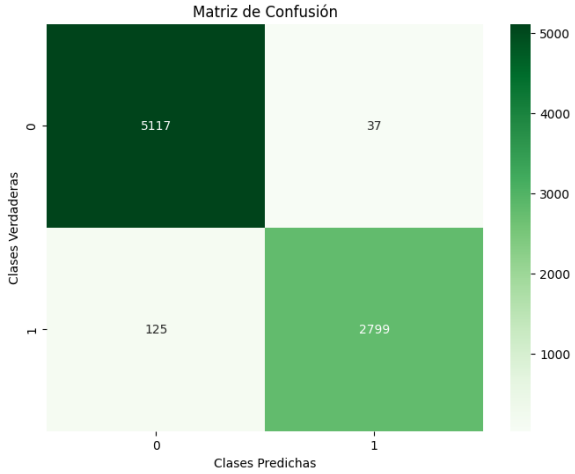
Tabla 5.2: Resultados algoritmo k-NN

	Sin balanceo	**Undersampling**	**Oversampling**	**Smote**
k	51	51	50	51
PRECISION (0)	0.96	0.97	0.98	0.98
PRECISION (1)	0.99	0.98	0.99	0.99
RECALL (0)	1	0.99	0.99	0.99
RECALL (1)	0.93	0.94	0.96	0.96
F1-SCORE (0)	0.98	0.98	0.98	0.98
F1-SCORE (1)	0.96	0.96	0.97	0.97
ACCURACY	0.97	0.97	0.98	0.98

Aunque el modelo con oversampling y smote tienen los mismos resultados, se toma el k-NN balanceado con smote como el mejor ya que toma un vecino más, y sus curvas de aprendizaje convergen en un valor aceptable (Figura 5.6a).



(a) Curvas de aprendizaje con smote



(b) Resultados con el método smote

Figura 5.6: k-NN con smote

A continuación, se presentan los resultados del mejor modelo (balanceado con smote) de la otra base de datos considerada (Tabla 5.3 y Figura 5.7).

Como se puede observar en los resultados, el modelo clasifica bastante bien la clase mayoritaria (0), sin embargo, se aprecia un descenso de la calidad del modelo en las predicciones de ambas clases, sobre todo, en la de la clase minoritaria (1). En comparación con los resultados obtenidos en el entrenamiento y testeo con la primera base de datos, se puede concluir que el modelo no ha llegado a generalizarse del todo bien.

Tabla 5.3: Resultados algoritmo k-NN

	Smote
PRECISION (0)	0.92
PRECISION (1)	0.65
RECALL (0)	0.84
RECALL (1)	0.79
F1-SCORE (0)	0.88
F1-SCORE (1)	0.71
ACCURACY	0.83

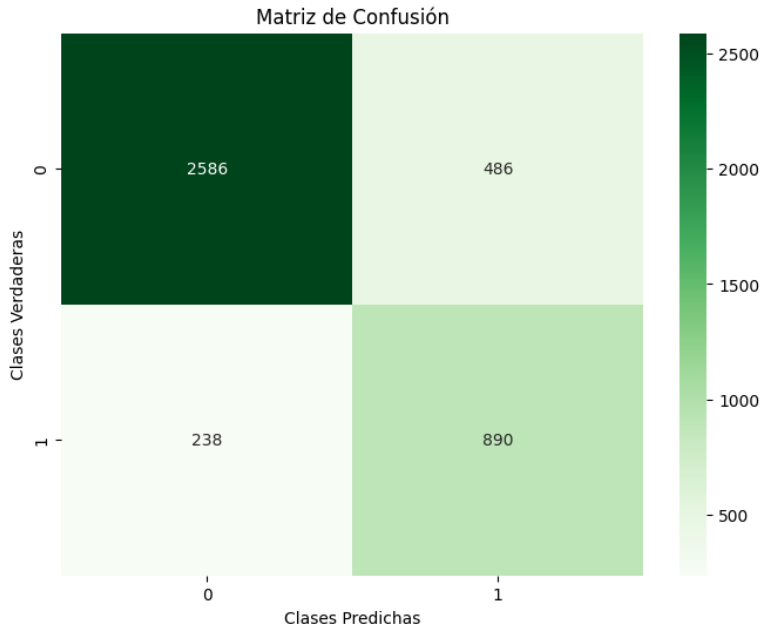


Figura 5.7: Resultados k-NN con smote segunda base de datos

5.5. Resultados Árbol de Decisión

Consideramos los cuatro escenarios anteriores. En la búsqueda de hiperparámetros hemos obtenido los mismos resultados para los cuatro modelos:

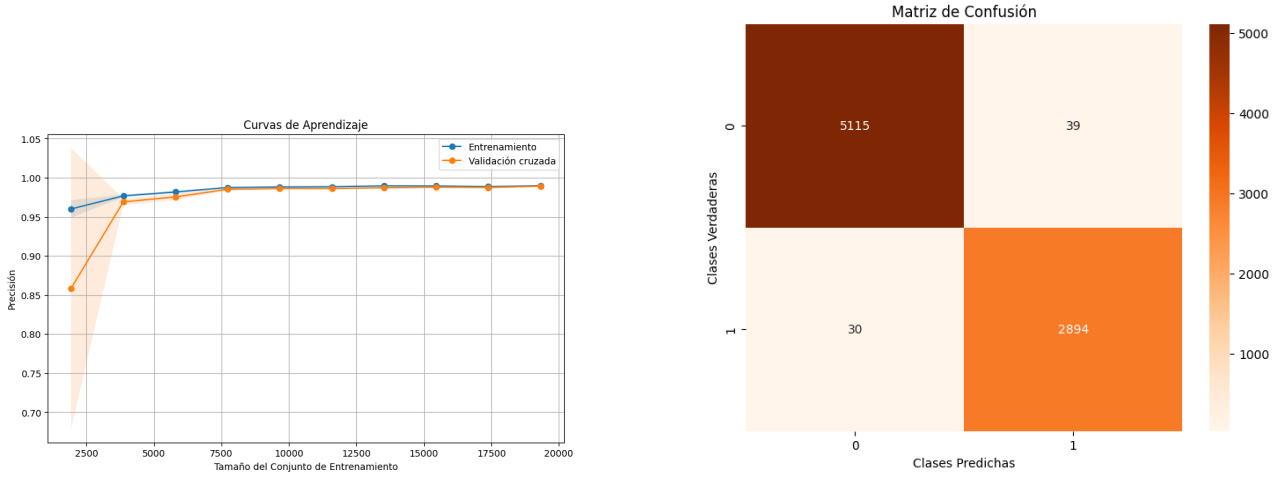
- **Profundidad máxima del árbol:** 5
- **Número mínimo de muestras requeridas para dividir un nodo:** 50
- **Número mínimo de muestras requeridas en un nodo hoja:** 50

Los valores obtenido en la validación del conjunto test se presentan a continuación (Tabla 5.4):

Tabla 5.4: Resultados algoritmo k-NN

	Sin balanceo	**Undersampling**	**Oversampling**	**Smote**
PRECISION (0)	0.99	0.99	0.99	0.99
PRECISION (1)	0.99	0.99	0.99	0.99
RECALL (0)	0.99	0.99	0.99	0.99
RECALL (1)	0.99	0.99	0.99	0.99
F1-SCORE (0)	0.99	0.99	0.99	0.99
F1-SCORE (1)	0.99	0.99	0.99	0.99
ACCURACY	0.99	0.99	0.99	0.99

Aunque en todos los modelos se tienen los mismos resultados, se toma el árbol de decisión balanceado con smote como el mejor ya que sus curvas de aprendizaje convergen en un valor aceptable y al principio del entrenamiento (Figura 5.8a).



(a) Curvas de aprendizaje con smote

(b) Resultados con el método smote

Figura 5.8: Árbol de decisión con smote

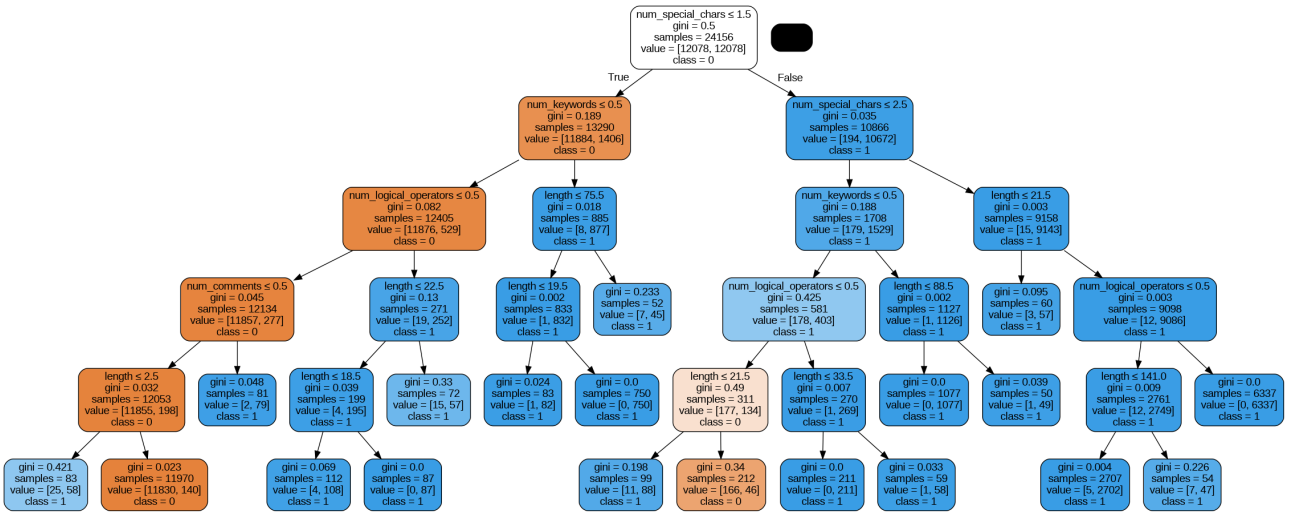


Figura 5.9: Árbol de decisión con el método smote

A continuación, se presentan los resultados del mejor modelo (balanceado con smote) de la otra base de datos considerada (Tabla 5.5 y Figura 5.10).

Tabla 5.5: Resultados del árbol de decisión

	Smote
PRECISION (0)	0.96
PRECISION (1)	0.74
RECALL (0)	0.88
RECALL (1)	0.91
F1-SCORE (0)	0.92
F1-SCORE (1)	0.81
ACCURACY	0.89

Al igual que en el caso anterior, el modelo clasifica bastante bien la clase mayoritaria (0), sin embargo, se aprecia un descenso de la calidad del modelo en las predicciones de ambas clases, sobre todo, en la de la clase minoritaria (1), pero menor que con el modelo k-NN. En comparación con los resultados obtenidos en el entrenamiento y testeo con la primera base de datos, se puede concluir que el modelo no ha llegado a generalizarse del todo bien, aunque el resultado es mejor que con el anterior.

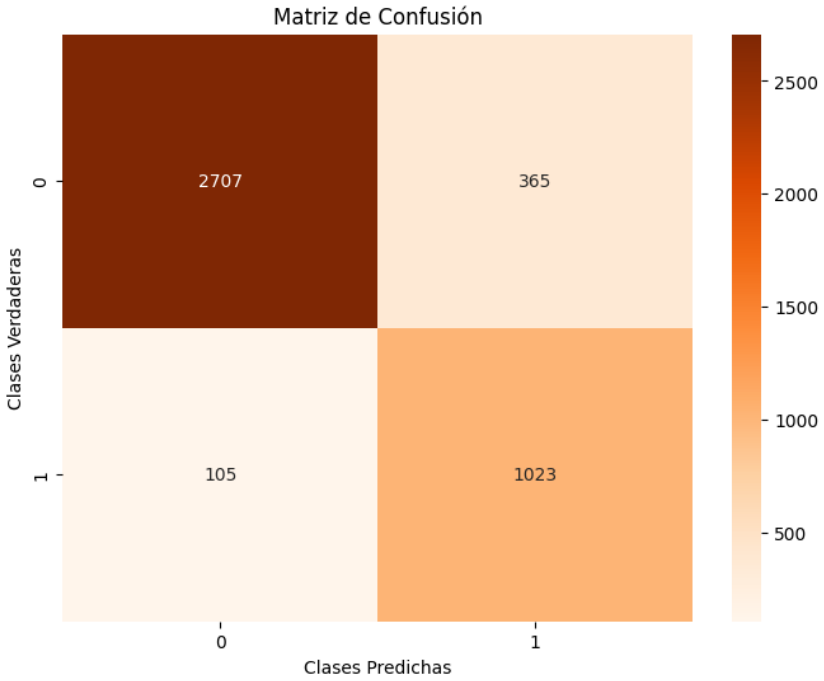


Figura 5.10: Resultados árbol de decisión con smote segunda base de datos

5.6. Resultados Regresión Logística

Consideramos los cuatro escenarios de siempre con el conjunto de test. Los valores del parámetro de regularización son bastante altos, por lo que el modelo no está infraajustado y se mantiene fiel al entrenamiento sin caer en el sobreajuste. Los resultados se presentan de la siguiente forma (Tabla 5.6):

Tabla 5.6: Resultados de la regresión logística

	Sin balanceo	**Undersampling**	**Oversampling**	**Smote**
C	3792.69	1438.45	206.91	206.91
PRECISION (0)	0.99	0.99	0.99	0.99
PRECISION (1)	1	0.99	0.99	0.99
RECALL (0)	1	1	1	1
RECALL (1)	0.98	0.98	0.98	0.99
F1-SCORE (0)	0.99	0.99	0.99	0.99
F1-SCORE (1)	0.99	0.99	0.99	0.99
ACCURACY	0.99	0.99	0.99	0.99

Aunque el modelo con oversampling y smote tienen casi los mismos resultados, se toma modelo balanceado con smote como el mejor ya que está más regularizado, lo que puede ayudar a generalizar más el modelo para datos no vistos, y mejora un poco el valor de recall de la clase 1 (Figura 5.11).

A continuación, se presentan los resultados del mejor modelo (balanceado con smote) de la otra base de datos considerada (Tabla 5.7 y Figura 5.12).

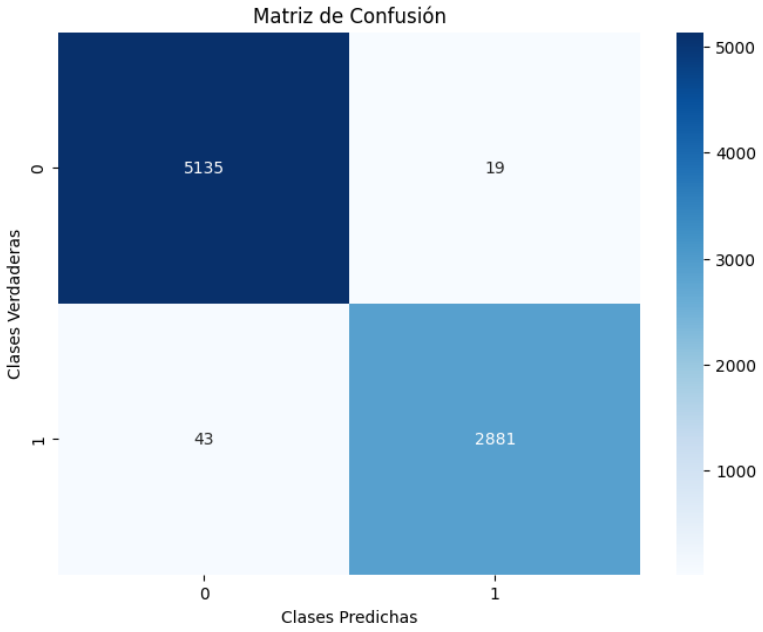


Figura 5.11: Resultados regresión logística con smote

Tabla 5.7: Resultados de la regresión logística

	Smote
PRECISION (0)	0.95
PRECISION (1)	0.98
RECALL (0)	0.99
RECALL (1)	0.87
F1-SCORE (0)	0.97
F1-SCORE (1)	0.92
ACCURACY	0.96

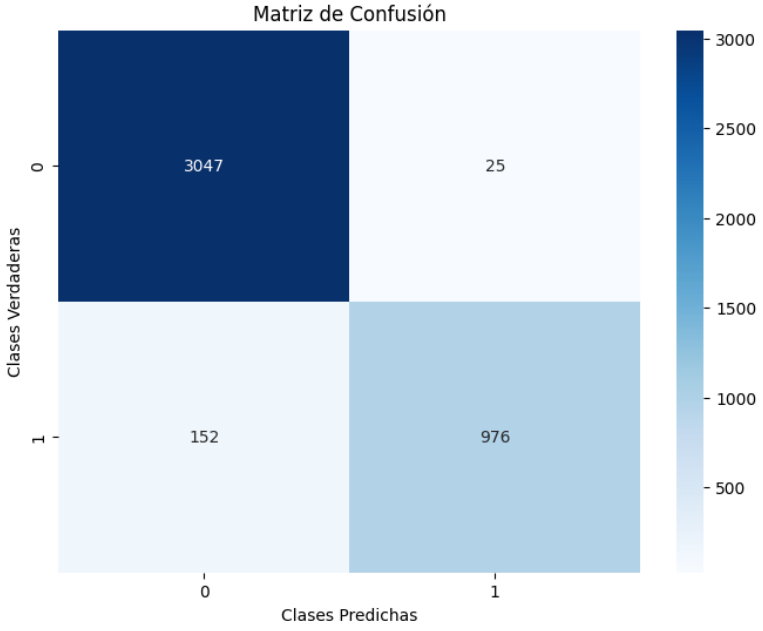


Figura 5.12: Resultados regresión logística con smote segunda base de datos

Como se puede observar en los resultados, el modelo clasifica bastante bien ambas clases, aunque tiende a clasificar peor la clase minoritaria, es decir, clasifica como inyecciones SQL como no inyección, lo cual es bastante peligroso. Sin embargo, en comparación a los otros modelos, este ha mantenido la calidad con la nueva base de datos, cuyas sentencias tienen ligeras diferencias con las de la base de datos de entrenamiento. Por ahora, se trata del modelo que mejor generaliza la detección de inyecciones SQL.

5.7. Resultados Support Vector Machine

Consideramos los cuatro escenarios de siempre con el conjunto de test. Los valores del parámetro de regularización también son bastante altos. Los resultados se presentan de la siguiente forma (Tabla 5.8):

Tabla 5.8: Resultados de la regresión logística

	Sin balanceo	**Undersampling**	**Oversampling**	**Smote**
C	3792.69	10000	10000	3792.69
PRECISION (0)	0.99	0.99	0.99	0.99
PRECISION (1)	1	1	1	0.99
RECALL (0)	1	1	1	1
RECALL (1)	0.99	0.99	0.99	0.99
F1-SCORE (0)	1	1	1	1
F1-SCORE (1)	0.99	0.99	0.99	0.99
ACCURACY	0.99	0.99	0.99	0.99

Aunque el modelo con oversampling y undersampling tienen los mejores resultados, se toma modelo balanceado con smote como el mejor ya que está más regularizado y los demás valores son similares (Figura 5.13).

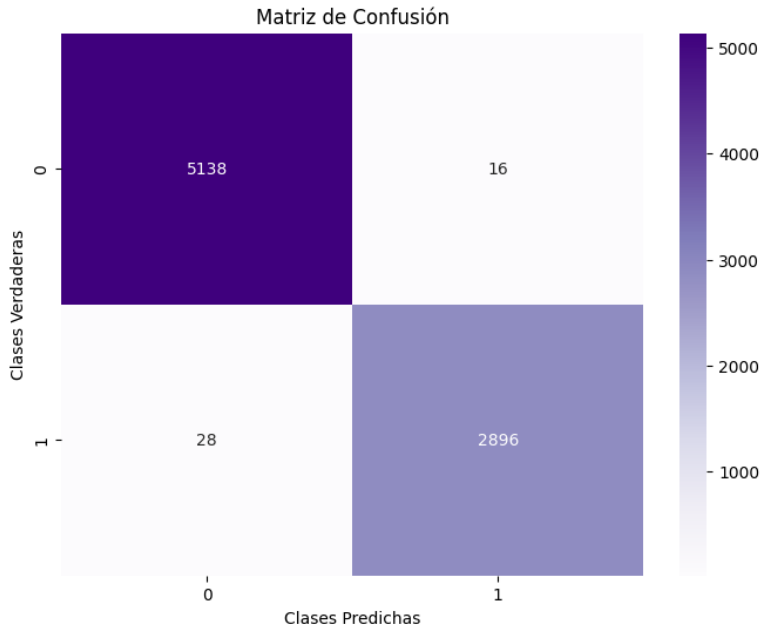


Figura 5.13: Resultados support vector machine con smote

A continuación, se presentan los resultados del mejor modelo (balanceado con smote) de la otra base de datos considerada (Tabla 5.9 y Figura 5.14).

Tabla 5.9: Resultados de support vector machine

	Smote
PRECISION (0)	0.97
PRECISION (1)	0.97
RECALL (0)	0.99
RECALL (1)	0.91
F1-SCORE (0)	0.98
F1-SCORE (1)	0.94
ACCURACY	0.97

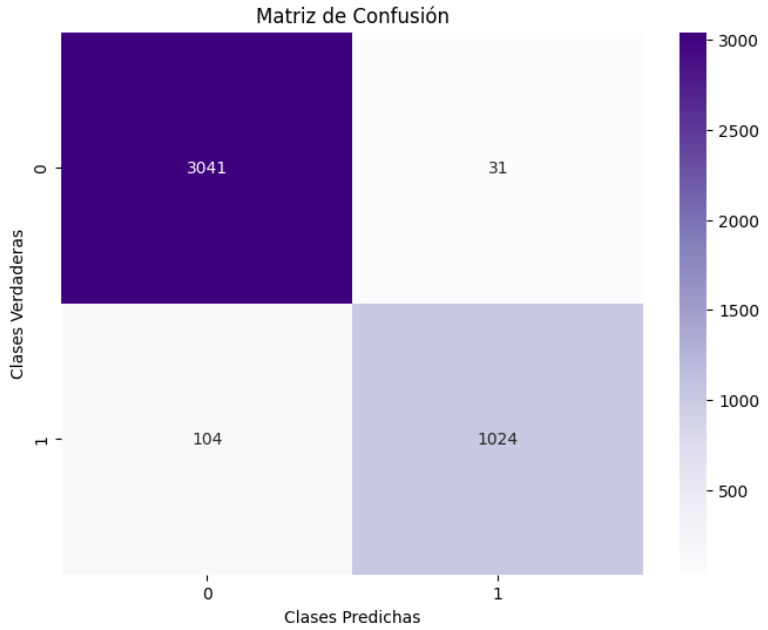


Figura 5.14: Resultados support vector machine con smote segunda base de datos

Como se puede observar en los resultados, el modelo clasifica bastante bien ambas clases, aunque tiende a clasificar un poco peor la clase minoritaria. Sin embargo, en comparación a los otros modelos, este también ha mantenido la calidad con la nueva base de datos y es un poco mejor que el modelo de regresión logística. Se puede concluir que se trata del modelo que mejor generaliza la detección de inyecciones SQL.

5.8. Resumen de mejores modelos

Con todos los modelos se ha tomado como mejor método de balanceo el método smote. A continuación, se resumen los resultados de estos mejores modelos con las dos bases de datos, donde, B1, es la base de datos que se ha usado para el entrenamiento y, B2, la que se usó para comprobar la generalización (Tablas 5.10 y 5.11).

Tabla 5.10: Resumen de resultados de los modelos I

	B1 k-NN	**B2 k-NN**	**B1 DT**	**B2 DT**
PRECISION (0)	0.98	0.92	0.99	0.96
PRECISION (1)	0.99	0.65	0.99	0.74
RECALL (0)	0.99	0.84	0.99	0.88
RECALL (1)	0.96	0.79	0.99	0.91
F1-SCORE (0)	0.98	0.88	0.99	0.92
F1-SCORE (1)	0.97	0.71	0.99	0.81
ACCURACY	0.98	0.83	0.99	0.89

Tabla 5.11: Resumen de resultados de los modelos II

	B1 LR	**B2 LR**	**B1 SVM**	**B2 DT**
PRECISION (0)	0.99	0.95	0.99	0.97
PRECISION (1)	0.99	0.98	0.99	0.97
RECALL (0)	1	0.99	1	0.99
RECALL (1)	0.99	0.87	0.99	0.91
F1-SCORE (0)	0.99	0.97	1	0.98
F1-SCORE (1)	0.99	0.92	0.99	0.94
ACCURACY	0.99	0.96	0.99	0.97

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

En el proyecto se ha abordado el problema de la clasificación de inyecciones SQL utilizando diversos algoritmos de aprendizaje automático: k-Nearest Neighbors (k-NN), Árboles de Decisión, Regresión Logística y Máquinas de Vectores de Soporte (SVM); y diferentes métodos de balanceo de clases: undersampling, oversamplig y smote. Tras un exhaustivo proceso de implementación y evaluación, se ha determinado que el modelo basado en SVM ha sido el que mejor ha generalizado y obtenido los mejores resultados en la tarea de clasificación, tanto de la primera base de datos, con la que se llevó a cabo el proceso de entrenamiento; como de la segunda, ligeramente diferente.

Aunque los modelos basados en k-NN y Árboles de Decisión mostraron buenos resultados durante la fase de entrenamiento, su rendimiento disminuyó considerablemente al ser evaluados en la base de datos adicional. Esto sugiere que estos modelos no generalizaron adecuadamente a datos no vistos, posiblemente debido a su menor capacidad para capturar la complejidad y las variaciones presentes en las inyecciones SQL. El modelo de regresión logística obtuvo un mejor resultado y fue capaz de generalizar mejor.

El modelo SVM destacó en términos de precisión y capacidad de generalización. Este rendimiento superior fue confirmado mediante la evaluación del modelo en una base de datos distinta, donde logró clasificar las sentencias con un notable éxito. La capacidad de SVM para manejar eficientemente los casos positivos y negativos de inyecciones SQL subraya su eficacia en la detección de este tipo de amenazas. Cabe destacar, también, que los mejores resultados en todos los modelos se determinaron con el método de balanceo smote, en particular, el modelo SVM con smote.

Uno de los aspectos cruciales que contribuyeron al éxito del modelo fue el preprocesado de los datos. Este proceso incluyó la limpieza de datos, comprensión de estos y la conversión de características textuales en representaciones numéricas adecuadas para los algoritmos de clasificación. Además, la búsqueda de hiperparámetros, llevada a cabo mediante técnicas como GridSearch y validación cruzada, permitió ajustar finamente los modelos para optimizar su rendimiento. Estas etapas fueron esenciales para mejorar la precisión y robustez del modelo, y subrayan la importancia de un pipeline de datos bien estructurado en proyectos de aprendizaje automático.

En definitiva, se ha cumplido con el objetivo principal de clasificar las inyecciones SQL utilizando algoritmos de machine learning.

6.2. Conclusions

The project has addressed the problem of classifying SQL injections using different machine learning algorithms: k-Nearest Neighbors (k-NN), Decision Trees, Logistic Regression and Support Vector Machines (SVM); and different class balancing methods: undersampling, oversamplig and smote. After an exhaustive implementation and evaluation process, it has been determined that the SVM-based model has been the one that best generalized and obtained the best results in the classification task, both for the first database, with which the training process was carried out, and for the second, slightly different one.

Although the models based on k-NN and Decision Trees showed good results during the training phase, their performance decreased considerably when evaluated on the additional database. This suggests that these models did not generalise adequately to unseen data, possibly due to their reduced ability to capture the complexity and variations present in SQL injections. The logistic regression model performed better and was able to generalise better.

The SVM model excelled in terms of accuracy and generalisability. This superior performance was confirmed by evaluating the model on a different database, where it was able to classify the statements with remarkable success. SVM's ability to efficiently handle both positive and negative SQL injections underlines its effectiveness in detecting this type of threat. It is also worth noting that the best results in all models were determined with the smote balancing method, in particular the SVM model with smote.

One of the crucial aspects that contributed to the success of the model was the preprocessing of the data. This process included data cleaning, data understanding and the conversion of textual features into numerical representations suitable for the classification algorithms. In addition, hyperparameter search, carried out using techniques such as GridSearch and cross-validation, allowed fine-tuning of the models to optimise their performance. These steps were essential to improve model accuracy and robustness, and underline the importance of a well-structured data pipeline in machine learning projects.

In short, the main objective of classifying SQL injections using machine learning algorithms has been achieved.

6.3. Propuestas de mejora y líneas de investigación futuras

A pesar de los resultados obtenidos, existen varias áreas en las que se podrían realizar mejoras adicionales.

La inclusión de características más sofisticadas y específicas del dominio, como patrones de consulta SQL y análisis de sintaxis, podría mejorar aún más la capacidad del modelo para detectar inyecciones SQL.

La combinación de múltiples modelos a través de técnicas de ensemble, como el boosting o el stacking, podría potenciar el rendimiento global y la robustez del sistema de detección. También se podría investigar el uso de modelos de aprendizaje profundo, como

redes neuronales, que podrían capturar patrones más complejos en los datos textuales de las consultas SQL.

Sería interesante implementar técnicas que permitan una mayor interpretabilidad de las decisiones del modelo SVM, lo cual es crucial en contextos de seguridad para entender por qué una consulta fue clasificada como maliciosa.

Por último, establecer un sistema continuo de actualización con nuevas muestras de datos para asegurar que el modelo se mantenga relevante y eficaz ante nuevas técnicas de inyección SQL.

6.4. Agradecimientos

Esta investigación resulta del Proyecto Estratégico SCITALA (C064/23), fruto del convenio de colaboración suscrito entre el Instituto Nacional de Ciberseguridad (INCIBE) y la Universidad de La Laguna. Esta iniciativa se realiza en el marco de los fondos del Plan de Recuperación, Transformación y Resiliencia, financiados por la Unión Europea (Next Generation).

Bibliografía

[1] IBM. (2023). ¿Qué es un ciberataque?. Recuperado de https://www.ibm.com/es-es/topics/cyber-attack#:~:text=IBM-,%C2%BFQu%C3%A9%20es%20un%20ciberataque%3F,sistema%20inform%C3%A1tico%20o%20dispositivo%20digital.

[2] NPR. (2021). The SolarWinds Attack: The Story Behind The Hack. Recuperado de https://www.npr.org/2021/04/20/989015617/the-solarwinds-attack-the-story-behind-the-hack.

[3] Delta Protect. (2023). Ciberataques: Qué son y cómo prevenirlos. Recuperado de https://www.deltaprotect.com/blog/ciberataques-que-son-y-como-prevenirlos.

[4] Telefónica. (s.f.). ¿Qué es la ciberseguridad y por qué es importante?. Recuperado de https://www.telefonica.com/es/sala-comunicacion/blog/que-es-la-ciberseguridad-y-por-que-es-importante/.

[5] Newtral. (2023). Ciberataques en España 2023. Recuperado de https://www.newtral.es/ciberataques-espana-2023/20240327/.

[6] Reuters. (2022, mayo 10). Factbox: The cyber war between Ukraine, Russia. Recuperado de https://www.reuters.com/world/europe/factbox-the-cyber-war-between-ukraine-russia-2022-05-10/.

[7] DragonJAR. (s.f.). OWASP Top Ten Project en Español. Recuperado de https://www.dragonjar.org/owasp-top-ten-project-en-espanol.xhtml.

[8] Kaspersky. (s.f.). ¿Qué es una inyección SQL (SQL Injection)?. Recuperado de https://latam.kaspersky.com/resource-center/definitions/sql-injection.

[9] Delta Protect. (s.f.). Inyección SQL. Recuperado de https://www.deltaprotect.com/blog/inyeccion-sql.

[10] Candelario Brito, A. (2023). Fundamentos de la Ciencia de Datos [Trabajo de Fin de Grado, Universidad de La Laguna] https://riull.ull.es/xmlui/handle/915/33179.

[11] IBM. (s.f.). ¿Qué es un árbol de decisión?. Recuperado de https://www.ibm.com/es-es/topics/decision-trees.

[12] Moreno Vega, M. (2023). Árboles de clasificación [Transparencias, Universidad de La Laguna].

[13] IBM. (s.f.). ¿Qué es la regresión logística?. Recuperado de https://www.ibm.com/es-es/topics/logistic-regression.

[14] Bravo, F. (s.f.). *Support Vector Machines*. Recuperado de https://felipebravom.com/teaching/svm.pdf.

[15] Torres, A. (s.f.). *Máquinas de Vectores de Soporte (SVM)*. Recuperado de https://albertotb.com/curso-ml-R/Rmd/07-svm/07-svm.html.