



**FACULTAD DE INGENIERÍA, ARQUITECTURA Y  
URBANISMO**

**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS**

**TESIS**

**ANÁLISIS COMPARATIVO DE ALGORITMOS DE  
APRENDIZAJE AUTOMÁTICO PARA  
IDENTIFICAR ATAQUES DE INYECCIÓN SQL A  
BASE DE DATOS EN APLICACIONES WEB**

**PARA OPTAR EL TÍTULO PROFESIONAL DE  
INGENIERO DE SISTEMAS**

**Autor**

**Bach. Castro Fernández, Levi Ronald**

**Orcid: <https://orcid.org/0000-0001-9861-6847>**

**Asesor**

**Dr. Chirinos Mundaca, Carlos Alberto**

**Orcid: <https://orcid.org/0000-0002-6733-8992>**

**Línea de Investigación**

**Infraestructura, Tecnología y Medio Ambiente**

**Pimentel - Perú**

**2022**

**ANÁLISIS COMPARATIVO DE ALGORITMOS DE APRENDIZAJE  
AUTOMÁTICO PARA IDENTIFICAR ATAQUES DE INYECCIÓN SQL A  
BASE DE DATOS EN APLICACIONES WEB.**

**Presentado por:**

---

**Castro Fernández Levi Ronald**  
**Autor**

---

**Dr. Chirinos Mundaca Carlos Alberto**  
**Asesor**

---

**Mg. Vásquez Leyva Oliver**  
**Presidente de jurado**

---

**Mg. Bances Saavedra David Enrique**  
**Secretario de jurado**

---

**Mg. María Noelia Sialer Rivera**  
**Vocal de jurado**

## **Dedicatoria**

Dedico este trabajo a Dios por darme la oportunidad de seguir disfrutando de la vida para seguir luchando por las metas trazadas, a mis padres, a mi esposa y a mis hijos por acompañarme en este proyecto profesional que me he trazado y que lo estoy cumpliendo con el desarrollo de este trabajo de investigación.

## **Agradecimientos**

Especialmente, para la Universidad Señor de Sipán, por brindarme la oportunidad de continuar mi formación profesional; seguido a los docentes, quienes compartieron sus enseñanzas que me brindaron las herramientas para seguir creciendo profesionalmente.

## Resumen

Un alto porcentaje de las organizaciones públicas como privadas actualmente han migrado sus actividades del mundo del papel hacia lo digital, implicando que la información se encuentre disponible en la red. Personas con propósitos mal intencionados con intereses de conocer o apoderarse de la información, se aprovechan de las vulnerabilidades de las aplicaciones implementadas por las organizaciones, para lograr su objetivo aplican diversos tipos de técnicas y métodos para lograr acceder a la información contenida en base de datos aprovechando las deficiencias de las aplicaciones web, apoyados de inyección de código SQL desde los elementos de un formulario o a través de la URL.

Por lo tanto, en esta investigación se plantea un análisis comparativo de los algoritmos de aprendizaje automático para la mitigación de los ataques de inyección SQL, comprende las etapas de clasificación de los algoritmos según su rendimiento en cuanto a precisión, luego la clasificación de los ataques de inyección SQL, extracción de los datos para su procesamiento y aplicación de los algoritmos de aprendizaje automático seleccionados para el análisis de los datos clasificados. La clasificación de los ataques de inyección SQL a base de datos, se construyó una tabla de clasificación por tipo de ataque y el nivel de riesgo que significan las firmas, para la implementación de los algoritmos de aprendizaje automático se realizó en el entorno de trabajo de Phyton, se utilizó librerías para la implementación de los algoritmos de aprendizaje automático AdaBoost, SVM y Decision Tree, Phyton en el entorno de trabajo Jupiter Notebook y scikit-learn demostraron ser un buen entorno para la implementación de los algoritmos por la existencia de múltiples librerías.

Se evaluó la precisión de tres algoritmos de aprendizaje automático poniendo a prueba con la data extraída después de recogerlo de los ataques realizados en un entorno de un sitio web, se encontró que el algoritmo Decision Tree demostró una mejor precisión obteniendo el 100% al momento de realizar el análisis.

*Palabras clave: algoritmos de aprendizaje automático, AdaBoost, Árbol de Decisiones, inyección SQL, Support Vector Machines.*

## **Abstract**

A high percentage of public and private organizations have currently migrated their activities from the paper world to the digital world, implying that the information is available on the network. People with ill-intentioned purposes with interests to know or take possession of the information, take advantage of the vulnerabilities of the applications implemented by the organizations, to achieve their goal they apply various types of techniques and methods to gain access to the information contained in database taking advantage of the deficiencies of web applications, supported by SQL code injection from the elements of a form or through the URL.

Therefore, in this research a comparative analysis of machine learning algorithms for the mitigation of SQL injection attacks is proposed, it comprises the stages of classification of algorithms according to their performance in terms of accuracy, then the classification of SQL injection attacks, extraction of data for processing and application of the selected machine learning algorithms for the analysis of the classified data. The classification of SQL injection attacks to database, a classification table was built by type of attack and the level of risk signified by the signatures, for the implementation of machine learning algorithms was performed in the Python working environment, libraries were used for the implementation of machine learning algorithms AdaBoost, SVM and Decision Tree, Python in the working environment Jupiter Notebook and scikit-learn proved to be a good environment for the implementation of the algorithms by the existence of multiple libraries.

The accuracy of three machine learning algorithms was evaluated by testing with the data extracted after collecting it from the attacks performed in a website environment, it was found that the Decision Tree algorithm showed a better accuracy obtaining 100% at the time of performing the analysis.

**Keywords:** machine learning algorithms, AdaBoost, Decision Tree, SQL injection, Support Vector Machines.

## Índice

I. INTRODUCCIÓN .....	10
1.1. Realidad Problemática .....	10
1.2. Antecedentes de Estudio .....	12
1.3. Teorías relacionadas al tema .....	18
1.4. Formulación del Problema .....	38
1.5. Justificación e importancia del estudio .....	38
1.6. Hipótesis .....	39
1.7. Objetivos .....	39
1.7.1. Objetivos General .....	39
1.7.2. Objetivos Específicos .....	39
II. MATERIAL Y MÉTODO .....	39
2.1. Tipo y Diseño de Investigación. ....	39
2.1.1. Tipo de investigación .....	39
2.1.2. Diseño de investigación .....	40
2.2. Población y muestra.....	40
2.3. Variables, Operacionalización.....	41
2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad. ..	43
2.5. Procedimientos de análisis de datos.....	43
2.6. Criterios éticos. ....	46
2.7. Criterios de Rigor científico .....	47
III. RESULTADOS.....	48
3.1. Resultados en Tablas y Figuras.....	48
3.2. Discusión de resultados .....	54
3.3. Aporte práctico. ....	57
IV. CONCLUSIONES Y RECOMENDACIONES .....	88
4.1. Conclusiones.....	88
4.2. Recomendaciones .....	89
REFERENCIAS .....	91
ANEXOS .....	96

## Índice de Figuras

Figura 1. Los 10 principales ataques web. (OWASP, 2017) .....	11
Figura 2. Gráfico de flujo de señal del perceptrón. (Haykin, 2008).....	28
Figura 3. Hiperplano como límite de decisión para un problema de clasificación de patrones bidimensional y de dos clases. (Shalev & Ben, 2014) .....	30
Figura 4. Modelo de regresión lineal (Shalev & Ben, 2014) .....	31
Figura 5. Árboles de decisión. (Charris, y otros, 2014) .....	32
Figura 6. kNN para un problema de 3 clases con $k = 5$ .....	34
Figura 7. Estructura gráfica de un modelo de Naive Bayes .....	35
Figura 8. Ilustración de SVM. Tomado de (Batta, Singh, Li, Ding, & Trajkovic, 2018) ...	37
Figura 9. Tiempo de respuesta promedio según el algoritmo implementado. ....	48
Figura 10. Consumo de CPU de algoritmos de aprendizaje Automático.....	49
Figura 11 Consumo de memoria expresados en Megabytes. ....	50
Figura 12. Ventana principal de OWASP ZAP 2.9.0 .....	70
Figura 13. Formulario de acceso.....	70
Figura 14 Conjunto de datos sin procesar recogidos con Owasp Zap. ....	72
Figura 15. Muestra del dataset construido en formato CSV .....	75



## Índice de tablas.

Tabla 1 Población de algoritmos de aprendizaje automático. ....	40
Tabla 2 Muestra de algoritmos de aprendizaje automático seleccionados para el desarrollo de la investigación .....	41
Tabla 3 Matriz de operacionalización de variables.....	42
Tabla 4 Matriz de confusión. ....	47
Tabla 5 Matriz de confusión del algoritmo SVM .....	51
Tabla 6 Matriz de confusión del algoritmo AdaBoost .....	51
Tabla 7 Matriz de confusión del algoritmo Decision tree .....	52
Tabla 8 Medidas de rendimiento de los algoritmos de aprendizaje automático seleccionados.....	53
Tabla 9 Clasificación de los ataques de Inyección SQL.....	59
Tabla 10 Caracterización del Dataset .....	73
Tabla 11 Descripción de la extracción de datos para el Dataset.....	74
Tabla 12 Calificación de los niveles de precisión de los algoritmos de aprendizaje automático en la detección de ataques de inyección SQL .....	76
Tabla 13 Evaluación de los algoritmos de aprendizaje automático en la detección de ataques de inyección SQL.....	77
Tabla 14 Algoritmos seleccionados según el grado de precisión.....	78
Tabla 15 Parámetros que se utilizaron al implementar el algoritmo SVM. ....	80
Tabla 16 Parámetros que se utilizaron al implementar el algoritmo SVM. ....	83
Tabla 17 Parámetros que se utilizaron al implementar el algoritmo SVM. ....	85
Tabla 18 Parámetros utilizados para los algoritmos.....	88
Tabla 19. Evaluación de la población de algoritmos de aprendizaje automático para determinar la muestra.....	99
Tabla 20 Matriz de revisión de papers para la selección de la muestra de algoritmos de aprendizaje automático. ....	101

## **I. INTRODUCCIÓN**

### **1.1. Realidad Problemática**

Hoy en día casi todos los sistemas y datos están digitalizados y almacenados en base de datos gestionados por Sistemas de Gestión de Base de Datos. Por eso, es importante conocer la situación y el nivel de riesgo de los activos en todos los niveles organizacionales públicos, privados, independientes o individuales. Existen muchas amenazas cibernéticas diferentes o ataques que pueden afectar la continuidad del negocio o la misión. Si la conciencia de estas amenazas se comparte entre organizaciones podría utilizar como una alerta temprana y predecir una preparación para hacer frente a las nuevas amenazas (Kokkonen y Hämäläinen, 2016). Entonces, para hacer frente se tiene que implementar mecanismos que orienten a la protección de los datos e información con base al planteamiento de exploraciones continuas y a partir de ellas plantear los cambios y mejoras que sean necesarios.

La gestión de seguridad en los sistemas de información se ha convertido en una tarea desafiante, la mayoría de las fallas de seguridad en la información ocurren debido a violaciones de los controles por parte del personal que interactúa con los sistemas, seguido por la disponibilidad de los datos y la información. Esto sugiere que, la gestión de la seguridad de la información solo puede garantizarse adecuadamente si el énfasis va más allá de los controles técnicos e incorpora los procesos de negocio y cuestiones organizativas. La gestión de seguridad de la información se ocupa principalmente de cuestiones estratégicas, tácticas y operativas que rodean la planificación, el análisis, el diseño, la implementación y el mantenimiento del programa de seguridad de la información de una organización. Un alto porcentaje de las vulneraciones es producto de los ataques a los sistemas que presentan la información en línea. (Romero, et al. 2018)

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 los curiosos o (hackers, crackers u otro experto en el tema) que desean saber que hay almacenado en la base de datos para explotarlos, aplican técnicas de ataques de inyección estructurada de lenguaje de consulta

SQL, la cual es una técnica de inyección de código en la que se insertan sentencias SQL maliciosas en la base de datos SQL simplemente usando navegadores web. (Romero, 2019)

El ataque de inyección SQL puede causar daños graves en una base de datos SQL, como es la pérdida de datos, revelar información confidencial o incluso cambiar los valores de los datos. Este tipo de ataque ha sido calificado como el ataque número uno en el top ten del Proyecto de Seguridad de Aplicaciones Web (OWASP, 2017). Esto nos conlleva a desarrollar la siguiente investigación en la cual se pretende evaluar el algoritmo de aprendizaje automático más pertinente para detectar ataques de inyección SQL a base de datos desde aplicaciones web,

→	OWASP Top 10 - 2017
→	A1:2017-Injection
→	A2:2017- Broken Authentication
↘	A3:2017- Sensitive Data Exposure
U	A4:2017- XML External Entities (XXE) [Nuevo]
↘	A5:2017- Broken Acces Control [Fusionado]
↗	A6:2017- Security Misconfiguration
U	A7:2017- Cross-Site Scripting (XSS)
⊗	A8:2017- Insecure Deserialization [Nuevo, Comunidad]
→	A9:2017- Using Components with Known Vulnerabilities
⊗	A10:2017- Insufficient Logging&Monitoring [Nuevo, Comunidad]

Figura 1. Los 10 principales ataques web. (OWASP, 2017)

Como vemos en la Figura 1. El principal tipo de ataque que lo clasifica Owasp en el top 10 es Injection, para enfrentar estos problemas de ataques de inyección SQL, desde aplicaciones web, se tiene a los algoritmos de aprendizaje automático; por tanto, con la presente investigación se busca determinar la

precisión de los algoritmos de aprendizaje automático para predecir vulnerabilidades de las aplicaciones web desarrolladas para las instituciones educativas públicas del Perú, las mismas que se encuentran desarrolladas en software libre y que avizoran la siguiente problemática al momento de la implementación: framework de desarrollo con configuraciones básicas por defecto, tipos de datos incorrectamente filtrados, no se realiza la validación de los formularios, no existen restricciones de acceso a los datos, ni restricciones basadas en roles, entre otras malas prácticas utilizadas al momento de programar aplicaciones usando software libre (OWASP, 2017). Entonces, se busca recoger datos y luego procesarlos a través de algoritmos de aprendizaje automático, para ello se tomó como datos cualitativos las URL y se procedió a cuantificar de acuerdo al tipo de inyección detectado en la cadena de consulta.

## **1.2. Antecedentes de Estudio**

Mishra (2019), realizó la investigación “SQL Injection Detection Using Machine Learning”, en San Jose State University, en donde se precisó que las aplicaciones basadas en web que aceptan información crítica de los usuarios que luego almacenan esta información en base de datos. Estas aplicaciones conectadas a base de datos son susceptibles a todo tipo de amenazas de seguridad de la información debido a su acceso a través de Internet. En esta investigación realizada se presentó una propuesta de implementación de un algoritmo llamado Gradient Boosting Classifier de ensemble machine, basado en enfoques de aprendizaje para clasificar y detectar ataques de inyección SQL. Se implementaron dos algoritmos de clasificación de aprendizaje automático el clasificador Naïve Bayes y el clasificador de aumento de gradiente. El clasificador de Bayes proporciona resultados con una precisión del 92,8%. Mientras que Gradient Boosting Classifier logró un 97.4% de precisión, concluyendo que el Clasificador Gradient presenta una mejor precisión que el Naive Bayes. El modelo de aprendizaje automático se puede mejorar su rendimiento con la creación de nuevas características.

Almseidin, Szilveszter, Al-kasassbeh (2017), realizaron la investigación: “Evaluation of machine learning algorithms for intrusion detection system”

presentado en la 15th International Symposium on Intelligent Systems and Informatics (SISY), en donde definieron que los ataques de inyección afectan a muchos sitios web de las empresas e instituciones que mantienen en línea sus servicios, las políticas de implementación de los sitios presentan ciertas deficiencias, lo que provoca que los atacantes logren el objetivo de vulnerar la seguridad del servicio accediendo a la información sensible. Es por ello que se planteó la implementación de los algoritmos de aprendizaje automático J48, Bosque aleatorio, árbol aleatorio, tabla de decisiones, MLP, ingenuo, Bayes y Bayes Network para evaluar y precisar el modelo del sistema de detección de intrusiones en los sistemas informáticos. Los resultados obtenidos demostraron que no existe un único algoritmo de aprendizaje automático que puede manejar eficientemente todos los tipos de ataques. Los resultados lo demostraron de la siguiente manera: la tabla de decisiones alcanzó el valor falso negativo más bajo de (0.002), pero fue lejos de la detección de la tasa de precisión más alta. En el otro lado, el clasificador de red Bayes tuvo el valor más alto, detectando correctamente los paquetes normales. Bosque al azar clasificador registró la tasa de precisión más alta 93.77%, con el valor RMSE más pequeño y tasa de falsos positivos. Parece que el clasificador de bosque aleatorio presenta un rendimiento aceptable parámetros excepto el parámetro falso negativo. Todos los clasificadores de aprendizaje automático pudieron construir sus modelos de entrenamiento en un período aceptable de tiempo excepto el MLP.

Dong et al. (2018), realizaron la investigación “An adaptive system for detecting malicious queries in web attacks”, en la University of Chinese Academy of Sciences, planteando que las cadenas de consulta de solicitud web que pasan por el protocolo HTTP siempre son manipuladas por los atacantes para acceder a los datos confidenciales de las empresas, también buscan controlar los servidores y aplicaciones web de las organizaciones. Entonces, frente a este problema plantearon enfoques que buscaron la detección de consultas maliciosas, proponiendo un sistema adaptativo para la detección mediante procesos de adaptación para mitigar los ataques de inyección de código basados en la web, que son la mayoría de los ataques web que reciben las aplicaciones que se encuentran en línea, para consolidar la propuesta se realizó aplicando la

metodología de análisis de consultas. Los resultados obtenidos demostraron que la propuesta planteada supera los métodos de detección de ataques web existentes con una precisión del 99.50%. También se considera que el número de consultas maliciosas recogidas es 3.07 veces mayor que el método popular de aprendizaje adaptativo de máquina de vectores de soporte. Entonces, como conclusión se tiene que las consultas maliciosas obtenidas se pueden usar para actualizar la biblioteca de firmas de detección de ataques de inyección, permitiendo que el sistema adaptativo sea una propuesta relevante para mitigar los ataques de inyección SQL.

Oliveira et al. (2018), realizaron la investigación “Fuzzy neural networks to create an expert system for detecting attacks by SQL Injection.”, universidad Vale do Rio Verde. Los ataques de inyección SQL dirigidos a los servicios web públicos, administrados por el gobierno estatal, provocan que la información y su accesibilidad se vea afectada en la disponibilidad para el usuario final. Por esta razón propusieron la implementación de un sistema que tuvo como base a los modelos híbridos profundos, el sistema propuesto trabaja con el uso de reglas difusas que facilitaron la construcción de los ataques de inyección SQL dirigidos a obtener los datos de las aplicaciones conectadas a las bases de datos. Los resultados obtenidos después de aplicar el análisis con las redes neuronales demostraron que es factible para la mitigación de ataques de inyección SQL a partir del entrenamiento del modelo, considerando el uso de reglas difusas, con la precisión de clasificación de las intrusiones dentro del margen de la desviación estándar. Este modelo propuesto servirá como modelo para que los países pueden asumir como referencia la protección de la información y los datos, también creando nuevos escenarios para la automatización de identificar los ataques de inyección SQL.

Ogbomon, Buchanan, y Fan (2017), en su investigación titulada “Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention”, presentado en Symposium on Integrated Network and Service Management (IM), en la que se precisó que los ataques de inyección SQL siguen siendo uno de las modalidades de vulneración preferidos por los intrusos para atacar las

aplicaciones web y extraer sin autorización la información confidencial de las bases de datos, provocando en las organizaciones pérdidas muy elevadas. Para hacer uso predictivo alternativo de Machine Learning, proporcionando de esta manera una minería funcional y escalable a big data en detección y prevención de SQL. Para hacer uso de los algoritmos de aprendizaje automático en escenarios combinados con Big Data en un proceso reciente que busca integrarse, esto debido a la falta de disponibilidad del conjunto de datos para usarlos con patrones e historiales que faciliten determinar un ataque; entonces, ahí se tiene que hacer un entrenamiento de los clasificadores de ataques de inyección SQL. La propuesta fue generar conjunto de datos que tenían ciertos patrones que determinaban los ataques eran parte de una cadena de inyección SQL. Después de aplicar la propuesta se determinó una precisión de 0.986 lo que indica que el modelo tenía un excelente desempeño en la predicción de ataques de inyección SQL. Con el desarrollo de este trabajo demostraron que la aplicación del análisis predictivo a la detección y prevención de SQLIA en el contexto de big data con la aplicación de Support Vector Machines demostró un excelente resultado que se evalúa empíricamente en la matriz de confusión y el gráfico ROC presentados anteriormente.

Tang, Qiu, Huang, Lian, y Liu (2020), en su investigación realizada "Detection of SQL injection based on artificial neural network", en la University, Shanghai, China. La inyección SQL, es un tipo de ataque web común, ha sido un problema de seguridad de red desafiante que causa anualmente millones de dólares en pérdidas financieras en todo el mundo, así como una gran cantidad de fugas de datos de privacidad de los usuarios. En el trabajo se planteó el desarrollo de un método de detección de inyección SQL de alta precisión basado en red neuronal, para ello en un primer momento se adquirió los datos de registro de acceso a URL de usuario auténticos de un proveedor de servicios de Internet, con esto se buscó garantizar que la investigación sea desarrollada en un entorno real promoviendo la clasificación efectiva y práctica. Después de recogida y clasificada la información, procedieron a realizar las pruebas del modelo para determinar datos normales y datos infectados con código de inyección SQL. Con base en los resultados obtenidos diseñaron ocho tipos de características y entrenaron el

modelo de la red de neuronas artificiales. La precisión del modelo se refleja por encima del 95%. Entonces, la aplicación de este modelo en la detección de inyección SQL demostró ser efectivo en la mitigación de inyección SQL, también se considera muy importante el método de extracción de características propuesto a otros modelos de redes neuronales

Hasan, Balbahaith y Tarique (2019), en su investigación titulada “Detection of SQL Injection Attacks: A Machine Learning Approach”, investigación realizada en la universidad Technology, University of Science & Technology of Fujariah. La metodología de aprendizaje automático que presentaron fue en función de evaluar los algoritmos más usuales en la detección de inyección de código SQL, como resultado de esa evaluación se determinó que los algoritmos tradicionales tienen escaso nivel de respuesta en la mitigación de este tipo de ataques. Por lo que, en el estudio se propuso la aplicación de algoritmos de aprendizaje automático basados en algoritmos heurísticos que permitieron aumentar el nivel de prevención de estos tipos de ataques que buscan la vulneración de la seguridad de las aplicaciones, como población se usó un conjunto de datos de 616 URL que contenían inyección de código SQL, para el entrenamiento se tuvo que usar 23 máquinas con escenarios diferentes donde se evaluaron el rendimiento de los clasificadores de aprendizaje automático. De estos clasificadores, se seleccionaron cinco mejores clasificadores basados en su precisión de detección y desarrollar un gráfico aplicación de interfaz de usuario basada en cinco clasificadores. Después de la puesta a prueba se determinó que los algoritmos Ensemble Boosted and Bagged Trees classifiers tuvieron la mejor precisión con un 93,8%. El área debajo de las curvas características de funcionamiento del receptor para ambos se encuentra por encima de 0,9, lo que indica que el sistema realiza la detección con una alta precisión permitiendo detectar los ataques y mitigarlos antes de que se afecte los sistemas de base de datos en línea.

Pattewar, et all. (2019), en su investigación: “SQL Injection Detection Using Machine Learning”. Los programadores de software utilizan las consultas SQL para automatizar las funciones de base de datos como parte de la programación de aplicaciones que interactúan con base de datos, las consultas utilizadas se apoyan de argumentos que validan que se regresen registros solicitados al



momento de ejecutar una consulta. Los atacantes buscan aprovechar las vulnerabilidades a través de la inyección SQL, invadiendo el formulario de acceso. La propuesta de los autores estuvo con base de la aplicación de algoritmos de aprendizaje automático de clasificación, los cuales se implementaron para mitigar el problema, los algoritmos de aprendizaje automático implementados fueron el Clasificador Naive Bayes y el Gradiente. Después de aplicar la propuesta se llegó a la conclusión que el clasificador Naive Bayes proporcionó resultados con una precisión del 92,8%, brindando una mayor precisión, debido a que implementan múltiples clasificadores simples para mejorar el error; por lo tanto, se selecciona el Clasificador de aumento de gradiente del aprendizaje en conjunto para implementarlo en la inyección SQL problema de clasificación. Quedando como una propuesta para la ejecución en la mitigación de los ataques de inyección SQL hacia las bases de datos.

Xie, Ren, Fu, Xu, y Guo (2017), en su investigación “SQL Injection Detection for Elastic-Pooling CNN-based Web Applications”, los datos de una empresa pueden ser uno de sus activos más importantes y, a menudo, críticos para el desarrollo y la supervivencia de la empresa. Los ataques de inyección SQL, vienen ocupando un lugar importante entre los principales riesgos que se tiene que considerar al momento de implementar aplicaciones web, los autores plantean el desarrollo de un método de detección de inyección SQL basado en Elastic-Pooling CNN y realizan la comparación con métodos tradicionales utilizados para la detección de inyección SQL. Este método plantea la generación de modelos de matrices bidimensionales fijas sin truncar los datos y de esta manera llevar a cabo eficiente de los ataques de inyección de códigos SQL que se utilizan para vulnerar la información de las aplicaciones web. Para su aplicación en otros escenarios se tiene que definir las características de coincidencia irregular, con ello se logra la identificación de nuevos ataques y es más difícil de eludir gracias a la definición de las reglas.

Kamtuo & Soomlek , (2017), en su investigación: “Machine Learning for SQL Injection Prevention on Server-Side Scripting”, investigación desarrollada para la universidad Khon Kaen University, La inyección SQL es la vulnerabilidad de

aplicaciones web más común. El desarrollo de software puede generar involuntariamente la vulnerabilidad durante la fase de desarrollo. El aseguramiento desde la fase de diseño e implementación es importante para prevenir vulnerabilidades. Los autores proponen un marco de prevención para mitigar los ataques de inyección de SQL, para ello utilizaron una plataforma de compilación y el aprendizaje automático. Lo concerniente al aprendizaje automático se describió como el núcleo de la propuesta respaldando la predicción de inyección de código SQL, para ello se utilizó una población de conjuntos de 1.100 datos de vulnerabilidades que permitieron el entrenamiento del modelo de aprendizaje automático basados en algoritmos machine learning. Los resultados indicaron que el algoritmo de aprendizaje automático árbol de decisiones es el mejor modelo en términos de tiempo de procesamiento, mayor eficiencia en la predicción. Por lo que se recomienda el uso de este algoritmo para la prevención de ataques de inyección SQL.

### **1.3. Teorías relacionadas al tema**

#### **1.3.1. Seguridad informática**

La seguridad en los sistemas informáticos es un factor determinante para la protección de datos e información. La intención de la seguridad informática es la protección de los recursos digitales de las organizaciones, como los datos, la información, el hardware y el software. Mediante la selección y aplicación de salvaguardas adecuadas, la seguridad facilita el acercamiento a las metas establecidas por las organizaciones al proteger sus activos materiales, financieros, la posición social, legal, los trabajadores y sus demás recursos físicos e intangible. Desafortunadamente, a veces se considera que la seguridad frustra la misión de la organización al imponer reglas y procedimientos molestos y mal seleccionados a los usuarios, gerentes y sistemas. Por el contrario, las reglas y procedimientos de seguridad bien elegidos no existen por sí mismos; se implementan para proteger activos importantes y, por lo tanto, respaldan la misión organizativa general. (Guttman y Roback, 1995)

### **1.3.2. Intrusiones en sistemas informáticos**

Los sistemas informáticos son vulnerables a los ataques de los intrusos que buscan acceso no autorizado, para ello en los sistemas que están funcionando en línea, disponibles las 24 horas son el objetivo de los atacantes, se lleva a cabo un ataque transmitiendo código malicioso que intenta vulnerar el sistema informático, utilizando argumentos que buscan forzar el acceso directo vulnerando la seguridad. El procesamiento del tráfico de Internet que lleva este argumento de gran tamaño desencadena un desbordamiento de búfer en la aplicación, lo que permite que el código malicioso se ejecute en la computadora de destino (Fillatre & Nikiforov, 2011). Los atacantes emplean intencionalmente implementaciones de códigos que buscan el control y acceso a los sistemas para apoderarse de la información, su código malicioso que superpuestos antes de la transmisión y luego transmitir los paquetes fuera de orden.

Para evitar las intrusiones, se implementan medidas de seguridad para mitigar el acceso de sin autorización. Medidas que necesitan ser implementadas en los dos niveles de utilización de software como equipamiento físico. Implementando desde contraseñas seguras para el acceso a los sistemas y otras medidas que se opte. Podría pensarse que la facilidad en el acceso ya cumple con el delito porque conlleva al acceso a los datos contenidos en los sistemas de información. Los accesos se llevan a cabo con diversos fines, como la modificación, copia, publicación no autorizada y otros tratamientos que el atacante crea conveniente (Hernández V. , 2018).

### **1.3.3. Ataques informáticos**

OWASP Foundation (2017), presenta a los ataques de inyección SQL (Structured Query Language) en el primer orden; los ataques de inyección SQL, NoSQL, OS o LDAP suceden al momento del envío de datos, datos que no son los que el sistema está esperando a un intérprete, lo cual si en muchos de los casos forman parte de un comando o sentencia. Los datos nocivos que ha implementado el atacante pueden mentir al intérprete para lograr ejecutar comandos que permite el acceso forzado a los datos. Todo ello acompañado

de la Pérdida de una autenticación autorizada, violando las aplicaciones pasando por alto todas las funciones implementadas para la autenticación y gestión de sesiones los cuales pueden no haberse implementadas correctamente, ello facilitando a los intrusos los usuarios y passwords, token de sesiones, o buscar otras fallas de implementación para acceder e incurrir en falsificación de la identidad de otros usuarios.

OWASP Foundation también describe la vulneración de datos sensibles, donde bastantes aplicaciones web y APIs no brindan la protección adecuada considerando la sensibilidad de los datos, información como es la financiera, salud, militar, entre otros. Los intrusos pueden sustraer o modificar los datos que no tienen una protección adecuada, lo cual facilita a los atacantes realizar estafas con el sistema financiero, suplantación de identidad u otros delitos. Estos datos sensibles necesitan de métodos de aseguramiento adicionales, como el cifrado en almacenamiento y tránsito. Seguido por las entidades Externas XML (XXE), lo cual explica que la tecnología de implementación web que usan las aplicaciones web son muy antiguas o están mal configuradas lo cual evalúan referencias a entidades externas. Los entes externos maliciosamente son implementados para revelar archivos internos almacenados en servidores los cuales no son actualizados periódicamente, ello provoca que se logre ejecutar código de forma remota. (OWASP, 2017)

#### **1.3.4. Ataques de inyección SQL**

La inyección de SQL es una vulnerabilidad de seguridad que ocurre en las capas de la base de datos de una aplicación. Es el acto de pasar código SQL a aplicaciones web interactivas que se emplean en servicios de bases de datos. La inyección SQL es un método para explotar la base de datos de la aplicación web. Se realiza inyectando las declaraciones SQL como una cadena de entrada para obtener un acceso no autorizado a una base de datos. La inyección de SQL es una vulnerabilidad grave que conduce a un alto nivel de compromiso, generalmente la capacidad de ejecutar cualquier consulta de base de datos. Es un ataque de aplicación basado en web que se conecta a los back-end de la base de datos y permite eludir el firewall. La ventaja del

código inseguro y la validación de entrada incorrecta es asegurada por el atacante para ejecutar comandos SQL no autorizados (Clarke, 2012). El objetivo es hacer que el sistema de base de datos ejecute código dañino que pueda revelar información confidencial. Por lo tanto, puedo decir que el ataque de inyección SQL es un acceso no autorizado a la base de datos. (Hernández V. , 2018)

### **1.3.5. Clasificación de los ataques de Inyección SQL**

Los tipos de ataques de inyección se clasifican de la siguiente manera.

#### **1.3.5.1. Orden sabio.**

Los ataques de este tipo se subdividen en tres tipos de vulneraciones que un atacante puede realizar.

- a. Primer orden de ataque de inyección SQL:** Los ataques de inyección de primer orden son cuando el atacante recibe el resultado deseado de inmediato, ya sea por respuesta directa de la aplicación con la que está interactuando o por algún otro mecanismo de respuesta, como correo electrónico, etc.
  
- b. Segundo orden de ataque de inyección SQL:** El ataque de inyección de segundo orden es la realización de un código malicioso inyectado en una aplicación por un atacante, pero no activado inmediatamente por la aplicación. Las entradas maliciosas son sembradas por el atacante en un sistema o base de datos. Esto se usa para activar indirectamente un SQLIA que se usa más adelante. El atacante generalmente analiza en dónde se utilizará posteriormente la entrada y, por lo tanto, elabora su ataque. La inyección de segundo orden deja un delicado trabajo de detección y prevención. Esto se debe a que el punto de inyección es diferente del punto donde realmente se manifiesta el ataque. (Alonso, Guzmán, Laguna, & Martín, s/f)

- c. Ataque de inyección SQL lateral:** Esta es una técnica que se utiliza para comprometer las bases de datos de Oracle de forma remota. El ataque explota algunos tipos de datos comunes, incluidos fecha y número, que no reciben ninguna entrada del usuario y, por lo tanto, normalmente no se consideran explotables. La base de datos puede ser manipulada por un atacante simplemente usando un poco de codificación creativa. (Clarke, 2012)

#### 1.3.5.2. Ataque ciego de inyección de SQL

Cuando la aplicación web es vulnerable a una inyección de SQL, pero los resultados de la inyección están ocultos para el atacante, se utiliza SQLA ciego. La página mostrada puede ser diferente a la original. Esto depende de los resultados de una declaración lógica inyectada en la declaración SQL legítima llamada para esa página. El SQLIA ciego permite al agente de amenazas 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. (Bonaccorso, 2017) la inyección de SQL ciego se puede clasificar además como:

- a. Inyección SQL ciega de base clásica:** Este ataque se basa en el análisis de una expresión lógica verdadera / falsa. Si la expresión es verdadera, la aplicación web devolverá cierto contenido, y si es falsa, la aplicación devolverá contenido.
- b. Inyección SQL ciega basada en errores:** La inyección SQL ciega basada en errores es la técnica más rápida de explotación de la inyección SQL. Lo mejor de este método es que la información valiosa de varios SGBD se puede almacenar en los mensajes de error en caso de recibir una expresión SQL ilegal. Esta técnica se puede utilizar si la aplicación vulnerable devuelve algún error en el procesamiento de la expresión SQL en el DBMS. (Bonaccorso, 2017)

**d. Inyección SQL doble ciego:** La inyección SQL doble ciego es una técnica en la que todos los mensajes de error y consultas vulnerables se excluyen de la página devuelta por la aplicación web y los resultados de la solicitud no influyen en la página devuelta. La explotación de este tipo de vulnerabilidad utiliza únicamente retrasos en el procesamiento de consultas SQL; es decir, es falso si una consulta SQL se ejecuta inmediatamente, pero si se ejecuta con un retraso de N segundos, entonces es verdadero. (Bonaccorso, 2017)

### 1.3.5.3. Contra la base de datos

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

**a. Manipulación SQL Tautología:** el objetivo de un ataque basado en tautología es inyectar código en una o más declaraciones condicionales de modo que la evaluación sea siempre verdadera. En este tipo de SQLIA, un atacante explota un campo inyectable que se utiliza en el condicional WHERE de una consulta; es decir, las consultas siempre devuelven resultados tras la evaluación de un parámetro condicional WHERE. Todas las filas de la tabla de la base de datos a las que se dirige la consulta se devuelven mientras se transforma el condicional en una tautología. (Bonaccorso, 2017)

Ejemplo: en este ejemplo, un atacante envía “ ’ or 1=1 - -” para el campo de entrada de inicio de sesión, la entrada enviada para los otros campos es irrelevante. La consulta resultante es: `SELECT cuentas FROM usuarios WHERE acceso=’ or 1=1 -- AND clave=’ AND pin=`

El código inyectado en el condicional (`OR 1 = 1`) transforma la cláusula WHERE completa en una tautología.

**b. Manipulación SQL de interferencia:** en este ataque, la consulta se modifica en la forma de una acción que se ejecuta en función de la respuesta a una pregunta verdadera / falsa sobre los valores de los

datos en la base de datos. En este tipo de inyección, el atacante intenta atacar un sitio lo suficientemente seguro como para no proporcionar comentarios aceptables a través de mensajes de error de acceso a la base de datos cuando una inyección ha tenido éxito. (Bonaccorso, 2017)

El atacante debe utilizar un método diferente para obtener la respuesta de la base de datos, ya que los mensajes de error de acceso a la base de datos no están disponibles para él. En esta situación, el atacante inyecta comandos en el sitio y luego observa cómo cambia la función / respuesta del sitio web. Al observar el comportamiento cambiante del sitio, el atacante puede extrapolar no solo los parámetros vulnerables, sino también información adicional sobre los valores en la base de datos. Los investigadores han informado que con estas técnicas han podido lograr una tasa de extracción de datos de 1B/s. (Bonaccorso, 2017)

Ejemplo: Considerar dos posibles inyecciones en el panel de acceso. El primero es usando "Usuario" 1=0 - -" y el segundo, "Usuario' and 1=1 - ". Estas inyecciones dan como resultado las dos consultas siguientes, la primera construida de la siguiente manera: `SELECT Usuarios FROM user WHERE usuario='NombUsuario' and 1=0 -- ' AND Clave=' ' AND pin=0`, y la otra consulta es: `SELECT Usuarios FROM user WHERE usuario=' NombUsuario' and 1=1 -- ' AND Clave=' ' AND pin=0`.

En el primer escenario, la aplicación es segura y la entrada para el inicio de sesión se valida correctamente. En este caso, ambas inyecciones devolverían mensajes de error de inicio de sesión y el atacante sabría que el parámetro de inicio de sesión no es vulnerable. En el segundo escenario, la aplicación es insegura y el parámetro de inicio de sesión es vulnerable a la inyección. La persona o robot envía la primera inyección y en consecuencia recibe un mensaje como error de acceso de sesión, porque siempre se evalúa como falso. Sin



embargo, el atacante no sabe si esto se debe a que la aplicación validó la entrada correctamente y bloqueó el intento de ataque o porque el ataque en sí provocó el error de inicio de sesión. Luego, el atacante envía la segunda consulta, que siempre se evalúa como verdadera. Si en este caso no hay un mensaje de error de inicio de sesión, entonces el atacante sabe que el ataque se produjo y que el parámetro de inicio de sesión es vulnerable a la inyección. (Bonaccorso, 2017)

**c. Consultas básicas de unión:** con esta técnica, el usuario malintencionado engaña al servidor para que devuelva datos que no estaban destinados a ser devueltos por los desarrolladores. En este ataque, un atacante explota un parámetro vulnerable para cambiar el conjunto de datos devuelto para una consulta determinada. Los atacantes hacen esto inyectando una declaración de la forma: UNION SELECT <resto de la consulta inyectada>. Dado que el atacante controla la segunda consulta inyectada por completo, puede usarla para recuperar información de una tabla específica. El resultado de este ataque hace que la base de datos devuelva un conjunto de datos que es la unión de los resultados de la primera consulta original y los resultados de la segunda consulta inyectada. (Bonaccorso, 2017)

**d. Consultas complementarias:** en este SQLIA, los atacantes no tienen como objetivo modificar la consulta; intentan incluir consultas nuevas y distintas en la consulta original. Esta base de datos de resultados para recibir múltiples consultas SQL y puede resultar extremadamente dañina.

Ejemplo: si el atacante ingresa ""'; drop table usuarios - -", en el campo de paso, la aplicación genera la consulta: SELECT cuentas FROM usuarios WHERE acceso='doe' AND clave=""; drop table usuarios -- ' AND pin=123. Después de la ejecución de la primera consulta, la base de datos reconocería el delimitador de consulta (";") y procedería con la segunda consulta inyectada. La ejecución de la segunda consulta

provocaría la eliminación de "usuarios" de la tabla, lo que probablemente dañaría información valiosa (Bonaccorso, 2017)

#### **1.3.5.4. Inyección de código.**

En relación a los ataques de inyección de código, son aquellos que intentan agregar instrucciones o comandos SQL adicionales a la instrucción SQL existente. Este tipo de ataque se usa con frecuencia contra aplicaciones de Microsoft SQL Server, pero rara vez funciona con una base de datos Oracle. Este ataque se puede clasificar como:

**a. Consultas con LIKE:** el atacante intenta manipular la declaración SQL utilizando consultas similares. En consecuencia, la entrada del usuario incorporada en un parámetro de consulta LIKE puede subvertir la consulta, complicar la coincidencia LIKE y, en muchos casos, evitar el uso de índices, lo que ralentiza sustancialmente una consulta. Con algunas iteraciones, una consulta LIKE comprometida podría lanzar un ataque de denegación de servicio al sobrecargar la base de datos.

```
Ejemplo: $sub = mysql_real_escape_string ("%something"); //
still%something mysql_query("SELECT * FROM mensajes WHERE
usuarios LIKE '{$usuarios}%");
```

**b. No coincide el número de columna:** para acumular el conocimiento de este tipo de inyección, considere un ejemplo aleatorio. Deje que la consulta original sea:

```
SELECT nombre_producto FROM todos_productos WHERE
nombre_producto like '&Chairs&', entonces la vulneración se acerca
como la siguiente consulta: SELECT nombre_producto FROM
todos_productos WHERE nombre_producto like " UNION ALL
SELECT 9,9 FROM SysObjects WHERE " = "
```

La consulta anterior daría errores que indican que hay una falta de coincidencia en el número de columnas y su tipo de datos en la unión de la tabla SysObjects y las columnas que se especifican usando 9. El error "El tipo de operando no coincide" se debe principalmente al tipo

de datos desajuste en la cláusula de unión causada por la cadena inyectada. Otro error que puedo ver es "Todas las consultas en una instrucción SQL que contiene un operador UNION deben tener el mismo número de expresiones en su lista de destino" se debe a que el número de columnas no coincide. (Bonaccorso, 2017)

Después de varias pruebas y errores, una declaración como esta puede tener éxito: `SELECT nombre_producto FROM todos_productos WHERE nombre_producto like " UNION ALL SELECT 9,9, 9,' Text', 9 FROM SysObjects WHERE " = "`. Lo resultante sería el conjunto de datos de la consulta anterior mostrará todas las filas en la tabla SysObjects y también mostrará valores de fila constantes para cada fila en la tabla SysObjects definida en la consulta.

- c. Cláusula adicional WHERE:** Surge el caso en el que puede haber una condición WHERE adicional en la declaración SL que se agrega después de la cadena inyectada. Ejemplo: `SELECT nombres, apellidos, titulo from trabajadores WHERE ciudad_origen = ' "& strCity &" ' AND pais = 'peru' "`. En el primer intento después de inyectar la cadena, la consulta resultante se vería así: `SELECT nombres, apellidos, titulo from trabajadores WHERE ciudad = 'lima' UNION ALL Select otrafila from otratabla WHERE 1 = 1 AND pais = 'peru'`

La consulta anterior generará un mensaje de error como este: Nombre de columna no válido 'ciudad' por qué 'otratabla' no tiene una columna llamada 'ciudad'.

- d. Insertar – Sub seleccionar consulta:** El uso de una consulta de inserción avanzada puede ayudar al atacante a acceder consecuentemente a los datos contenidos en la base de datos. Ejemplo: `INSERT INTO usuarios_sist values ( ' " ' + (SELECT TOP 1 usuario FROM usuarios_sist ) + ' ' , 'user@dasa.yeah' , '653214526')`. Siempre que esta información se muestra al usuario, normalmente se ubica en páginas donde los usuarios pueden editar la información del usuario. En el ataque anterior, se mostrará el primer valor en

FieldName en lugar del nombre de usuario. Si no se utiliza TOP 1, aparecerá un mensaje de error “subselect returned too many rows”. El atacante puede revisar todos los registros usando NOT in cláusula (Clarke, 2012).

### 1.3.6. Algoritmos de aprendizaje automático

#### 1.3.6.1. Perceptrón

En relación de los algoritmos de aprendizaje automático podemos mencionar el perceptrón de Rosenblatt, que se construye alrededor de una neurona no lineal, a saber, el modelo McCulloch-Pitts de una neurona, este modelo neuronal funciona como un mezclador lineal destinado por un limitador rígido. El nodo adicionador del modelo neuronal calcula una combinación lineal de las entradas empleadas en las sinapsis, y también incorpora un sesgo aplicado ampliamente. La adición resultante, es decir, el campo local inducido, se aplica a un limitador. El Perceptrón es un algoritmo iterativo que construye una secuencia de vectores  $w^{(1)}$ ;  $w^{(2)}$ ; ... Inicialmente,  $w^{(1)}$  se establece como el vector de todos ceros. En la iteración  $t$ , el perceptrón encuentra un ejemplo  $i$  que está mal etiquetado por  $w^{(t)}$ , es decir, un ejemplo para el cual  $\text{signo}(w^{(t)}; x_i) \neq y_i$  (Haykin, 2008). Entonces, el Perceptrón actualiza  $w^{(t)}$  añadiéndole la instancia  $x_i$  escalada por la etiqueta  $y_i$ . Es decir,  $w^{(t+1)} = w^{(t)} + y_i x_i$ . El objetivo es tener  $y_i(w, x_i) > 0$ , para todo  $i$  y se observa que:  $y_i(w^{(t+1)}, x_i) = y_i(w^{(t)} + y_i x_i, x_i) = y_i(w^{(t)}, x_i) + ||x_i||^2$

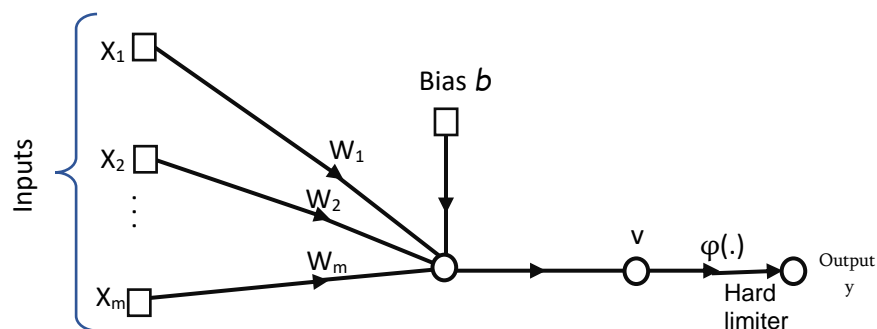


Figura 2. Gráfico de flujo de señal del perceptrón. (Haykin, 2008)

En consecuencia, la neurona da como resultado 1 si el limitador de la entrada es positivo y -1 cuando es negativa. En el modelo de gráfico de flujo de señal de la figura 2, los pesos sinápticos del perceptrón se denotan

por  $w_1, w_2, \dots, w_m$ . En consecuencia, las entradas aplicadas al perceptrón se denotan por  $w_1, w_2, \dots, w_m$ . El sesgo aplicado externamente se denota con  $b$  (Shalev & Ben, 2014). A partir del modelo, encontramos que la entrada del limitador duro, o campo local inducido, de la neurona es:  $v = \sum_{i=1}^m w_i x_i + b$

El objetivo del perceptrón es clasificar correctamente el grupo de estímulos aplicados externamente  $x_1, x_2, \dots, x_m$ , en una de dos clases,  $c_1$  o  $c_2$ . Se utiliza la regla que permite decidir en cuanto a la clasificación asignando el punto visto por las entradas del algoritmo  $x_1, x_2, \dots, x_m$  en la clase que corresponde a  $c_1$  si el resultado del algoritmo perceptrón es  $+1$  y lo correspondiente a la clase  $c_2$  si el resultado es  $-1$ . Para desarrollar una idea del comportamiento de un clasificador que corresponde a los patrones, se debe trazar una guía con las regiones que se evaluarán y definirán la decisión en el espacio de señal  $m$ -dimensional abarcado por las múltiples variables que se necesita para procesar  $x_1, x_2, \dots, x_m$  (Shalev & Ben, 2014). Existe dos regiones de decisión separados por un hiperplano, que se define por:  $v = \sum_{i=1}^m w_i x_i + b = 0$

Como se aprecia en la figura 3 para el caso de dos variables de entrada  $x_1$  y  $x_2$ , para las cuales el límite de decisión toma la forma de una línea recta. Un punto  $(x_1, x_2)$  que se encuentra por encima de la línea límite se asigna a la clase  $c_1$ , y un punto  $(x_1, x_2)$  que se encuentra debajo de la línea límite se asigna a la clase  $c_2$ . Tenga en cuenta también que el efecto del sesgo  $b$  es simplemente desplazar el límite de decisión fuera del origen. Los pesos sinápticos  $w_1, w_2, \dots, w_m$  del perceptrón pueden adaptarse iteración por iteración (Shalev & Ben, 2014).

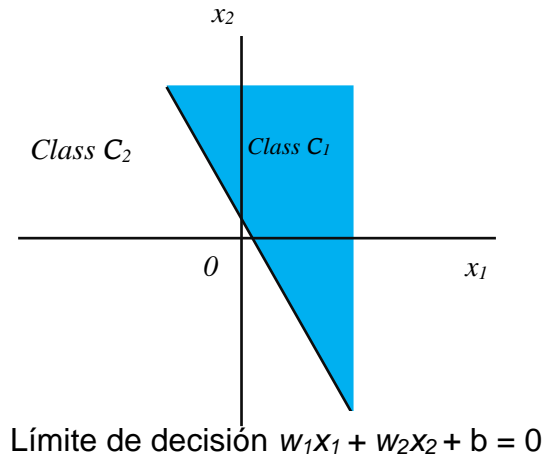


Figura 3. Hiperplano como límite de decisión para un problema de clasificación de patrones bidimensional y de dos clases. (Shalev & Ben, 2014)

### 1.3.6.2. Regresión lineal

La regresión lineal es una técnica que se utiliza en la estadística común con la finalidad de encontrar la relación entre variables \ "explicativas" y algún resultado con valor real. Considerado como un problema de aprendizaje, el conjunto de dominios  $X$  es un subconjunto de  $\mathbb{R}^d$ , para algunos  $d$ , y el conjunto de etiquetas  $Y$  es el conjunto de números reales. Nos gustaría aprender una función lineal  $h: \mathbb{R}^d \rightarrow \mathbb{R}$  que mejor se aproxima a la relación entre nuestras variables. La clase de hipótesis de predictores de regresión lineal es simplemente el conjunto de funciones lineales. (Shalev & Ben, 2014)

$$H_{reg} = L_d = \{x \rightarrow (w, x) + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

La definición de una función de pérdida para la regresión. Si bien en la clasificación la definición de la pérdida es sencilla, ya que  $\ell(h; (x; y))$  simplemente indica si  $h(x)$  predice correctamente y o no, en regresión, si el peso del bebé es de 3 kg, tanto las predicciones 3.00001 kg y 4 kg son \ "incorrectas" ", pero claramente preferiríamos lo primero sobre lo último. Por lo tanto, necesitamos definir cuánto seremos \ "penalizados" por la diferencia entre  $h(x)$  e  $y$ . Una forma común es usar la función de pérdida al cuadrado. (Shalev & Ben, 2014)

$$l(h, (x, y)) = (h(x) - y)^2.$$

Para esta función de pérdida, la función de riesgo empírico se denomina Error cuadrático medio.  $L_s(h) = \frac{1}{m} \sum_{i=1}^m (h(x_i) - y_i)^2$ .

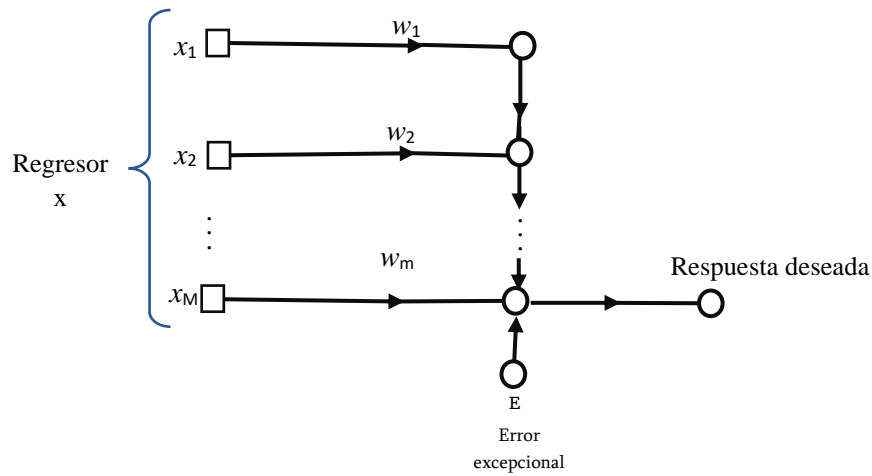


Figura 4. Modelo de regresión lineal (Shalev & Ben, 2014)

### 1.3.6.3. Árbol de decisión

El algoritmo de árboles de decisión es utilizado para las clasificaciones, así como también en las predicciones, este modelo realiza la clasificación de la información considerando los datos asignados como entrada, seguidamente con los datos ingresados pasa por los nodos del árbol que son tomados como los atributos y los posibles valores son los que van entres los ramos entre nodos. (Raschka & Mijarlili, 2017).

La construcción de un algoritmo comience con la construcción del árbol y posteriormente se procede a realizar la poda, esto con la finalidad de hacerlo más eficaz, esta técnica de poda permite la reducción del árbol, la complejidad temporal de la construcción va a ser en función de la magnitud del tamaño de los datos, los atributos y la forma de los datos y posteriormente se considera la forma final del árbol (Charris, y otros, 2014). Existen múltiples árboles de decisión, pero los más utilizados y conocidos en la comunidad son los ID3, C4.5 y ACR. La aplicación de este tipo de algoritmos se da en diversos escenarios, lo importante de este tipo de algoritmos es la efectividad en la clasificación que realiza y la facilidad que nos da el de comprender los resultados. Entonces siempre se basa en el diseño de un árbol que inicia por la raíz, luego seguido por las ramas.

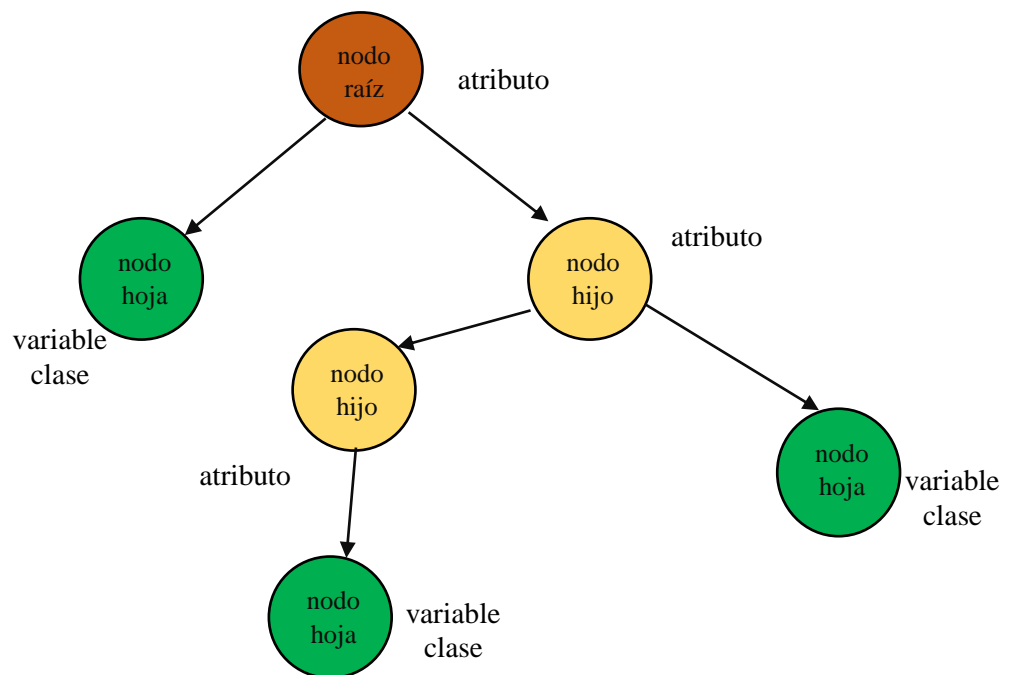


Figura 5. Árboles de decisión. (Charris, y otros, 2014)

El árbol de decisión empieza por la raíz, después las ramas con dirección a la izquierda y derecha y finaliza en las hojas. Para la lectura se tiene que tener en cuenta la siguiente forma: cada nodo intermedio representa a las características de los datos que se van a procesar, para ello consideremos a la raíz como la característica más importante y seguidamente las hojas representan a las clases. En este algoritmo se lleva a cabo la agrupación de datos considerando los valores de atributos que se tiene. Entonces para ello se tiene en cuenta los atributos que tienen mejor distinción de los demás, luego se procede a realizar la recursión hasta lograr todos los datos sean parte de un subconjunto o pertenezcan a la una misma clase. (Raschka & Mijarlili, 2017)

La Implementación de algoritmos de árboles de decisión: las implementaciones se realizan considerando la base, entonces se presente las siguientes implementaciones que se puede realizar: ID3 (Iterative Dichotomiser 3): fue implementado por Ross Quinlan. Este algoritmo crea un árbol con varios caminos y localiza para cada nodo la particularidad categórica que logra obtener mejor ganancia de los demás nodos. A partir de esto se realizan técnicas aplicando la poda en la mejora



y generalizando para los datos que no han sido revisados aún. La otra implementación es el C4.5, que es una evolución de ID3, se diferencia por quitar la restricción de categorización de atributos, haciendo uso del concepto de entropía. (Raschka & Mijarlili, 2017)

#### **1.3.6.4. Neighbors K-Neighbors**

Este es uno de los algoritmos muy conocidos y utilizado en el reconocimiento de patrones y las correlaciones, su aplicación se puede encontrar en el campo financiero para la prevención de delitos de esta naturaleza, sus implementaciones se aplican en la identificación de spam, también es aplicable en el campo de la medicina donde permite determinar si los pacientes ingresan con algún infarto y de esta manera llegue a conclusiones futuras en probables nuevos ataques de infarto. (Bonaccorso, 2017)

Su implementación es bastante sencilla para ser aplicado a la clasificación, los datos para este algoritmo no se asumen como distribuidos, porque bien sabemos que en un caso real no sucede, los datos nunca están ordenados según alguna distribución teórica, por lo que es una buena opción elegir este tipo de algoritmos para realizar el trabajo de clasificación (Bonaccorso, 2017). A menos que se indique lo contrario, la métrica de la distancia predeterminada de los algoritmos KNN es la distancia euclidiana (también llamada distancia L2), que calcula la

distancia entre dos puntos,  $x^{[a]}$  y  $x^{[b]}$ : 
$$d(x^{[a]}, x^{[b]}) = \sqrt{\sum_{j=1}^m (x_j^{[a]} - x_j^{[b]})^2}.$$

El reconocimiento de patrones es una técnica que es abordado por el algoritmo KNN el cual no usa parámetros importantes y es un algoritmo de aprendizaje supervisado. Las reglas de clasificación las generan las muestras al momento de realizar los entrenamientos sin datos adicionales. El algoritmo KNN busca la predicción considerando las muestras de acuerdo a las K muestras seleccionadas para el entrenamiento que corresponden a las vecinas más cercanas con la

prueba, y se juzga considerando la categoría a la que corresponde mejores probabilidades en las categorías. (Bonaccorso, 2017)

El trabajo que realiza el algoritmo KNN al momento de clasificar los elementos de muestras  $X$  es: suponga que hay  $j$  categorías de entrenamiento  $C_1, C_2, \dots, C_j$  y la adición de muestras de entrenamiento es  $N$  considerando la disminución de las propiedades, se convierten en vector de propiedades de dimensión  $m$ . Hacer que la muestra  $X$  sea el mismo vector de características del formulario  $(X_1, X_2, \dots, X_m)$ , teniendo como base a las partes designadas como entrenamiento. En el cálculo de similitudes entre todas las partes segmentadas para las muestras de entrenamiento y  $X$ . Se elige la  $i$ ésima muestra  $d_i$  ( $d_{i1}, d_{i2}, \dots, d_{im}$ ) como ejemplo, la similitud  $SIM(X, d_i)$  es la siguiente. (Bonaccorso, 2017)

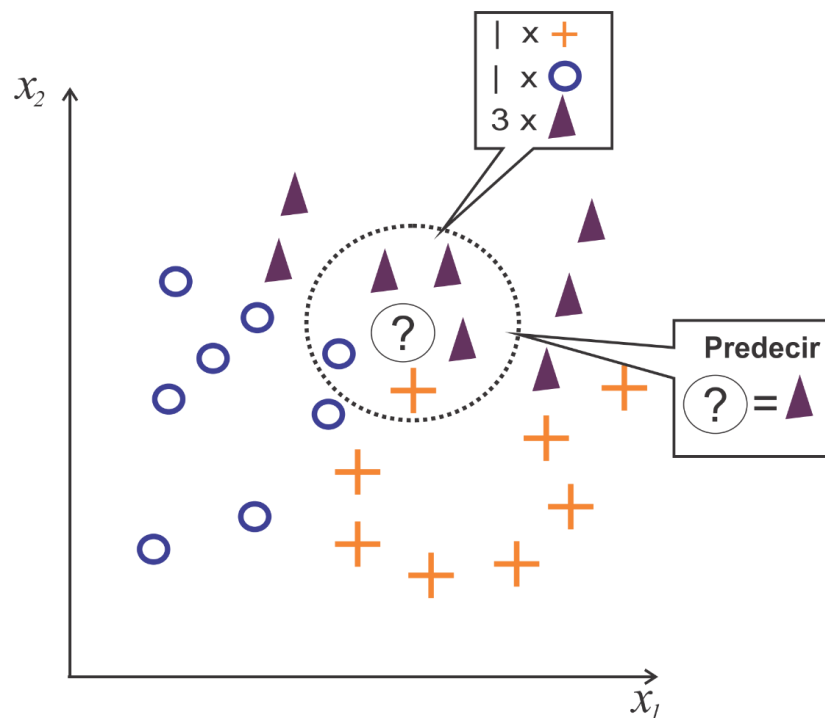


Figura 6. kNN para un problema de 3 clases con  $k = 5$

### 1.3.6.5. Naive Bayes: (clasificador bayesiano)

Este algoritmo de clasificación se utiliza muy a menudo y los resultados son exitosos cuando se trata información de clasificación y análisis de textos, el funcionamiento se basa en el teorema de Bayes en el que se

basa, funciona en base a probabilidades condicionadas es decir en función a algo que ya ocurrió anteriormente, de esta manera podremos calcular lo que ocurrirá(Murphy, 2006), su fórmula es:  $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$

Se detalla de la siguiente manera:  $P(h)$  es la posibilidad de ejecutar positivamente la hipótesis  $h$ ;  $P(D)$  revela la posibilidad de observación de los datos preparados para el entrenamiento  $D$ .  $P(D|h)$  posibilidad de verificar los datos del conglomerado de  $D$  considerando un conjunto de datos de verificación de la hipótesis  $h$ .  $P(h|D)$  es una posibilidad que se tienen después de  $h$  al haber observado el conjunto de entrenamiento  $D$ . (Murphy, 2006). Supongamos que queremos clasificar un documento en una de las clases  $C$  (por ejemplo,  $C = 2$  y las clases son "spam" y "no spam"). Una representación muy simple, llamada modelo de la bolsa de palabras, es ignorar el orden de las palabras y simplemente contar el número de veces que ocurre cada palabra. Supongamos que hay palabras  $D$  en el idioma (siempre podemos hacer cumplir esto mapeando todas las palabras fuera del vocabulario en un símbolo especial como unk, para palabra desconocida). Entonces, un documento se puede representar como un  $p$ -vector de conteos (un histograma de frecuencia de palabras). Sea  $X = k$  significa que la palabra aparece exactamente  $k$  veces, para  $k = 0: K - 1$ ; para simplificar, diremos que esta palabra tiene cuenta  $k$ . (Si la palabra aparece más de  $K - 1$  veces en un documento, la trataremos como si ocurriera  $K - 1$  veces; aquí  $K$  es un límite superior elegido por el usuario). En este caso, podemos representar la clase condicional densidad como producto de multinomios (Murphy, 2006)

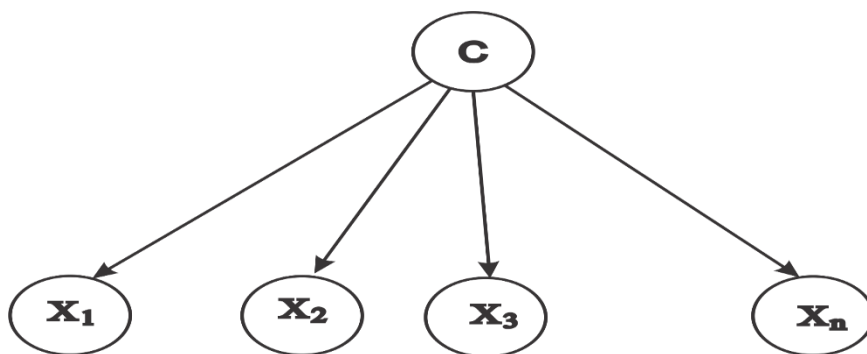


Figura 7. Estructura gráfica de un modelo de Naive Bayes

### 1.3.6.6. Support Vector Machine (SVM).

El Algoritmo de aprendizaje automático Linear Support Vector Machine, es un clasificador discriminativo utilizado para tareas de clasificación y regresión. SVM define un hiperplano de separación para asignar las variables de destino en categorías distintas, como algoritmo de aprendizaje supervisado, SVM es un clasificador no probabilístico que se utiliza para problemas de clasificación y en aplicaciones de reconocimiento de patrones, es una versión modificada de la regresión logística. (Batta, Singh, Li, Ding, & Trajkovic, 2018)

Para un conjunto de datos dado  $x$  con  $n$  número de datos de entrenamiento, SVM encuentra el hiperplano de margen máximo que separa diferentes clases de datos:

$$X = (x_n, y_n), x_n \in \mathbb{R}^p, y_n \in \{-1, 1\}, \forall n = 1, 2, \dots, N$$

Donde  $x_n$  es el vector de entrada, mientras que  $y_n$  representa al valor de salida (1 o -1). Entonces el vector de decisión que delimita las dos clases viene determinado por:

$$w^T \cdot x + b = 0$$

Donde  $w^T$  representa al vector de ponderación óptimo y  $b$  representa al sesgo. Entonces para los datos separados lineales, se usan los márgenes que se definen como:

$$w^T \cdot x + b = 1 \text{ y } w^T \cdot x + b = -1$$

La distancia entre márgenes está dada por  $2/\|w^T\|$ , entonces la función objetivo busca la minimización de  $\|w^T\|$ , es poco posible separar en la práctica los datos que se están entrenando. Entonces considerando que sea  $C$  el parámetro que regularizará y definirá la separación de las dos clases, finalmente el hiperplano se adquiere por minimización (Batta, Singh, Li, Ding, & Trajkovic, 2018).

$$C \sum_{n=1}^n \zeta_n + \frac{1}{2} \|w\|^2$$

Con las restricciones  $\text{tny}(x_n) > 1 - \zeta_n$ ,  $n=1, \dots, N$ ; considerando que  $\text{tn}$  es el dato objetivo y  $\zeta_n$  representa a las variables, que permite que la formulación del problema se realice utilizando el multiplicador doble Lagrangian  $\beta$  como:

$$\max \sum_{n=1}^N \beta_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \beta_n \beta_m y_n y_m (X_n, X_m)$$

La ecuación planteada está restringida por:

$$0 \leq \beta_i \leq C \quad \forall i = 1, 2, \dots, n, \quad \text{and} \quad \sum_{i=1}^n \beta_i y_i = 0$$

En la figura 12 se puede apreciar los datos ubicados en los puntos de clasificación regular representados por círculos y los datos con clasificación irregular representados por estrellas que representan datos anómalos los cuales representan a una clasificación correcta, de igual manera se tiene los datos clasificados incorrectamente, también representados por círculos. (Batta, Singh, Li, Ding, & Trajkovic, 2018)

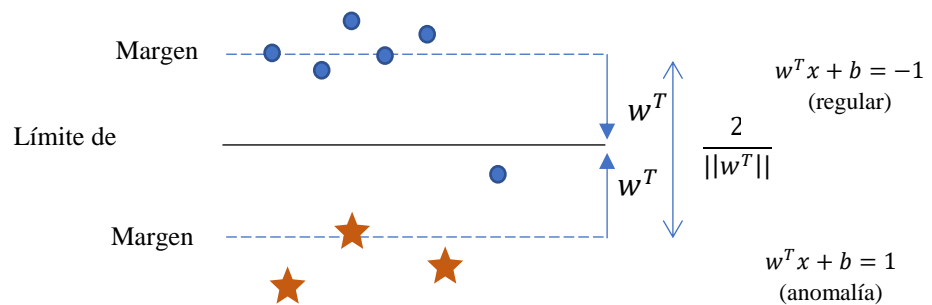


Figura 8. Ilustración de SVM. Tomado de (Batta, Singh, Li, Ding, & Trajkovic, 2018)

#### **1.4. Formulación del Problema**

¿Qué algoritmo de aprendizaje automático presenta mejor precisión en la detección de ataques de inyección SQL a base de datos en aplicaciones Web?

#### **1.5. Justificación e importancia del estudio**

Con el desarrollo de la presente investigación se buscó brindar una solución a las instituciones que ahora están migrando sus actividades al mundo digital, para ello es importante que la implementación de las Tecnologías que utilizarán en el despliegue de sus sistemas cuente con un nivel de seguridad para proteger la información que estará disponible en Internet. Las empresas que utilicen sitios empresariales, sitios de negocios, informativos u otro tipo de actividad, en consecuencia, también se despliega base de datos, entonces se tiene que determinar la mejora de la seguridad de la información que la organización prevea a sus clientes.

Sin embargo, el aspecto tecnológico debe permitir trabajar conjuntamente con los avances recientes que la tecnología está generando, consecuentemente al despliegue de sistemas de información existe un alto número de personas curiosas, malintencionadas que buscan acceder hacia la data que se está compartiendo, por lo tanto, implementan mecanismos para vulnerar la seguridad aprovechando las malas configuraciones de las páginas web que se han construido. Por lo tanto, para hacer frente a este tipo de problemática en esta investigación se propuso la implementación de algoritmos de aprendizaje automático que analicen y determinen los ataques de inyección SQL generadas desde las aplicaciones web para recuperar la información contenida en las bases de datos.

Debido a esta situación se propuso implementar la comparación de los algoritmos de aprendizaje automático, los cuales permitieron contrarrestar los ataques de inyección SQL. De allí la importancia de utilizar mecanismos de seguridad que contrarresten ataques informáticos de tipo inyección SQL que son utilizados para explotar la información de las bases de datos.

## **1.6. Hipótesis**

El algoritmo Decisión Tree presenta mejor precisión para la detección de ataques de inyección SQL a base de datos en aplicaciones Web.

## **1.7. Objetivos**

### **1.7.1. Objetivos General**

Determinar el algoritmo de aprendizaje automático que presenta mejor precisión en la detección de ataques de inyección SQL a base de datos en aplicaciones Web.

### **1.7.2. Objetivos Específicos**

- a. Clasificar los ataques de inyección SQL a base de datos.
- b. Implementar el Dataset para el aprendizaje automático.
- c. Analizar comparativamente los algoritmos de aprendizaje automático para identificar ataques de inyección SQL a base de datos en aplicaciones Web.
- d. Seleccionar los algoritmos de aprendizaje automático para identificar ataques de inyección SQL.
- e. Implementar los algoritmos de aprendizaje automático seleccionados.

## **II. MATERIAL Y MÉTODO**

### **2.1. Tipo y Diseño de Investigación.**

#### **2.1.1. Tipo de investigación**

El presente trabajo de investigación se considera de tipo cuasi experimental, cuantitativa y tecnológica aplicada, investigación que busca la implementación de tecnologías. Investigación que es de naturaleza propia de la Ingeniería vinculada a la innovación tecnológica promoviendo el desarrollo de conocimiento útil que brinde soluciones específicas a los problemas de estudio que son parte de las necesidades de la sociedad actual (Hernández, Fernández, & Baptista, 2014). La investigación tecnológica se desarrolla siguiendo el método científico para validar las hipótesis planteadas, en consecuencia, el resultado obtenido es una aplicación

práctica que permitirá ser aplicado para solucionar problemas o plantear mejoras que ayuden a las organizaciones y a las personas que interactúan con sistemas presentes con sus servicios en Internet.

### 2.1.2. Diseño de investigación

El diseño que se enfoca el presente trabajo de investigación pertenece al diseño cuasi experimental, se buscó la manipulación de datos obtenidos a partir del análisis que se realizó, manipulando deliberadamente la variable dependiente que permitió el analizar el resultado. (Hernández, Fernández, & Baptista, 2014)

## 2.2. Población y muestra.

### 2.2.1. Población

La población está determinada por 15 algoritmos de aprendizaje automático utilizado en diversas investigaciones, los cuales se analizaron para determinar el desarrollo de la investigación.

Tabla 1

*Población de algoritmos de aprendizaje automático.*

<b>Nro</b>	<b>Algoritmo de aprendizaje automático</b>
1	Logistic Regression
2	Support Vector Machine
3	Gradient Boosting Classifier
4	Decision Tree
5	K-Nearest Neighbor
6	Random Forest
7	Naïve Bayes
8	AdaBoost
9	Perceptron
10	Polynomial Support Vector Machine
11	Redes Neuronales
12	Dicotomizador Iterativo (ID3)
13	TensorFlow Linearclassifie
14	Hidden Markov Model
15	Deep ANN



### **2.2.2. Muestra**

Se consideró como muestra tres algoritmos de aprendizaje automático que tienen mayor nivel de precisión en la detección de inyecciones SQL, la selección se realizó de considerando la revisión bibliográfica identificando el escenario donde fueron aplicados. El criterio que se consideró para la selección de los algoritmos de muestra han sido el grado de precisión. (ver anexo 03 y 04)

Tabla 2

*Muestra de algoritmos de aprendizaje automático seleccionados para el desarrollo de la investigación*

---

**Muestra:**

---

1. Support Vector Machine
  2. Decision Tree
  3. AdaBoost
- 

## **2.3. Variables, Operacionalización.**

### **2.3.1. Variable independiente.**

Algoritmos de aprendizaje automático

### **2.3.2. Variable dependiente.**

Detección de ataques de inyección SQL

Tabla 3

Matriz de operacionalización de variables.

<b>Variable independiente</b>	<b>Dimensión</b>	<b>Indicadores</b>	<b>Formula</b>	<b>Técnicas, Instrumentos</b>
<b>Algoritmos de aprendizaje automático</b>	Tiempo	Tiempo de respuesta	$TR = TF - TI$	Observación  Ficha de observación
	Rendimiento	Consumo de CPU	$CPU = \frac{TCPU}{NCPU}$	
		Consumo de memoria.	$cm = \sum_j^n \frac{cm_j}{n}$	
	Clasificación	Tasa de clasificaciones correctas	$T_{respuesta} = T_{final.} - T_{inicio}$	
<b>Variable dependiente</b>	<b>Dimensión</b>	<b>Indicadores</b>	<b>Formula</b>	<b>Técnicas, Instrumentos</b>
<b>detección de ataques de inyección SQL</b>	Rendimiento	Precisión	$Precisión = \frac{VP}{VP+FP}$	Observación  Ficha de observación
		Recall	$R = \frac{VP}{VP + FN}$	
	F-Score	$F = (2) \frac{P * R}{P + R}$		
	AUC	$AUC = \sum_{i \in (P+N)} \frac{(TPR_i + TPR_{i-1})(FPR_i - FPR_{i-1})}{2}$		

## **2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.**

### **2.4.1. Técnica de recolección de datos.**

#### **a. Observación.**

Esta técnica se ha empleado para observar los criterios de precisión, exactitud, F-Score, AUC de los algoritmos de aprendizaje automático de las pruebas de verificación.

### **2.4.2. Instrumentos de recolección de datos**

#### **a. Ficha de observación**

El instrumento utilizado para el desarrollo de la investigación, evaluación y recojo de datos, concerniente a los objetivos específicos, instrumento que se determinaron las variables específicas y se usaron para consignar los datos con la finalidad de brindar sugerencias para el análisis más conveniente.

## **2.5. Procedimientos de análisis de datos.**

Para determinar las métricas de evaluación de los algoritmos de aprendizaje automático en la identificación de ataques de inyección SQL se realizará con el instrumento de un registro de datos que permite evaluar los indicadores de consumo de CPU, consumo de memoria, las fórmulas que permiten la recolección de datos se determinan a continuación.

a. Fórmula para determinar el tiempo de respuesta:

$$TR = TF - TI$$

Donde:

**TR:** tiempo de respuesta.

**TF:** tiempo final

**TI:** tiempo inicial.

b. Para determinar el consumo de CPU mediante la fórmula:

$$CPU = \frac{TCPU}{NCPU}$$

Donde:

CPU : determina el consumo de CPU,

TCPU : tiempo que utiliza el CPU en el procesamiento de los datos proporcionados.

NCPU : determina el número de instrucciones que procesa el CPU en un determinado tiempo.

c. Para determinar el consumo de memoria:

$$CM = \sum_j^n \frac{CM_j}{n}$$

Donde:

$CM$ : porcentaje de uso de la memoria

$CM_j$ : porcentaje de uso de memoria durante la prueba  $j$

$n$ : corresponde a la totalidad de pruebas

**La variable de ataques de inyección SQL se miden en función a los indicadores como Recall, F-Score, AUC, Precisión.**

a. El recall, corresponde a la totalidad de predicciones verdaderas y correctas en consultas anómalas SQL:

$$Recall = \frac{VP}{VP + FN}$$

Donde:

VP: corresponde a los verdaderos positivos

VN: corresponde a los verdaderos negativos

FP: corresponde a los falsos positivos

FN: corresponde a los falsos negativos

- b. La exactitud es el porcentaje de consultas anómalas SQL verdaderas que el modelo va a evaluar y a partir de ello determinar el grado de exactitud.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

Donde:

VP: corresponde a los verdaderos positivos

VN: corresponde a los verdaderos negativos

FP: corresponde a los falsos positivos

FN: corresponde a los falsos negativos

- c. F-Score cuando se alcanza un balance entre la precisión y la exhaustividad y así determinar la existencia del riesgo al detectar consultas anómalas SQL:

$$F - Score = 2 * \frac{Precisión * Recall}{Precisión + Recall}$$

Donde:

P: Es la precisión

R: Recall

- d. La curva de ROC, permite calcular el desempeño del modelo, evaluando la exactitud y precisión, para ello se utiliza los siguientes parámetros:

Tasa de Verdaderos Positivos (TPR)

$$TPR = \frac{VP}{VP + FN}$$

Donde:

VP = cantidad de Verdaderos Positivos

FN = cantidad de Falsos Negativos

- e. Tasa de Falsos Positivos (FPR)

$$FPR = \frac{FP}{FP + VN}$$

Donde:

FP = cantidad de Falsos Positivo

VN = cantidad de Verdaderos Negativos

El AUC alcanza a determinar el área por debajo de ROC de la detección en todos los puntos iniciales de clasificación. El resultado se interpreta como como clasificación positivo aleatorio y clasificación negativo aleatorio.

$$AUC = \sum_{i \in (P+N)} \frac{(TPR_i + TPR_{i-1})(FPR_i - FPR_{i-1})}{2}$$

Para determinar la matriz de confusión requiere analizar las reglas de predicción para analizar el nivel de desempeño del algoritmo de aprendizaje automático, para ello se utilizará la técnica de instrumentos físicos o digitales haciendo uso de un instrumento para el registro digital; instrumento que será empleado para el recojo de información cuando se ejecuten los algoritmos de aprendizaje automático en el lenguaje Python a través de Skylearn para analizar la precisión, exactitud, F-Score, Curva de ROC y Área Bajo la Curva.

## **2.6. Criterios éticos.**

La ética en la investigación es un factor muy importante, por tal motivo los criterios éticos asumidos en el desarrollo del proceso investigativo son los siguientes:

### **a. Criterio de confiabilidad**

La confiabilidad es imprescindible en las investigaciones porque permite determinar que la información utilizada en investigación es confiable y puede ser utilizada por otros investigadores generando resultados equivalentes al estudio original. También se tuvo en cuenta los datos personales utilizados en la investigación los cuales fueron obtenidos legalmente haciendo uso del profesionalismo evitando de los a los involucrados en la investigación.

### **b. Criterio de Conformabilidad**

El resultado y conclusiones afirmativas resultado del proceso investigativo han sido demostrados a través de la ejecución de consulta de bibliografía de base de datos de prestigio como Scopus, Sciencie Direct, entre otros, de pruebas de los algoritmos utilizados se realizó considerando las librerías que

permitían emitir los resultados que fueron requeridos para el informe de la presente investigación.

## 2.7. Criterios de Rigor científico

- a. **Confiabilidad:** para determinar la confiabilidad se utilizará la estadística a través de los métodos y herramientas de recolección de datos que permitan evaluar las métricas de rendimiento en clasificación de ataques de inyección SQL.
- b. **Validez:** la presente investigación se realizó la validez de tipo lógico teniendo como base los modelos matemáticos y estándares.

Las fórmulas matemáticas utilizadas son:

Error cuadrático medio

$$ECM = \frac{1}{N} \sum_{(x,y) \in D} (y - p(x))^2$$

(Google developers, 2021)

Donde: (x, y) representa al conjunto de datos, x representa al conjunto de atributos de la variable, y representa a la etiqueta que se está trabajando. D representa al conjunto de datos que contiene las etiquetas de los pares (x, y), finalmente N es el número de datos contenidos en D.

Tabla 4  
*Matriz de confusión.*

		Pronóstico	
		Positivo	Negativo
Exploración	Positivo	Verdadero Positivo	Falso Negativo
	Negativo	Falso Positivo	Verdadero Negativo

### III. RESULTADOS

#### 3.1. Resultados en Tablas y Figuras

Se realizó las pruebas de verificación para determinar los ataques de inyección SQL, para esta acción se consideraron indicadores como el tiempo de respuesta, el consumo de CPU y consumo de memoria, los resultados que se obtuvieron fueron a partir de la implementación de los algoritmos de aprendizaje automático en el lenguaje Python usando librerías Sklearn y propias del lenguaje.

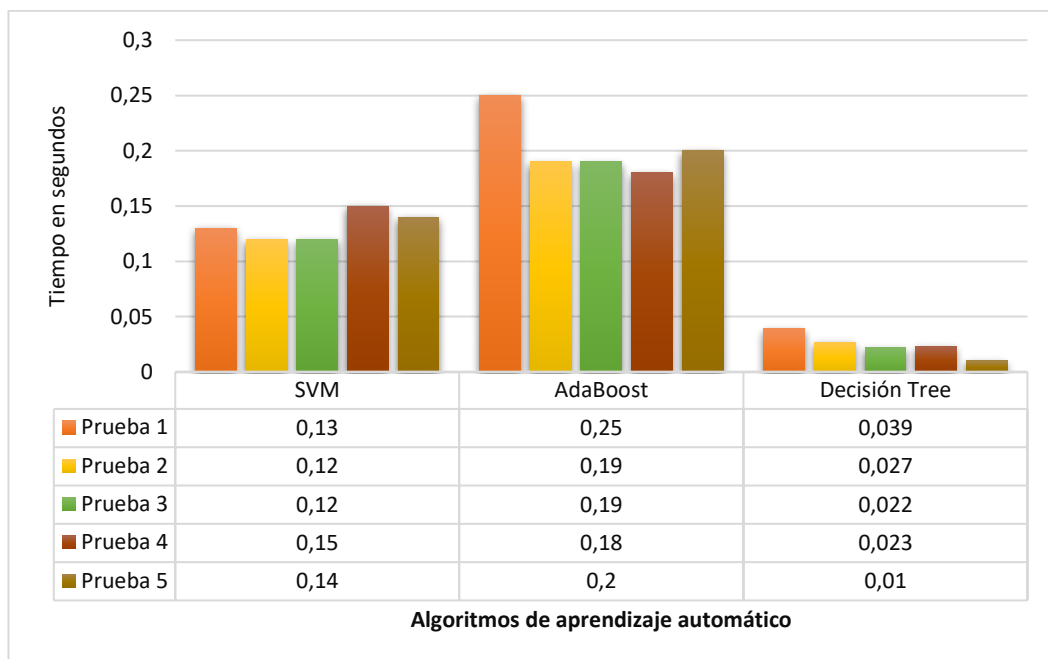


Figura 9. Tiempo de respuesta promedio según el algoritmo implementado.

Fuente: elaboración propia.

En la figura 9 se presenta los datos registrados para cada algoritmo según el tiempo utilizado en las cinco pruebas realizadas, los resultados determinaron que el algoritmo SVM utilizó el tiempo promedio de 0.132 segundos para realizar la identificación de los datos, mientras que el algoritmo AdaBoost utilizó un tiempo promedio de 0.202 segundos para realizar la identificación de los ataques de inyección SQL, el algoritmo que con menor tiempo de respuesta fue Decisión Tree con 0.0264 segundos, entonces el algoritmo que utiliza más tiempo utiliza para identificar ataques de tipo inyección SQL es el



AdaBoost, mientras que el que utiliza menor tiempo es el Decision Tree, lo cual determina su alto grado de precisión al momento de ejecutarse.

### Indicador consumo de CPU

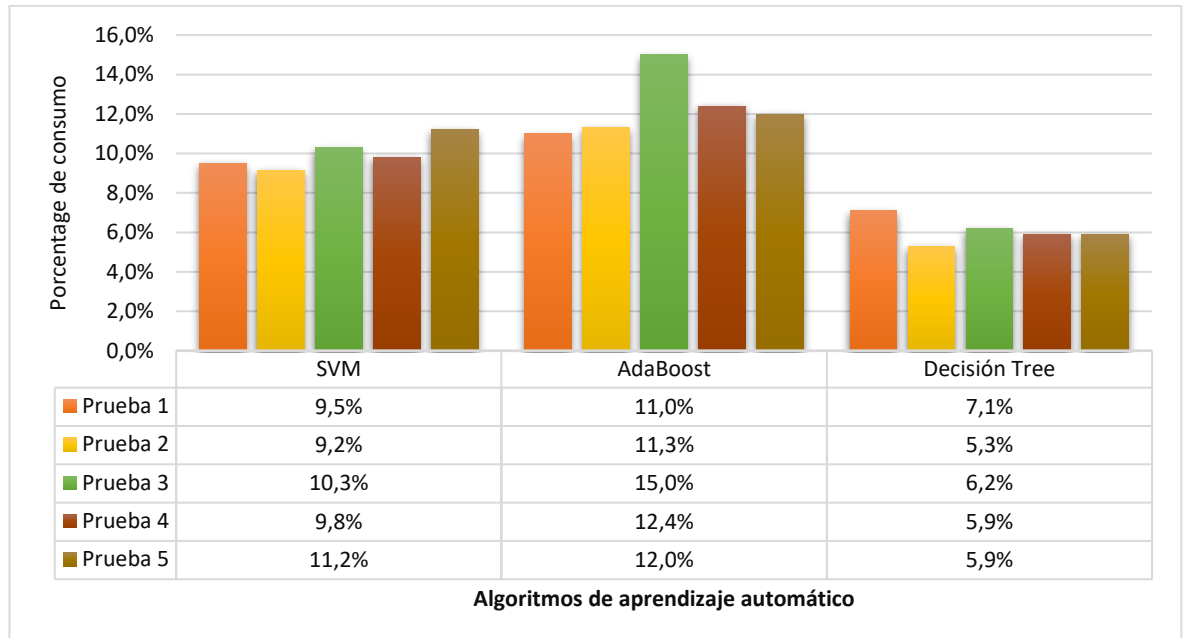
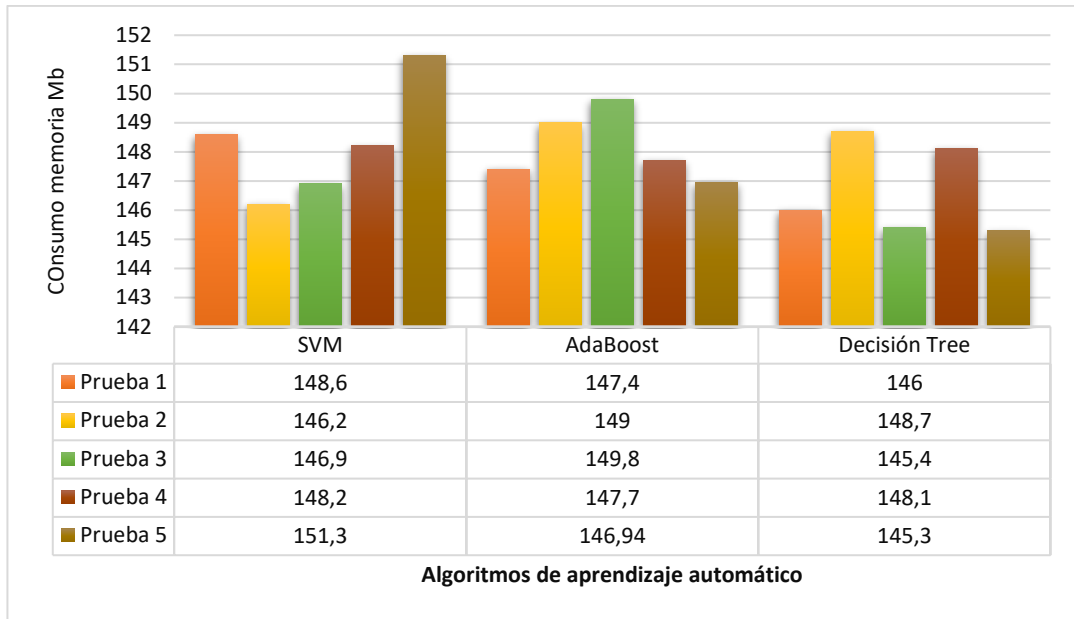


Figura 10. Consumo de CPU de algoritmos de aprendizaje Automático

Fuente: elaboración propia.

En la figura 10, los resultados estadísticos demuestran que el algoritmo SVM utiliza mayor cantidad de recursos de CPU al momento de procesar los datos, consumiendo en promedio de las cinco pruebas realizadas 10% grados de CPU, el algoritmo AdaBoost tiene un consumo de 12.34 % de CPU, el algoritmo Decision Tree ha consumido en promedio de 6.08 % de CPU, lo cual lo ubica como el algoritmo de menor consumo de CPU, permitiendo determinarlo como uno de los algoritmos con mejor eficiencia para ser entrenado en la detección de inyección SQL.



*Figura 11* Consumo de memoria expresados en Megabytes.

*Fuente:* elaboración propia.

En la figura 11, se presenta el nivel de consumo de memoria de los algoritmos de aprendizaje automático implementados para la identificación de inyección SQL, la unidad de medida se expresa en Megabytes, de este gráfico se determina cual es el algoritmo con menos consumo de recurso de memoria. El algoritmo de SVM de las cinco pruebas realizadas presenta un promedio de 148.24 Mb de consumo de memoria al momento de ejecutarse, seguido por una diferencia de 148.168 Mb de diferencia por el algoritmo AdaBoost con un consumo promedio de memoria de 146.7 Mb. Y el algoritmo que presenta menor consumo de recurso memoria es el Decisión Tree con un promedio de 146.7 Mb, puntuación que lo ubica en el mejor algoritmo para la detección de inyección SQL.

Los resultados en cuanto a precisión de los algoritmos de aprendizaje automático seleccionados para la detección de ataques de inyección SQL se consideró la tasa de verdaderos positivos y falso positivos comparando con la cantidad total de elementos positivos predichos, seguido por indicadores como exactitud, precisión, Recall, F-Score y AUC; el entrenamiento de datos para la clasificación respectiva.

Resultados de la ejecución de los algoritmos de aprendizaje automático presentados en matriz de confusión

Tabla 5

*Matriz de confusión del algoritmo SVM*

		<b>Clasificación</b>	
		Consulta normal	Consulta Anómala
<b>Reales</b>	Consulta normal	1077	54
	Consulta Anómala	60	497

*Fuente:* elaboración propia.

La población total de registros de análisis del dataset fue de 8439, de los cuales el algoritmo Support Vector Machines detectó 1137 consultas normales y 551 consultas anómalas. En cuanto a las consultas normales logró predecir 1077 consultas correctas y 60 consultas con error esto porque el modelo predijo consultas normales, siendo en realidad consultas anómalas, en el siguiente cuadrante se presenta la predicción realizada de 497 consultas de forma correcta, mientras que 54 consultas se predijeron de manera errónea debido a la predicción del modelo como consulta anómala.

Tabla 6

*Matriz de confusión del algoritmo AdaBoost*

		<b>Clasificación</b>	
		Consulta normal	Consulta Anómala
<b>Reales</b>	Consulta normal	1168	1
	Consulta Anómala	3	516

*Fuente:* elaboración propia.

La población total de registros de análisis del dataset fue de 8439, de los cuales el algoritmo AdaBoost detectó 1171 consultas normales y 517 consultas anómalas. En cuanto a las consultas normales logró predecir 1168 consultas correctas y 3 consultas con error esto porque el modelo predijo consultas normales, siendo en realidad consultas anómalas, en el siguiente cuadrante se presenta la predicción realizada de 516 consultas de forma correcta, mientras que 1 consultas se predijeron de manera errónea debido a la predicción del modelo como consulta anómala.

Tabla 7

*Matriz de confusión del algoritmo Decision tree*

		<b>Clasificación</b>	
		Consulta normal	Consulta Anómala
<b>Reales</b>	Consulta normal	1164	0
	Consulta Anómala	0	524

*Fuente: elaboración propia.*

La población total de registros de análisis del dataset fue de 8439, de los cuales el algoritmo AdaBoost detectó 1164 consultas normales y 524 consultas anómalas. En cuanto a las consultas normales logró predecir 1164 consultas correctas y 0 consultas con error esto porque el modelo predijo consultas normales, siendo en realidad consultas anómalas, en el siguiente cuadrante se presenta la predicción realizada de 524 consultas de forma correcta, mientras que 0 consultas se predijeron de manera errónea debido a la predicción del modelo como consulta anómala.

Resultados del entrenamiento al conjunto de datos según cada indicador y algoritmo clasificador:

Tabla 8

Medidas de rendimiento de los algoritmos de aprendizaje automático seleccionados.

Algoritmo Clasificador	Nro de prueba	Exactitud	Precisión	Recall	F-Score	AUC
Support Vector Machine	Prueba 1	73.18 %	90.1 %	89.2 %	89.7 %	92.2%
	Prueba 2	95.5 %	97.7 %	84.7 %	90.7 %	91.8 %
	Prueba 3	85 %	99.7 %	85 %	91.8 %	92.5 %
	Prueba 4	94.1 %	93.7 %	86.7 %	90.1 %	91.9 %
	Prueba 5	96.8 %	99.5 %	88.8 %	93.9 %	94.3 %
<b>Promedio</b>		<b>88.92%</b>	<b>96.14%</b>	<b>86.88%</b>	<b>91.24%</b>	<b>92.54%</b>
AdaBoost	Prueba 1	99.76 %	99.80 %	99.42%	99.61%	99.66%
	Prueba 2	99.7 %	99.8 %	99.4 %	99.6 %	99.4 %
	Prueba 3	99.6 %	99.7 %	99.4 %	99.6 %	99.6 %
	Prueba 4	99.7 %	99.8 %	99.4 %	99.6 %	99.4 %
	Prueba 5	99.5 %	99.6 %	99.5 %	99.7 %	99.6 %
<b>Promedio</b>		<b>99.65%</b>	<b>99.74%</b>	<b>99.42%</b>	<b>99.62%</b>	<b>99.53%</b>
Árbol de decisión	Prueba 1	100 %	100 %	100 %	100 %	100 %
	Prueba 2	100 %	100 %	100 %	100 %	100 %
	Prueba 3	100 %	100 %	100 %	100 %	100 %
	Prueba 4	100 %	100 %	100 %	100 %	100 %
	Prueba 5	100 %	100 %	100 %	100 %	100 %
<b>Promedio</b>		<b>100 %</b>	<b>100 %</b>	<b>100 %</b>	<b>100 %</b>	<b>100 %</b>

Fuente: elaboración propia.

Los datos muestran que los algoritmos AdaBoost y Decision Tree obtuvieron mejores resultados en los indicadores de rendimiento para la detección de ataques de inyección SQL, esto se debe a que el total de predicciones correctas supera en grandes cantidades a las predicciones erróneas; el umbral de clasificación es un factor determinante para lograr un alto nivel de detección, mientras que el umbral sea mayor la clasificación será más rigurosa, por lo tanto la evaluación de verificar si una consulta es normal o anómala.

El algoritmo Support Vector Machine utilizando un kernel lineal para la evaluación de los datos, considerando que la clasificación de los datos están basados en la

etiquetas que determina los valores de consultas normales y anómalas para es el que obtuvo la evaluación más baja en cuanto a precisión en las cinco pruebas realizadas, entonces con la implementación del kernel linear SVM separó las consultas normales de las consultas anormales permitiendo predecir un número significativo de consultas como erróneas, determinando de esta manera el nivel de precisión a un con un 96.14%.

El algoritmo de AdaBoost mantiene el porcentaje de AUC en 99.53 % con relación del algoritmo que el algoritmo SVM, entonces lo ubica en un algoritmo que tiene el umbral más certero de división entre consultas normales y anómalas, existen errores de predicción correctamente, pero estos niveles del umbral de clasificaciones se pueden aumentar o disminuir. Si los resultados de AUC fuese del 50% o menos la conclusión sería que el algoritmo implementado no logra diferenciar entre consultas normales y anómalas.

La precisión está definida por la cantidad de consultas marcadas como anómalas, mientras que el Recall se determina por la cantidad de consultas normales que se predijeron correctamente. Entonces la precisión se ve relacionado estrechamente con el Recall, también se tiene que considerar al F-Score que es una media armónica entre el Recall y la precisión, siempre se tiene que considerar que el F-Score no debe exceder a la precisión.

### **3.2. Discusión de resultados**

El presente trabajo se diferencia de los trabajos realizados por Hasan, Balbahaith y Tarique, quienes en su investigación proponen el recojo de información de la URL para construir su dataset, procedieron a eliminar información redundante, vectorizando las cadenas originales, creando de esta manera matrices, posteriormente se puso a prueba los algoritmos implementados en el entorno de evaluación MATLAB, logrando como resultado que el modelo SVM alcanzó una precisión de 94,49% y Decision Tree con una precisión de 98,7%. Entonces la investigación que se planteó con cuatro pasos: **el primer paso** consistió en realizar la clasificación de los ataques de inyección SQL considerando la tipología de ataques, describiendo las principales características y el nivel de riesgo que se

considere el tipo de ataque, una vez clasificado los tipos de ataques se procedió con **el segundo paso**, que consistió en recuperar la información para la construcción del dataset, se experimentó con una herramienta implementada por Owasp, denominada Owasp ZAP que permitió hacer la carga del Payload seleccionado (base datos de ataques), se procedió a ejecutar la aplicación para atacar un sitio web implementado, los resultados de los ataques realizados se recogieron en una hoja de cálculo, luego se procedió con la clasificación según las características presentes en la cadena URL (operadores lógicos, comandos SQL), estas características fueron codificadas en un archivo de formato CSV y etiquetadas como consulta normal y consulta anómala. **El tercer paso** del método propuesto consistió en el análisis de fuentes bibliográficas con estudios relacionados a los ataques de inyección SQL, luego se procedió con la selección de los algoritmos de aprendizaje automático considerando el grado de precisión como resultado de los estudios realizados, de la evaluación se obtuvo a los algoritmos con mayor precisión a SVM, AdaBoost y Decision Tree. **El cuarto paso del método** se procedió a la implementación de los algoritmos de aprendizaje automático, para este paso se procedió a preparar el entorno de desarrollo tomando como Base la plataforma Anaconda para la implementación de los algoritmos en el lenguaje python usando librerías de scikit-learn, NumPy, SciPy y matplotlib, los algoritmos fueron implementados con los parámetros que permitieron evaluar los ataques de inyección SQL. Los resultados obtenidos con el algoritmo SVM fue una precisión del 96.14%, seguido por el algoritmo AdaBoost con una precisión de 99.74% y el algoritmo con mejor precisión fue Decision Tree que alcanzó el 100% de precisión.

Otro estudio comparable es el realizado por los autores Xie, Ren, Fu, Xu, y Guo, en su investigación "SQL Injection Detection for Elastic-Pooling CNN-based Web Applications", centraron su investigación en la detección de la inyección SQL, también para la implementación del modelo procedieron inicialmente a recopilar los datos para la construcción de un dataset a partir del cual se puso a prueba los algoritmos de aprendizaje automático logrando resultados para el algoritmo decision tree del 98.64% de precisión en cuanto a detección de inyección SQL, el algoritmo SVM alcanzó el 97.84% de precisión. Entonces se coincide con esta

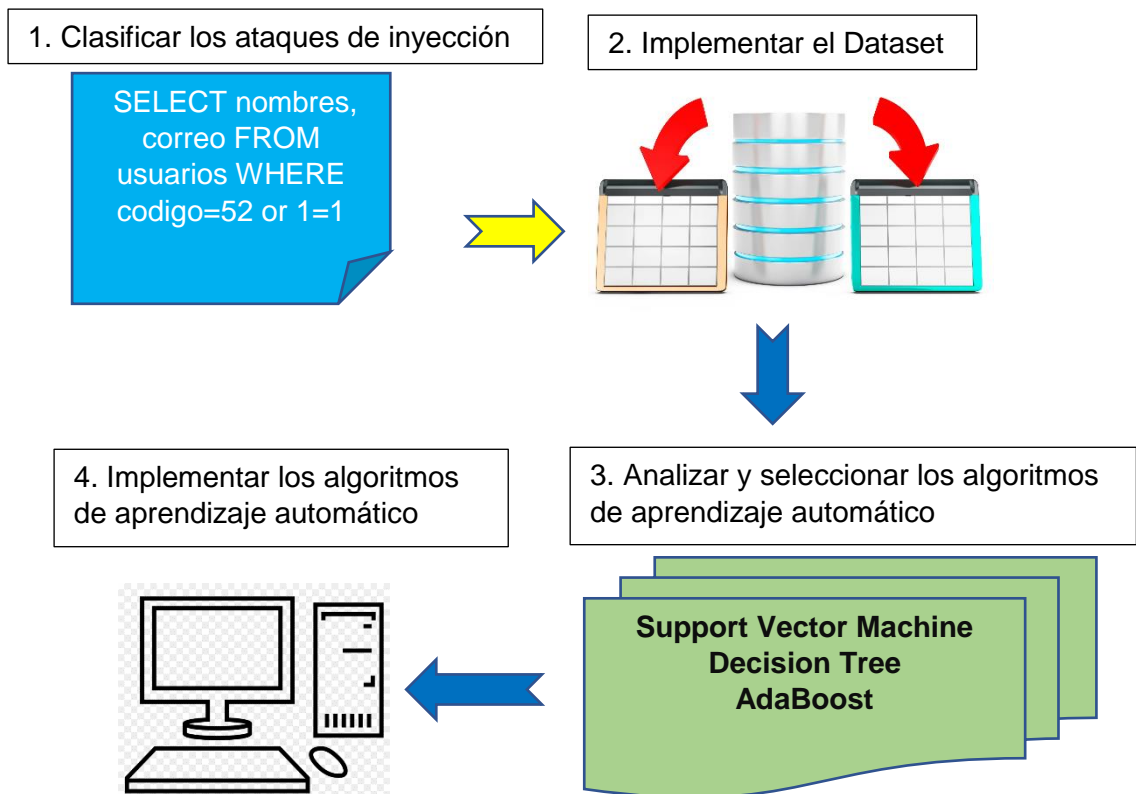
investigación en cuanto al algoritmo decision tree con mayor precisión en la detección de ataques de inyección SQL

Otro de los estudios es el realizado por (Bhagwani, y otros), investigación que buscaron detectar ataques de inyección SQL en Cloud mediante el aprendizaje automático en escenarios utilizando cargas útiles obtuvieron resultados para el algoritmo de aprendizaje automático AdaBoost de 99.7% de precisión, para Decision Tree obtuvieron una precisión del 99.4, en el método propuesto primero crearon las características de las consultas SQL, estableciendo las reglas, en segundo momento entrenaron los modelos de aprendizaje automático y en tercer momento probaron y compararon su resultado en cuanto a precisión.

Otro de los estudios comparables es el de los autores (Cui , He, Yao, & Shi) investigación que buscaron la detección de URL maliciosas con extracción de funciones basadas en aprendizaje automático, los autores realizaron un clasificación de los tipos de inyección SQL asignando valores numéricos para cada uno de los parámetros considerados, al aplicar los algoritmos obtuvieron una precisión para el algoritmo Decisión tree del 99.8% y el algoritmo SVM del 99.3% en el análisis de datos que contenían consultas de inyección SQL enviadas a través de la URL, se puede concluir que los algoritmos obtuvieron un alto nivel de precisión en ambos algoritmos. También se tiene el estudio de (Imtiazul, Hasan, Golam, Imtiaz, & Sarwat, 2020) que realizaron un estudio para generar, detectar y mitigar ciberataques en las redes inteligentes, evaluaron el rendimiento en cuanto a precisión de un grupo significativo de algoritmos de aprendizaje automático, uno de los algoritmos seleccionados para el estudio fue AdaBoost, el cual presentó un nivel de precisión de 85.9%, lo cual determina que el algoritmo de aprendizaje automático presenta un nivel por debajo de los resultados obtenidos en el presente estudio.



### 3.3. Aporte práctico.



Para la clasificación de los ataques de inyección SQL se realizó la consulta en las bases de datos especializadas IEEE Xplore, Scopus, Web of Science, en la búsqueda se utilizó el filtro del año de publicación desde el año 2017 hasta el año 2021, también se utilizó las palabras claves de búsqueda como: SQL injection, Classification SQL injection, entre las principales.

Así mismo, se realizó la clasificación de los ataques de inyección SQL, tomando como referencia a Khanna y Verma (2018), quienes realizaron una clasificación considerando por tipos de ataques, los ataques de inyección que hacen uso de operadores lógicos en una consulta SQL de tipo tautología lo ubicaron en el nivel de riesgo medio, el resultado es peligroso para la información que se almacena en una BD, el siguiente tipo son las consultas lógicamente incorrectas, ubicándolo en un nivel de riesgo bajo; seguido por las consultas de unión que son un tipo de ataques de inyección SQL que buscan obtener resultados

a partir de la unión de las consultas y forzando el uso de comentarios para desactivar la codificación programada.

También, se tienen las consultas de unión con un nivel de riesgo medio; seguido por las consultas acumuladas con nivel de riesgo alto, este tipo de ataque de inyección SQL es devastador para la información contenida en una base de datos, al ejecutarse da lugar a la eliminación de tablas de base de datos; para ello se hace uso de comandos como Drop table. Otro de los ataques de inyección SQL de nivel de riesgo alto son los procedimientos almacenados que alteran la ejecución obteniendo resultados favorables para el atacante. La codificación alternativa es otro tipo de ataques utilizados para explotar inyecciones SQL que son considerados de nivel de riesgo alto, finalmente la clasificación de ataques de inyección SQL a través de inferencias que contiene a la inyección por blindaje y ataques por tiempos.

Tabla 9

*Clasificación de los ataques de Inyección SQL*

Tipo de ataque	Descripción	Consulta Normal	Consultas inyectadas	Nivel de riesgo
Tautological Attacks	<p>Ataque de inyección SQL que busca cambiar la condición en la cláusula WHERE de la consulta SQL para que el resultado se evalúe como verdadera independientemente de la condición original asignada en el lenguaje de programación. Un ataque tautológico de forma "OR 1 = 1" determina que la cláusula WHERE a evaluarse como verdadera. Los ataques tautológicos son los más utilizados para eludir la autenticación y extraer datos.</p>	<pre>SELECT * FROM usersist WHERE nomb_user = 'nomb_user' AND clave = 'pass'</pre> <p>En esta consulta se espera un usuario y clave de acceso.</p> <p>Uuario: kikeCa Clave: Enriq42351</p>	<p>Usuario : ' OR 1 = 1 -- SELECT * FROM usersist WHERE nomb_user = " OR 1 = 1 -- AND clave = 'pass'</p>	Medio
Piggybacked queries	<p>En este tipo de ataque se busca inyectar consultas SQL adicionales</p>		<p>Al incorporar la siguiente cadena.</p>	Alto

incrustando en la consulta original. El atacante no busca el cambio de la consulta original, lo que busca es incorporar otra consulta paralela para ser ejecutadas por el SGBD como consulta por lotes.

```

SELECT * FROM Usuario : kikeCa'; DROP TABLE usuarios
usersist WHERE --
nomb_user = 'juan'
AND password = 'Jsas12'

```

La consulta quedaría de la siguiente forma:

```

SELECT * FROM usersist WHERE
nomb_user = ' kikeCa'; DROP TABLE
usuarios --' AND password = 'pass'

```

En la consulta se incorporó la consulta para eliminar la tabla usuarios.

Illegal or Logically Incorrect Queries Este método se usa palabras clave o parámetros no válidos, para que la consulta SQL resultante sea ilegal o incorrecta, dando como resultado un error.

```

SELECT * FROM
usersist WHERE
nomb_user = 'juan'
AND password =
'Jsas12'

```

Con este tipo de ataque se busca obtener información de la base de datos, como el nombre, usuarios,

Bajo Con la incorporación de los siguientes datos en el campo usuario "juan' ABCD" and "xyz" en el campo contraseña, el mensaje de error que se muestra será:

Bajo

nombres de las tablas, columnas, etc. Este tipo de inyección SQL es un paso preliminar para obtener datos sobre la estructura de la base de datos para ser utilizado en otros ataques de inyección.

```
SELECT * FROM usersist WHERE
nomb_user = "juan' ABC" and password =
"xyz"
```

Código de error: 1064

Tiene un error en su sintaxis SQL; compruebe el manual correspondiente a su versión del servidor MySQL para conocer la sintaxis correcta para usar cerca de 'ABCD AND passwd = 'abcd' en la línea 1

UNION  
based  
Attacks

La palabra clave UNION se utiliza al unir resultados de varias sentencias SELECT en un único conjunto de resultados. En los ataques de inyección SQL basados en UNION, el atacante inyecta una consulta adicional utilizando la palabra clave UNION

```
SELECT id, nombre,
apellidos, correo
FROM usuarios
WHERE id = 'juan'
UNION SELECT 1
FROM dual -- AND
passwd = '432423' "
```

Código de error: 1222

Las sentencias SELECT utilizadas tienen un número diferente de columnas

Medio

para introducir datos de otras tablas o columnas en el conjunto de resultados para que puedan mostrarse en la página web.

```
nombre' UNION SELECT 1,
table_schema, table_nombre, 4
FROM information_schema.tables
WHERE table_schema
NOT IN ('information_schema', 'mysql',
'test') LIMIT 1,1 --
```

Blind  
Injection  
Attacks

Cuando se suprimen los mensajes de error en base de datos, el atacante no puede reunir la información necesaria para elaborar los posteriores ataques de inyección. Frente a esta situación, el atacante inyecta comandos SQL y observa si hay un cambio en la respuesta de la página web. Relacionando cuidadosamente las respuestas con los vectores de inyección, es decir, cuando el comportamiento permanece igual y

[http://www.iesppjsch/libro\\_detalle.php?pid=24](http://www.iesppjsch/libro_detalle.php?pid=24) Bajo

Esto causaría un error de sintaxis en la consulta SQL generada por el código PHP. Como los mensajes de error no se suprimen, MySQL mostraría el siguiente mensaje de error:

código de error: 1064  
Tiene un error en su sintaxis SQL; consulte el manual correspondiente a su versión del servidor MySQL para conocer

cuando cambia, el atacante puede hacer inferencias sobre los parámetros vulnerables e información adicional sobre la estructura de la base de datos.

#### Boolean-based Blind Attacks

En la inyección ciega basada en booleanos, el atacante diseña los vectores de ataque para hacer preguntas de verdadero/falso al servidor. Cuando la respuesta es verdadera, la página web se muestra normalmente, pero cuando la respuesta es falsa, la visualización de la página cambia significativamente.

#### Time-based Blind Attacks

[http://www.iesppjsch/matr\\_detalle.php?id=24](http://www.iesppjsch/matr_detalle.php?id=24)

la sintaxis correcta a utilizar cerca de la línea "24"

El atacante intenta primero acceder a la siguiente URL:

[http://www.iesppjsch/matr\\_detalle.php?id=24+AND+1+=+1](http://www.iesppjsch/matr_detalle.php?id=24+AND+1+=+1)

Cada signo '+' en la cadena de consulta es la forma codificada en la url de un carácter de espacio. Por lo tanto, el parámetro id lleva ahora el valor de la cadena "24 AND 1 = 1".

[http://www.iesppjsch/matr\\_detalle.php?id=24-SLEEP\(10\)](http://www.iesppjsch/matr_detalle.php?id=24-SLEEP(10))

Bajo

Bajo

[http://www.iesppjsch/matr\\_detalle.php?id=24](http://www.iesppjsch/matr_detalle.php?id=24)

Este tipo de ataques se recurre a otra forma de inferencia elaborando el ataque con construcciones if/then con retrasos de tiempo dentro de ellas. Lo que se busca es medir el tiempo que tarda en llegar una respuesta, a partir de la cual se pueden sacar conclusiones. Para ello se usan palabras reservadas como SLEEP, WAITFOR DELAY 'hh:mm:ss' y WAITFOR TIME 'hh:mm:ss'.

En este caso el parámetro id llevará el valor de la cadena "24 - SLEEP(10)" que se utilizará para construir la consulta.

```
SELECT * FROM matrícula WHERE
código_producto = 24 - SLEEP(10)
```

Tras la ejecución, la llamada a la función SLEEP() hará que la consulta se detenga durante 10 segundos. Cuando la función SLEEP(10) se complete con éxito, devolverá cero, por lo que el id del producto seguirá siendo el mismo que el 24.

Bajo

### Heavy Query Attacks

En este tipo de ataques el atacante busca simular retrasos de tiempo utilizando consultas pesadas

[http://www.iesppjsch/matr\\_detalle.php?id=24](http://www.iesppjsch/matr_detalle.php?id=24)

```
SELECT count(*) FROM
information_schema.columns x,
```



dentro del ataque. Una consulta pesada es una consulta que es costosa y tomará un tiempo notable para ejecutarse en el motor de la base de datos. Al incorporar múltiples uniones entre dichas tablas del sistema, el vector de ataque puede hacerse aún más pesado. Para ello, el atacante elige tablas del sistema que se sabe que contienen un gran número de filas.

Stored Procedure Attacks Un procedimiento almacenado es un conjunto de consultas SQL con un nombre asignado que se

information\_schema.columns y,  
information\_schema.columns z

Esta consulta tarda '23 segundos en completarse, lo que equivale a una llamada a la función SLEEP(23). Puede observarse que la consulta une implícitamente la tabla information\_schema.columns consigo misma tres veces sin cláusula WHERE, lo que la hace bastante pesada. Sin embargo, el tiempo de ejecución de una consulta pesada puede variar en función del número de filas que contenga la tabla elegida, que a su vez está influenciado por factores como el tamaño de la base de datos, el rendimiento del servidor, etc.

```
CREATE PROCEDURE acceso
(nombUser VARCHAR(255), clave
VARCHAR(255)) BEGIN SET @qry =
```

Alto

almacena en forma compilada en el servidor de la base de datos. Los procedimientos almacenados separan la lógica de negocio compleja del código de la aplicación, generalmente toman parámetros, realizan consultas SQL de acuerdo con la lógica de negocio y devuelven los resultados a la aplicación web.

#### Alternate Encoding Attacks

Para ejecutar este ataque se modifica el vector de ataque utilizando un método de codificación diferente para evadir la detección de los sistemas de detección de intrusos o eludir las

[http://www.iesppjsch/matr\\_detalle.php?id=24](http://www.iesppjsch/matr_detalle.php?id=24)

```
CONCAT("SELECT id, nomb, apell,  
correo FROM usuarios", " WHERE id = ",  
nomb, " AND clave = ", clave, "");
```

```
PREPARE stmt FROM @qry; EXECUTE  
stmt; END
```

El código del procedimiento almacenado construye la consulta SQL dinámicamente concatenando los valores pasados por los parámetros. Por lo tanto, es vulnerable a los ataques de inyección SQL de la misma manera.

CHAR(79,82,32,49,32,61,32,49) en MySQL es la forma codificada ASCII de "OR 1 = 1". Los códigos ASCII también pueden darse en forma hexadecimal. A veces también se utilizan otras funciones

funciones de desinfección. Las funciones de sanitización buscan la presencia de caracteres especiales como las comillas simples o caracteres de comentario. La codificación de estos caracteres de forma diferente en el vector de inyección vector permite al atacante explotar la vulnerabilidad que puede no ser posible por métodos directos, se hace uso en sus vectores de inyección, como la codificación ASCII, hexadecimal, binaria y unicode.

Second-order Injection Attacks

La inyección de código SQL malicioso que se almacena en la base de datos como normales para luego ser ejecutados como consulta dinámica. Este tipo de ataques que llegan a la base de datos a través de consultas

[http://www.iesppjsch/matr\\_detalle.php?id=24](http://www.iesppjsch/matr_detalle.php?id=24)

como "SHUTDOWN" puede codificarse utilizando los códigos hexadecimales de cada carácter como CONCAT(CHAR(0x53485554), CHAR(0x444F574E)).

```
UPDATE usuario SET clave = 'Levi1210'
WHERE nombre_usuario = 'levir' OR
nombre_usuario LIKE '%%' -- ' AND clave
= 'XXkkll'}"
```

INSERT, con esto se envía entradas que contienen código SQL a través de campos de formulario. Estas entradas se almacenan silenciosamente en la base de datos como cualquier otro dato.

---

Fuente: Khanna y Verma (2018), Sheykhkanloo (2017).

La recopilación de los datos para el procesamiento a través de los algoritmos de aprendizaje automático, primero se desarrolló una aplicación de tipo repositorio de documentos utilizando para ello tecnología PHP, AJAX, JavaScript y HTML5; como SGBD se utilizó a MySQL, se creó la base de datos con la codificación UTF-8 y las tablas con sus respectivos campos, el sistema funciona bajo la plataforma de XAMPP. El sitio web utilizado para la simulación de ataques tiene un cuadro de búsqueda en la página principal, también se incluyó el acceso a través de un formulario login y un panel de administración para que se registren cada entidad de la base de datos. Los formularios implementados permitieron la realización de las pruebas de inyección SQL.

Para la realización de los ataques se utilizó la herramienta OWASP ZAP 2.9.0 implementada por la Fundación OWASP que nos brinda el acceso a su proyecto de código abierto destinada a elevar el nivel de seguridad de las aplicaciones web, se procedió a la descarga desde el sitio web oficial OWASP, luego se realizó la instalación respectiva, una vez instalado se procedió a programar los ataques automatizados de acuerdo a los siguientes parámetros de política de escaneo, para ello se consideró solamente los parámetros Inyección SQL avanzada con un umbral “Alto” y una fuerza de ataque modo “Loco”, procediendo a continuación a configurar los parámetros SQL Injection con un umbral “Alto” y una fuerza de ataque modo “Loco”, se ingresó la URL del sitio alojado en el servidor local administrado por XAMPP, y se procedió a ejecutar el análisis.

Con la configuración realizada para la ejecución de los ataques automatizados se logró obtener la mayor cantidad de registros para posteriormente ser evaluados aplicando algoritmos de aprendizaje automático. El registro de ataques fue exportado a un archivo CSV para su análisis respectivo.

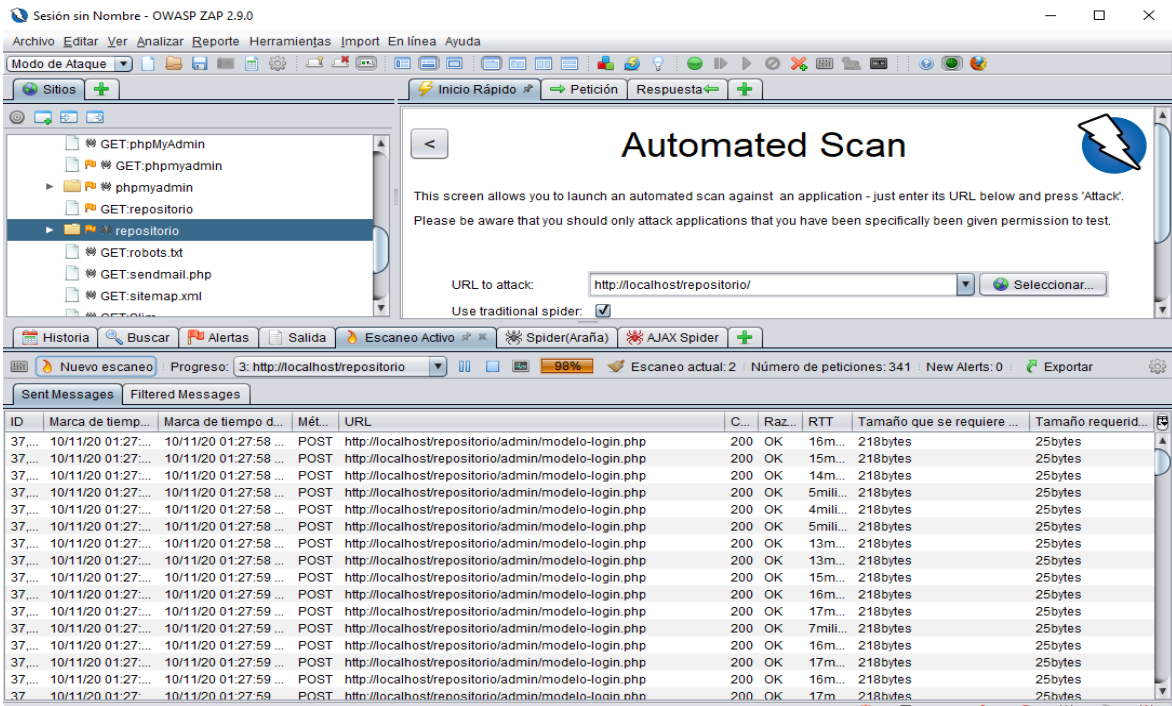


Figura 12. Ventana principal de OWASP ZAP 2.9.0

## Preparación del dataset para el análisis con los algoritmos de aprendizaje automático seleccionados

Los registros que se recogieron de los ataques automatizados con OWASP ZAP, los ataques de inyección SQL se realiza alterando los datos al momento enviar datos desde los formularios hacia la base de datos, los ataques más comunes son los que involucran verificación de acceso de usuario y password el cual se realiza en un formulario.

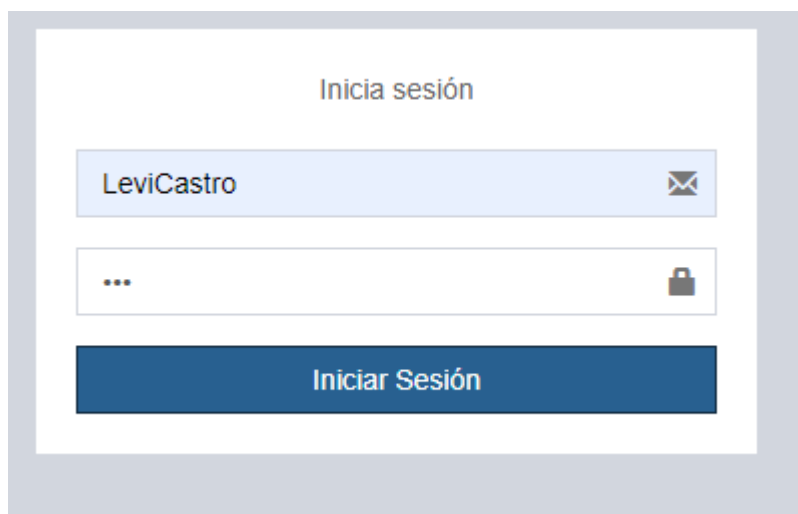
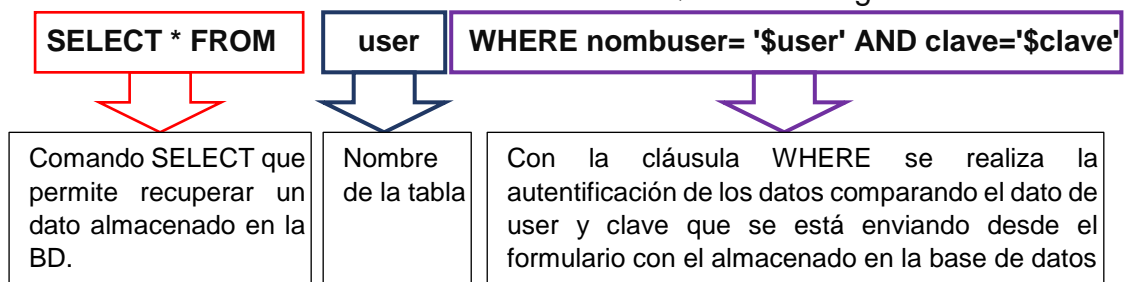


Figura 13. Formulario de acceso.

A través de los formularios se envían datos que luego son recepcionados y gestionados desde una Base de Datos con comandos SQL, en este caso presentado en la figura 11, el formulario gestiona la identificación de usuarios para lo cual se envían dos parámetros a la base de datos que corresponde al usuario y contraseña, para esto se utiliza el comando `SELECT * FROM`, luego se hace referencia a la tabla `usuarios`, seguido se usa la cláusula `WHERE` y se compara los datos contenidos en la tabla y lo que se está enviando desde el formulario. Y donde esperamos se cumpla la sentencia con los siguientes datos:

**Usuario : LeviCastro**  
**Password : FeCastroLe**

La construcción de la consulta normal en SQL es de la siguiente estructura:



Y donde esperamos se cumpla una sentencia considerando que los datos que se presentan a continuación corresponden a los que están almacenados en la tabla `usuarios`, por lo que como resultado nos da un acceso correcto al sistema. Ahora manipulando la consulta anterior, incorporando una inyección de código SQL, la consulta quedaría construida de la siguiente forma:

**SELECT \* FROM user WHERE nombuser = '\$user ' AND clave = " OR " = "**

Código de inyección SQL incorporada en la consulta, donde se está comparando el clave vacío con vacío y como resultado se obtiene una verdad, por lo tanto, se ingresaría al sistema autenticado con el usuario LeviCastro

Para la implementación del Dataset se consideró la clasificación de los tipos de inyección SQL, luego se procedió a la recopilación de datos de inyección usando herramientas como Owasp Zap, los datos resultantes de la aplicación de ataques contienen información con características como Método de Petición (Post y Get), Código de suceso, URL, RTT (marca de tiempo), Tiempo requerido para el encabezamiento, tiempo requerido para el cuerpo.

Método	URL	Código	RTT	Tamaño que se requiere para el encabezamiento	Tamaño requerido para el cuerpo
POST	http://localhost/repositorio/admin/modelo-login.php	200	5	199	25
GET	http://localhost/repositorio/admin/login.php	200	9	338	3691
GET	http://localhost/repositorio/admin/js/sweetalert2.all.min.js	200	3	270	63907
GET	http://localhost/phpmyadmin/view_create.php?db=sistematicasme	200	256	1115	58159
GET	http://localhost/phpmyadmin/view_create.php?db=sistematicasme	200	266	1115	58065
GET	http://localhost/phpmyadmin/view_create.php?db=bdusuario&prin	200	305	1115	52005
GET	http://localhost/phpmyadmin/view_create.php?db=bdusuario&prin	200	338	1115	53430
GET	http://localhost/phpmyadmin/view_create.php?db=bdusuario&prin	200	347	1115	53670
GET	http://localhost/phpmyadmin/view_create.php?db=bdusuario&prin	200	240	1115	54730
GET	http://localhost/phpmyadmin/view_create.php?db=ajax_regiones&	200	209	1115	55248
GET	http://localhost/phpmyadmin/view_create.php?db=ajax_regiones&	200	390	1115	55670
GET	http://localhost/phpmyadmin/view_create.php?db=ajax_regiones&	200	307	1115	56280
GET	http://localhost/phpmyadmin/view_create.php?db=ajax_regiones&	200	291	1115	57042
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fwww.p	200	2109	1136	201
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fwww.p	200	127	1136	202
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fwww.p	200	78	1136	205
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fwww.p	200	124	1136	194
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fphp.net	200	2156	1136	214
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fphp.net	200	249	1136	208
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fphp.net	200	127	1136	210
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fphp.net	200	134	1136	206
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fdev.my	200	120	1136	246
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fdev.my	200	112	1136	249
GET	http://localhost/phpmyadmin/url.php?url=https%3A%2F%2Fdev.my	200	86	1136	247

Figura 14 Conjunto de datos sin procesar recogidos con Owasp Zap.



Después de recopilar los datos, se procedió a determinar la codificación de las características considerando los aportes de (Dueñas, 2020) para finalmente transformarlo en un formato CSV para ser analizado con los algoritmos de aprendizaje automático utilizando scikit-learn. Se parte de la siguiente codificación.

Tabla 10

*Caracterización del Dataset*

<b>Columna1</b>	<b>Method</b>	<b>Método: 1=Post, 2=Get</b>
Columna2	Long_URL	Longitud de la URL: representa el tamaño de la consulta realizada.
Columna3	Num_Caract_Esp	Cantidad de caracteres especiales en la consulta: (=, \$, %, +, --)
Columna4	Commands_SQL	Número de comandos SQL en la cadena de la URL como Select, Query, Sleep, Waifor, Concat, union etc.
Columna5	Succes_cod	Error interno del servidor = 500, Página de error = 404, Página encontrada Ok=200
Columna6	RTT	Tiempo de respuesta en milisegundos de las solicitudes de consultas.
Columna7	Size_Encabezado	Tamaño que se requiere para el encabezamiento expresado en bytes
Columna8	Size_Body	Tamaño requerido para el cuerpo expresado en bytes
Columna9	Label	Campo que determina si la petición es: 0= Consulta Normal, 1= Consulta anómala

Fuente: elaboración propia

Tabla 11

Descripción de la extracción de datos para el Dataset.

Method	URL	Succes_Co d	RT T	Size_ Encabezad o	Size_ body	Label
POST	https://localhost/repositorio/admin/modelo- login.php?query=query%29%29%29+UNION+ALL+SELECT+NULL%2CNULL%2CNULL%2CN ULL%2CNULL--+	200	5	218	25	
1	127	15	4			
↑ Method	↑ Long_URL	↑ Num_Caract._esp.	↑ Commands_SQL			

Fuente: elaboración propia.

Después de la caracterización de las columnas del Dataset, se determina que la primera columna corresponde al método que puede tomar un valor numérico de 1 o 2, la longitud de la URL se determinó realizando el conteo de toda la cadena de caracteres contenido, el número de caracteres especiales se determinó realizando una búsqueda en toda la cadena de los caracteres que se utiliza en las inyecciones SQL (+, --, %, =, ==), los datos para la columna commmads\_SQL se realizó una búsqueda de los comandos inyectados en la URL para generar una inyección SQL (UNION, SELECT, NULL, ALL, SLEEP, WAITFOR, etc), el código de suceso representa (500=error interno del servidor, Página de error = 404, Ok=200). El RTT es el tiempo en milisegundos que se necesita para la consulta, el tamaño requerido para el encabezamiento y el tamaño requerido para el cuerpo son datos expresados en bytes que OWASP ZAP proporciona en su informe. Para la columna Label se determina en función del número de caracteres especiales y los comandos SQL encontrados en la consulta. Si existen caracteres especiales y comandos SQL en la consulta, en la etiqueta se considera el valor de 1 que representa una URL maliciosa, si en la cadena no existe caracteres especiales y comandos SQL la etiqueta recibe el valor de 0, lo cual determina que es una URL benigna. Después de la decodificación de los datos se procedió a construir el dataset en formato CSV.

Method	Long_URL	Num_Caract_Esp	Commands_SQL	Succes_cod	RTT	Size_Encabezado	Size_Body	Label
2	189	23	5	200	5	275	41583	1
2	144	19	4	200	6	277	106344	1
1	202	31	6	200	5	220	4934	1
2	143	18	4	200	5	277	121200	1
2	85	8	1	200	2	271	200	1
2	190	30	6	200	5	220	4861	1
2	97	11	3	200	27	193	7289	1
2	93	9	3	200	18	271	200	1
2	100	10	3	200	3	271	200	1
2	185	28	6	200	15	220	4861	1
2	137	16	4	200	5	277	121200	1
2	137	16	4	200	6	277	121200	1
2	185	31	6	200	21	220	4861	1
2	190	33	6	200	5	220	4861	1
2	177	29	6	200	4	220	4861	1
1	191	27	6	200	31	220	4934	1
2	125	13	3	200	6	277	106344	1
2	176	25	5	200	5	193	4288	1

Figura 15. Muestra del dataset construido en formato CSV

Para la selección de los algoritmos de aprendizaje automático, se realizó la consulta en las bases de datos especializadas IEEE Xplore, Scopus, Web of Science; en la búsqueda se utilizó el filtro del año de publicación desde el año 2016 hasta el año 2021, también se utilizó las palabras claves de búsqueda como: SQL injection attack, Machine Learning, machine learning and SQL injection attack. Los artículos consultados demostraron resultados a partir de la experimentación con la temática consultada o aplicadas a casos con similares al estudio que se realizó.

Así mismo, se realizó la clasificación de los resultados proponiendo las escalas según el nivel de precisión de los algoritmos de aprendizaje automático y los que demostraron una precisión menor o igual al 90% fueron considerados de “Baja precisión”, los algoritmos que obtuvieron de 91% hasta 96% de precisión fueron considerados de “Precisión aceptable” y los que obtuvieron una precisión mayor de 96% fueron considerados de “Alta precisión”. Lo cual se puede apreciar en la siguiente tabla con las valoraciones respectivas de acuerdo al grado de precisión.

Tabla 12

*Calificación de los niveles de precisión de los algoritmos de aprendizaje automático en la detección de ataques de inyección SQL*

<b>Precisión</b>	<b>Calificación</b>	<b>Código</b>
<= 90 %	Baja precisión	BP
> 90 % y <= 96 %	Precisión aceptable	PA
> 96%	Alta precisión	AP

Fuente: elaboración propia considerando los resultados de la revisión bibliográfica.

Posteriormente a la calificación de los niveles de precisión de los algoritmos de aprendizaje automático en la detección de ataques de inyección SQL, se procede a presentar la tabla de evaluación N° 12 en la cual se presenta la precisión, los escenarios donde fueron aplicados, los investigadores que plantearon la investigación y por último la calificación que se asignó según el nivel de precisión de los algoritmos de aprendizaje automático.

Tabla 13

*Evaluación de los algoritmos de aprendizaje automático en la detección de ataques de inyección SQL.*

<b>Nro</b>	<b>Algoritmo de aprendizaje automático</b>	<b>Precisión</b>	<b>Escenario de aplicación</b>	<b>Investigador (es)</b>	<b>Calificación</b>
1	Logistic Regression	70 %	Clasificación y visualización de ataques web mediante encabezados HTTP y técnicas de aprendizaje automático	(Enciso & Camargo, 2020)	BP
2	Support Vector Machine	98.5 %	Predicción de vulnerabilidades web en aplicaciones web basadas en aprendizaje automático	(Cui , He, Yao, & Shi, 2018)	AP
3	Gradient Boosting Classifier	95.16 %	Clasificación automatizada de ataques de aplicaciones web para la detección de intrusiones	(Bhagwani, y otros, 2019)	PA
4	Decision Tree	99.40 %	Detección de URL maliciosas con extracción de funciones basada en aprendizaje automático	(Cui , He, Yao, & Shi, 2018)	AP
5	K-Nearest Neighbor	98.22 %	Detección de ataques de inyección SQL basada en un algoritmo TFIDF mejorado	(Li & Zhang, 2019)	AP
6	Random Forest	97 %	Clasificación automatizada de ataques de aplicaciones web para la detección de intrusiones	(Bhagwani, y otros, 2019)	AP
7	Naïve Bayes	94.20 %	Predicción de vulnerabilidades web en aplicaciones web basadas en aprendizaje automático	(Cui , He, Yao, & Shi, 2018)	BP
8	AdaBoost	99.7 %	Detectar ataques de inyección de SQL en Cloud SaaS mediante el aprendizaje automático	(Dharitri, Gohil, & Halabi, 2020)	AP
9	Perceptron	94 %	Codificación numérica para ataques de inyección SQL.	(Ogbomon, Buchanan, & Fan, 2017)	BP
10	Polynomial Support Vector Machine	93.58 %	Detección y clasificación de amenazas y ataques persistentes avanzados utilizando la máquina de vectores de soporte.	(Chu, Lin, & Chang, 2019)	BD

11	Redes Neuronales	95.30%	Inyección SQL	(Zhang, 2019)	BP
12	Dicotomizador Iterativo (ID3)	91.66%	Comparativa y análisis de algoritmos de aprendizaje automático para la predicción de ataques SQL	(Burón, 2015)	BP
13	Xgboost	89.51%	Aprendizaje supervisado para la detección de amenazas web mediante clasificación	(Islam, Islam, Ahmed, Iqbal, & Shahriyar, 2019)	BP
14	Markov	73.87%	Aplicación de Modelos de Markov y Máquinas SVM al Reconocimiento de intrusiones web	(Troyano, Díaz, & Enriquez, 2002)	BP
15	Regresión Ordinaria Por Mínimos Cuadrados	90.00%	Regresión lineal en analítica de ataques en aplicaciones web	(Olivieri, 2016)	BP

Fuente: Elaboración propia.

En la tabla N° 14 se presenta los tres algoritmos seleccionados con mejor precisión para la detección de inyecciones SQL.

Tabla 14

*Algoritmos seleccionados según el grado de precisión*

Nro	Algoritmo	Precisión
01	Support Vector Machine	98.5 %
02	Decisión Tree	99.40 %
03	AdaBoost	99.7 %

Fuente: elaboración propia.

El escenario utilizado para la implementación de los algoritmos de aprendizaje automático fue en el entorno de Anaconda, utilizando las herramientas de Jupyter Notebook, importando librerías para el análisis de datos con python, librería de pandas, numpy, scikit-learn, tensorflow y librerías de los algoritmos seleccionados Decision Tree, Support Vector Machines y AdaBoost.

El primer análisis que se realizó fue con el algoritmo de aprendizaje automático Support Vector Machine (SVM), es un algoritmo supervisado, utilizado para realizar clasificaciones o regresiones. El algoritmo SVM permite dar solución a problemas binarios y problemas con más clases, la finalidad es buscar el hiperplano que separe las clases que se evalúan con un margen mayor.

El algoritmo SVM un clasificador discriminativo que es utilizado para el desarrollo de tareas de clasificación y regresión, SVM es un clasificador no probabilístico que se utiliza para problemas de clasificación y en aplicaciones de reconocimiento de patrones, es una versión modificada de la regresión logística. (Batta, Singh, Li, Ding, & Trajkovic, 2018). La fórmula que representa es la siguiente:

$$d(\bar{x}) = \sum_{i=1}^m \alpha_i y_i K(\bar{x}_i, \bar{x}) + b$$

Considerando que:

$\alpha_i$ : parámetros que se consideran en el algoritmo

$\bar{x}_i$ : representa a los datos que se proporcionará.

$K(\bar{x}_i, \bar{x})$ : representa la función kernel que estructura las muestras en el espacio dimensional.

Para el desarrollo del algoritmo SVM propuesto para esta investigación se consideró un Kernel lineal.

$$K(x, x') = x \cdot x'$$

Donde:

$x$ : representa a los atributos de entrenamiento.

El tipo de SVM seleccionado fue el C-SVC, C representa el parámetro de regularización considerando el error de entrenamiento y el modelo de complejidad, para el desarrollo de la propuesta se asignó el valor de C=100. La implementación del algoritmo en Python se realizó de la siguiente forma:

Tabla 15

Parámetros que se utilizaron al implementar el algoritmo SVM.

Algoritmos	Parámetros utilizados	Valor(es)
Support Vector Machine	Tipo de SVM	LinearSVC, C=100
	Función del kernel	Lineal: $K(x, x') = x \cdot x'$

La implementación del algoritmo se procedió a realizar la carga y lectura de los datos organizados en formato CSV.

```
#importamos las librerías
import pandas as pd
import numpy as np
from timeit import timeit
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn import metrics
from sklearn.metrics import classification_report
from math import sin
from time import time
from sklearn.metrics import accuracy_score
#importamos la base de datos
df_datos=pd.read_csv("DataSet_SQL.csv")
```

Una vez leído las librerías necesarias para trabajar con los datos y el algoritmo, se procede a separar los datos en arrays para luego crear los datos de entrenamiento y los datos de prueba, se toma como datos de prueba el 20% y el 80% para el entrenamiento.



```
#Separa datos
```

```
X = np.array(df_datos.drop(['Num_Caract_Esp', 'Long_URL', 'Label'], 1))  
y = np.array(df_datos['Label'])  
tiempolni = time()  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  
print('Son {} para entrenar {} para prueba'.format(X_train.shape[0], X_test.shape[0]))
```

Como resultado se tiene que los datos para el estudio es de 8438 registros, entonces para el entrenamiento se eligieron 6751 datos para el entrenamiento, mientras que la prueba se utilizaron 1688 datos, para esto se aplica la ley de Pareto que se plantea el 80% y el 20%. El siguiente paso corresponde a implementar la librería del algoritmo SVM utilizando el tipo de kernel LinearSVC, con un parámetro de regularización de C=100, con una tasa de aprendizaje de 1

```
#----Modelo-----
```

```
model_algoritmo = LinearSVC()  
model_algoritmo.fit(X_train, y_train)  
Y_pred = model_algoritmo.predict(X_test)
```

Finalmente se imprimen los valores:

```
tiempoFin = time()  
tiempoTotal = tiempoFin - tiempolni  
print("Precisión:", metrics.precision_score(y_test, Y_pred))  
print("El tiempo de ejecucion fue:", tiempo_ejecucion)  
print('Accuracy: {}'.format(algoritmo.score(X_train, y_train)))  
print("Matriz confusión :\n", metrics.confusion_matrix(y_test, Y_pred))  
print("AUC:", metrics.roc_auc_score(y_test, Y_pred))  
print("Recall:", metrics.recall_score(y_test, Y_pred))  
print("F-Score:", metrics.f1_score(y_test, Y_pred))  
print("ROC:", metrics.roc_curve(y_test, Y_pred))  
import os  
import psutil  
pid = os.getpid()  
py = psutil.Process(pid)  
memoryUse = py.memory_info()[0]/2.**30  
print('Memoria Usada:', memoryUse)
```

**Algoritmo de aprendizaje automático Decision Tree**, algoritmo aprendizaje supervisado, permite la resolución de problemas de regresión y clasificación, permitiendo crear modelos de entrenamiento para predecir clases a partir de reglas de decisión simple tomando datos anteriores. Entonces para realizar las predicciones de etiquetas de clases de los registros se parte desde la raíz del árbol, posteriormente se realiza la comparación de valores del atributo raíz con el atributo del registro y se continua con la rama siguiente y posteriormente se pasa a los siguientes nodos. Para iniciar el algoritmo se inicia con la medida de la selección de atributos, que facilita la segmentación de los datos. La ganancia de información utiliza la entropía antes de proceder a realizar la segmentación de datos, el algoritmo ID3 utiliza la ganancia de información basándose en la siguiente fórmula.

$$Info(D) = - \sum_{i=1}^m p_i \log_2 p_i$$

Entonces, la constante  $P_i$  es la posibilidad de que tupla arbitraria en  $D$  forme parte de la clase  $C_i$ .

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(A)$$

Otro de los puntos que se considera importante es la ratio de ganancia de la información, en este aspecto tomaremos como base al algoritmo J48, con este algoritmo se determinó la ganancia de la información en el análisis a partir del dataset que se utilizó. Considerando para el análisis la siguiente fórmula.

$$SplitInfo_A(D) = \sum_{j=1}^v \frac{|D_j|}{D} \times \log_2 \left( \frac{|D_j|}{D} \right)$$

Entonces la relación que se tiene con respecto a la ganancia se define como:

$$GainRatio(A) = \frac{Gain(A)}{splitInfo_A(D)}$$

Esta fórmula da como resultado a la elección del atributo con ganancia más elevada, el cual sirve como atributo de segmentación. Ahora se muestra la implementación del algoritmo. Para el desarrollo del algoritmo decisión tree propuesto para esta investigación se consideró los parámetros.

Tabla 16

Parámetros que se utilizaron al implementar el algoritmo Decision Tree.

Algoritmos	Parámetros utilizados	Valor(es)
Decision Tree	max_depth (Profundidad máxima del árbol)	5
	Criterion (Mide la impureza de una división)	Gini $G_m = \sum_{k=1}^K p_{mk} (1 - p_{mk})$
	random_state (Grado de aleatoriedad del estimador)	123

En la implementación del algoritmo se procedió a importar las librerías, carga y lectura de los datos.

```
#importamos las librerías
import pandas as pd
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.tree import plot_tree
from timeit import timeit
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import classification_report
from math import sin
from time import time
from sklearn.metrics import accuracy_score
#importamos la base de datos.
df_datos=pd.read_csv("DataSet_SQL.csv")
```

Una vez leído las librerías necesarias para trabajar con los datos y el algoritmo, se procede a separar los datos de entrenamiento y los datos de prueba, se toma como datos de prueba el 20% y el 80% para el entrenamiento.

```
#X_Medio, y_Medio, test_size=0.2, random_state=123
tiempolni = time()
Medio = df_datos[['Method', 'Long_URL', 'Num_Caract_Esp', 'Commands_SQL',
'Label']]
X_Medio = np.array(Medio.drop(['Label'], 1))
y_Medio = np.array(Medio['Label'])
X_train, X_test, y_train, y_test = train_test_split(X_Medio, y_Medio, test_size=0.2, random_state=123)
```

La cantidad de datos evaluados para el estudio es de 8438 registros, entonces para el entrenamiento se eligieron 6751 datos para el entrenamiento, mientras que la prueba se utilizaron 1688 datos, para esto se aplica la ley de Pareto que se plantea el 80% y el 20%. Posteriormente se procedió a la aplicación del algoritmo Decisión Tree, ajustando los parámetros `max_depth = 5`, `criterion = 'gini'`, `random_state = 123`

```
#Modelo
algoritmo_1 = DecisionTreeClassifier(max_depth=5, criterion = 'gini', random_state=123)
algoritmo_1.fit(X_train, y_train)
Y_pred = algoritmo_1.predict(X_test)
```

Finalmente se imprimen los valores:

```
tiempoFin = time()
tiempo_ejecucion = tiempoFin - tiempoIni
print("Precisión:", metrics.precision_score(y_test, Y_pred))
print("El tiempo de ejecucion fue:", tiempo_ejecucion)
print('Accuracy: {}'.format(algoritmo.score(X_train, y_train)))
print("Matriz confusión :\n", metrics.confusion_matrix(y_test, Y_pred))
print("AUC:", metrics.roc_auc_score(y_test, Y_pred))
print("Recall:", metrics.recall_score(y_test, Y_pred))
print("F-Score:", metrics.f1_score(y_test, Y_pred))
print("ROC:", metrics.roc_curve(y_test, Y_pred))
import os
import psutil
pid = os.getpid()
py = psutil.Process(pid)
memoryUse = py.memory_info()[0]/2.**30
print('Memoria Usada:', memoryUse)
```

## Algoritmo de aprendizaje automático AdaBoost

Es un clasificador de refuerzo de conjuntos, lo interesante de este algoritmo es su capacidad de combinar múltiples clasificadores, Adaboost trabaja con métodos interactivos, lo cual le permite obtener resultados de alta precisión. Entonces para trabajar con AdaBoost se tiene que asignar los pesos para los clasificadores y posteriormente entrenar la muestra de datos en las interacciones que se realicen para así asegurar las predicciones con precisión de las situaciones inusuales.

La ecuación final para la clasificación se puede representar como:

$$F(x) = \text{sign}\left(\sum_{m=1}^M \theta_m \int m(x)\right),$$

Donde  $f_m$  representa el  $m$ ésimo clasificador débil y  $\theta_m$  es el peso correspondiente. Es exactamente la combinación ponderada de  $M$  clasificadores débiles. Todo el procedimiento del algoritmo AdaBoost se puede resumir de la siguiente manera. Dado un conjunto de datos que contiene  $n$  puntos, donde:  $x_i \in R^d, Y_i \in \{-1,1\}$ . Aquí  $-1$  denota la clase negativa mientras que  $1$  representa la positiva. Inicialice el peso para cada punto de datos como:  $w(x_i, y_i) = \frac{1}{n}, i = 1, \dots, n$ .

Para el desarrollo del algoritmo AdaBoost propuesto para esta investigación se consideró el parámetro  $n\_estimators$  con un valor de 50, para este caso se usa el máximo número de estimadores para el refuerzo respectivo, si el algoritmo logra el ajuste deseado, el procedimiento del desarrollo del aprendizaje se detiene en el punto antes de utilizar todos los estimadores, combinando con el  $learning\_rate$  se logra mejorar el equilibrio.

Tabla 17

*Parámetros que se utilizaron al implementar el algoritmo AdaBoost.*

Algoritmos	Parámetros utilizados	Valor(es)
AdaBoost	$n\_estimators$	50
	$learning\_rate$	1

(peso aplicado a cada clasificador en las interacciones)

Fuente: elaboración propia

En la implementación del algoritmo se procedió a importar las librerías, carga y lectura de los datos.

```
#importamos las librerías
import pandas as pd
import numpy as np
from timeit import timeit
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from math import sin
from time import time
df_datos=pd.read_csv("DataSet_SQL.csv")
```

Una vez leído las librerías necesarias para trabajar con los datos y el algoritmo, se procede a separar los datos de entrenamiento y los datos de prueba, se toma como datos de prueba el 20% y el 80% para el entrenamiento.

```
tiempoIni = time()
X = np.array(df_datos.drop(['Num_Caract_Esp', 'Long_URL','Label'], 1))
y = np.array(df_datos['Label'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
print('Son {} data entrenamiento y {} data prueba'.format(X_train.shape[0], X_test.shape[0]))
```

Posteriormente se procedió a la aplicación del algoritmo AdaBoost, ajustando los parámetros `n_estimators` con un valor máximo de 50 estimadores y el `learning_rate` o tasa de aprendizaje con un valor de 1.

```
# Modelo
algoritmo = AdaBoostClassifier(n_estimators=50, learning_rate=1)
modelo = algoritmo.fit(X_train, y_train)
y_pred = modelo.predict(X_test)
tiempoFin = time()
tiempo_ejecucion = tiempoFin - tiempoIni
```

Finalmente se imprimen los valores:

```
tiempoFin = time()
tiempoTotal = tiempoFin - tiempoIni
print("Precisión:",metrics.precision_score(y_test, Y_pred))
print("El tiempo de ejecucion fue:",tiempo_ejecucion)
print('Accuracy: {}'.format(algoritmo.score(X_train, y_train)))
print("Matriz confusión :\n",metrics.confusion_matrix(y_test, Y_pred))
print("AUC:",metrics.roc_auc_score(y_test, Y_pred))
print("Recall:",metrics.recall_score(y_test, Y_pred))
print("F-Score:",metrics.f1_score(y_test, Y_pred))
print("ROC:",metrics.roc_curve(y_test, Y_pred))
import os
import psutil
pid = os.getpid()
py = psutil.Process(pid)
memoryUse = py.memory_info()[0]/2.**30
print('Memoria Usada:', memoryUse)
```

Entonces, los parámetros muy importantes para este algoritmo son: **base\_estimator**, el cual es considerado como un aprendiz débil y que sirve para el entrenamiento del modelo, este parámetro utiliza el clasificador del árbol de decisiones como aprendiz débil; el segundo parámetro **n\_estimators**, relacionado con el número de aprendices débiles utilizados en el entrenamiento de forma iterativa y el tercer parámetro es **learning\_rate**, ayuda a mejorar el peso de los aprendices débiles, usa el valor numérico entero de 1.

Tabla 18  
*Parámetros utilizados para los algoritmos.*

Algoritmos	Parámetros utilizados	Valor(es)
Decision Tree	max_depth	5
	criterion	Gini $G_m = \sum_{k=1}^K p_{mk}(1 - p_{mk})$
	random_state	123
Support Vector Machine	Tipo de SVM	LinearSVC, C=100
	Función del kernel	Lineal: $K(x, x') = x \cdot x'$
AdaBoost	learning_rate <small>Tasa de aprendizaje</small>	1
	n_estimators <small>Número de impulso</small>	50

Fuente: elaboración propia

#### IV. CONCLUSIONES Y RECOMENDACIONES

##### 4.1. Conclusiones.

- a. El estudio planteó la clasificación de los ataques de inyección SQL a base de datos, se tomó como base la clasificación realizada por (Khanna & Verma, 2018) y (Sheykhkanloo, 2017), para ello se construyó el Dataset que significan las firmas, datos que facilitaron la implementación posterior del dataset.
- b. En el estudio se planteó la implementación de un dataset elaborado a partir del recojo de datos de inyección SQL generado por la herramienta OwasZAP, luego se realizó la identificación de elementos que corresponden a una consulta inyectada con comandos SQL para vulnerar la seguridad de las bases de datos desde una aplicación Web para el entrenamiento y ejecución de los algoritmos de aprendizaje automático para la detección de inyección SQL.
- c. En el estudio se planteó el uso del entorno de trabajo de Phyton el cual cuenta con librerías que facilitan la implementación de los algoritmos AdaBoost, SVM y Decision Tree, Phyton en el entorno de trabajo Jupiter



Notebook y scikit-learn demostraron ser un buen entorno para la implementación de los algoritmos por la existencia de múltiples librerías.

- d. Se realizó el análisis y selección de algoritmos de aprendizaje automático considerando con base al nivel de precisión en la detección de ataques de inyección SQL, se construyó una tabla para evaluar los algoritmos, la revisión de la bibliografía, después de realizar el análisis comparativo de los algoritmos con base a resultados obtenidos en diversas fuentes se llegó a determinar que los algoritmos Adaboost, SVM y Decision Tree tenían los mejores resultados en cuanto a precisión.
- e. Después de generar el dataset, se procedió con la implementación de los algoritmos de aprendizaje automático seleccionados como es el algoritmo SVM, AdaBoost y Decision Tree, el siguiente paso fue ejecutar los algoritmos teniendo como resultados promedio de las 5 pruebas realizadas del algoritmo Support Vector Machine con un kernel lineal con una Exactitud promedio del 88.92 %, el nivel de precisión 96.14%, el Recall 86.88 % y el F-Score de 91.24 %, lo cual demuestra que este algoritmo alcanzó la puntuación más baja, seguido por el algoritmo AdaBoost con un nivel de exactitud promedio de las cinco pruebas realizadas del 99.65 %, el nivel de precisión de este algoritmo se ubicó en un 99.74 %, con un Recall de 99.42% y un F-Score de 99.62%, datos que lo ubican en un buen nivel de precisión en la detección de inyección SQL.
- f. Finalmente se pudo determinar que el algoritmo de aprendizaje automático que alcanzó al máximo nivel de precisión fue el Decisión Tree con el 100 % de exactitud y el 100 % de precisión, el 100% de Recall y 100 % de F-Score. Entonces se puede concluir que el algoritmo de mejor rendimiento para la detección de inyección SQL es Decision Tree.

#### **4.2. Recomendaciones**

- a. Para seleccionar los algoritmos de aprendizaje automático se tiene que realizar búsqueda de información en revistas indexadas como IEEE, ACM Library, Scopus, ScienceDirect, considerando un rango de fecha y

utilizando las palabras de búsqueda como SQL injection attack, Machine Learning, machine learning and SQL injection attack.

- b. Para la clasificación de los ataques de inyección SQL a base de datos debe realizarse primero tomando como referencia la búsqueda de aportes de autores de investigaciones publicadas en IEEE Xplore, Scopus, Web of Science; utilizando el filtro búsqueda palabras como: SQL injection, Classification SQL injection y luego codificar en una tabla los cada uno de los aspectos encontrados para la realización del análisis posterior.
- c. Para la implementación de los algoritmos es importante utilizar herramientas de Jupyter Notebook, combinando el trabajo con las librerías de scikit-learn, Pandas, Numpy, Metrics e importando los algoritmos seleccionados Decission Tree, Support Vector Machines y AdaBoost.
- d. Determinar otras métricas para el dataset que ayuden a optimizar la detección de la inyección SQL, aumentando la cantidad de casos de inyección SQL para ser analizados.

## REFERENCIAS

- Almseidin, M., Alzubi, M., Kovacs, S., & Alkasassbe, M. (2017). Evaluation of Machine Learning Algorithms for Intrusion Detection System. *IEEE*. doi:10.1109 / SISY.2017.8080566
- Almseidin, M., Szilveszter, K., & Al-kasassbeh, M. (2017). Evaluation of machine learning algorithms for intrusion detection system. *ResearchGate*, 277-282. doi:10.1109/SISY.2017.8080566
- Alonso, J., Guzmán, A., Laguna, P., & Martín, A. (s/f). *Ataques a BB.DD., SQL Injection*. Catalunya. Obtenido de [https://d1wqtxts1xzle7.cloudfront.net/40002441/Ataques-a-bases-de-datos.pdf?1447526544=&response-content-disposition=inline%3B+filename%3DAtaques\\_a\\_bases\\_de\\_datos.pdf&Expires=1601016371&Signature=Oz0jMgfP4GSg6S2gTPHjGmlecMxPBmszSGGICHxDq8L3faaBMQhbfoLSgL](https://d1wqtxts1xzle7.cloudfront.net/40002441/Ataques-a-bases-de-datos.pdf?1447526544=&response-content-disposition=inline%3B+filename%3DAtaques_a_bases_de_datos.pdf&Expires=1601016371&Signature=Oz0jMgfP4GSg6S2gTPHjGmlecMxPBmszSGGICHxDq8L3faaBMQhbfoLSgL)
- Batta, P., Singh, M., Li, Z., Ding, Q., & Trajkovic, L. (2018). Evaluation of Support Vector Machine Kernels for Detecting Network Anomalies. *IEEE*. doi:10.1109/ISCAS.2018.8351647
- Bhagwani, H., Negi, R., Dutta, A., Handa, A., Kumar, N., & Kumar, S. (2019). Clasificación automatizada de ataques a aplicaciones web para la detección de intrusos. *Springer*, 123-141. doi:10.1007/978-3-030-35869-3\_10
- Bonaccorso, G. (2017). *Machine Learning Algorithms*. Birmingham: Pack Publishing Ltd.
- Charris, L., Henriquez, C., Hernandez, S., Jimeno, L., Guillen, O., & Moreno, S. (2014). Análisis comparativo de algoritmos de árboles de decisión en el procesamiento de datos biológicos. *Revista I+D en TIC*, 26-34.
- Chu, W., Lin, C., & Chang, K. (2019). Detection and Classification of Advanced Persistent Threats and Attacks Using the Support Vector Machine. *Applied Sciences*, 2-16. doi:10.3390/app9214579
- Clarke, J. (2012). *SQL Injection Attacks*. Amsterdam: Elsevier.
- Clarke, J. (2012). *SQL Injection Attacks and Defense*. Waltham: Elsevier. Obtenido de

- <https://doc.lagout.org/security/SQL%20INJECTION%20SECOND%20EDITIOn/SQL%20INJECTION%20SECOND%20EDITIOn.pdf>
- Cui , B., He, S., Yao, X., & Shi, P. (2018). Malicious URL detection with feature extraction based on machine learning. *InderScience*, 166-178.  
doi:10.1504/IJHPCN.2018.094367
- Dharitri, T., Gohil, R., & Halabi, T. (2020). Detectar ataques de inyección de SQL en Cloud SaaS mediante el aprendizaje automático. *IEEE*, 141-150.  
doi:10.1109 / BigDataSecurity-HPSC-IDS49724.2020.00035
- Dong, Y., Zhang, Y., Ma, H., Wu, Q., Liu, Q., Wang, K., & Wang, W. (2018). An adaptive system for detecting malicious queries in web attacks. *SpringerLink*. doi:<https://doi.org/10.1007/s11432-017-9288-4>
- Durai, K., & Baskaran, K. (2019). Decision tree classification – N tier solution for preventing SQL injection attack on websites. *Int. J. Enterprise Network Management*, 253-271. doi:10.1504/ijenm.2019.103155
- Enciso, N., & Camargo, J. (2020). Classification and Visualization of Web Attacks Using HTTP Headers and Machine Learning Techniques. *Springer*, 243-256. doi:10.1007/978-3-030-46785-2
- Fillatre, L., & Nikiforov, I. (2011). Fault Detection and Isolation in Large-Scale IP Network. *Elsevier*.
- Google developers. (2021). *Aprendizaje automático*. Obtenido de <https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss?hl=es>
- Guaman, R. (2016). Seguridad en Aplicaciones web para Sistemas de Gestión Académica. *Revista Tecnológica ESPOL*.
- Gupta, R., Tanwar, S., Tyagi, S., & Kumar, N. (2020). Machine Learning Models for Secure Data Analytics: A taxonomy and threat model. *ElseVier*, 406-440. doi:10.1016/j.comcom.2020.02.008
- Guttman, B., & Roback, E. (1995). An Introduction to Computer Security: The NIST Handbook. *NIST*.
- Hasan, M., Balbahaith, Z., & Tarique, M. (2019). Detection of SQL Injection Attacks: A Machine Learning Approach. *IEEE*. doi:10.1109 / ICECTA48151.2019.8959617

- Haykin, S. (2008). *Neural Networks and Learning Machines*. New Jersey: Pearson Prentice.
- Hernández, R., Fernández, C., & Baptista, M. (2014). *Metodología de la investigación*. México: Mc Graw Hill. Obtenido de <http://observatorio.epacartagena.gov.co/wp-content/uploads/2017/08/metodologia-de-la-investigacion-sexta-edicion.compressed.pdf>
- Hernández, V. (2018). *Acceso ilícito a sistemas informáticos*. Obtenido de <https://blog.hernandez-vilches.com/intrusion-informatica-acceso-ilicito/>
- Imtiazul, N., Hasan, M., Golam, M., Imtiaz, A., & Sarwat, A. (2020). Machine Learning in Generation, Detection, and Mitigation of Cyberattacks in Smart Grid: A Survey. *IEEE*. Obtenido de <https://arxiv.org/pdf/2010.00661.pdf>
- Kamtuo, K., & Soomlek, C. (2017). Machine Learning for SQL Injection Prevention on Server-Side Scripting. *IEEE*. doi:10.1109 / ICSEC.2016.7859950
- Khalid, M., Farooq, H., Iqbal, M., Alam, T., & Rasheed, K. (2019). Predicción de vulnerabilidades web en aplicaciones web basadas en aprendizaje automático. *Springer*, 473-484. doi:10.1007/978-981-13-6052-7\_41
- Khanna, S., & Verma, A. (2018). Classification of SQL Injection Attacks Using Fuzzy Tainting. *Springer*, 463-469. doi:10.1007/978-981-10-3373-5\_46
- Kokkonen, T., & Hämäläinen, T. (2016). Model for Sharing the Information of Cyber Security Situation Awareness between Organizations. *IEEE*. doi:10.1109 / ICT.2016.7500406
- Li, Y., & Zhang, B. (2019). Detection of SQL Injection Attacks Based on Improved TFIDF Algorithm. *Journal of Physics*, 1-8. doi:10.1088/1742-6596/1395/1/012013
- Mishra, S. (2019). SQL Injection Detection Using Machine Learning. *IEEE*. doi:<https://doi.org/10.31979/etd.j5dj-ngvb>
- Murphy, K. (2006). 2006. *Naive Bayes classifiers*.
- Ogbomon, S., & Buchanan, W. (2016). Numerical Encoding to Tame SQL Injection Attacks. *IEEE*, 1253-1256. doi:10.1109 / NOMS.2016.7502997

- Ogbomon, S., Buchanan, W., & Fan, L. (2017). Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention. *IEEE*. doi:10.23919/INM.2017.7987433
- Oliveira , L., Adriano, G., Souza, V., Silva , V., Silva, T., Junio , A., & Campos , P. (2018). Fuzzy neural networks to create an expert system for detecting attacks by SQL Injection. *IJoFCS*, 8-21. doi:http://dx.doi.org/10.5769/J201801001
- OWASP. (2017). *OWASP Top 10-2017: The Ten Most Critical Web Application Security Risks*. Obtenido de [https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010-2017%20\(en\).pdf](https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010-2017%20(en).pdf)
- Pattewar, T., Patil, H., Patil, H., Patil , N., Taneja , M., & Wadile, T. (2019). Detection of SQL Injection using Machine Learning: A Survey. *IRJET*. Obtenido de [https://d1wqtxts1xzle7.cloudfront.net/61273198/IRJET-V6I114220191119-123579-856wj2.pdf?1574228479=&response-content-disposition=inline%3B+filename%3DIRJET\\_Detection\\_of\\_SQL\\_Injection\\_using\\_M.pdf&Expires=1601184491&Signature=aBg5~EmT6F28IBpiold2HhPtuibvjPpC](https://d1wqtxts1xzle7.cloudfront.net/61273198/IRJET-V6I114220191119-123579-856wj2.pdf?1574228479=&response-content-disposition=inline%3B+filename%3DIRJET_Detection_of_SQL_Injection_using_M.pdf&Expires=1601184491&Signature=aBg5~EmT6F28IBpiold2HhPtuibvjPpC)
- Raschka, S., & Mijarlili, V. (2017). *Python Machine Learning*. Birmingham: Pack Publishing Ltd.
- Romero, D. (2019). Compilación de tipos de ataque a base de datos: inyecciones SQL. *Redtis*.
- Romero, M., Figueroa, G., Vera, D., Álava, J., Parrales, G., Murillo, Á., & Castillo, M. (2018). *Introducción a la seguridad informática y el análisis de vulnerabilidades*. Alicante: 3Ciencias. Obtenido de <https://www.3ciencias.com/wp-content/uploads/2018/10/Seguridad-inform%C3%A1tica.pdf>
- Ruiz, A., López, J., & Delgado, J. (2019). The Social Component of the Hybrid Threat and its Detection with Bayesian Models. *Scielo*. doi:https://doi.org/10.17141/urvio.25.2019.3997
- Salgado, R. (2013). *Websec*. Obtenido de [https://www.websec.ca/kb/sql\\_injection#MySQL\\_Default\\_Databases](https://www.websec.ca/kb/sql_injection#MySQL_Default_Databases)

- Shalev, S., & Ben, D. (2014). *Understanding Machine Learning: from theory to algorithms*. Paris: Hardback.
- Sheykhkanloo, N. (2017). A Pattern Recognition Neural Network Model for Detection and Classification of SQL Injection Attacks. *International Journal of Cyber Warfare and Terrorism*, 1443-1453.
- Tang, P., Qiu, W., Huang, Z., Lian, H., & Liu, G. (2020). Detection of SQL injection based on artificial neural network. *E/Sevier*. doi:<https://doi.org/10.1016/j.knosys.2020.105528>
- Xie, X., Ren, C., Fu, Y., Xu, J., & Guo, J. (2017). SQL injection detection for web applications based on Elastic-Pooling CNN. *IEEE*. doi:10.1109 / ACCESS.2019.2947527
- Zhang, K. (2019). A Machine Learning Based Approach to Identify SQL Injection Vulnerabilities. *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1286-1287. Obtenido de <https://ieeexplore.ieee.org/abstract/document/8952467>

# ANEXOS

## Anexo 1 : Resolución de aprobación del proyecto

### FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO RESOLUCIÓN N° 2322-2020/FIAU-USS

Pimentel, 17 de noviembre de 2020

#### VISTOS:

El Acta de reunión N° 2610-2020 del Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS remitida el 12 de noviembre de 2020 mediante oficio N° 0237-2020/FIAU-IS-USS de la Dirección de Escuela profesional de INGENIERÍA DE SISTEMAS, y;

#### CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y tesis son aprobados por el Comité de Investigación y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El periodo de vigencia de los mismos será de dos años, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24° señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; es individual o en pares para obtener un título profesional. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, según documentos de vistos el Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS acuerda aprobar la modificación de los temas de Tesis a cargo de los estudiantes y/o egresados que se detallan en el anexo de la presente Resolución.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

#### SE RESUELVE:

**ARTÍCULO 1°: MODIFICAR**, el tema de la Tesis perteneciente a la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE, a cargo de los estudiantes y/o egresados del Programa de estudios de **INGENIERÍA DE SISTEMAS** según se detalla en el anexo de la presente Resolución.

**ARTÍCULO 2°: MODIFICAR**, la Resolución de Facultad con la que se asigna Asesor especialista en el extremo del tema de la tesis quedando tal como se detalla en el anexo de la presente Resolución.

**ARTÍCULO 3°: DEJAR SIN EFECTO**, toda Resolución emitida por la Facultad que se oponga a la presente Resolución.

#### REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

  
 Dr. Mario Fernando Ramos Moscol  
Decano - Facultad de Ingeniería,  
Arquitectura y Urbanismo  
UNIVERSIDAD SEÑOR DE SIPÁN S.A.C.

  
 MBA. María Noelia Sialer Rivera  
Secretaría Académica / Facultad de Ingeniería,  
Arquitectura y Urbanismo  
UNIVERSIDAD SEÑOR DE SIPÁN S.A.C.

Cc: Interesado, Archivo



**FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO  
RESOLUCIÓN N°2322-2020/FIAU-USS**

Pimentel, 17 de noviembre de 2020

**ANEXO**

<b>N°</b>	<b>APELLIDOS Y NOMBRES</b>	<b>TEMA DE TESIS PRIMIGENIO</b>	<b>TEMA DE TESIS MODIFICADO</b>
1	CABREJOS TORRES RAMIRO	INCIDENCIA DE LA METODOLOGÍA MAGERIT V3 EN LA SEGURIDAD DE INFORMACIÓN DE LA EMPRESA DECO INTERIORS SAC	INFLUENCIA DE LA METODOLOGÍA MAGERIT V3 EN LA SEGURIDAD DE INFORMACIÓN DE LA EMPRESA DECO INTERIORS SAC.
2	MARRUFO SALAZAR YOAN JOHEL	MEJORA DE LA USABILIDAD Y ACCESIBILIDAD EN PORTALES WEB EN INSTITUCIONES EDUCATIVAS DEL NIVEL SECUNDARIO	EVALUACIÓN DE LA USABILIDAD Y ACCESIBILIDAD EN PORTALES WEB DE INSTITUTOS DE EDUCACIÓN SUPERIOR PEDAGÓGICA DEL PERÚ
3	CAMONES AGUIRRE OSCAR BRYAN	EVALUACION DE LA VIABILIDAD DE IMPLEMENTACION DE TECNICA BASADA EN ALGORITMO GENÉTICO Y EL FRAMEWORK WS-ATTACKER PARA DETECTAR ATAQUES A SERVICIOS WEB EN UN AMBIENTE DE COMPOSICIÓN DINÁMICO	EVALUACIÓN DE TÉCNICA BASADA EN REGLAS Y ALGORITMO GENÉTICO PARA DETECTAR ATAQUES A SERVICIOS WEB EN UN AMBIENTE DE COMPOSICIÓN DINÁMICO
4	ALVAREZ GONZAGA BRAULIO RICARDO	ANÁLISIS COMPARATIVO DE TÉCNICAS DE EXTRACCIÓN, TRANSFORMACIÓN Y CARGA DE DATOS APLICADAS A BUSINESS INTELLIGENCE	ANÁLISIS COMPARATIVO DE TÉCNICAS DE MINERÍA DE DATOS APLICADAS A BUSINESS INTELLIGENCE
5	SANDOVAL ODAR WILLIAM	COMPARACIÓN DE LAS TÉCNICAS DE SEGMENTACIÓN OPTIMIZACIÓN DE ENJAMBRES DE PARTÍCULAS Y AGRUPAMIENTO JERÁRQUICO EN AMBIENTES NO CONTROLADOS DE PLANTACIONES DE ARROZ	COMPARACIÓN DE ALGORITMOS DE SEGMENTACIÓN DE IMÁGENES DIGITALES DE PLANTAS DE ARROZ EN AMBIENTES NO CONTROLADOS
6	MOGOLLÓN GARCÍA MANUEL ESTEBAN	EVALUACIÓN DE HERRAMIENTAS DE SEGURIDAD INFORMÁTICA COMO SOPORTE DE LA NORMA ISO 27001 PARA GARANTIZAR LA GESTIÓN DE LA SEGURIDAD DE LA INFORMACIÓN EN UNA MUNICIPALIDAD DEL PERÚ	DESARROLLO DE UN MODELO DE GESTIÓN DE RIESGOS BASADO EN LA METODOLOGÍA MAGERIT PARA MINIMIZAR LOS RIESGOS DE LA IMPLANTACIÓN Y USO DE TI EN UNA MUNICIPALIDAD DEL PERÚ
7	CASTRO FERNÁNDEZ LEVI RONALD	IDENTIFICACIÓN DE INTRUSIONES A BASE DE DATOS DESDE APLICACIONES WEB UTILIZANDO LOS ALGORITMOS DE APRENDIZAJE AUTOMÁTICO	ANÁLISIS COMPARATIVO DE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA IDENTIFICAR ATAQUES DE INYECCIÓN SQL A BASE DE DATOS EN APLICACIONES WEB
8	CALDERON TENORIO CESAR ROLEN	MODELO DE CONTROL AUTOMÁTICO PARA EL PROCESO DEL RIEGO EN EL CULTIVO DE LA PAPA EN EL CENTRO POBLADO DE CHAQUIL DISTRITO DE LA ESPERANZA PROVINCIA DE SANTA CRUZ REGION DE CAJAMARCA	DESARROLLO DE UN MODELO DE CONTROL AUTOMÁTICO BASADO EN INTELIGENCIA ARTIFICIAL PARA EL PROCESO DEL RIEGO EN EL CULTIVO DE LA PAPA PERUANA

**Anexo 02: Formato de recojo de información**

<b>Consumo de CPU</b>			
<b>Ítems</b>		<b>Valor</b>	
Tiempo de Uso			
Velocidad			
Procesos activos			
<b>Consumo de Memoria</b>			
<b>Ítems</b>		<b>Valor</b>	
Tiempo de uso			
Disponibilidad			
En caché			
<b>Promedio de Tiempo de respuesta</b>			
<b>Ítems</b>		<b>Valor</b>	
Velocidad			
Tiempo de uso			
CPU			
Memoria			
Disco			
		<b>Resultados Clasificación</b>	
		Consulta Normal	Consulta Anómala
<b>Resultados Reales</b>	Consulta Normal		
	Consulta Anómala		
<b>Ítems</b>		<b>Valor</b>	
Verdadero positivo (VP)			
Falso Positivo (FP)			
Verdadero Negativo (VN)			
Falso Negativo (FN)			
<b>Ítems</b>		<b>Valor</b>	
Exactitud			
Precisión			
F-Score			
AUC			

### Anexo 03. Evaluación de la población de algoritmos de aprendizaje automático

Tabla 19.

*Evaluación de la población de algoritmos de aprendizaje automático para determinar la muestra*

<b>Nro</b>	<b>Algoritmo de aprendizaje automático</b>	<b>Precisión</b>	<b>Escenario de aplicación</b>	<b>Investigador (es)</b>
1	Logistic Regression	70 %	Classification and Visualization of Web Attacks Using HTTP Headers and Machine Learning Techniques	(Enciso & Camargo, 2020)
2	Support Vector Machine	98.5 %	Malicious URL detection with feature extraction based on machine learning	(Cui , He, Yao, & Shi, 2018)
3	Gradient Boosting Classifier	95.16 %	Automated Classification of Web-Application Attacks for Intrusion Detection	(Bhagwani, y otros, 2019)
4	Decision Tree	99.40 %	Malicious URL detection with feature extraction based on machine learning	(Cui , He, Yao, & Shi, 2018)
5	K-Nearest Neighbor	98.22 %	Detection of SQL Injection Attacks Based on Improved TFIDF Algorithm	(Li & Zhang, 2019)
6	Random Forest	97 %	Automated Classification of Web-Application Attacks for Intrusion Detection	(Bhagwani, y otros, 2019)

7	Naïve Bayes	94.20 %	Malicious URL detection with feature extraction based on machine learning	(Cui , He, Yao, & Shi, 2018)
8	AdaBoost	99.7 %	Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning	(Dharitri, Gohil, & Halabi, 2020)
9	Perceptron	94 %	Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention	(Ogbomon, Buchanan, & Fan, 2017)
10	Polynomial Support Vector Machine	93.58 %	Detection and Classification of Advanced Persistent Threats and Attacks Using the Support Vector Machine	(Chu, Lin, & Chang, 2019)
11	Redes Neuronales	95.30%	Detection of SQL injection based on artificial neural network	(Tang, Qui, Huang, & Liu, 2020)
12	Dicotomizador Iterativo (ID3)	98 %	Encoded Flow Features for Network Intrusion Detection in Internet of Things	(Siddiqui & Boukerche, 2020)
13	TensorFlow Linearclassifie	90.8%	Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning	(Tripathy, Gohil, & Halabi , 2020)
14	Hidden Markov Model	89 %	Web attack forensics based on network traffic behavior characteristics and URLs	(Sun, Zhu, Li, & Li, 2018)
15	Deep ANN	93.4%	Detecting SQL Injection Attacks in Cloud SaaS using Machine Learning	(Tripathy, Gohil, & Halabi , 2020)

#### Anexo 04. Matriz de revisión de papers.

Tabla 20

Matriz de revisión de papers para la selección de la muestra de algoritmos de aprendizaje automático.

<b>N°</b>	<b>Año de publicación</b>	<b>Autores</b>	<b>Link</b>	<b>Título original del artículo</b>	<b>Base de datos</b>	<b>Tema que aborda</b>	<b>Problema que enfrenta</b>	<b>El método de solución que propuso</b>	<b>Detalle del Método propuesto</b>	<b>Que resultados obtuvo</b>	<b>Conclusiones relevantes</b>
1	2019	Nicolas Ricardo Enciso and Jorge E. Camargo	10.1070/978-3-03-03003-0-46785-2	Classification and Visualization of Web Attacks Using HTTP Headers and Machine Learning	Springer	Clasificación y visualización de ataques web	Ataques web mediante cabeceras	identificar ataques web como XSS, CRLF e inyección SQL utilizando un conjunto de datos que contiene elementos normales y anómalos.	identificar ataques web como XSS, CRLF e inyección SQL utilizando un conjunto de datos que contiene elementos normales y anómalos. metodología	Para el algoritmo SVM Gaussian, utilizaron un parámetro C igual a 1,11 y una gamma igual a 0,09. Para el bosque aleatorio, el número de parámetros del estimador se	Los resultados se muestran una buena clasificación, precisión y exactitud, especialmente en los casos de bosque aleatorio y SVM

g  
Techniq  
ues

propuesta utiliza fijó en 100, el gaussiano.  
técnicas de mejor No  
reducción algoritmo fue el obstante,  
dimensional bosque los índices  
para la aleatorio, con de  
visualización un 83,2% de clasificación  
(PCA, t-SNE) y precisión y un no son  
algoritmos de área bajo la demasiado  
aprendizaje curva (AUC) elevados  
automático del 93,3%, como para  
(SVM, Naive seguido por el decir que  
Bayes, random SVM Gaussian son muy  
forest, regresión con un 79,4% y precisos y  
logística) para un 87,09% de pueden  
realizar la AUC. Los utilizarse en  
clasificación de demás, como un entorno  
las URL Naive Bayes de software  
contenidas en en ambos de  
las cabeceras tipos, no dieron producción.  
HTTP resultados de ataques  
concluyentes web.

con un 65% de precisión. Una idea clave que puede ser

2 2018 Cui, [10.](#) Maliciou Inde Clasific Análisis Este Mediante el Basándose en Se  
 B., [15](#) s URL rsci ación de enfoque del análisis de las los resultados compararon  
 He, [04/](#) detectio enc de trafico problema características de las pruebas, tres  
 S., [IJ](#) n with e URLS URL basado en el de las URL el clasificador algoritmos  
 Yao, [HP](#) feature Ent con malicios aprendizaje maliciosas, de árbol de de  
 X., & [CN](#) extractio erpri técnica o a busca que la utilizaron dos decisión se de clasificación  
 Shi, [.20](#) n based ses s de través distribución algoritmos de decisión se y el  
 P. [18.](#) on Ltd. aprend de de los valores algoritmos de mostró algoritmo de  
[09](#) machine izaje aprendi de las análisis preferible para árbol de  
[43](#) learning autom zaje característica estadístico su uso en el decisión es  
[67](#) ático automat s para de las basados en el algoritmo final el que que  
 ico instancias aprendizaje por de y se incluyó demostró el  
 maliciosas es gradientes y en el modelo mejor

---

Una diferente de extracción de detección. rendimiento. solicitud la de las automática de En la En de URL instancias características aplicación comparació puede benignas basada en un práctica n con los tratarse benignos y el nivel de umbral aplicación métodos de como un conjunto de sigmoideal para práctica, este detección problem datos de resolver el sistema se ha tradicionale a de formación problema de la desplegado en s, este clasifica comparte el identificación de línea en sistema ción mismo características empresas de puede binaria proceso de eficaces. Internet muy detectar con binaria con el conocidas en precisión las en la conjunto de China que se URL que los datos de especializan maliciosas casos prueba. En el en seguridad que tienen positivo proceso de de la red. característic s son construcción Procesa un as difusas. petición del modelo de enorme El modelo es de detección, se número de de URL probaron tres URLs, detección maliciosos algoritmos de aproximadame no se basa

---



---

as o entrenamiento  
malicioso o y  
as o seleccionamos  
vulnerables el mejor  
y como modelo  
los de detección  
casos práctico.  
negativo como modelo  
s son de detección  
benigno práctico.  
s.

nte 2TB en una sola  
diarios. Lo característico  
hemos a ni en una  
probado con lista negra,  
una lista de sino que es  
URLs un enfoque  
maliciosas en integral que  
línea utiliza todas  
encontradas las  
entre 15 de característico  
octubre de as para el  
2015 y el 29 de aprendizaje  
octubre de automático.  
2015, y mostró Además,  
buen este  
rendimiento. enfoque se  
La precisión es ha  
superior al desarrollado  
98,7% con solo en un  
modestos proyecto  
práctico y

---

								falsos positivos.	desplegado en línea para proteger grandes volúmenes de tráfico web.		
3	2019	Hars h Bhag wani, Rohit Negi, Aneet Kuma r Dutta , 9-	<a href="#">10.</a> <a href="#">10</a> <a href="#">07/</a> <a href="#">97</a> <a href="#">8-</a> <a href="#">3-</a> <a href="#">03</a> <a href="#">0-</a> <a href="#">35</a> <a href="#">86</a> <a href="#">9-</a>	Automat ed Classific ation of Web- Applicati on Attacks for Intrusio n	Spri nger  atizada de ataque s a en aplicac iones web para la	Clasific ación autom atizada de ataques basados en registro s para lograr una	detecció n y clasifica ción de ataques basados en registro s para lograr una	Detección y clasificación de ataques basados en registros basados en registros para lograr una mayor precisión en la detección, considerando el	El método propuesto para la detección y clasificación de ataques basados en registros para TPR, FPR, TNR y FNR para todos los clasificadores en cada	El rendimiento en términos de exactitud, precisión, recuperación, puntuación F1, TPR, FPR, TNR y FNR para todos los clasificadores en cada	Por lo tanto, para detectar estos ataques, implementa mos un enfoque de dos fases, es decir, un análisis

Anand Handa(B), Nitesh Kumar, and Sandeep Kumar Shukla

[3](#)  
[10](#)

Detection

detección de intrusos  
mayor precisión en la detección.

experimento en conjunto de un sitio web en datos de funcionamiento, ataques. Sobre se propuso la base de división en dos fases: la detección y clasificación basada en registros fuera de línea y la implementación de los ataques XSS, SQLi, Path Traversal y LDAP. En el caso de ataques XSS, Random Forest logra la mayor

basado en el registro de línea utilizando seleccionamos varios modelos de aprendizaje automático La tabla 8 y una muestra los resultados de implementación en línea de la versión de análisis del registro. El sistema de análisis de registros en línea lee las solicitudes

precisión, un HTTP de los 99,38%, con archivos de un con un FPR registro bajo. También cada 15 en el caso de segundos los ataques para SQLi, Random clasificarlas Forest obtiene en ataques mejores o solicitudes resultados con normales. una precisión Para ello, el del 97,91%. sistema con una analiza esas precisión del peticiones y 97,91%. Los las pasa a clasificadores diferentes Random modelos de Forest y aprendizaje Gradient automático Boosting para la consiguen la

										mayor precisión del 99,28% para los ataques Path Traversal. En el caso de los ataques LDAP.	
4	2018	Cui, B., He, S., Yao, X., & Shi, P.	<a href="#">10.</a> <a href="#">15</a> <a href="#">04/</a> <a href="#">1J</a> <a href="#">HP</a> <a href="#">CN</a> <a href="#">.20</a> <a href="#">18.</a> <a href="#">09</a> <a href="#">43</a> <a href="#">67</a>	Malicious URL detection with feature extraction based on machine learning	Indexing of malicious URLs with Entrees Ltd.	Classification of traffic URLs with malicious URLs techniques of learning automatic	Análisis de tráfico URL malicioso a través de aprendizaje automático	La distribución de los valores de las características maliciosas para las instancias de aprendizaje automático benignas y el	Mediante el análisis de las características de las URL maliciosas, se utilizaron dos algoritmos de decisión de análisis estadístico basados en el aprendizaje por gradientes	el Basándose en los resultados de las pruebas, el clasificador de árbol de decisión se mostró preferible para su uso en el algoritmo final y se incluyó en el modelo	Se compararon tres algoritmos de clasificación y el algoritmo de decisión se demostró el mejor

---

Una solicitud de URL puede tratarse como un problema a clasificar binaria en que los casos positivos son peticiones de URL maliciosas

conjunto de datos de formación comparte el mismo proceso con el conjunto de datos de prueba. En el proceso de construcción del modelo de detección, se probaron tres algoritmos de entrenamiento y seleccionamos el mejor como modelo

de extracción automática de características basada en un nivel de umbral sigmooidal para resolver el problema de identificación de características eficaces.

de detección. rendimiento. En la En aplicación práctica aplicación práctica, este sistema se ha desplegado en línea en empresas de Internet conocidas en China que se especializan en seguridad de la red. Procesa un enorme número de URLs, aproximadame

comparación con los métodos de detección tradicionales, este sistema puede detectar con precisión las URL maliciosas que tienen características difusas. El modelo de detección no se basa

---

---

as o de detección  
malicios práctico.  
as o como modelo  
vulnera de detección  
bles y práctico.  
los  
casos  
negativo  
s son  
benigno  
s.

nte 2TB en una sola  
diarios. Lo característic  
hemos a ni en una  
probado con lista negra,  
una lista de sino que es  
URLs un enfoque  
maliciosas en integral que  
línea utiliza todas  
encontradas las  
entre 15 de característic  
octubre de as para el  
2015 y el 29 de aprendizaje  
octubre de automático.  
2015, y mostró Además,  
buen este  
rendimiento. enfoque se  
La precisión es ha  
superior al desarrollado  
98,7% con solo en un  
modestos proyecto  
práctico y

---

---

falsos positivos.	desplegado en línea para proteger grandes volúmenes de tráfico web.
----------------------	--

---



5	2019	Yingbo Li and Bin Zhan	<a href="#">10.</a> <a href="#">10</a> <a href="#">88/</a> <a href="#">17</a> <a href="#">42-</a> <a href="#">65</a> <a href="#">96/</a> <a href="#">13</a> <a href="#">95/</a> <a href="#">1/0</a> <a href="#">12</a> <a href="#">01</a> <a href="#">3</a>	Detectio n of SQL Injection Attacks Based on Improve d TFIDF Algorith m	IOP Publ ishi ng Injectio n SQL	Detecti on of SQL ataques de inyecció n SQL	Detecci ón de ataques de inyecció n SQL	El método de detección de SQLIAs basado en una herramienta de segmentación de palabras que se encuentran en el conjunto de datos y las detectamos utilizando el algoritmo TFIDF SVM, que son de mucho más alto que los métodos de la literatura. Esto se debe a una mayor precisión, tasa de recuperación	El método se realizó en primer lugar el uso de una herramienta de segmentación de palabras que se encuentran en el conjunto de datos y las detectamos utilizando el algoritmo TFIDF SVM, que son de mucho más alto que los métodos de la literatura. Esto se debe a una mayor precisión, tasa de recuperación	La precisión, la recuperación y la puntuación F del método propuesto son del 99,08%, 99,34% y 99,21% respectivamente cuando se puede mejorar la tasa de detección de los SQLIA. El método propuesto tiene una mayor precisión, tasa de recuperación	Los resultados experimentales muestran que el algoritmo TFIDF propuesto puede mejorar la tasa de detección de los SQLIA. El método propuesto tiene una mayor precisión, tasa de recuperación
---	------	------------------------	---	--	--	---	--	---	---	--	---

completamos el ITFIDF de 32 n y  
preprocesamien caracteres puntuación  
to de datos. A sensibles, la F en  
continuación, el longitud de la comparació  
conjunto de sentencia SQL n con otros  
datos se divide y la frecuencia métodos  
en conjunto de de palabras similares.  
entrenamiento y clave se toman En el futuro  
conjunto de como puntos nos  
prueba. Al de centraremos  
mismo tiempo, características. en cómo  
el conjunto de mejorar los  
entrenamiento En métodos de  
se utiliza para comparación aprendizaje  
entrenar el con KNN y automático  
modelo de Decision Tree. para  
clasificación El clasificador obtener un  
SVM y el SVM tiene mejor  
conjunto de mayor modelo de  
pruebas se precisión, entrenamien  
utiliza para

verificar el recuperación y to en el  
modelo puntuación. futuro  
generado.

6	2019	Hars h Bhag wani, Rohit Negi, Aneet Kuma r Dutta , Anan d Hand a(B), Nites h Kuma r, and Sand eep	<a href="#">10.</a> <a href="#">10</a> <a href="#">07/</a> <a href="#">97</a> <a href="#">8-</a> <a href="#">3-</a> <a href="#">03</a> <a href="#">0-</a> <a href="#">35</a> <a href="#">86</a> <a href="#">9-</a> <a href="#">3</a> <a href="#">10</a>	Automat ed Classific ation of Web- Applicati on Attacks for Intrusio n Detectio n	Spri nger	Clasific ación autom atizada de ataque s a en aplicac iones s para web lograr para la una detecci ón de precisió n en la detecció n.	detecció n y clasifica ción de ataques basados en registro s para lograr una mayor precisió n en la detecció n.	Detección y clasificación de ataques basados en registros basados en registros para lograr una mayor precisión en la detección, y LDAP. En el caso de los análisis experimento en ataques XSS, basado en el registro funcionamiento, Forest logra la fuera de se propuso la mayor línea división en dos precisión, un utilizando fases: la 99,38%, con varios detección y un con un FPR modelos de clasificación bajo. También aprendizaje basada en en el caso de automático registros fuera los ataques y una	El método propuesto para mejor rendimiento. La tabla 8 muestra los ataques, basados en resultados de implementa mos un enfoque de dos fases, es decir, un análisis basado en el registro fuera de línea utilizando varios modelos de aprendizaje automático y una	Por lo tanto, para detectar estos ataques, implementa mos un enfoque de dos fases, es decir, un análisis basado en el registro fuera de línea utilizando varios modelos de aprendizaje automático y una
---	------	---	---	---	--------------	--	--	---	---	---

---

Kuma  
r  
Shukl  
a

de línea y la implementación en línea. SQLi, Random Forest obtiene mejores resultados con una precisión del 97,91%. El sistema con una precisión del 97,91%. Los clasificadores Random Forest y Gradient Boosting consiguen la mayor precisión del 99,28% para los ataques Path Traversal. implementa ción en línea de la versión de análisis del registro. El sistema de análisis de registros en línea lee las solicitudes HTTP de los archivos de registro cada 15 segundos para clasificarlos en ataques o solicitudes normales.

---

7 2018 Cui, [10.](#) Maliciou Inde Clasific Análisis La Mediante el Basándose en Se  
 B., [15](#) s URL rsci ación de distribución análisis de las los resultados compararon  
 He, [04/](#) detectio enc de trafico de los valores características de las pruebas, tres  
 S., [IJ](#) n with e URLS URL de las de las URL el clasificador algoritmos  
 Yao, [HP](#) feature Ent con malicios característica maliciosas, de árbol de de  
 X., & [CN](#) extractio erpri técnica o a s para de las utilizaron dos decisión se de clasificación  
 Shi, [.20](#) n based ses s de través instancias algoritmos de decisión se y el  
 P. [18.](#) on Ltd. aprend de maliciosas es algoritmos de mostró algoritmo de  
[09](#) machine izaje aprendi diferente de análisis preferible para árbol de  
[43](#) learning autom zaje la de las estadístico su uso en el decisión es  
[67](#) ático automat instancias basados en el algoritmo final el que que  
 ico benignas aprendizaje por de y se incluyó demostró el  
 benignos y el gradientes y en el modelo mejor  
 Una conjunto de extracción de detección. rendimiento.  
 solicitud datos de automática de En la En  
 de URL formación características aplicación comparació  
 puede comparte el basada en un práctica n con los  
 tratarse mismo nivel de umbral aplicación métodos de  
 como un proceso de sigmoidal para práctica, este detección  
 problem con el resolver el sistema se ha tradicionale  
 a de conjunto de problema de la desplegado en s, este

clasifica datos de identificación de línea en sistema  
 ción prueba. En el características empresas de puede  
 binaria proceso de eficaces. Internet muy detectar con  
 binaria construcción conocidas en precisión las  
 en la del modelo de China que se URL  
 que los detección, se especializan maliciosas  
 casos probaron tres en seguridad que tienen  
 positivo algoritmos de de la red. característic  
 s son entrenamient Procesa un as difusas.  
 petición o y enorme El modelo  
 es de seleccionamo número de de  
 URL s el mejor URLs, detección  
 malicios como modelo aproximadame no se basa  
 as o de detección nte 2TB en una sola  
 malicios práctico. diarios. Lo característic  
 as o como modelo hemos a ni en una  
 vulnera de detección probado con lista negra,  
 bles y práctico. una lista de sino que es  
 los URLs un enfoque  
 casos maliciosas en integral que  
 negativo línea utiliza todas

s son  
benigno  
s.

encontradas las  
entre 15 de característic  
octubre de as para el  
2015 y el 29 de aprendizaje  
octubre de automático.  
2015, y mostró Además,  
buen este  
rendimiento. enfoque se  
La precisión es ha  
superior al desarrollado  
98,7% con solo en un  
modestos proyecto  
falsos práctico y  
positivos. desplegado  
en línea  
para  
proteger  
grandes  
volúmenes  
de tráfico  
web.



8	2020	Dhari tri Tripat hy, Rudr arajsi nh Gohil, and Talal Halab i	<a href="#">10.</a> <a href="#">11</a> <a href="#">09/</a> <a href="#">Big</a> <a href="#">Da</a> <a href="#">ta</a> <a href="#">Se</a> <a href="#">cur</a> <a href="#">ity-</a> <a href="#">HP</a> <a href="#">SC</a> : <a href="#">ID</a> <a href="#">S4</a> <a href="#">97</a> <a href="#">24.</a> <a href="#">20</a> <a href="#">20.</a> <a href="#">00</a> <a href="#">03</a> <a href="#">5</a>	Detectin g SQL Injection Attacks in Cloud SaaS using Machine Learnin g	IEE E	Ataque s de inyecci ón SQL	Los intrusos pueden causar graves pérdida s a las instituci ones inyectan do cargas útiles malicios as. Esto afecta a las relacion es de confianz a entre	El método propuesto para detectar ataques SQLi se divide en seis fases principales, definición del problema, recogida y limpieza de datos, ingeniería de característica s, entrenamient o del modelo y evaluación. que son de uso potencial. La visualización del PCA de	Se crean características personalizadas como la longitud y la distribución de bytes, la operación de lectura o escritura, palabra clave y los caracteres no imprimibles, luego se Clasifican las características personalizadas que son de uso potencial. La visualización del PCA de	En el proceso de aprendizaje, la aplicación lee el conjunto de datos de entrenamiento desde un archivo csv y la los datos son utilizados por los el método de aprendizaje del clasificador. A una partir del vector de características generado, se realiza la clasificación. Con 5 de	El aprendizaje automático tiene un enorme potencial en cibersegurid ad. La mayoría de los clasificadore s obtienen precisión cercana al 98%. Un patrón claro es que el clasificador aleatorio
---	------	---	--	---	----------	--	---	---	--	--	---

---

<p>el cliente y el proveedor de servicios.</p>	<p>entradas maliciosas y no maliciosas utilizaron la métrica chi-cuadrado. En la precisión de los modelos aumenta cuando se incorporan más características. La característica de longitud evalúa la longitud de la entrada. La característica de caracteres de puntuación contiene</p>	<p>AdaBoostClassifier, SGD, RandomForest, DecisionTree, Tensorflow Linear classifier, Tensorflow boosted tree, y Deep ANN lograron una precisión del 99,3%, 99,1%, 99,80%, 99,523%, 97,3%, 99,5%, 97,6%, respectivamente. Con 7 características, el</p>	<p>supera a todos los demás en el conjunto de datos y alcanza una precisión del 99,8%. La longitud de la entrada, el número de caracteres de puntuación y el número de bytes distintos son las tres mejores características</p>
--	--	---	---

---

---

número de caracteres de puntuación en una carga que incluye < y '. La característica Byte describe el valor mínimo de byte de los estándares UTF-8 dentro de la entrada.

de sifier, SGD, as RandomForest encontradas , decisionTree, . Las operaciones de escritura de maliciosas son más críticas que las operaciones de lectura.

Tensorflow boosted tree, y Deep ANN las lograron un 99,4%, 99,3%, 99,86%, 99,560%, 97,4%, 99,5%, 97,7% de precisión, respectivamente. Los resultados de clasificación de resultados de

---

---

clasificación de los enfoques de aprendizaje automático se analizados mediante la exactitud, la precisión, la recuperación, la puntuación F1, la sensibilidad y especificidad en la aplicación. Los modelos finales optimizados basados en las métricas que se prueban con

---

---

datos de prueba fuera de la muestra. Los modelos ofrecen una mayor precisión con 10 características que con 5 y 7 características. El clasificador NaiveBayes, como resultado se tiene un 95,8% de que es mayor que el valor anterior.

---

9	2017	Solo mon Ogbo mon Uwag bole Willia m J. Buch anan, Lu Fan	<a href="#">10.</a> <a href="#">23</a> <a href="#">91</a> <a href="#">9/1</a> <a href="#">N</a> <a href="#">M.</a> <a href="#">20</a> <a href="#">17.</a> <a href="#">79</a> <a href="#">87</a> <a href="#">43</a> <a href="#">3</a>	Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention	IEE E	aprendizaje automático aplicado a la Detección y prevención de ataques de inyección SQL	tendencias de conjuntos de datos que contienen datos extraídos de patrones de ataque conocidos, parte de ataques SQLIA muestra la prevalencia de esta forma de ataque y, por lo tanto, la capacidad de	generación de un conjunto de datos que contiene los patrones de ataque conocidos, incluyendo tokens SQL y símbolos presentes en los puntos de inyección. Además, como caso de prueba, construimos una aplicación	Se generó un conjunto de datos que contiene los patrones de ataque conocidos, incluyendo tokens SQL y símbolos presentes en los puntos de inyección. Además, se construyó una aplicación web que espera la lista de palabras del diccionario como variables vectoriales para	Se obtuvo que el algoritmo CM (exactitud: 0,986, precisión: 0,974, recall: 0,997 y puntuación F1: 0,985) con un umbral de 0,5, pero este cálculo se repite en todos los bins de puntuación/ umbrales entre 0,0 y 1 para los valores del gráfico ROC de la Fig. 4. El	En este trabajo se demostró que la aplicación del análisis predictivo a la detección y prevención de SQLIA en el contexto de big data con un excelente resultado que se evalúa empíricamente en la matriz de
---	------	---	---	--	-------	---	--	--	--	--	--

proteger web que demostrar ROC se confusión y  
 la base espera la lista cantidades representa con el gráfico  
 de datos de palabras masivas de una relación de ROC  
 de back- del datos de la tasa de presentados  
 end de diccionario aprendizaje. falsos positivos anteriormen  
 SQLIA como (FPR) en te. Al  
 en la era variables El clasificador (especificidad) comparar  
 del big vectoriales entrenado se frente a la tasa este artículo  
 data para despliega como de verdaderos con los  
 sigue demostrar un servicio web positivos (TPR) trabajos  
 siendo cantidades que se consume en el eje (TPR) existentes,  
 un tema masivas de en una en el eje Y la  
 de datos de aplicación dot (sensibilidad). metodología  
 actualid aprendizaje NET En el eje de propuesta  
 ad. personalizada abscisas, un aquí es  
 que implementa valor más alto funcional en  
 una interfaz de implica un mal un contexto  
 programación rendimiento, de big data,  
 de aplicaciones mientras que lo que falta  
 (API) de proxy en el eje Y un en los  
 web para valor más alto trabajos

interceptar y de 0,986 en el existentes  
predecir con Área Bajo la hasta ahora  
precisión SQLIA Curva (AUC) sobre  
en las alcanzado aquí SQLIA,  
solicitudes web con una según  
y así evitar que puntuación de nuestro  
las peticiones 0,5 indica un conocimient  
web maliciosas excelente o. El trabajo  
lleguen a la base rendimiento en futuro  
de datos base la predicción consiste en  
de datos back- de SQLIA. emplear un  
end protegida. clasificador  
multiclase  
para  
identificar y  
agrupar los  
diferentes  
tipos de  
SQLIA tal y  
como se  
prevén.



1 0	2019	Wen- Lin Chu 1, Chih- Jer Lin 2,* and Ke- Neng Chan g 2	10. 33 90/ p9 21 45 79	Detectio n and Classific ation of Advanc ed Persiste nt Threats and Attacks Using the Support Vector Machine	Appl . Sci	ataque a la red y de piratería informát ica de la informa ción	análisis de componentes principales (PCA) para el muestreo de característica s y la mejora de la eficiencia de la detección	de El método principal utiliza de los datos de auditoría de seguridad y las característic as complejas y dinámicas del comportami ento de las intrusiones, así como la optimización del rendimiento de la detección de APT se	Los resultados de los experimentos muestran que la máquina de vectores de soporte (SVM) tiene la tasa de reconocimient o más alta, alcanzando el 97,22% (para los subdatos entrenados comparan las ventajas y desventajas del uso de los clasificadores para detectar el conjunto de	La gran cantidad de datos de auditoría de seguridad y las característic as complejas y dinámicas del comportami ento de las intrusiones, así como la optimización del rendimiento de la detección de APT se
--------	------	---	--	---	---------------	---	---	--	---	---

---

datos,	el	ha
clasificador		convertido
admite	la	en una
máquina		importante
vectorial,	la	cuestión
clasificación		abierta.
ingenua	de	Según los
Bayes, el árbol		experimento
de decisión y las		s de este
redes		estudio, los
neuronales.		clasificadore
		s SVM-RBF
		y MLP-AS
		(N = 4)
		tienen la
		mejor tasa
		de
		reconocimie
		nto para
		NSL-KDD.

---

---

El uso de PCA para reducir la dimensión no ayudó a la tasa de reconocimiento, pero podría mejorar la velocidad de cálculo. Se recomienda utilizar los clasificadores SVM-RBF o MLP-AS (N = 4) para la detección de un ataque APT.

---

												El uso conjunto de SVM y PCA tiene la ventaja de acelerar el proceso de cálculo.
1	2020	Peng Tang, Weidong Qiu, a,* , Zhen g Huan	10. Detectio n of SQL injection based artificial neural network	ELS EVI ER	inyeccio nes SQL	la detección de inyecciones SQL de inyecciones SQL basado en una red neuronal artificial, consta de tres partes: preprocesami	En primer lugar, se extraen 8 tipos de características relevantes mediante el análisis de una gran cantidad de datos de inyecciones SQL y, a continuación, se	Los resultados de la prueba muestran que el efecto de detección del MLP utilizando la capa oculta de 3 capas es mejor que el MLP de 1 a capa, el MLP de 2 capas y el	Las características separadas en gran medida las sentencias SQL maliciosas de las normales, y ha			

g a,b  
,  
Huiju  
an  
Lian  
a  
,  
Guoz  
hen  
Liu

ento de utilizan muchos LSTM. logrado  
datos, datos reales Detectando buenos  
extracción de para entrenar el simultáneame resultados  
característica modelo de la red nte el mismo de  
s y neuronal. Por número de detección  
entrenamient último se URLs, el MLP mediante el  
o del modelo. comparan los de las 3 capas aprendizaje  
resultados del ocultas tarda de estas  
entrenamiento un 11% más de característic  
del modelo. En tiempo que el as. Sin  
este documento, de 1 capa de embargo, en  
se utilizan MLP, y el la detección  
diversos tiempo de del modelo  
modelos de detección del LSTM, la  
redes LSTM es capacidad  
neuronales para mucho mayor de  
el que el del MLP. reconocimie  
entrenamiento. En En términos nto de  
de tiempo de palabras  
ejecución, con clave y  
el aumento del caracteres

número de específicos  
capas ocultas es pobre, y  
capas ocultas, presta  
el tiempo atención a el  
empleado por reconocimie  
el modelo para nto de la  
detectar un relación  
dato es entre las  
también es palabras  
mayor. Entre clave y los  
ellos, MLP caracteres  
tarda una especiales,  
media de 66,3 así como los  
ms de 1 capa caracteres  
para detectar internos de  
un dato, 74 ms las palabras  
para el MLP de clave. En  
3 capas para Además, se  
de 3 capas adopta el  
para detectar código  
un dato, y ASCII para

37149,4 ms el  
para LSTM procesamiento  
para detectar el inicio de  
un dato. de cadenas en  
datos, lo que el  
demuestra que el experimento  
el tiempo de . En el  
detección de código  
LSTM es ASCII, los  
mucho más caracteres  
largo que el de especiales y  
MLP las letras  
son son  
similares, y  
la distinción  
del tamaño  
numérico no  
es obvia, lo  
que que la  
distinción

del tamaño numérico.

1	2020	Abdul	10.	Encode	IEE	Intrusi	Intrusió	transformar	Este	trabajo	En	la	Utilizando el
2		Jabb	11	d Flow	E	ón en	n en el	las	propone	el	clasificación	libro	de
		ar	09/	Feature		IOT	Internet	característica	método	binaria,	el	códigos,	se
		Siddi	CC	s for			de las	s basadas en	Codebooked	objetivo	es	miden	las
		qui,	NC	Network			Cosas	el flujo en	Encoded	Flow	detectar	desviacione	
		Azze	46	Intrusio				representacio	Features	muestras	s de	los	
		dine	10	n				nes más	(CEFF)	que	anómalas	flujos	de
		Bouk	8.2	Detectio				discriminativa	captura	los	(maliciosas)	tráfico	
		erche	02	n in				s.	patrones	clave	que pueden	(benignos y	
			0.9	Internet					de	las	pertenecer a	maliciosos)	
			04	of					características	diferentes	tipos	respecto a	
				Things					de	flujo	de ataque	y,	esos



---

51	benignas en un	por tanto, tener	patrones
95	libro de códigos discriminativo aprendido de forma no supervisada. El método propuesto se basa en la mejora del modelo Bag-of- Features (BoF) (BoF) en el contexto de los IDS basados en las características del flujo de la red. (en algunos trabajos, modelo BoF se	diferentes características de diferentes. Alrededor del 99,59% de las muestras benignas y el 97,97% de las muestras maliciosas se clasificaron correctamente, con una tasa de falsos positivos de alrededor del 2% y una tasa de falsos negativos del 0,4%. negativo	clave y se registran como característic as transformad as para entrenar un clasificador basado en máquinas de vectores de apoyo. Las evaluacione s experimenta les en dos conjuntos de datos

---

---

denomina también cuantificación vectorial.	del 0,4%. Las puntuaciones de precisión, recuerdo y f1 fueron: 0,9935, 0,9959, 0,9947 (para la clase benigna) y de 0,9872, 0,9797, 0,9834 (para la clase maliciosa), respectivamen te.	realistas recientes demostraro n la eficacia de los CEFF para la detección y de intrusiones, obteniendo de falsos positivos para diferentes tipos de ataques. tipos de ataques.
---	---	--

---

1	2020	Dhari	<a href="#">10.</a>	Detectin	IEE	Ataque	Los	El método	Se	crean	En el proceso	El
3		Tri	<a href="#">11</a>	g SQL	E	s de	intrusos	propuesto	características	de aprendizaje,	aprendizaje	
		Tripat	<a href="#">09/</a>	Injection		inyecci	pueden	para detectar	personalizadas	la aplicación	automático	
		hy,	<a href="#">Big</a>	Attacks		ón	causar	ataques SQLi	como la longitud	lee el conjunto	tiene un	
		Rudr	<a href="#">Da</a>	in Cloud		SQL	graves	se divide en	y la distribución	de datos de	enorme	
		arajsi	<a href="#">ta</a>	SaaS			pérdida	seis fases	de bytes, la	entrenamiento	potencial en	
		nh	<a href="#">Se</a>	using			s a las	principales,	operación de	desde un	cibersegurid	
		Gohil,	<a href="#">cur</a>	Machine			instituci	definición del	lectura o	archivo csv y	ad. La	
		and	<a href="#">ity-</a>	Learnin			ones	problema,	escritura, la	los datos son	mayoría de	
		Talal	<a href="#">HP</a>	g			inyectan	recogida y	palabra clave	utilizados por	los	
		Halab	<a href="#">SC</a>				do	limpieza de	SQL y los	el método de	clasificadore	
		i	<a href="#">:</a>				cargas	datos,	caracteres no	aprendizaje del	s obtienen	
			<a href="#">ID</a>				útiles	ingeniería de	imprimibles,	clasificador. A	una	
			<a href="#">S4</a>				malicios	característica	luego se	partir del vector	precisión	
			<a href="#">97</a>				as. Esto	s,	Clasifican	las de	cercana al	
			<a href="#">24.</a>				afecta a	entrenamient	características	características	98%. Un	
			<a href="#">20</a>				las	o del modelo	personalizadas	generado, se	patrón claro	
			<a href="#">20.</a>				relacion	y evaluación.	que son de uso	realiza la	es que el	
			<a href="#">00</a>				es de		potencial. La	clasificación.	clasificador	
			<a href="#">03</a>				confianz		visualización del	Con 5	de bosque	
			<a href="#">5</a>				a entre		PCA de de las	características,	aleatorio	

el cliente y el proveedor de servicios.

entradas maliciosas y no maliciosas utilizaron la métrica chi-cuadrado. En la precisión de los modelos aumenta cuando se incorporan más características. La característica de longitud evalúa la longitud de la entrada. La característica de caracteres de puntuación contiene

AdaBoostClassifier, SGD, RandomForest, DecisionTree, Tensorflow Linear classifier, Tensorflow boosted tree, y Deep ANN lograron una precisión del 99,3%, 99,1%, 99,80%, 99,523%, 97,3%, 99,5%, 97,6%, respectivamente. Con 7 características, el

supera a todos los demás en el conjunto de datos y alcanza una precisión del 99,8%. La longitud de la entrada, el número de caracteres de puntuación de bytes distintos son las tres mejores características

número de caracteres de puntuación en una carga que incluye La característica Byte describe el valor mínimo de UTF-8 dentro de la entrada.

de RandomForest en , decisionTree, . Las operaciones de escritura de maliciosas son más críticas que las operaciones de lectura.

SGD, as

Tensorflow

Linear classifier, Boosted tree, y Deep ANN

99,4%, 99,3%, 99,86%, 99,560%, 97,4%, 99,5%, 97,7% de precisión, respectivamente. Los resultados de clasificación de resultados de

clasificación de los enfoques de aprendizaje automático se analizados mediante la exactitud, la precisión, la recuperación, la puntuación F1, la sensibilidad y especificidad en la aplicación. Los modelos finales optimizados basados en las métricas que se prueban con

datos de prueba fuera de la muestra. Los modelos ofrecen una mayor precisión con 10 características que con 5 y 7 características. El clasificador NaiveBayes. El resultado es un 95,8% de que es mayor que el valor anterior

1 4	2018	Guozi SUN, Lei ZHU, Huak ang LI Wenj un LI	10. 11 09/ AV SS .20 18. 86 39 33 5	Web attack forensic s based network traffic behavio r charact eristics and URLs	IEE E	Ataque s web	Ataques web basado en las caracter ísticas del comport amiento del tráfico de la red y las URL	El propuesto divide en cuatro partes, como se muestra en la Figura 1: extracción de característica s de tráfico, identificación de anomalías mediante SVM, extracción de característica s energéticas mediante análisis wavelet, reconocimien	Se tráfico basado en el protocolo de red HTTP, los datos de entrenamiento del tráfico, y la atribución de [longitud del paquete HTTP, la longitud de la URL, el número de URI número, la longitud máxima del URI, la longitud media de la parte de valor, el número de parámetros URI, la longitud media de la	Al añadir HMM a la regla de forense de ataque web basado en la combinació n de las característic as del comportami ento del tráfico de la red y las URL. Se obtuvieron buenos resultados en el conjunto de datos experimenta les, lo que	EL modelo forense de ataque web basado en la combinació n de las característic as del comportami ento del tráfico de la red y las URL. Se obtuvieron buenos resultados en el conjunto de datos experimenta les, lo que
--------	------	--	---	--	----------	-----------------	---	--	--	---	---



---

to de ataques parte de Comparamos demostró la  
XSS y de parámetros URI, nuestro eficacia del  
inyección la longitud total método con método. En  
SQL basados de la parte de otros métodos. el siguiente  
en modelo parámetros URI, El experimento paso,  
RF. la IP de origen, demuestra que trataremos  
el puerto de nuestro de encontrar  
origen, la IP de método una manera  
destino, se basado en las de no tener  
toman como una características que  
secuencia de del seleccionar  
características, comportamient manualment  
y cada flujo es o del tráfico de e la función,  
etiquetado. A la red y la URL y utilizar el  
continuación, se propuesto en método de  
utiliza el modelo este trabajo es aprendizaje  
SVM para eficaz para el profundo  
identificar las reconocimient para el  
anomalías. o de ataques análisis  
web y ha forense de  
logrado los ataques

---

---

buenos resultados. XSS y de inyección

Decision tree SQL. El logra una objetivo es precisión de mejorar la 92%, seguido precisión del de HMM con reconocimie un 89% y otros nto y ahorrar métodos con tiempo.

96% de Consulte a precisión [9] para seleccionar esta función.

---

15	2020	Dhari tri Tripat hy, Rudr arajsi nh Gohil, and Talal Halab i	<a href="#">10.</a> <a href="#">11</a> <a href="#">09/</a> <a href="#">Big</a> <a href="#">Da</a> <a href="#">ta</a> <a href="#">Se</a> <a href="#">cur</a> <a href="#">ity-</a> <a href="#">HP</a> <a href="#">SC</a> : <a href="#">ID</a> <a href="#">S4</a> <a href="#">97</a> <a href="#">24.</a> <a href="#">20</a> <a href="#">20.</a> <a href="#">00</a> <a href="#">03</a> <a href="#">5</a>	Detectin g SQL Injection Attacks in Cloud SaaS using Machine Learnin g	IEE E	Ataque s de inyecci ón SQL	Los intrusos pueden causar graves pérdida s a las instituci ones inyectan do cargas útiles malicios as. Esto afecta a las relacion es de confianz a entre	El método propuesto para detectar ataques SQLi se divide en seis fases de bytes, la operación definición del problema, recogida y limpieza de datos, ingeniería de característica luego se Clasifican las características personalizadas que son de uso potencial. La visualización del PCA de de las	Se crean características personalizadas como la longitud y la distribución de bytes, la operación de lectura o escritura, la palabra clave SQL y los caracteres no imprimibles, luego se Clasifican las características personalizadas que son de uso potencial. La visualización del PCA de de las	En el proceso de aprendizaje, la aplicación lee el conjunto de datos de entrenamiento desde un archivo csv y la los datos son utilizados por los el método de aprendizaje del clasificador. A una partir del vector de características generado, se realiza la clasificación. Con 5 de bosque	El aprendizaje automático tiene un enorme potencial en cibersegurid ad. La mayoría de los clasificadore s obtienen precisión cercana al 98%. Un patrón claro es que el clasificador aleatorio
----	------	---	--	---	----------	--	---	---	--	---	---

el cliente y el proveedor de servicios.

entradas maliciosas y no maliciosas utilizaron la métrica chi-cuadrado. En la precisión de los modelos aumenta cuando se incorporan más características. La característica de longitud evalúa la longitud de la entrada. La característica de caracteres de puntuación contiene el AdaBoostClassifier, SGD, RandomForest, DecisionTree, Tensorflow Linear classifier, Tensorflow boosted tree, y Deep ANN lograron una precisión del 99,3%, 99,1%, 99,80%, 99,523%, 97,3%, 99,5%, 97,6%, respectivamente. Con 7 características, el AdaBoostClassifier supera a todos los demás en el conjunto de datos y alcanza una precisión del 99,8%. La longitud de la entrada, el número de caracteres de puntuación de bytes distintos son las tres mejores características.

número de sifier, SGD, as  
caracteres de RandomForest encontradas  
puntuación en , decisionTree, . Las  
una carga útil Tensorflow operaciones  
que incluye < y '. Linear de escritura  
La característica classifier, maliciosas  
Byte mínimo Tensorflow son más  
describe el valor boosted tree, y críticas que  
mínimo de byte Deep ANN las  
de los lograron un operaciones  
estándares 99,4%, 99,3%, de lectura.  
UTF-8 dentro de 99,86%,  
la entrada. 99,560%,  
97,4%, 99,5%,  
97,7% de  
precisión,  
respectivamen  
te. Los  
resultados de  
clasificación de  
resultados de

clasificación de los enfoques de aprendizaje automático se analizados mediante la exactitud, la precisión, la recuperación, la puntuación F1, la sensibilidad y especificidad en la aplicación. Los modelos ofrecen una mayor precisión con 10 características

que con 5 y 7 características.

También evaluamos la relevancia de las

características con el clasificador

Naive Bayes.

El resultado es un 95,8% de que es mayor que el valor anterior