GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DETECCIÓN DE ATAQUES A REDES Y SISTEMAS DE INFORMACIÓN EMPLEANDO TÉCNICAS DE DEEP LEARNING

Autor: Miriam Colino Ruipérez

Director: Miguel Ángel Sanz Bobi

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Detección de ataques a redes y sistemas de información empleando técnicas de Deep Learning

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/2023 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Miriam Colino Ruipérez Fecha: 30/05/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por SANZ BOBI MIGUEL ANGEL - ***6599** el día 30/05/2023 con un certificado emitido por AC FNMT Usuarios

Fdo.: Miguel Ángel Sanz Bobi Fecha: 30/05/2023

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DETECCIÓN DE ATAQUES A REDES Y SISTEMAS DE INFORMACIÓN EMPLEANDO TÉCNICAS DE DEEP LEARNING

Autor: Miriam Colino Ruipérez

Director: Miguel Ángel Sanz Bobi

Madrid

Agradecimientos

A toda mi familia, por todo el apoyo incondicional que me han dado siempre y especialmente en estos años del grado.

A mi director, Miguel Ángel, por su ayuda a lo largo de todo este trabajo.

DETECCIÓN DE ATAQUES A REDES Y SISTEMAS DE INFORMACIÓN EMPLEANDO TÉCNICAS DE DEEP LEARNING

Autor: Colino Ruipérez, Miriam.
Director: Sanz Bobi, Miguel Ángel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto se han construido y entrenado diversos modelos de Machine Learning con el objetivo de detectar tráfico benigno y malicioso generado por ataques en redes de comunicaciones como fuerza bruta web, SQL injection o fuerza bruta XSS. Se han analizado los resultados individuales utilizando distintas métricas y se ha construido un ensemble con el fin de robustecer la detección de anomalías en el tráfico observado.

Palabras clave: Machine Learning, NIDS, hiperparámetros, precisión, ensemble

1. Introducción

Actualmente, el uso de las redes de telecomunicaciones que soportan las distintas aplicaciones y recursos informáticos que el mundo utiliza ha sufrido un incremento exponencial. Esto hace que se haya vuelto una tarea fundamental el asegurar la confidencialidad, fiabilidad, integridad y disponibilidad de la información que fluye a través de las redes de telecomunicación. Para ello, innumerables empresas han desarrollado diversas tecnologías de protección de estas infraestructuras de telecomunicación para poder detectar, analizar y responder a posibles ataques protegiendo dichas infraestructuras. Sin embargo, a la par que las empresas incrementan sus esfuerzos en protegerse, los ciberdelincuentes utilizan técnicas más avanzadas para atacar a dichas redes. Aquí es donde se vuelve crucial la investigación, el desarrollo y la implementación de técnicas avanzadas de protección, siendo las técnicas de Machine Learning, y más concretamente Deep Learning las más destacadas.

2. Definición del Proyecto

Este trabajo persigue los siguientes objetivos:

- Explorar distintas fuentes desde donde extraer datos que contengan las características y la clasificación de distintos escenarios de ataques maliciosos y benignos para poder analizarlos con distintos modelos.
- Aprendizaje y uso de varios modelos de Machine y Deep Learning con el objetivo de construir y entrenar distintos modelos: SGD, Regresión Logística, Árbol de decisión, Random Forest, KNN y Red neuronal.
- Propuesta de modelos de identificación y clasificación de ataques intrusos y optimización de sus hiperparámetros para mejorar la precisión y eficiencia.
- Validación de las clasificaciones obtenidas mediante métricas como la precisión general, precisión por clase, F1-score, curvas ROC y matrices de confusión.

- Propuesta de entorno de colaboración conjunta de los modelos desarrollados. De esta forma, se consigue una compensación de los errores y puede conseguirse un modelo más robusto para la detección de intrusiones.
- Comparación del rendimiento de todos los modelos para observar sus fortalezas y debilidades para futuros proyectos.

3. Descripción del modelo

Los datos que se manejarán, obtenidos de [1] [2],contienen diversidad de características de ataques tipo Benigno, Fuerza bruta – Web, Fuerza bruta -XSS y SQL injection. Primero, se realizará un filtrado de datos al dataset y se aplicará la técnica de SMOTE para balancear las clases creando muestras sintéticas. Con los datos ya procesados, se realizará la separación entre un conjunto de entrenamiento y conjunto de prueba. Se procederá a la construcción y entrenamiento de los modelos, además de un ajuste de sus hiperparámetros. Los modelos multiclase que se han elegido para este trabajo han sido: SGD (Descenso de gradiente estocástico), Regresión Logística, Árbol de decisión, Random Forest, KNN (K vecinos más cercanos) y Red neuronal. Todos ellos son testeados posteriormente con el conjunto de prueba, obteniendo así las predicciones. Se utilizarán varias métricas de evaluación del rendimiento de los modelos como la matriz de confusión, curva Roc, F1 score y precisión por clases. Con los modelos con mejor precisión para la clasificación de los ataques se realizará un ensemble, técnica que consiste en la combinación de predicciones para obtener un modelo más robusto. Se ha demostrado en varios trabajos [3] [4] que, con esta estrategia, se consigue una alta efectividad. Finalmente, se compararán y evaluarán los resultados obtenidos entre todos los modelos construidos para obtener el algoritmo con mejor capacidad de clasificación de ataques.

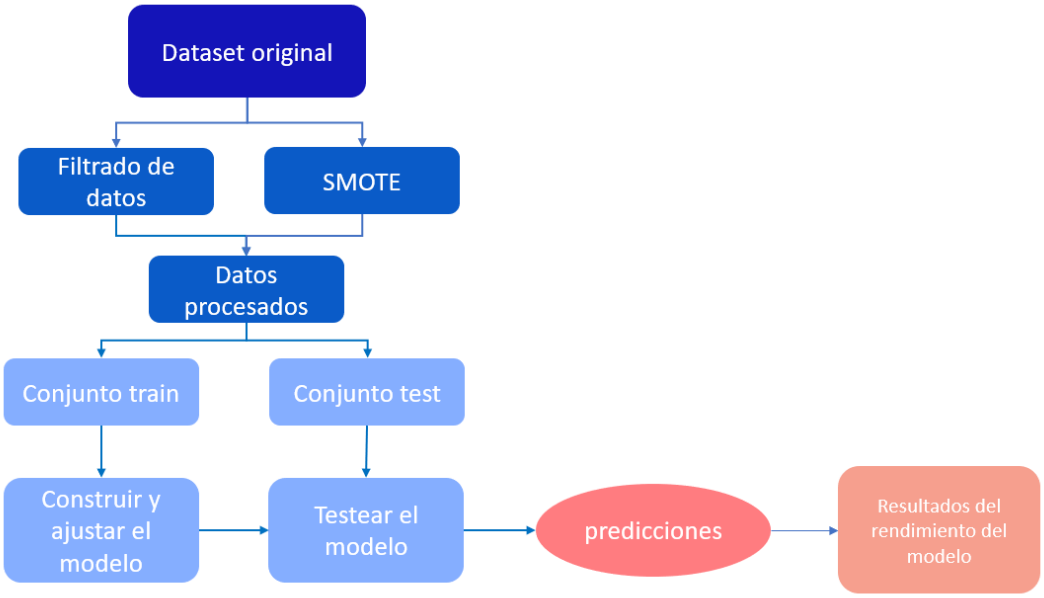


Ilustración 1 Diagrama desarrollo del proyecto

4. Resultados

Los resultados obtenidos se muestran en las siguientes ilustraciones. En la primera ilustración, se muestra la precisión de cada modelo con el tiempo de ejecución en horas. En la segunda ilustración, se muestra la precisión por clase (ataque) de los tres modelos con mejor rendimiento para la clasificación, junto a la precisión media y la del ensemble. De esta forma, se demuestra la efectividad del ensemble al haber creado un modelo más robusto para la detección de los cuatro tipos de ataques que se estudiaban.

Modelo	Precisión (%)	Tiempo (horas)
Regresión Logística	83.135122	0.062919
SGD	80.662324	0.007545
Árbol decisión	97.813603	0.018899
Random Forest	97.299555	0.012715
KNN	98.970487	1.649002
Red neuronal	90.550625	1.291269
Ensemble	98.455946	0.190283

Ilustración 2 Resultados obtenidos en el proyecto (1)

Tipo de Clase	Precisión Árbol decisión	Precisión Random Forest	Precisión KNN	Precisión Media	Precisión Ensemble	¿Conseguida mejora con Ensemble?
Bening	99.989286	99.137701	99.995657	99.707548	99.994209	Sí
Brute Force - Web	94.138296	99.661078	98.413504	97.404293	98.595344	Sí
Brute force - XSS	99.707707	99.871478	99.489164	99.689450	99.761145	Sí
SQL injection	97.660119	90.651863	97.984080	95.432021	95.600507	Sí

Ilustración 3 Resultados obtenidos en el proyecto (2)

5. Conclusiones

A la vista de los resultados obtenidos, se concluye que se han cumplido todos los objetivos propuestos. La construcción de un ensemble con los tres mejores modelos : KNN, Árbol de decisión y Random Forest, ha demostrado una alta eficacia, al tener una precisión mayor (98,455946 %) que dos de ellos. Se ha observado que KNN es el modelo con mejor rendimiento en la clasificación de estos datos de ataques en tráfico de red con una precisión de 98,97047%.

6. Referencias

[1] «CSE-CIC-IDS2018 on AWS,» Canadian Institue for Cybersecurity, [En línea]. Available: https://www.unb.ca/cic/datasets/ids-2018.html . [Último acceso: 30 mayo 2023].

[2] «A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018),» aws, [En línea]. Available: https://registry.opendata.aws/cse-cic-ids2018/ . [Último acceso: 30 mayo 2023].

[3] A. M. M. D. V. S. S. Saikat Das*, «DDoS Intrusion Detection through Machine,» Department of Computer Science The University of Memphis, 2019. [En línea]. Available: https://drsaikatdas.com/papers/ddos_ensemble.pdf . [Último acceso: 30 mayo 2023].

[4] M. M. S. U. N. Ngamba Thockchom, «A novel ensemble learning-based model for network intrusion detection,» Complex Intell. Syst, 3 Abril 2023. [En línea]. Available: https://link.springer.com/article/10.1007/s40747-023-01013-7 . [Último acceso: 30 mayo 2023].

DETECTION OF NETWORK AND INFORMATION SYSTEMS ATTACKS USING DEEP LEARNING TECHNIQUES

Author: Colino Ruipérez, Miriam.
Supervisor: Sanz Bobi, Miguel Ángel.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this project, several Machine Learning models have been built and trained to detect benign and malicious traffic generated by attacks in communication networks such as Web brute force, SQL injection or XSS brute force. Individual results have been evaluated using different metrics and an ensemble has been built in order to strengthen the detection of anomalies in the observed traffic.

Keywords: Machine Learning, NIDS, hyperparameters, accuracy, ensemble

1. Introduction

Currently, the use of telecommunication networks that support the different applications and computing resources that the world uses has increased exponentially. This has made it a fundamental task to ensure the confidentiality, reliability, integrity, and availability of the information flowing through telecommunication networks. To this end, countless companies have developed various technologies to protect these telecommunication infrastructures in order to detect, analyze and respond to possible attacks by protecting these infrastructures. However, as companies increase their efforts to protect themselves, cybercriminals are using more advanced techniques to attack these networks. This is where research, development and implementation of advanced protection techniques becomes crucial, with Machine Learning techniques, and more specifically Deep Learning, being the most prominent.

2. Definition of the project

This work pursues the following objectives:

- Explore different sources from which to extract data containing the characteristics and classification of different malicious and benign attack scenarios in order to analyze them with different models.
- Learning and use of several Machine and Deep Learning models in order to build and train different models: SGD, Logistic Regression, Decision Tree, Random Forest, KNN and Neural Network.
- Proposal of intruder attack identification and classification models and optimization of their hyperparameters to improve accuracy and efficiency.
- Validation of the classifications obtained using metrics such as overall accuracy, class accuracy, F1-score, ROC curves and confusion matrices.
- Proposal for a collaboration environment for the models developed. In this way, error compensation is achieved and a more robust model for intrusion detection can be achieved.

- Comparison of the performance of all models to observe their strengths and weaknesses for future projects.

3. Description of the model

The data to be handled obtained from [1] [2] contains a variety of characteristics of Benign, Web brute force, XSS brute force and SQL injection attacks. First, a data filtering will be performed on the dataset and the SMOTE technique will be applied to balance the classes creating synthetic samples. With the data already processed, the separation between a training set and a test set will be performed. We will proceed to the construction and training of the models, in addition to an adjustment of their hyperparameters. The multiclass models that have been chosen for this work have been: SGD (Stochastic Gradient Descent), Logistic Regression, Decision Tree, Random Forest, KNN (K-nearest neighbors) and Neural Network. All of them are subsequently tested with the test set, thus obtaining the predictions. Several metrics will be used to evaluate the performance of the models such as confusion matrix, Roc curve, F1 score and class accuracy. With the models with the best accuracy for the classification of attacks, an ensemble will be performed, a technique that consists of combining predictions to obtain a more robust model. Works like [3] [4] have demonstrated the efficacy of this strategy. Finally, the results obtained among all the models built will be compared and evaluated to obtain the algorithm with the best attack classification capacity.

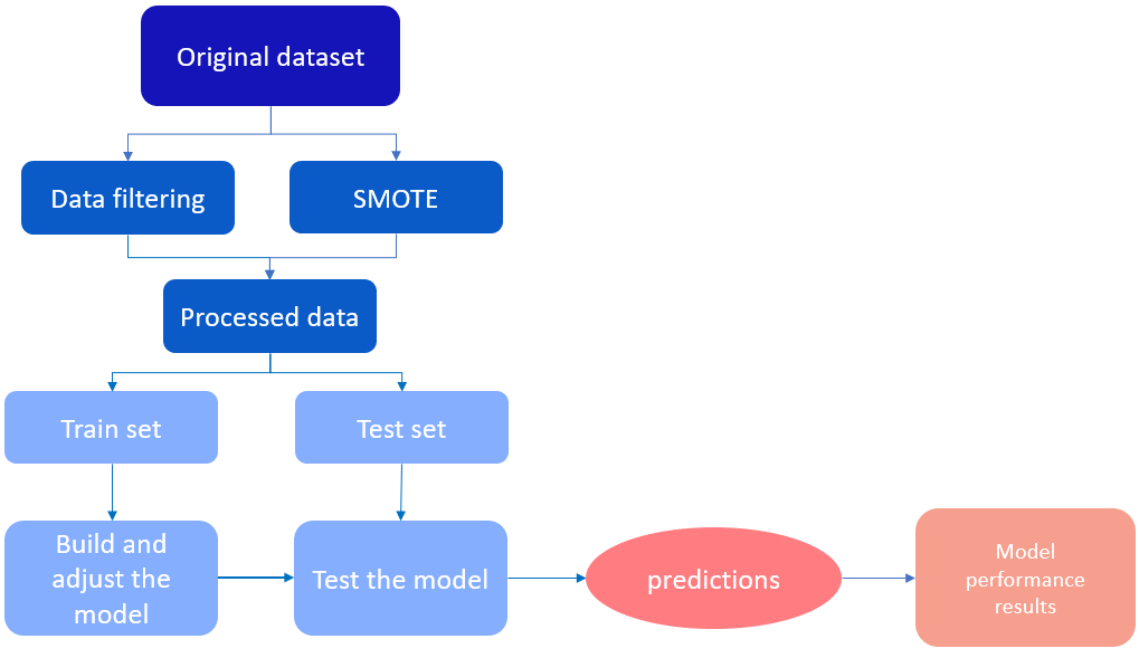


Ilustración 4 Project development diagram

4. Results

The results obtained in the project are shown in the following illustrations. In the first illustration, the accuracy of each model is shown with the execution time in hours. In the second illustration, the accuracy per class (attack) of the three best performing models for classification is displayed, along with the average accuracy and the ensemble's accuracy. This demonstrates the effectiveness of the ensemble by creating a more robust model for the detection of the four types of attacks studied.

Model	Accuracy (%)	Time (hours)
Logistic Regression	83.135122	0.062919
SGD	80.662324	0.007545
Decision Tree	97.813603	0.018899
Random Forest	97.299555	0.012715
KNN	98.970487	1.649002
Neuronal Network	90.550625	1.291269
Ensemble	98.455946	0.190283

Ilustración 5 Results obtained in the project (1)

Type of attack (class)	Decision Tree accuracy	Random Forest accuracy	KNN accuracy	Average accuracy	Ensemble accuracy	Improvement achieved with Ensemble?
Bening	99.989286	99.137701	99.995657	99.707548	99.994209	Yes
Brute Force - Web	94.138296	99.661078	98.413504	97.404293	98.595344	Yes
Brute force - XSS	99.707707	99.871478	99.489164	99.689450	99.761145	Yes
SQL injection	97.660119	90.651863	97.984080	95.432021	95.600507	Yes

Ilustración 6 Results obtained in the project (2)

5. Conclusions

In view of the results obtained, it is concluded that all the proposed objectives have been met. The construction of an ensemble with the three best models, namely KNN, Decision Tree and Random Forest, has shown a high efficiency, having a higher accuracy (98,455946 %) than two of them. It has been observed that KNN is the model with the best performance in the classification of these network traffic attack data with an accuracy of 98.97047%.

6. References

[1] «CSE-CIC-IDS2018 on AWS,» Canadian Institue for Cybersecurity, [En línea]. Available: https://www.unb.ca/cic/datasets/ids-2018.html . [Último acceso: 30 mayo 2023].

[2] «A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018),» aws, [En línea]. Available: https://registry.opendata.aws/cse-cic-ids2018/ . [Último acceso: 30 mayo 2023].

[3] A. M. M. D. V. S. S. Saikat Das*, «DDoS Intrusion Detection through Machine,» Department of Computer Science The University of Memphis, 2019. [En línea]. Available: https://drsaikatdas.com/papers/ddos_ensemble.pdf . [Último acceso: 30 mayo 2023].

[4] M. M. S. U. N. Ngamba Thockchom, «A novel ensemble learning-based model for network intrusion detection,» Complex Intell. Syst, 3 Abril 2023. [En línea]. Available: https://link.springer.com/article/10.1007/s40747-023-01013-7 . [Último acceso: 30 mayo 2023].

Índice de la memoria

Índice de figuras

Índice de abreviaturas

- ML: Machine Learning
- DL: Deep Learning
- IDS: Sistema de detección de intrusiones
- XSS: Ataque Cross-site scripting
- SGD: Algoritmo Descenso de gradiente estocástico
- KNN: Método de los k-vecinos más próximos

Capítulo 1. INTRODUCCIÓN

Actualmente, el uso de las redes de telecomunicaciones que soportan las distintas aplicaciones y recursos informáticos que el mundo utiliza ha sufrido un incremento exponencial. Todos los datos tanto a nivel personal como a nivel empresarial se transmiten a través de dichas redes. Esto las ha convertido en el objetivo prioritario de los ciberdelincuentes para poder robar de ellas dichos datos con distintos fines: económicos, geopolíticos, activismo hacking, etc. En muchos casos lo han conseguido, comprometiendo la confidencialidad de dichos datos y exigiendo grandes sumas de dinero por ellos, así como grandes perjuicios operativos a los usuarios. Esto hace que se haya vuelto una tarea fundamental el asegurar la confidencialidad, integridad y disponibilidad de la información que fluye a través de las redes de telecomunicación. Para ello, innumerables empresas han desarrollado diversas tecnologías de protección de estas infraestructuras de telecomunicación para poder detectar, analizar y responder ante posibles ataques protegiendo dichas infraestructuras. Sin embargo, a la par que las empresas incrementan sus esfuerzos en protegerse, los ciberdelincuentes utilizan técnicas más avanzadas para atacar a dichas redes. Aquí es donde se vuelve crucial la investigación, el desarrollo y la implementación de técnicas avanzadas de protección, siendo las técnicas de Machine Learning, y más concretamente Deep Learning las más destacadas. El Deep Learning o aprendizaje profundo es una red neuronal artificial estructurada en capas o niveles jerárquicos. Se puede asemejar al funcionamiento y estructura del cerebro humano y es una de las bases de la inteligencia artificial. Algunos ejemplos de este tipo de Machine Learning son los asistentes virtuales, los coches autónomos, clasificación de imágenes o reconocimiento de voz, entre otros. Para el entrenamiento de los algoritmos, se usarán datos públicos existentes, para aplicar un benchmarking de los resultados obtenidos y los futuros que se obtengan usando diversos métodos de análisis de datos.

1.1 MOTIVACIÓN DEL PROYECTO

Este proyecto está motivado por indagar y mejorar una estrategia de seguridad muy usada en la actualidad, distinto a un firewall que pueda aplicarse a todo el mundo tecnológico. Esta estrategia, llamada NIDS (Network Intrusion Detection System) es un software de seguridad fundamental para que se tenga capacidad de reacción en caso de ataque y es una de las principales líneas de defensas ante intrusos de los dispositivos tecnológicos. Monitoriza el tráfico de red y para ello, compara información de ataques que se tenga. Ataques hay de muchos tipos, unos ejemplos pueden ser ataques de denegación de servicio, escaneo de puertos, ataques de fuerza bruta o Fuzzing. Algunos sistemas comerciales son Snort o BlackICE.

La tecnología IDS/IPS (Intrusion Detection/Prevention System), que puede observarse en la ilustración a continuación [1], se está incorporando actualmente en los firewalls, pasándose estos a llamar Next Generation Firewalls. Empresas como Fortinet y Palo Alto los han desarrollado. A estos cortafuegos se les han añadido diversos algoritmos de Machine Learning y en concreto, de Deep Learning para detección de patrones de tráfico de red. Estos algoritmos pueden usar distintos paradigmas de aprendizaje como lo son el aprendizaje supervisado y no supervisado o el aprendizaje por refuerzo. Con esto se persigue robustecer la seguridad ante posibles amenazas.

Con este trabajo se persigue añadir un aporte a este área de inteligencia artificial con el uso de las nuevas técnicas que persiguen el fortalecimiento de la Ciberseguridad.

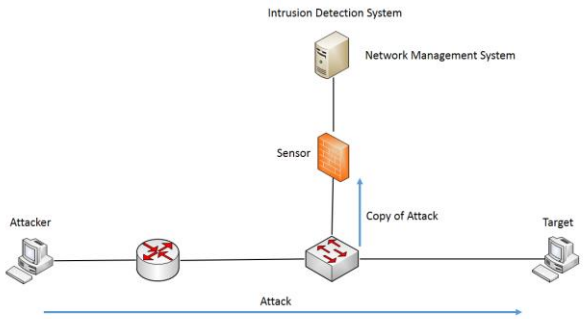


Ilustración 7 Intrusion Detection System

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este trabajo se han usado las siguientes tecnologías:

Lenguaje de programación Python: Además de ser la forma de programar más popular para emplear en Machine Learning, es fácil de implementar, tiene un alto rendimiento, eficiencia. Además, cuenta con varias librerías que se pueden aplicar para estas técnicas como Scikit-learn, Matplotlib, Pandas y Statsmodel, que serán utilizadas en este proyecto, entre otras.

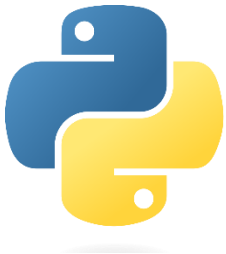


Ilustración 8 Logo de Python

TensorFlow: Se trata de una librería de código libre para Machine Learning (ML). Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. Permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos. [2]

Ilustración 9 Logo de TensorFlow

Jupyter Notebook: aplicación cliente-servidor lanzada en 2015 por la organización sin ánimo de lucro Proyecto Jupyter. Permite crear y compartir documentos web en formato JSON que siguen un esquema versionado y una lista ordenada de celdas de entrada y de salida. Estas celdas albergan, entre otras cosas, código, texto (en formato Markdown), fórmulas matemáticas y ecuaciones, o también contenido multimedia (Rich Media). El programa se ejecuta desde la aplicación web cliente que funciona en cualquier navegador estándar**.** [3]

Ilustración 10 Logo jupyter

Excel: Para el manejo de los datos, tanto de las fuentes como los futuros obtenidos.

Capítulo 3. ESTADO DE LA CUESTIÓN

Deep Learning es una método con múltiples aplicaciones en el mundo de la Ciberseguridad, entre ellas, están la detección de intrusos (IDS), analizar amenazas contra TPV móviles o para evitar amenazas Zero-Day.

Entre los diversos métodos de clasificación de ML aplicados para la detección de intrusiones en redes (NIDS) se encuentran los árboles de decisión, Random Forest, Naive Bayes y Multi layer Percepton. Los métodos de agrupamiento son del tipo k-means clustering o fuzzy clustering.

En el área de detección de intrusiones en la red, se han investigado y publicado una variedad ingente trabajos. Existen diversas publicaciones sobre detección de amenazas en redes IoT con ML, como por ejemplo el trabajo [4], en el cual se construyen y comparan modelos de Máquina de Vector de Soporte, Regresión Logística, árboles de decisión y red neuronal convolucional. El trabajo [5] analiza los modelos más comúnmente usados en ML y detección de ataques (SVM, RF, NB y DT) y observa sus rendimientos para clasificar tanto de forma binaria (ataque maligno o normal) o como multiclase, para detectar el tipo de ataque como DoS, Probe, Privilege o Access. Algoritmos como AdaBoost, Multilayer Perceptron, el clasificador cuadrático (QDA) y Naive Bayes, entre otros, fueron evaluados en el proyecto [6], utilizando el nivel de importancia de las características de Random forest para decidir cuáles usar en el resto de los modelos.

Se han usado técnicas basadas en el muestreo con Máquina de Vector de Soporte de Mínimos Cuadrados (LS-SVM) usando para la validación KDD-99, Redes de Creencia Profunda (DBN) y modelos basados en la red neuronal convolucional (CNN) con un método de reducción de características. Otros trabajos han consistido en utilizar una red neuronal profunda de alimentación hacia adelante (FFDNN), consiguiendo en el sistema una alta precisión. [7]

Este proyecto se basará en un trabajo de la Colorado Mesa University [8], en el cual se exploraba las capacidades de varios Deep-Learning frameworks para detectar y clasificar tráfico de intrusión a redes. En este, se partía de varios dataset csv publicados en 2018 que consistían en tráfico de red benigno o malicioso generados por ataques como fuerza bruta, DoS, Bot, Web e Infiltración. Con matrices de confusión se podía distinguir el número de muestras clasificadas correctamente en cada categoría. Se analizó la precisión del uso de distintos frameworks: fast.ai, Keras-TensorFlow y Keras-Theano, resultando la más precisa fast.ai siendo en la mayoría de los casos 99% precisa.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Existen diversos trabajos que exploran distintas técnicas de Machine Learning para detectar anomalías en el tráfico de red. No obstante, debido a la velocidad de evolución de los ataques y amenazas, es necesario seguir investigando y robusteciendo las capacidades de detección de los modelos. Además, el tamaño del tráfico global aumenta de manera exponencial por lo que es imprescindible contar con medidas de control y prevención cada vez mejores con el fin de salvaguardar la información que se transmite. El año en que se publica este trabajo está marcando un momento decisivo para la inteligencia artificial, ya que se están produciendo avances significativos que podrían sin duda, transformar la forma en que vivimos y trabajamos. Debido a este auge, es fundamental continuar desarrollando modelos con el objetivo de aprovechar su potencial.

En el área de detección de ataques a redes, los trabajos e investigaciones se han enfocado en gran medida en los algoritmos tradicionales de Árboles de decisión o Random forest, como el trabajo publicado [9], en el que se utiliza además vectores de soporte y XGBoost para clasificar ataques de un conjunto de datos IoT hacia dos dispositivos a través de una red inalámbrica en una smart home. Otro tipo de artículos se han dirigido a estudiar el desempeño de modelos como Ada Boost y MLP (Perceptron multicapa) junto a Random Forest, como el publicado [10] en el cual se usaban estos algoritmos y los datos SNMP-MIB, una combinación del protocolo de monitorización de redes SNMP y la base de información de administración MIB, y probar su efectividad en la detección de ataques a redes.

Sin embargo, para añadir un enfoque más amplio, en este trabajo se ha estudiado y entrenado más modelos como Regresión logística, SGD (Descenso de Gradientes Estocástico) o KNN (K vecinos más próximos), con el objetivo de comparar el rendimiento de cada uno usando las mismas métricas para cada uno y observar sus fortalezas y debilidades al clasificar cada clase. Considerar un mayor número de modelos es de gran ayuda para encontrar el más

correcto para los datos que se están analizando en cada caso. Además, debido a su creciente uso y utilidad, este trabajo propone un modelo de red neuronal, lo cual supone una técnica más avanzada para comparar las ventajas que aporta con los demás algoritmos utilizados en ML.

En este trabajo se construirá un ensemble una vez obtenidas las precisiones de los modelos. Esta técnica consiste en combinar las predicciones de los modelos con mejor rendimiento. Este enfoque permite obtener una mejor precisión que la individual de algunos de los modelos en cuestión al aprovechar las fortalezas y debilidades de cada uno al combinarlos. En la detección de ataques no existe una solución única para todos los casos, por lo que la utilización de un modelo ensemble puede ser de gran ayuda. Trabajos como los publicados [11] [12] demuestran la eficacia de esta técnica para conseguir un modelo más robusto para la detección de intrusiones.

Este desarrollo de modelos con altas precisiones puede ser de gran utilidad para empresas que busquen proteger su información y la de sus clientes ante intrusiones, prever futuros ataques y mejorar sus sistemas de protección.

4.2 OBJETIVOS

Lo que se quiere perseguir con este proyecto son los siguientes objetivos:

- Explorar distintas fuentes desde donde extraer datos que contengan las características y la clasificación de distintos escenarios de ataques maliciosos y benignos para poder analizarlos con distintos modelos.
- Aprendizaje y uso de varios modelos de Machine y Deep Learning con el objetivo de construir y entrenar distintos modelos: SGD, Regresión Logística, Árbol de decisión, Random Forest, KNN y Red neuronal.
- Propuesta de modelos de identificación y clasificación de ataques intrusos y optimización de sus hiperparámetros para mejorar la precisión y eficiencia.
- Validación de las clasificaciones obtenidas mediante métricas como la precisión general, precisión por clase, F1-score, curvas ROC y matrices de confusión.
- Propuesta de entorno de colaboración conjunta de los modelos desarrollados. De esta forma, se consigue una compensación de los errores y puede conseguirse un modelo más robusto para la detección de intrusiones.
- Comparación del rendimiento de todos los modelos para observar sus fortalezas y debilidades para futuros proyectos.

4.3 METODOLOGÍA

Para la gestión de los proyectos software existen distintas metodologías a adoptar. Este trabajo seguirá una metodología Agile (Ilustración 11 [13]). Con este enfoque de tipo iterativo, el proyecto se divide en etapas denominadas "sprints". Tras la realización de cada sprint se observan las mejoras que podrían emplearse para el siguiente. Aplicándolo a este proyecto, si durante el entrenamiento de los modelos se van encontrando mejoras ya sea en el ajuste de hiperparámetros o en la optimización de los procesos se aplicarán posteriormente en los modelos restantes. Esto conlleva una mayor eficiencia y rapidez, además de flexibilidad en los cambios que se realicen durante el proyecto.

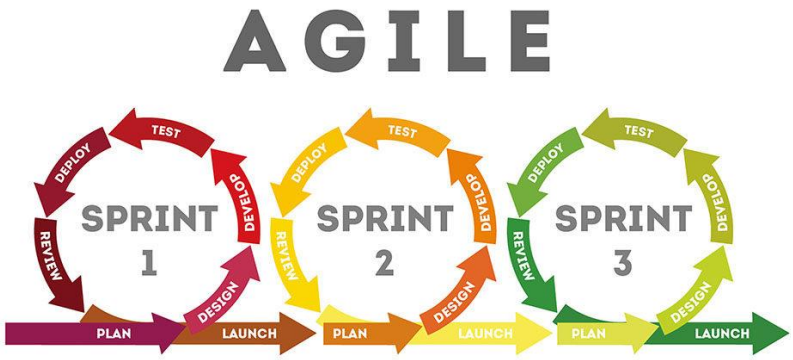


Ilustración 11 Metodología Agile

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

La planificación de este trabajo se refleja en el Diagrama de Gantt (Anexo II). En él, se muestra cómo se han gestionado las partes en las que se divide el proyecto a lo largo del periodo de tiempo en el que se ha desarrollado.

Con respecto a la estimación económica, el lenguaje de programación que se ha utilizado es Python, el cual es gratuito al ser de código abierto. Esto hace que no haya costes económicos en el proyecto software.
Sin embargo, si se quisiera acelerar el proceso de entrenamiento de los modelos se podría emplear herramientas como una unidad de procesamiento gráfico (GPU), lo que supondría un coste adicional. Asimismo, se podría almacenar el proyecto en la nube, con servicios como Amazon Web Service (AWS), los cuales tienen un sistema de pago y ofrecen una multitud de herramientas para los proyectos basados en Machine Learning.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

El sistema que se desarrolló en este proyecto se muestra en la siguiente ilustración:

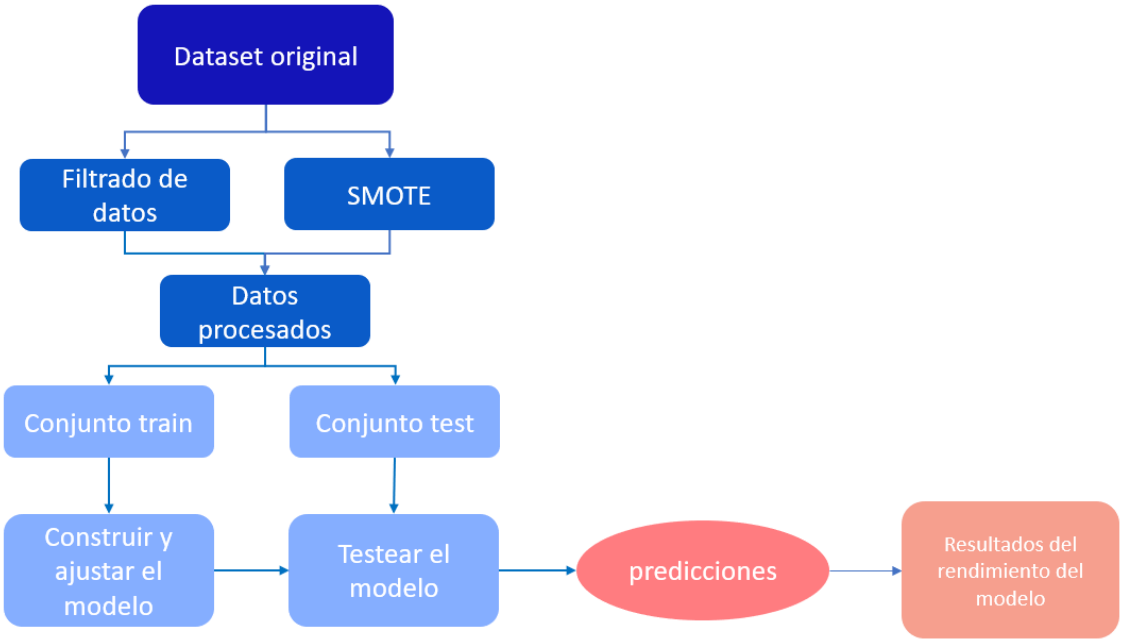


Ilustración 12 Sistema desarrollado

5.1 FUENTE DE DATOS Y EXPLORACIÓN INICIAL DE LOS MISMOS

Para llevar a cabo este trabajo se buscaron conjuntos de datos que fuesen conocidos y usados por otros autores ya que se deseaba contar con una fuente diversa y fiable de información. Finalmente se decidió usar un conjunto de datos de la Colorado Mesa University. Este dataset, denominado CSE-CIC-IDS2018, se obtuvo del Canadian Institute of Cybersecurity. [14] [15]

Las características presentes en el archivo Excel original que se mantuvieron para este proyecto son las siguientes:

Num.	**Nombre**	**Descripción**
1	Dst Port	Puerto de destino del paquete
2	Protocol	Protocolo utilizado por el paquete
3	Timestamp	Marca de tiempo del paquete
4	Flow Duration	Duración del flujo
5	Tot Fwd Pkts	Número total de paquetes en la dirección de envío
6	Tot Bwd Pkts	Número total de paquetes en la dirección de retorno
7	TotLen Fwd Pkts	Tamaño total del paquete en la dirección de envío
8	TotLen Bwd Pkts	Tamaño total del paquete en la dirección de retorno
9	Fwd Pkt Len Max	Máximo tamaño del paquete en la dirección de envío.
10	Fwd Pkt Len Min	Mínimo tamaño del paquete en la dirección de envío.
11	Fwd Pkt Len Mean	Tamaño medio del paquete en la dirección de envío
12	Fwd Pkt Len Std	Desviación estándar del tamaño del paquete en la dirección de envío

13	Bwd Pkt Len Max	Máximo tamaño del paquete en la dirección de retorno
14	Bwd Pkt Len Min	Mínimo tamaño del paquete en la dirección de retorno
15	Bwd Pkt Len Mean	Tamaño medio del paquete en la dirección de retorno
16	Bwd Pkt Len Std	Desviación estándar del tamaño del paquete en la dirección de retorno
17	Flow Pkts/s	Tasa de paquetes de flujo, es decir, número de paquetes transferidos por segundo
18	Flow IAT Mean	Tiempo medio entre dos flujos
19	Flow IAT Std	Desviación estándar del tiempo entre dos flujos
20	Flow IAT Max	Tiempo máximo entre dos flujos
21	Flow IAT Min	Tiempo mínimo entre dos flujos
22	Fwd IAT Tot	Tiempo total entre dos paquetes enviados en la dirección de envío.
23	Fwd IAT Mean	Tiempo medio entre dos paquetes enviados en dirección de envío
24	Fwd IAT Std	Desviación estándar del tiempo entre dos paquetes enviados en la dirección de envío
25	Fwd IAT Max	Tiempo máximo entre dos paquetes enviados en la dirección de envío
26	Fwd IAT Min	Tiempo mínimo entre dos paquetes enviados en la dirección de envío

27	Bwd IAT Tot:	Tiempo total entre dos paquetes enviados en la dirección de retorno
28	Bwd IAT Mean	Tiempo medio entre dos paquetes enviados en la dirección de retorno
29	Bwd IAT Std	Desviación estándar del tiempo entre dos paquetes enviados en la dirección de retorno
30	Bwd IAT Max	Tiempo máximo entre dos paquetes enviados en la dirección de retorno.
31	Bwd IAT Min	Tiempo mínimo entre dos paquetes enviados en la dirección de retorno
32	Subflow Fwd Pkts	Número medio de paquetes en un subflujo en la dirección de envío
33	Subflow Fwd Byts	Número medio de bytes en un subflujo en la dirección de envío.
34	Subflow Bwd Pkts	Número medio de paquetes en un subflujo en la dirección de retorno
35	Subflow Bwd Byts	Número medio de bytes en un subflujo en la dirección de retorno
36	Init Fwd Win Byts	Número de bytes enviados en ventana inicial en la dirección de envío
37	Init Bwd Win Byts	Número de bytes enviados en la ventana inicial en la dirección de retorno
38	Fwd Act Data Pkts	Número de paquetes con al menos 1 byte de carga útil de datos TCP en la dirección de envío
39	Fwd Seg Size Min	Tamaño mínimo del segmento observado en la dirección de envío
40	Active Mean	Tiempo medio que un flujo estuvo activo antes de pasar a inactivo
41	Active Std	Desviación estándar del tiempo que un flujo estuvo

		activo antes de pasar a inactivo
42	Active Max	Tiempo máximo que un flujo ha estado activo antes de pasar a inactivo
43	Active Min	Tiempo mínimo que un flujo ha estado activo antes de pasar a inactivo
44	Idle Mean	Tiempo medio de inactividad de un flujo antes de activarse
45	Idle Std	Desviación estándar del tiempo que un flujo estuvo inactivo antes de activarse
46	Idle Max	Tiempo máximo de inactividad de un flujo antes de activarse
47	Idle Min	Tiempo mínimo de inactividad de un flujo antes de activarse

Los tipos de ataques que se analizaron a partir del dataset son los siguientes:

- Ataque benigno.
- Ataque Fuerza Bruta - Web. Consiste en utilizar la fuerza bruta, es decir, probar todas las combinaciones posibles, sobre contraseñas, claves y nombres de usuario de un sitio web.
- Ataque Fuerza Bruta - XSS. Se trata de un ataque que se aprovecha de la vulnerabilidad XSS (Cross-site scripting) para insertar programas maliciosos y de esta forma, obtener el control del usuario o robar información .
- Ataque SQL Injection. Este ataque consiste en insertar un script SQL malicioso en un campo de entrada de texto con la intención de manipular o destruir una base de datos.

5.1.1 FILTRADO DE DATOS

El filtrado de datos es un proceso muy común en el análisis de datos para eliminar los datos innecesarios, cambiar los formatos de valores si se requiere o reducir el tamaño del conjunto para facilitar su utilización.

En este proyecto, se empezó con un filtrado de la información del dataset. Los datos se leyeron de un CSV mediante la biblioteca pandas. Se convirtieron las etiquetas de la columna 'Label', es decir, el tipo de ataque a números enteros. Así los nombres 'Bening', 'Brute Force -Web', 'Brute Force -XSS' y 'SQL injection', se reemplazaron por las clases 0, 1, 2 y 3 respectivamente. Se eliminaron las filas con valores NaN (Not a Number). Se separaron los datos en dos variables X e Y, la primera conteniendo todas las columnas (características) y la segunda que contiene la columna 'Label'.

5.1.2 SMOTE

Una vez obtenido los conjuntos X y Y, se detectó que el número de muestras de cada clase 'Label' era muy desigual. Una posible solución para corregirlo es utilizando la técnica SMOTE.

SMOTE (Synthetic Minority Over-sampling Technique) es una técnica de sobremuestreo utilizada en el procesamiento de datos para tratar la clase desbalanceada en un conjunto de datos de clasificación. Esto sucede cuando una de las clases tiene muchos más elementos que otra. Si se aplica un clasificador sin considerar este problema se podría obtener una predicción errónea de la clase minoritaria. SMOTE crea nuevos elementos sintéticos de esta clase minoritaria a partir de elementos existentes de dicha clase permitiendo que el modelo de aprendizaje automático presente un rendimiento mejor.

Se aplicó, por tanto, SMOTE a los conjuntos X e Y, y se comprobó que el número de muestras de las clases se igualaron:

```
N° de muestras clase '0' en Y antes de SMOTE: 1048009
N° de muestras clase '1' en Y antes de SMOTE: 362
N° de muestras clase '2' en Y antes de SMOTE: 151
N° de muestras clase '3' en Y antes de SMOTE: 53
```

```
N° de muestras clase '0' en Y después de SMOTE: 1048009
N° de muestras clase '1' en Y después de SMOTE: 1048009
N° de muestras clase '2' en Y después de SMOTE: 1048009
N° de muestras clase '3' en Y después de SMOTE: 1048009
```

5.1.3 SEPARACIÓN DATOS EN CONJUNTOS TRAIN Y TEST

Una vez con los datos preparados para su uso, es necesario hacer una separación en dos conjuntos para su análisis: conjunto de entrenamiento y conjunto de prueba. En este trabajo, se asignó el 67% de los datos para el conjunto de entrenamiento y el 33% restante para el de prueba. Una vez hecha la separación, se entrena y ajusta el modelo con el conjunto de entrenamiento, para después analizar cómo se comporta el modelo con datos nuevos, con los datos de prueba que aún no ha procesado.

Además, se estandarizaron las características tanto de entrenamiento como de prueba con la clase StandardScaler() para que todas las características tuvieran una escala similar. La media se hace 0 y la varianza de los datos se ajusta a 1. Se le suele denominar a esta técnica "normalización" de los datos.

```
N° de muestras clase '0' en Y_train_sm_df después de split: 702245
N° de muestras clase '1' en Y_train_sm_df después de split: 702707
N° de muestras clase '2' en Y_train_sm_df después de split: 701378
N° de muestras clase '3' en Y_train_sm_df después de split: 702334

N° de muestras clase '0' en Y_test_sm_df después de split: 345764
N° de muestras clase '1' en Y_test_sm_df después de split: 345302
N° de muestras clase '2' en Y_test_sm_df después de split: 346631
N° de muestras clase '3' en Y_test_sm_df después de split: 345675
```

5.2 MODELOS

Se construyeron diversos modelos de Machine y Deep Learning.

En el proceso de entrenamiento de cada modelo, se buscó una alta precisión, evitar el sobreajuste y prevenir la complejidad del modelo mediante el ajuste y optimización de los hiperparámetros correspondientes de cada caso estudiado.

5.2.1 CRITERIOS DE ANÁLISIS DE LOS MODELOS

Para analizar sus rendimientos se utilizaron distintas herramientas de análisis de modelos de clasificación como la matriz de confusión, el informe de clasificación o la curva ROC.

La matriz de confusión consiste en una tabla que muestra la cantidad de predicciones correctas e incorrectas realizadas por el modelo para cada clase. La diagonal principal está compuesta por las predicciones que acertó el modelo, tanto verdaderos positivos como verdaderos negativos, mientras que la diagonal secundaria está formada por los fallos del modelo en su predicción, estos son, los falsos positivos y falsos negativos.

El informe de clasificación resume el rendimiento del modelo y muestra la precisión, además de otras métricas como recall, F1-score y support, todas ellas para cada una de las clases. El recall es la proporción de verdaderos positivos sobre el total de verdaderos positivos y falsos negativos. F1-score es la medida ponderada de la precisión y el recall. Support indica el número de muestras de cada clase del conjunto de datos de prueba.

Por otro lado, la curva ROC grafica la tasa de verdaderos positivos en función de los falsos negativos en diferentes umbrales de decisión. El puntaje más relevante es el AUC (Área bajo la curva). Este número indica cuán bueno es el desempeño del modelo entre 0 y 1. Un valor de 1 significa una clasificación perfecta y seria la situación ideal. Un puntaje de 0.5 señala que el modelo clasifica aleatoriamente y no tiene capacidad de distinción de clases. A continuación, se muestra una visualización de ello. [16]

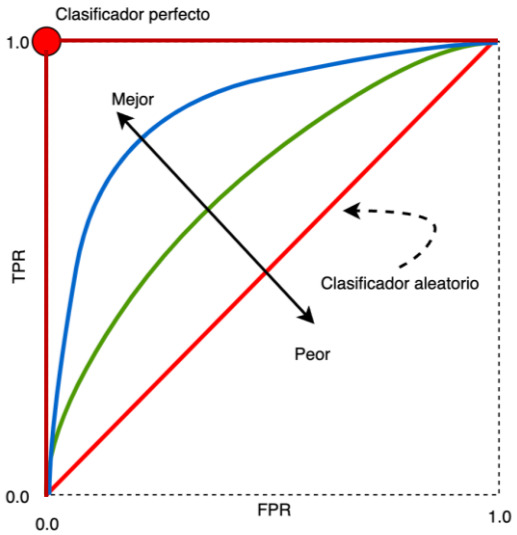


Ilustración 13 Curva ROC explicación

También se graficó la precisión por clase para cada modelo en un gráfico de barras. Esta representación permite comparar de forma visual y rápida los niveles de precisión para cada una de las clases.

5.2.2 REGRESIÓN LOGÍSTICA

La Regresión Logística es un algoritmo de clasificación utilizado para modelar la relación entre una variable de resultado binaria y una o más variables explicativas continuas o categóricas. Es uno de los algoritmos de ML más utilizados, ya que presenta ventajas como la velocidad y la simplicidad.

En este proyecto se utilizará la Regresión Logística Multiclase, ya que se manejan más de dos clases. Con ella, se puede predecir la probabilidad de pertenencia a cada una de las clases. Se utilizan técnicas como la codificación One vs All (uno contra todos) o One vs One (uno contra uno). Entre las ventajas de la regresión logística se encuentran la velocidad y la simplicidad.

En la siguiente ilustración [17], se muestra el diagrama de la Regresión Logística Multiclase. Las características de entrada se transforman en los datos z lineales, para luego aplicarles la función de activación 'softmax' y calcular sus probabilidades. La clase predicha para una entrada se obtiene en función de la probabilidad más alta.

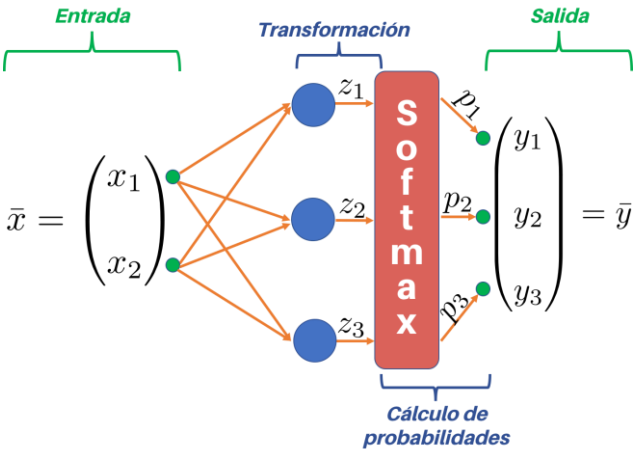


Ilustración 14 Diagrama Regresión multiclase

5.2.2.1 Resultados ajustes de los hiperparámetros

El parámetro 'random_state' representa la semilla que controla la aleatoriedad en varias partes del proceso de entrenamiento de un modelo, como en SMOTE, en la separación de datasets de train y test o al definir los algoritmos de ML. El valor que se utilizará para todos los modelos de este trabajo será 42 ya que es el mayormente utilizado en modelos de ML. La importancia de ser un valor fijo reside en utilizar el mismo número en cada ejecución para asegurar los mismos valores aleatorios si se cambia algún otro hiperparámetro.

Se probaron tanto el método de optimización de coeficientes "L-BFGS" como el solver "saga" para comparar sus resultados.

Además, el valor del número máximo de iteraciones también se modificó para observar su impacto en el modelo.

random_state	multi_class	max_iter	solver	tiempo (horas)	precision
42	multinomial	100	lbfgs	0.020517	79.569342
42	multinomial	300	lbfgs	0.062919	83.135122
42	multinomial	500	lbfgs	0.097137	83.085027
42	multinomial	1000	lbfgs	0.196798	82.994740
42	multinomial	100	saga	0.129947	74.457774
42	multinomial	300	saga	0.395091	76.734964
42	multinomial	500	saga	0.592345	77.709972
42	multinomial	1000	saga	1.169887	79.023791

Ilustración 15 Resultados evaluación hiperparámetros Regresión Logística

Se eligió el modelo entrenado con el método L-BFGS y 300 iteraciones, ya que se obtuvo la mayor precisión con esta combinación.

5.2.2.2 Métricas de evaluación del modelo y gráficas

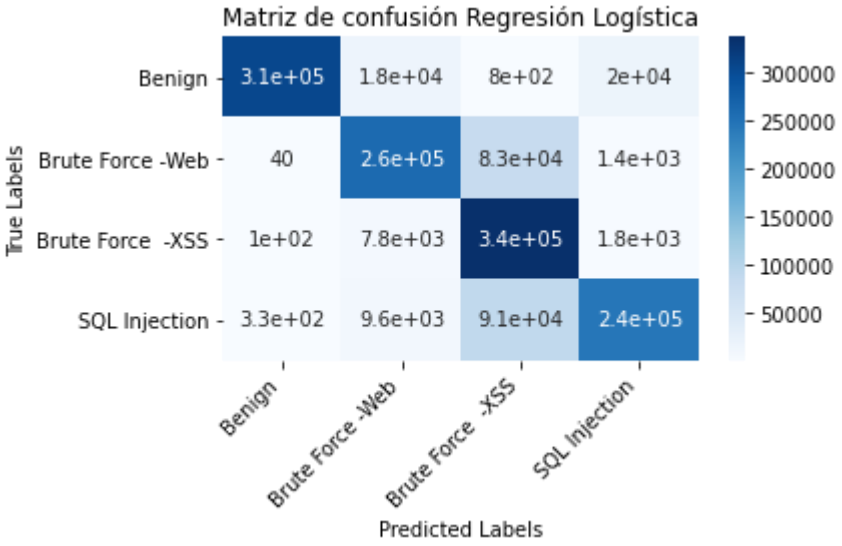


Ilustración 16 Matriz de confusión Regresión Logística

	precision	recall	f1-score	support
Benign	1.00	0.89	0.94	345764
Brute Force -Web	0.88	0.76	0.81	345302
Brute Force -XSS	0.66	0.97	0.79	346631
SQL Injection	0.91	0.71	0.80	345675
accuracy			0.83	1383372
macro avg	0.86	0.83	0.83	1383372
weighted avg	0.86	0.83	0.83	1383372

Ilustración 17 Informe de clasificación Regresión Logística

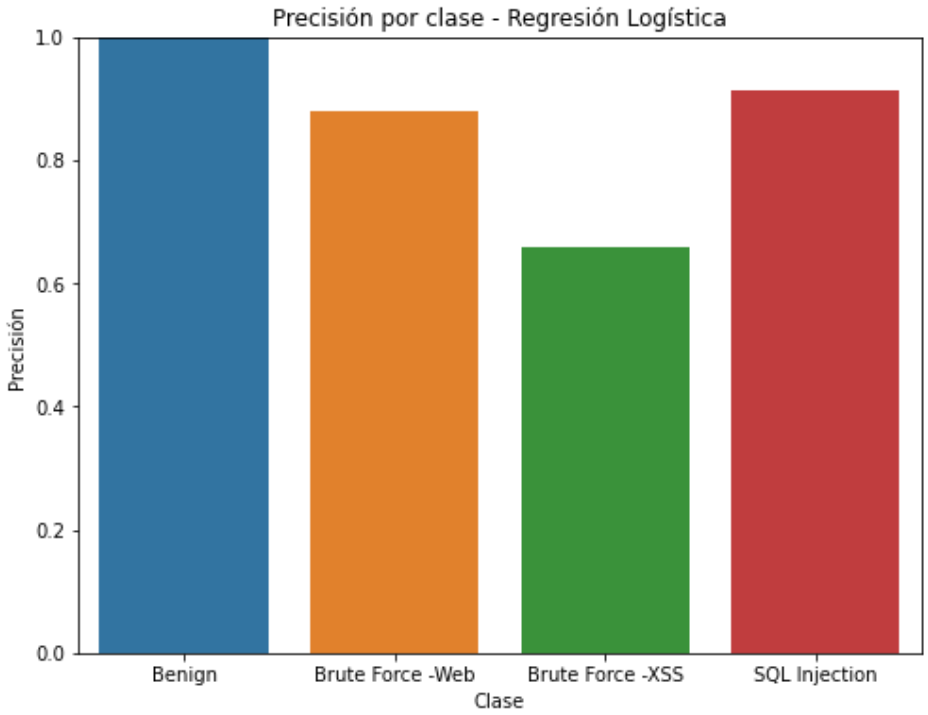


Ilustración 18 Precisión por clase Regresión Logística

En la matriz de confusión se observa que el modelo ha predicho correctamente 307351 muestras de la clase 0 (Ataque beningo), 260817 de la clase 1 (Ataque Brute Force -Web) , 336956 de la clase 2 (Ataque Brute Force -XSS) y 244944 de la clase 3 (Ataque SQL Injection). El reporte de clasificación indica que la precisión correspondiente a la clase 0 es del 100%, de la clase 1 del 88%, de la clase 2 el 66% y de la clase 3, 91%. La precisión ponderada es del 86% , al igual que el promedio ponderado de precisión, f1-score y recall. En general, los resultados indican que el modelo tiene un rendimiento bueno en la clasificación de las clases, excepto de la clase 2 donde se aprecia una menor precisión. Sin embargo, para considerarlo un modelo excelente en la clasificación de ataques se espera que tenga una precisión global mayor al 95%. Esta cifra denota una capacidad buena del modelo de identificar correctamente la mayoría de los ataques, y por tanto, el margen de error es bajo.

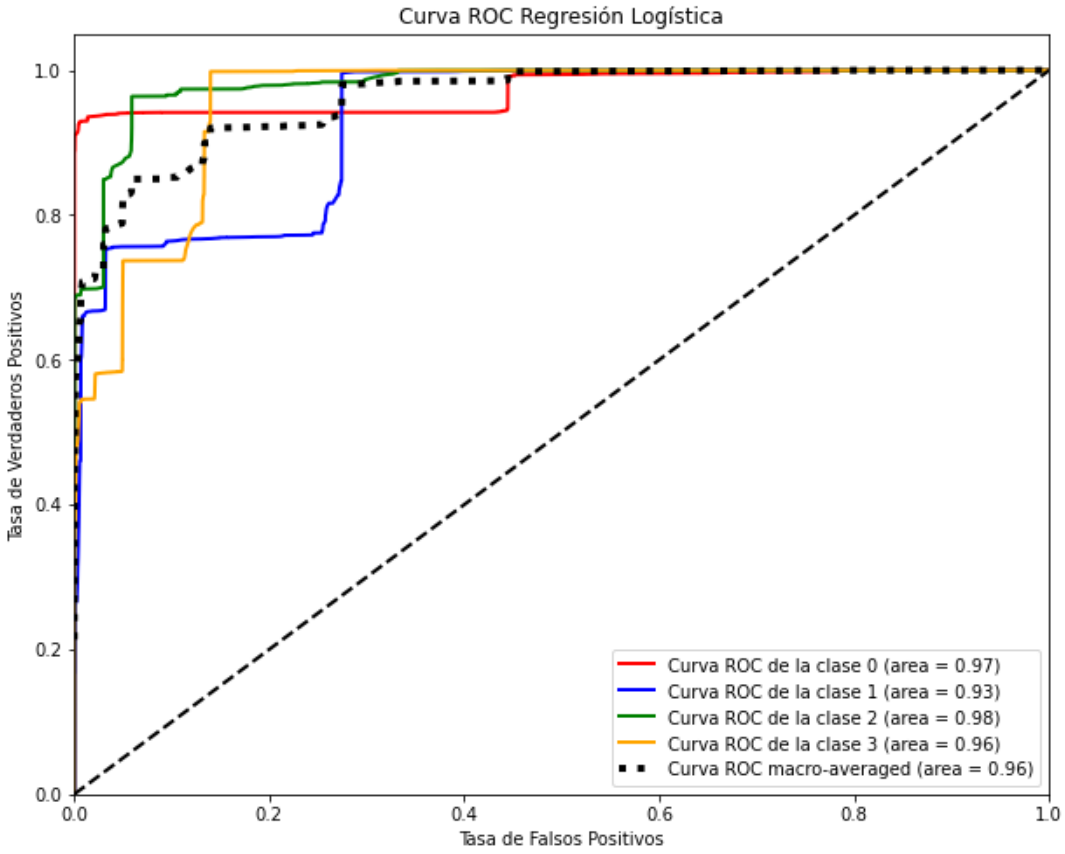


Ilustración 19 Curva ROC Regresión Logística

Como se aprecia en la curva ROC, las AUC (áreas bajo la curva) son altas para todas las clases. Además, la curva macro-average, que corresponde al promedio de todas es de 0.96 indicando un buen rendimiento en general.

5.2.3 SGD

El Descenso de Gradientes Estocástico es un algoritmo de optimización muy común en la práctica de aprendizaje automático, especialmente en el entrenamiento de redes neuronales debido a su eficiencia y simplicidad. Utiliza una muestra aleatoria de los datos en cada iteración con el objetivo de reducir el error o pérdidas calculando el gradiente de la función de pérdida y ajustando los parámetros del modelo. De esta forma, se previene el sobreajuste del modelo al usar distintos subconjuntos en cada iteración. Es un algoritmo de gran utilidad si se manejan grandes cantidades de datos. Entre las ventajas de utilizar este modelo se encuentran la escalabilidad y la facilidad de implementación.

5.2.3.1 Resultados ajustes de los hiperparámetros

Para el entrenamiento del modelo SGD, se probaron la función de pérdida "hinge" y "log", ambas utilizadas para problemas multiclase. La función "hinge" es comúnmente utilizada también para modelos SVM (Máquinas de vectores de soporte). La función "Log" (Logarithmic loss), también llamada "Cross-entropy loss" mide la entropía cruzada del error entre dos distribuciones de probabilidad. La entropía cruzada de un modelo perfecto sería cercana a 0. [18]

random_state	loss function	precisión	tiempo (horas)
42	hinge	80.662324	0.007545
42	log	76.842599	0.008505

Ilustración 20 Resultados evaluación hiperparámetros SGD

Se eligió el modelo entrenado con la función hinge, ya que se obtuvo más precisión con esta.

5.2.3.2 Métricas de evaluación del modelo y gráficas

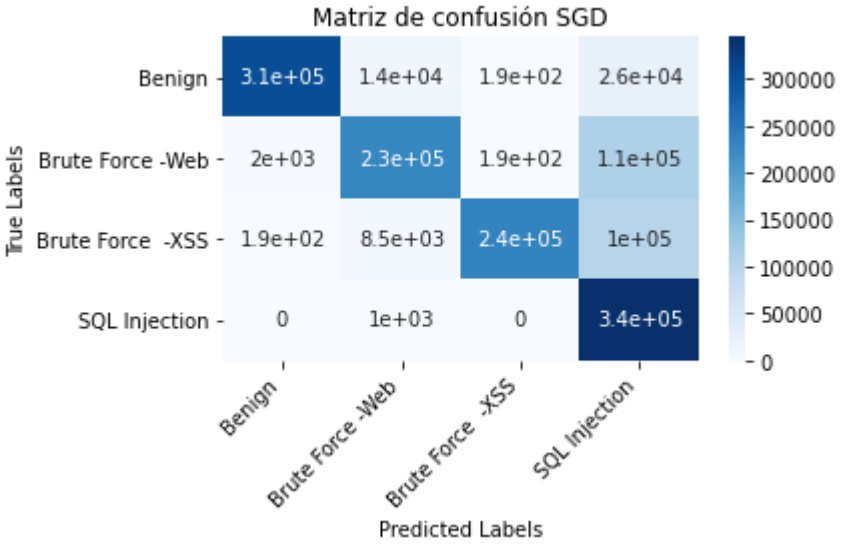


Ilustración 21 Matriz de confusión SGD

	precision	recall	f1-score	support
Benign	0.99	0.88	0.94	345764
Brute Force -Web	0.91	0.67	0.77	345302
Brute Force -XSS	1.00	0.68	0.81	346631
SQL Injection	0.59	1.00	0.74	345675
accuracy			0.81	1383372
macro avg	0.87	0.81	0.81	1383372
weighted avg	0.87	0.81	0.81	1383372

Ilustración 22 Informe de clasificación SGD

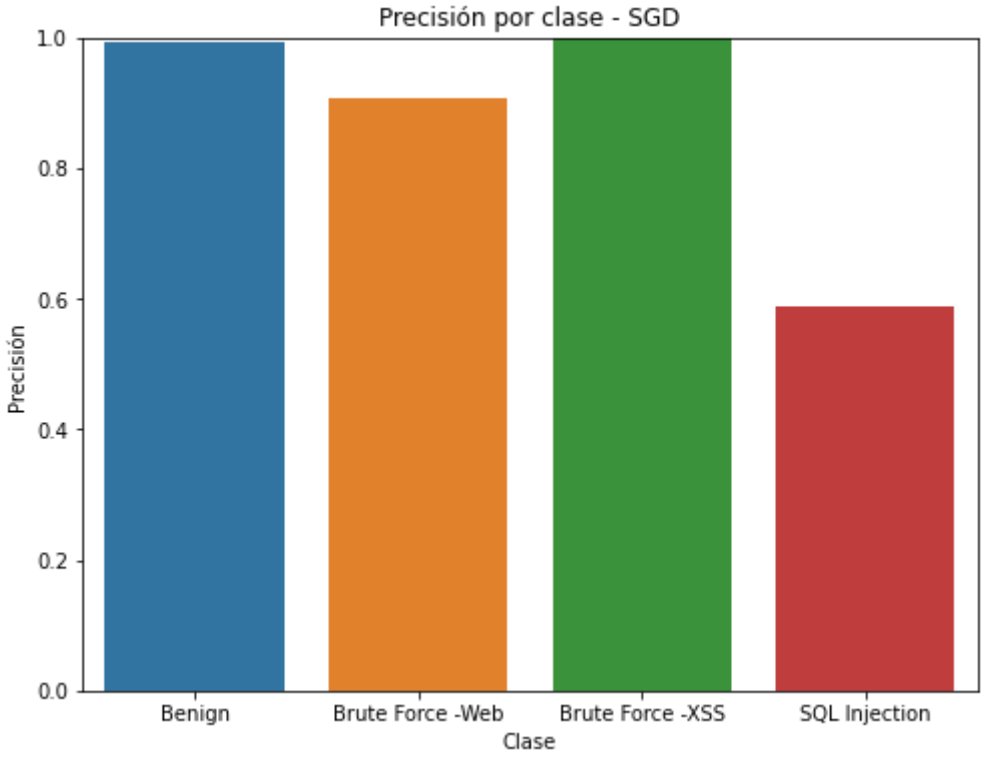


Ilustración 23 Precisión por clase SGD

En la matriz de confusión se observa que el modelo ha predicho correctamente 305681 muestras de la clase 'Benign', 230400 de la clase 'Brute Force -Web', 235111 de la clase 'Brute Force -XSS' y 344668 de la clase 'SQL Injection'. El reporte de clasificación indica que la precisión correspondiente a la clase 0 es del 99%, de la clase 1 del 91%, de la clase 2 el 100% y de la clase 3, 59%. La precisión ponderada es del 87 % , al igual que el promedio ponderado de precisión, f1-score y recall. Estos resultados sugieren que el modelo falla más clasificando la clase 3 en comparación con las demás clases, en la que muestra un mejor nivel de rendimiento.

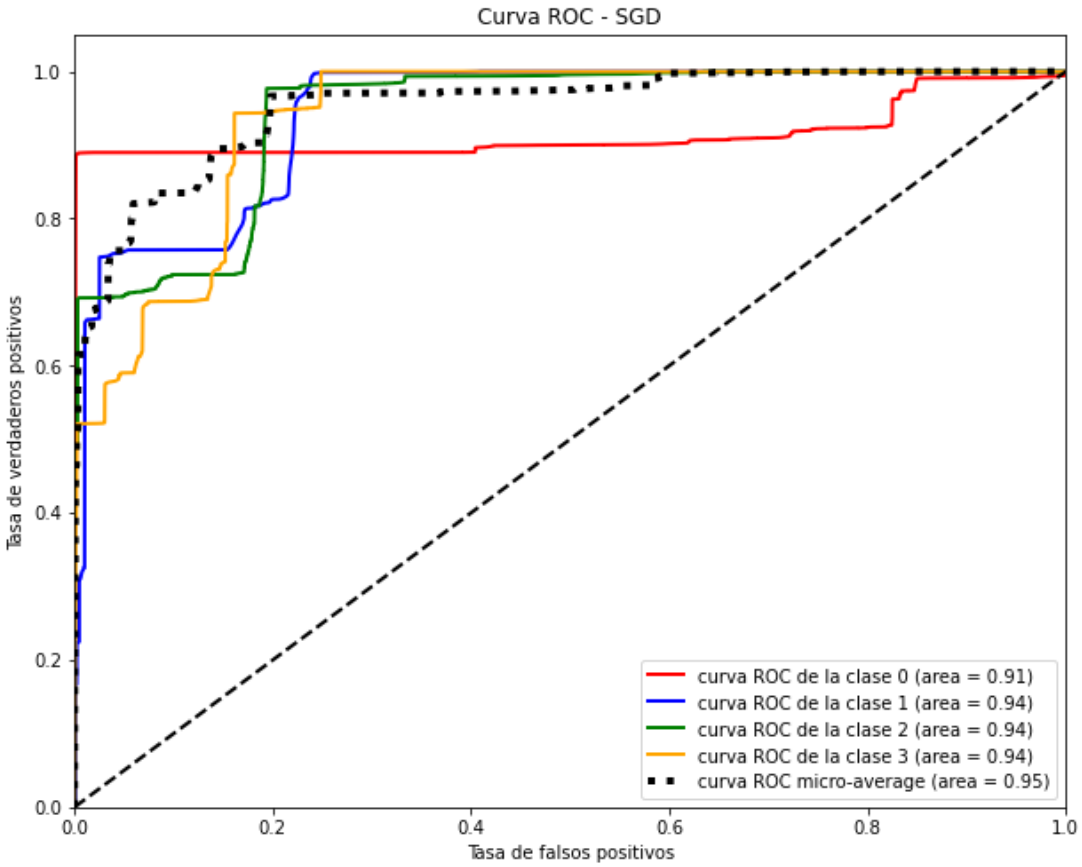


Ilustración 24 Curva ROC SGD

De forma parecida a la Regresión Logística, los valores de las áreas bajo la curva son altos, lo que indica un buen rendimiento en general. La AUC promedio es 0.95, lo que sugiere una buena capacidad de clasificación del modelo.

5.2.4 ÁRBOL DE DECISIÓN

El árbol de decisión es uno de los algoritmos más utilizados en Machine Learning tanto para tareas de regresión como de clasificación. Se representa como una estructura en forma de árbol con nodos y ramas. Cada ramificación representa una decisión en función de la elección tomada inicialmente. Cada nodo interno indica las características a considerar en una decisión y los nodos finales son los resultados de la decisión. Algunas ventajas que supone utilizar árboles de decisión son la versatilidad y el ser no paramétricos.

5.2.4.1 Resultados ajustes de los hiperparámetros

Uno de los parámetros a ajustar en la clase "DecisionTreeClassifier" es la máxima profundidad del árbol, que representa la cantidad de nodos del camino más largo desde la raíz a una de las hojas. Se fue variando hasta obtener una precisión mayor al 95%. La precisión se obtuvo con la clase "score" de la clase "DecisionTreeClassifier".

En la siguiente gráfica se observa la evolución de la precisión del árbol de decisión en función de la máxima profundidad del árbol. Como en max_depth =12 se estabiliza la precisión, se escogió este valor.

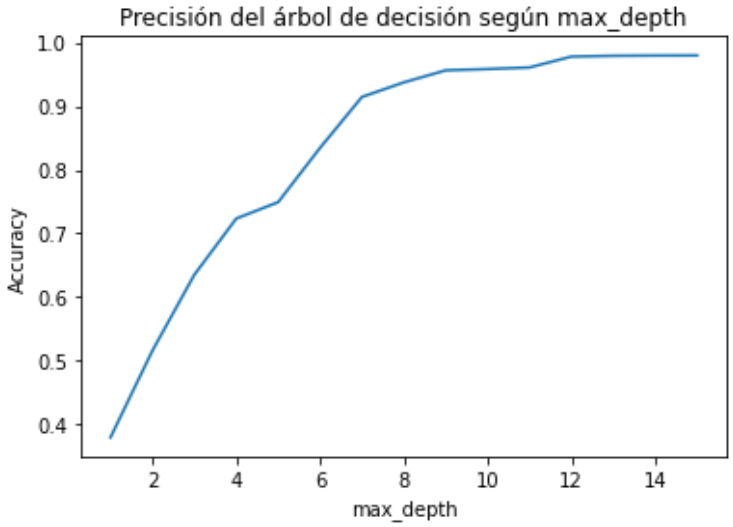


Ilustración 25 Precisión según máx profundidad Árbol de decisión

max_depth	random state	precisión	tiempo (horas)
1	42	0.379213	0.002858
2	42	0.515584	0.005125
3	42	0.634934	0.007678
4	42	0.723350	0.010047
5	42	0.749135	0.012308
6	42	0.834496	0.013338
7	42	0.914647	0.015244
8	42	0.937578	0.015929
9	42	0.956524	0.016637
10	42	0.958561	0.017540
11	42	0.961064	0.018376
12	42	0.978136	0.018899
13	42	0.979377	0.019339
14	42	0.979792	0.019506
15	42	0.979905	0.019620

Ilustración 26 Resultados evaluación hiperparámetros Árbol de decisión

A continuación, se muestra una imagen del árbol completo de 12 niveles y una de sus hojas.

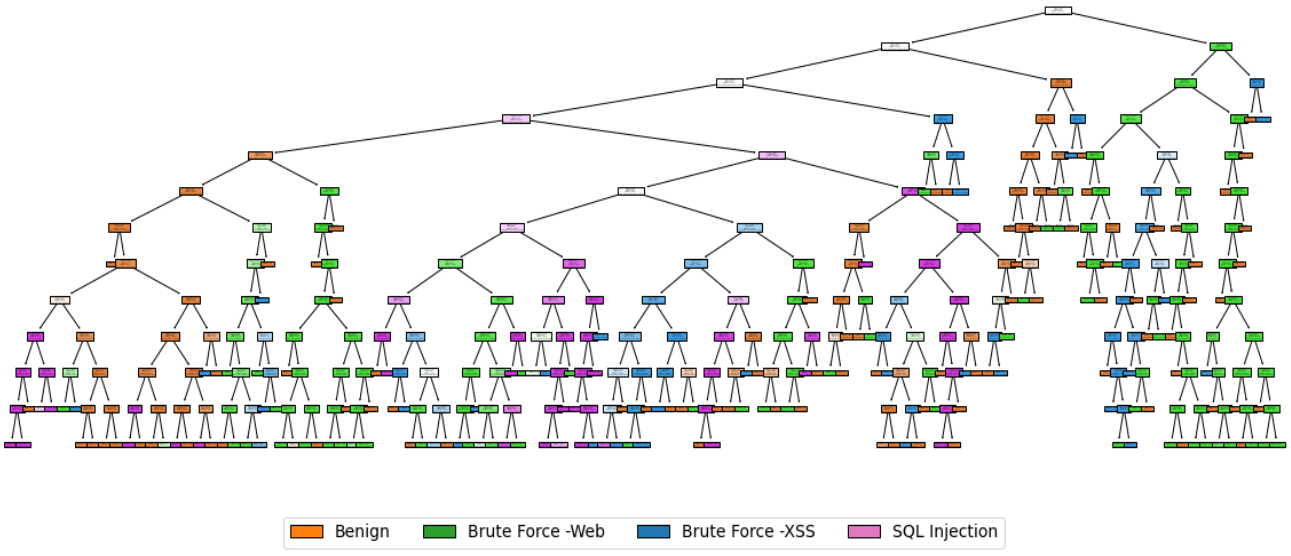


Ilustración 27 Árbol de decisión 12 niveles

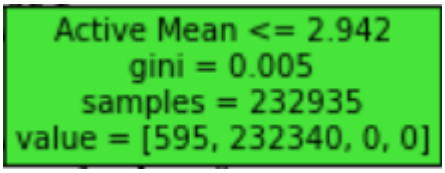


Ilustración 28 Hoja intermedia árbol decisión

Cada color que se muestra en la leyenda representa una clase identificada por el modelo en cada hoja.

Cada nodo del árbol contiene la siguiente información:

- Condición: criterio tomado por el modelo para tomar decisiones. Determinan cómo se dividen los datos. Las únicas hojas que no incluyen condición son las finales.
- Gini: grado de impureza (índice Gini). El valor 0 indica que el nodo es totalmente puro. En las hojas finales todos los valores Gini son menores a 0.005.
- Samples: número de muestras que se han considerado para llegar a ese nodo.
- Value: número de muestras en cada clase para ese nodo. [clase 0, clase 1, clase 2, clase 3]

Por ejemplo, en la hoja de la ilustración 24, se observa que el criterio que se ha elegido para ese nivel es un valor menor a 2.942 de la variable Active Mean. El índice Gini indica un grado de impureza muy bajo (0.005). Se han considerado 232935 muestras para ese nodo y el modelo lo ha clasificado como clase 1, que corresponde con el ataque Fuerza Bruta – Web, ya que el mayor número de las muestras (232340) pertenecen a esa clase.

El número total de hojas que conforman el árbol es de 181 hojas. Es importante destacar que el dataframe original tenía una dimensión de 1048575 filas y 48 columnas. Después del entrenamiento del modelo, se ha logrado simplificar la información en 181 nodos. Esta compresión de los datos es otra de las ventajas principales de este algoritmo.

5.2.4.2 Métricas de evaluación del modelo y gráficas

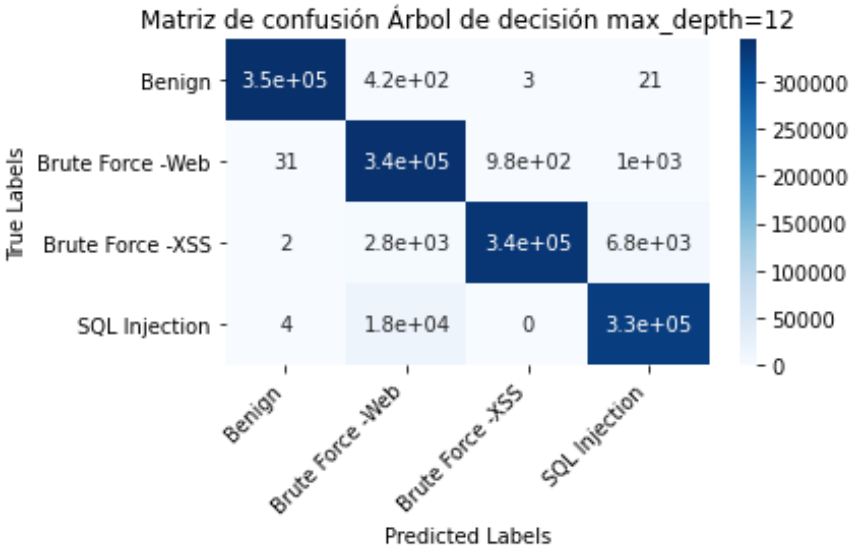


Ilustración 29 Matriz de confusión Árbol de decisión

	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	345764
Brute Force -Web	0.94	0.99	0.97	345302
Brute Force -XSS	1.00	0.97	0.98	346631
SQL Injection	0.98	0.95	0.96	345675
accuracy			0.98	1383372
macro avg	0.98	0.98	0.98	1383372
weighted avg	0.98	0.98	0.98	1383372

Ilustración 30 Informe de clasificación Árbol de decisión

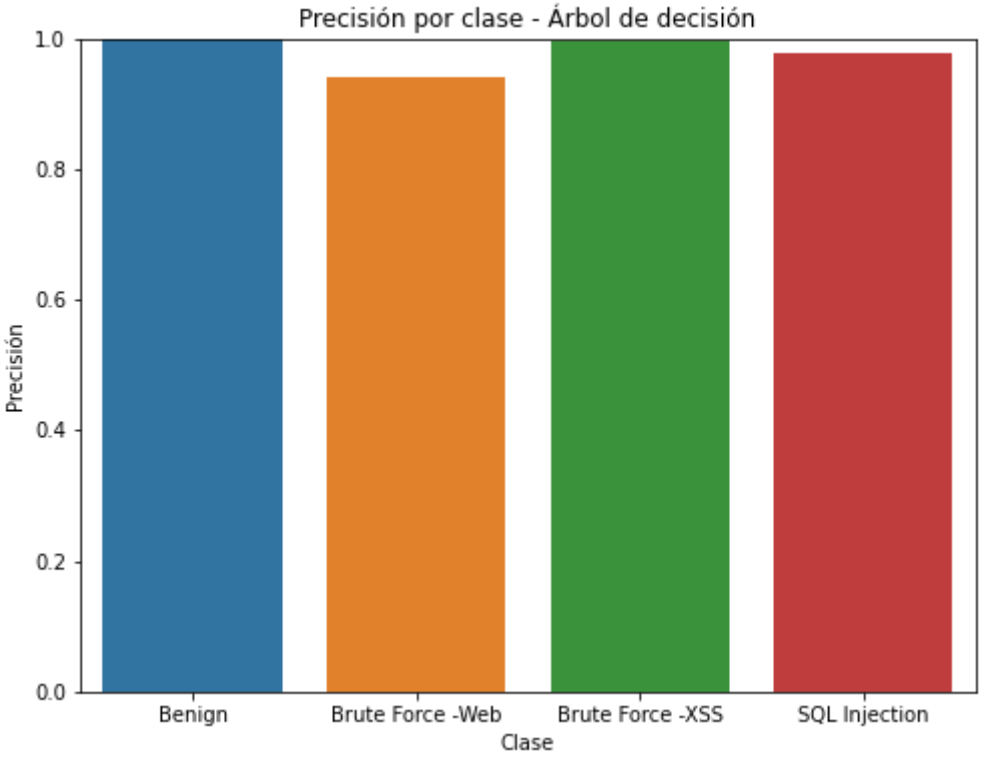


Ilustración 31 Precisión por clase Árbol de decisión

En la matriz de confusión se puede observar que el modelo ha predicho correctamente 345321 muestras de la clase 'Benign', 343264 de la clase 'Brute Force -Web', 337029 de la clase 'Brute Force -XSS' y 327512 de la clase 'SQL Injection'. El informe de clasificación indica que la precisión correspondiente a la clase 0 es del 100%, de la clase 1 del 94%, de la clase 2 el 100% y de la clase 3, 98%. Además, tanto la precisión ponderada como el promedio ponderado de precisión, f1-score y recall son del 98%. Según estos resultados, podemos concluir que es un modelo excelente con altas tasas de aciertos en todas las clases.

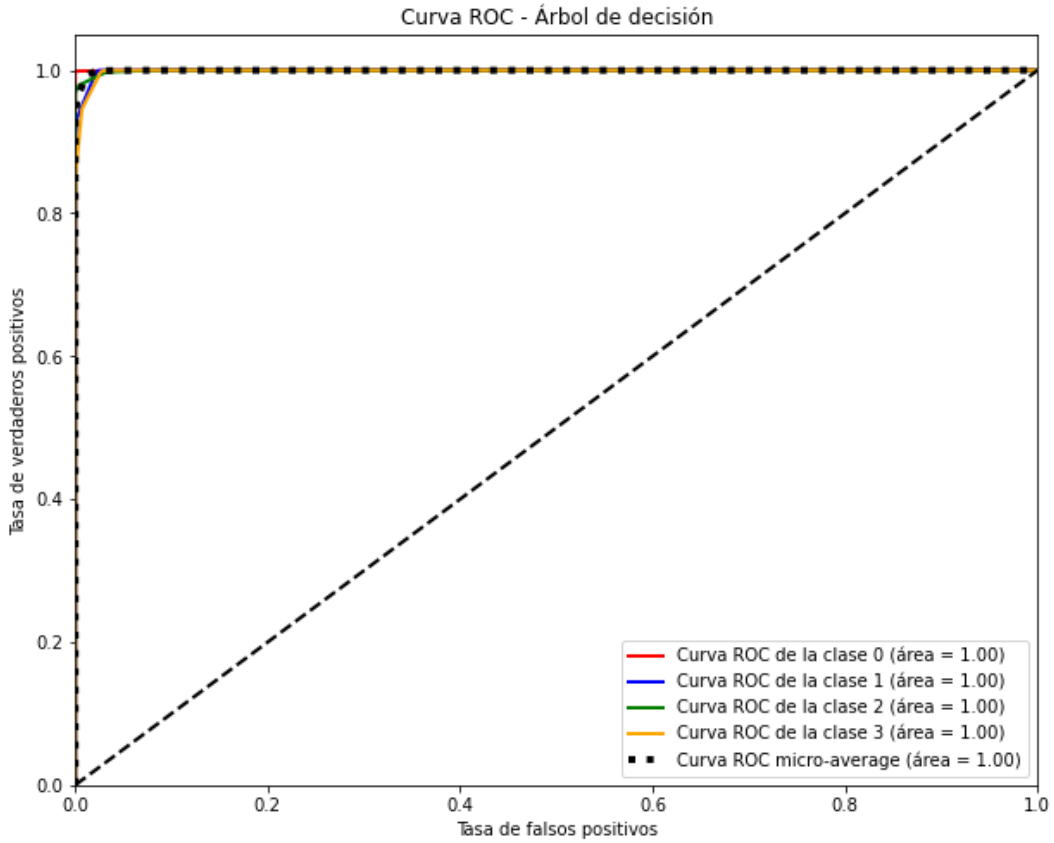


Ilustración 32 Curva ROC Árbol de decisión

La Curva ROC obtenida indica que el árbol de decisión es un modelo excelente para la clasificación de estos datos, lo que ya sugería la precisión global e individual por clases conseguida.

Además, se obtuvo una gráfica de la importancia de cada característica, para saber cuál tenía mayor peso o impacto en la predicción del modelo. Puede ayudar a simplificar el modelo al eliminar las características con menor impacto y de ese modo, hacerlo más eficiente. Además, puede ser de utilidad para futuros modelos.

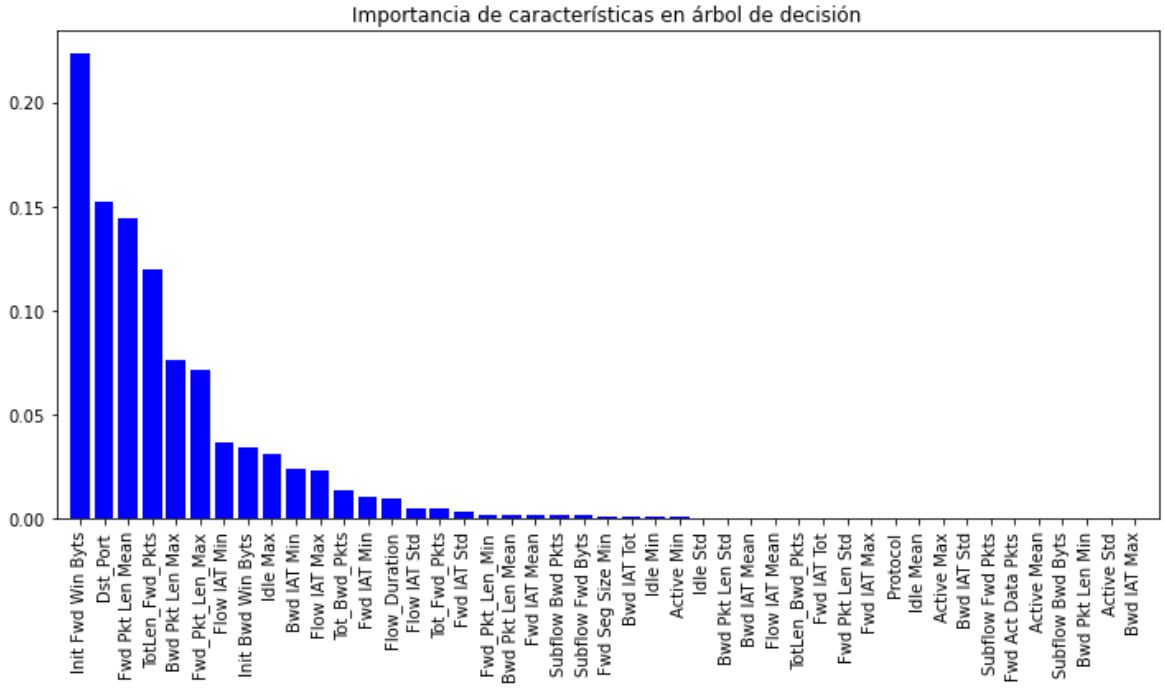


Ilustración 33 Importancia características Árbol de decisión

5.2.5 RANDOM FOREST

El conjunto de múltiples árboles de decisión se denomina Random Forest. Cada árbol es entrenado con un subconjunto aleatorio de características y de muestras de entrenamiento, evitando así el sobreajuste. Esto resulta en una precisión mayor ya que los errores se compensan al combinar los resultados. El resultado de la precisión final se toma con el voto mayoritario de los árboles que conforman el bosque.

A continuación, se muestra una imagen [19] que explica el algoritmo.

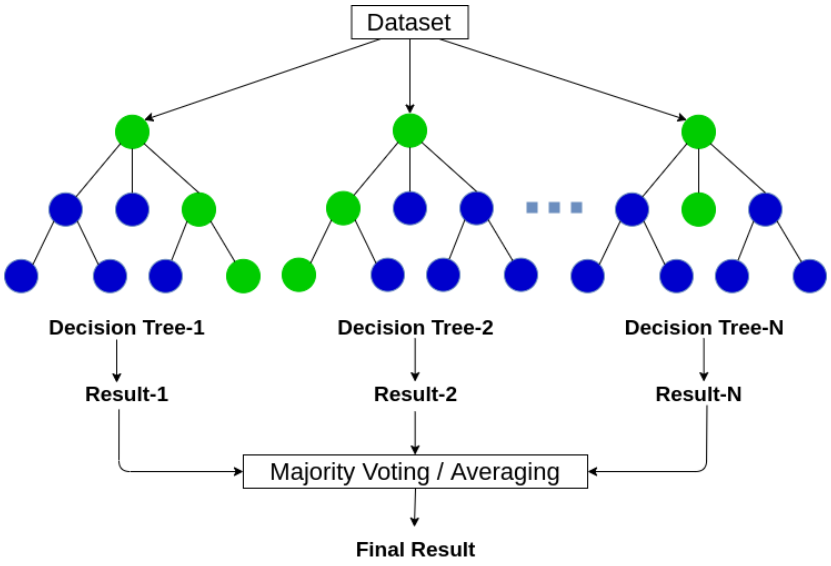


Ilustración 34 Diagrama Random Forest

5.2.5.1 Resultados ajustes de los hiperparámetros

Al definir el modelo, es necesario definir el número de estimaciones, que corresponde al número de árboles del bosque. Variando este parámetro y el número de columnas del conjunto de entrenamiento con el que se hace la predicción, se quería conseguir una alta precisión. Se eliminaron las 10 primeras columnas de los datos.

Para determinar el número de estimaciones adecuado para estos datos, se graficó la precisión en función del valor de estimaciones. La precisión se obtiene con la clase "score" de la clase "RandomForestClassifier".

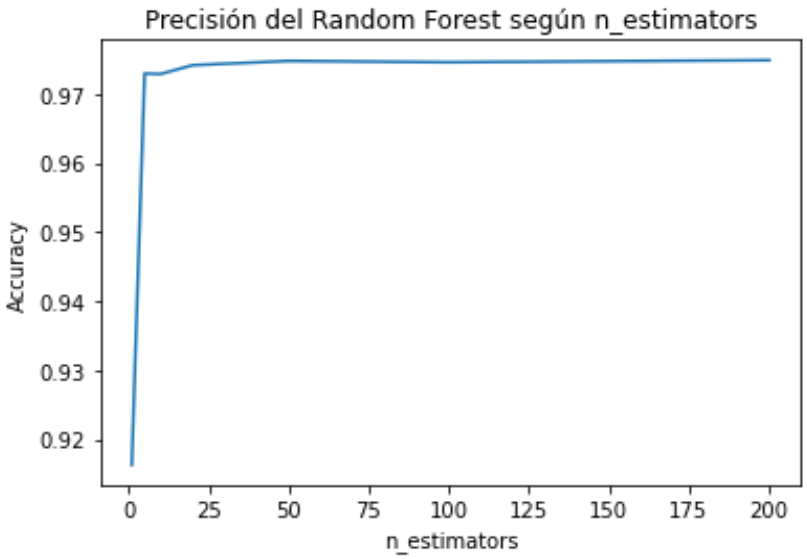


Ilustración 35 Precisión según número de estimaciones Random forest

n_estimators	precisión	tiempo (horas)
1	0.916333	0.002620
5	0.972995	0.012715
10	0.972938	0.026018
20	0.974216	0.051645
50	0.974827	0.128876
100	0.974640	0.255989
200	0.974939	0.515306

Ilustración 36 Resultados evaluación hiperparámetros Random Forest

Como no se obtenía una precisión significativamente mayor con más de 5 estimaciones, se escogió este valor, ya que equivale a una precisión bastante buena y simplifica la complejidad del modelo, además de evitar el sobreajuste del modelo.

5.2.5.2 Métricas de evaluación del modelo y gráficas

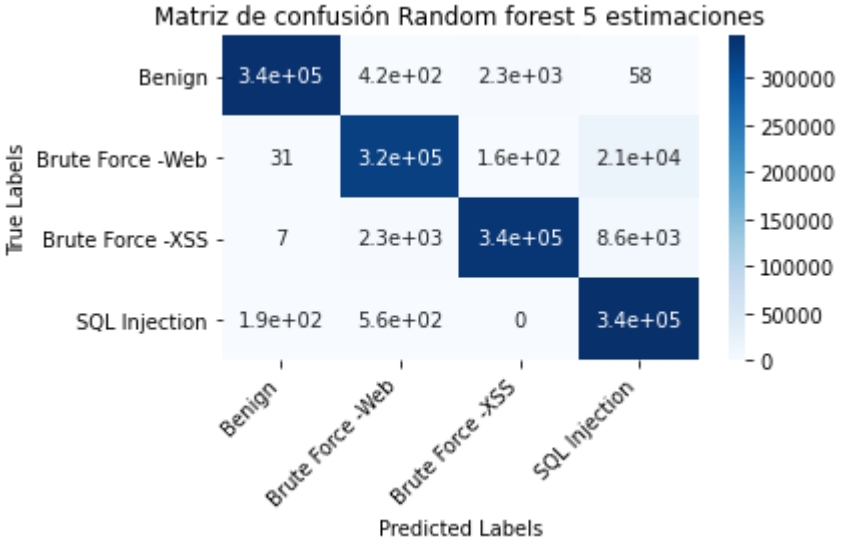


Ilustración 37 Matriz de confusión Random forest

	precision	recall	f1-score	support
Benign	1.00	0.99	1.00	345764
Brute Force -Web	0.99	0.94	0.96	345302
Brute Force -XSS	0.99	0.97	0.98	346631
SQL Injection	0.92	1.00	0.96	345675
accuracy			0.97	1383372
macro avg	0.98	0.97	0.97	1383372
weighted avg	0.98	0.97	0.97	1383372

Ilustración 38 Informe de clasificación Random forest

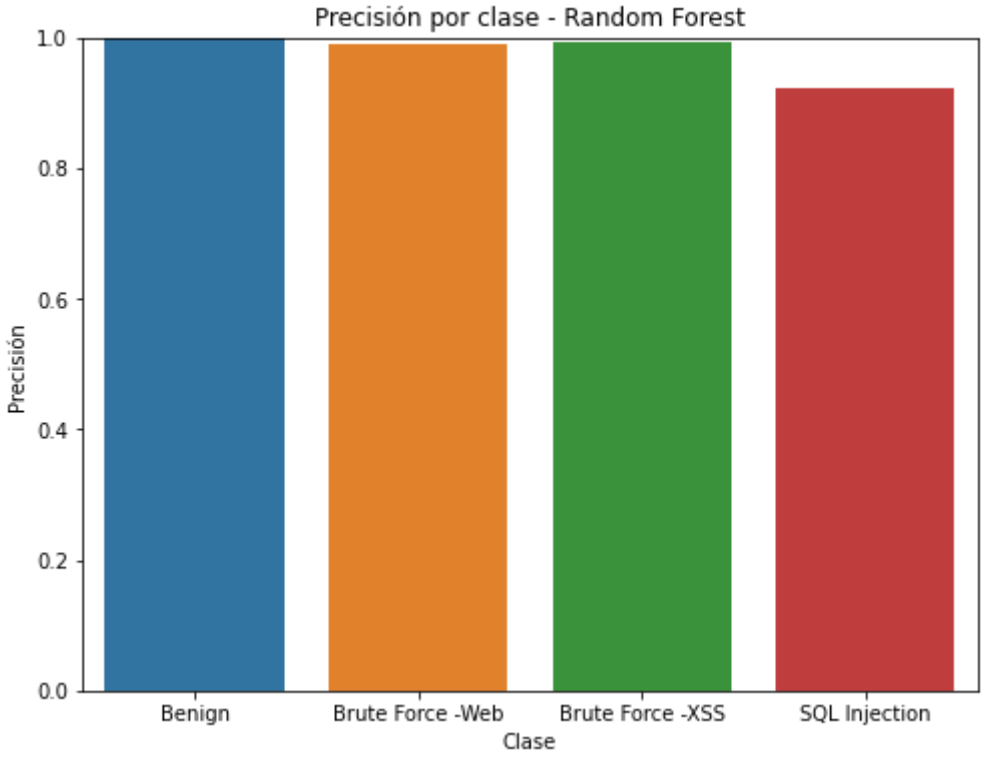


Ilustración 39 Precisión por clase Random Forest

En la matriz de confusión se observa que el modelo ha predicho correctamente 343022 muestras de la clase 'Benign', 324588 de la clase 'Brute Force -Web', 335689 de la clase 'Brute Force -XSS' y 344919 de la clase 'SQL Injection'. Según el informe de clasificación, la precisión correspondiente a la clase 0 es del 100%, de la clase 1 del 99%, de la clase 2 el 99% y de la clase 3, 92%. La precisión ponderada es del 98 % y el promedio ponderado de precision, f1-score y recall del 97%. De acuerdo con estos resultados, el Random forest entrenado es una buena opción de modelo para clasificar los ataques.

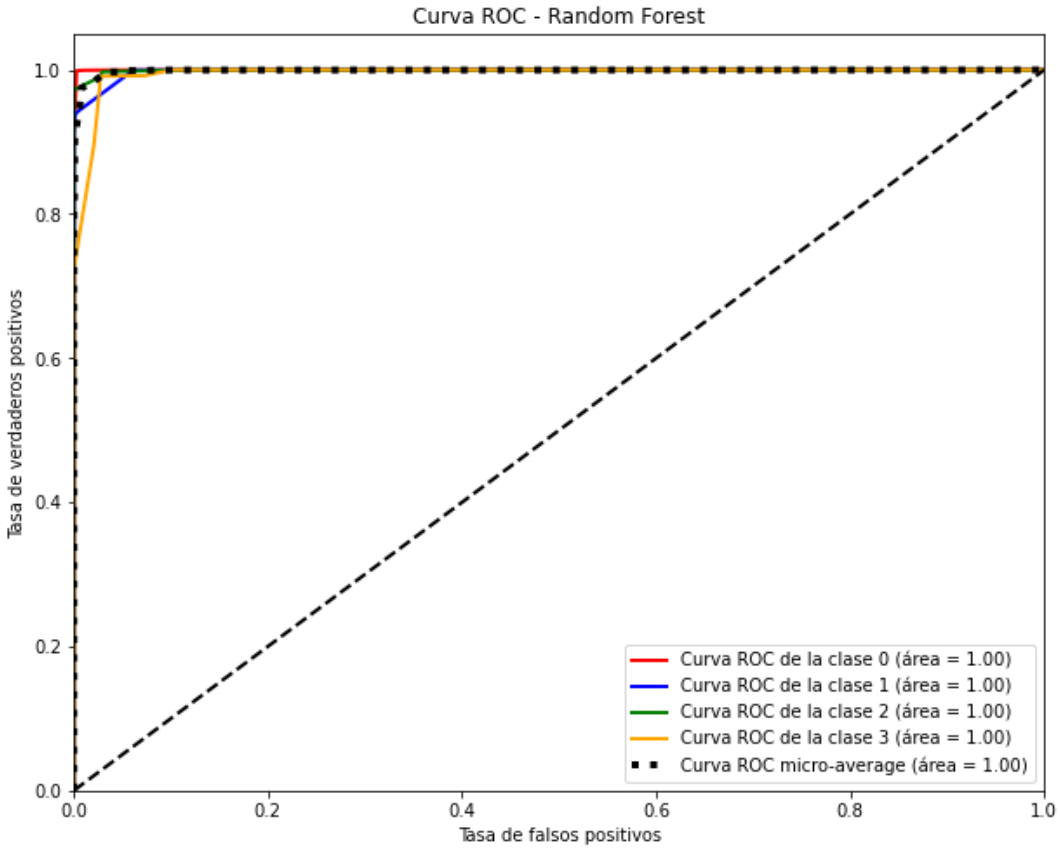


Ilustración 40 Curva ROC Random Forest

De forma muy similar al Árbol de decisión, las curvas ROC muestran que el modelo es óptimo para la tarea de clasificación de estos datos. Las curvas se aproximan a la esquina superior izquierda de la gráfica, lo que indica una alta tasa de verdaderos positivos y baja tasa de falsos positivos para los distintos umbrales.

Del mismo modo que con el árbol de decisión, se obtuvo una gráfica de la importancia de cada característica.

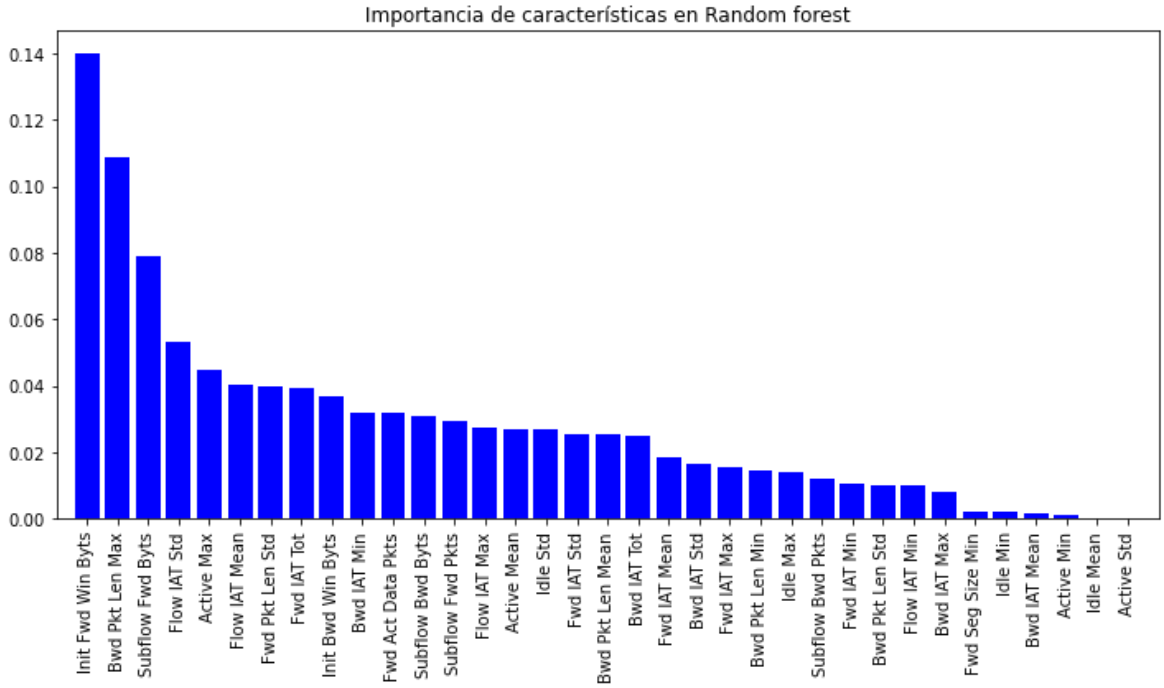


Ilustración 41 Importancia características Random forest

Si se comparan ambas gráficas de importancia (Ilustración 28 e Ilustración 36), puede apreciarse que el Árbol de decisión asigna importancias distintas a las características que el Random Forest. Cabe destacar que la forma en la que se calcula la importancia es diferente entre estos modelos. En el árbol de decisión, la importancia se mide por el coeficiente Gini o por el Information Gain, que es una medida que indica cuánta información proporciona una característica sobre una clase y cuánto mejora el modelo. Cuanto mayor es el information gain, mejor es la separación del árbol. [20]. En cambio, en el Random forest se calcula la importancia (con el atributo 'feature_importances_') como la media y desviación estándar de la acumulación de la disminución de la impureza dentro de cada árbol.

A continuación, se muestran las características que acumulan el 50% de la importancia total para ambos modelos.

Característica	Importancia Árbol decisión
Init Fwd Win Byts	0.222955
Dst_Port	0.152487
Fwd Pkt Len Mean	0.144311
Total importancia	0.519753

Ilustración 42 Características Árbol de decisión 50% importancia acumulada

Característica	Importancia Random Forest
Init Fwd Win Byts	0.154312
Flow IAT Max	0.092676
Subflow Bwd Byts	0.084258
Subflow Fwd Pkts	0.062950
Fwd IAT Mean	0.055225
Bwd Pkt Len Std	0.044478
Total importancia	0.493898

Ilustración 43 Características Random forest 50% importancia acumulada

Como se puede observar, el árbol de decisión atribuye una importancia total aproximada del 50% a sólo 3 características de las 45 que considera. Esto significa que esas características tienen un mayor peso en la consideración del algoritmo para la toma de decisiones en el entrenamiento. Por otra parte, Random forest necesita 6 variables de las 36 que toma en cuenta para alcanzar la mitad de importancia total. Ambos algoritmos consideran la característica "Init Fws Win Byts" (Número de bytes enviados en ventana inicial en la dirección de envío) como la más importante del conjunto, aunque con distinto valor de importancia. Otras similitudes entre asignación de importancia en ambos son:

- Flow IAT Max: Este atributo es considerado importante en ambos algoritmos, siendo el segundo más importante en el Random forest.
- Init Bwd Win Byts: Ambos modelos consideran esta importancia la octava más importante del total.
- Bwd Pkt Len Min, Active Std y Bwd IAT Std: Tanto el árbol como el bosque asignan una importancia cercana a cero a estas características.

5.2.6 KNN

KNN (K-Nearest Neighbors) es un algoritmo de aprendizaje supervisado no paramétrico utilizado tanto para clasificación como en regresión. Realiza una búsqueda de los k puntos más cercanos en el conjunto de entrenamiento a un punto de datos de prueba. El procedimiento que realiza el algoritmo es el siguiente:

1. Calcula la distancia entre el punto de datos de prueba y el resto de los datos del conjunto de entrenamiento en el espacio de características. Hay diversos tipos de distancias como la distancia Minkowski o la distancia Manhattan. Por defecto, con la biblioteca scikit-learn se utiliza la distancia euclidiana.
2. Selecciona los "k" puntos o vecinos más cercanos.
3. Se escoge la clase a la que pertenecerá ese punto por la votación de la mayoría de los vecinos más cercanos.

La versatilidad y la eficacia son algunas de las ventajas de emplear este algoritmo.

En la siguiente imagen [21] se muestra el funcionamiento del algoritmo.

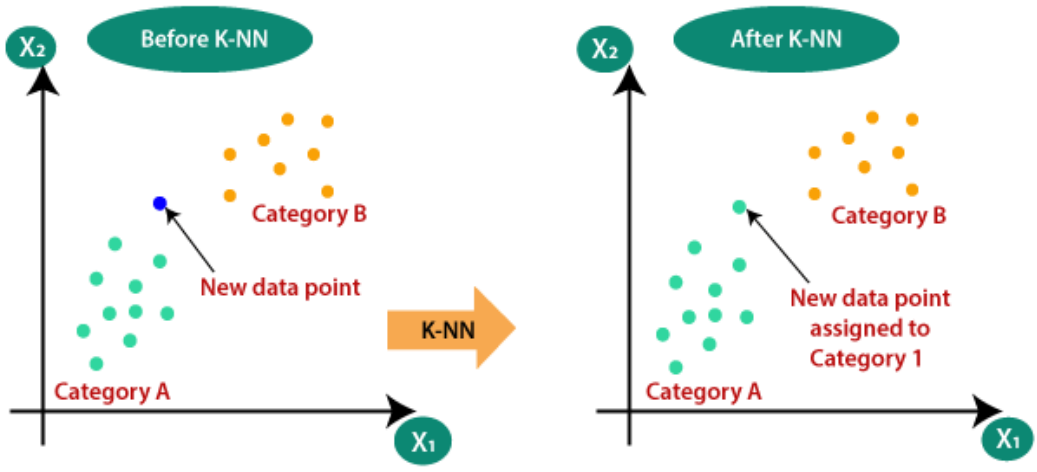


Ilustración 44 Algoritmo KNN

5.2.6.1 Resultados ajustes de los hiperparámetros

Para escoger el valor de k adecuado para el modelo, este se entrenó con varios valores y se observó que la precisión no aumentaba considerablemente con más de k=5 vecinos, por lo que se eligió este valor para el entrenamiento.

k	precisión	tiempo (horas)
2	98.796997	1.520823
5	98.970487	1.649002
7	98.976125	1.634998
10	98.968535	1.846955

Ilustración 45 Resultados evaluación hiperparámetros KNN

5.2.6.2 Métricas de evaluación del modelo y gráficas

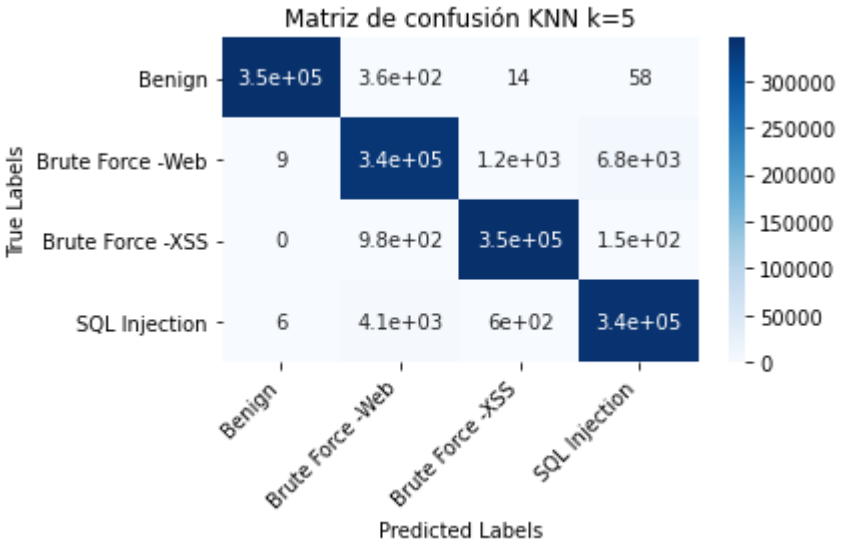


Ilustración 46 Matriz de confusión KNN

	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	345764
Brute Force -Web	0.98	0.98	0.98	345302
Brute Force -XSS	0.99	1.00	1.00	346631
SQL Injection	0.98	0.99	0.98	345675
accuracy			0.99	1383372
macro avg	0.99	0.99	0.99	1383372
weighted avg	0.99	0.99	0.99	1383372

Ilustración 47 Informe de clasificación KNN

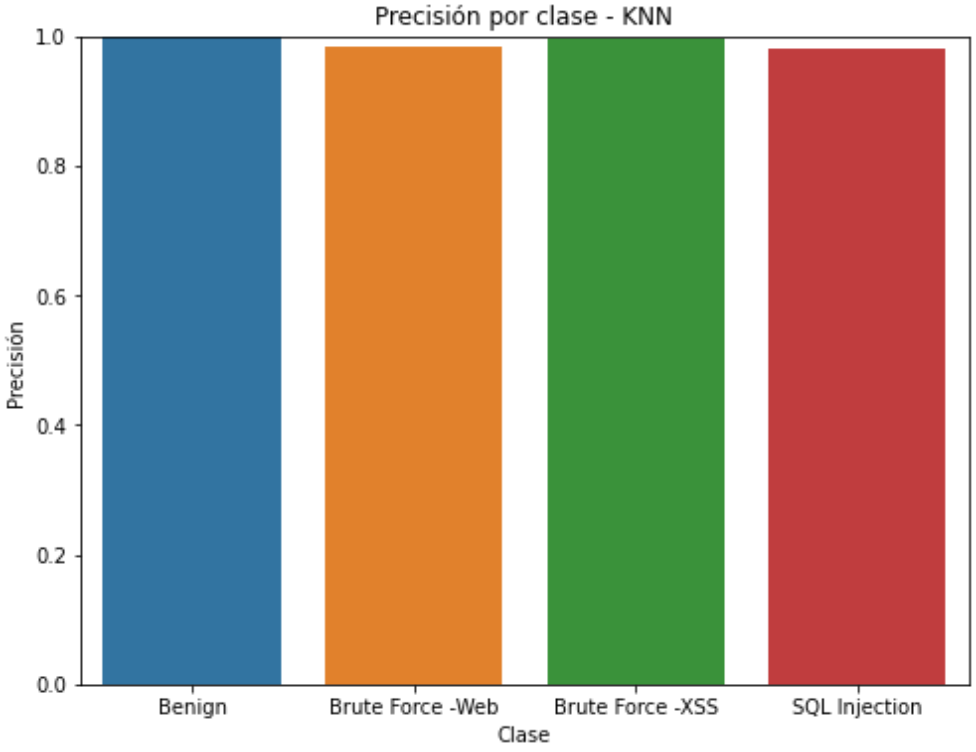


Ilustración 48 Precisión por clase KNN

En la matriz de confusión se observa que el modelo ha predicho correctamente 345335 muestras de la clase 'Bening', 337330 de la clase 'Brute-Force -Web' , 345500 de la clase 'Brute-Force -XSS' y 340965 de la clase 'SQL Injection'. El reporte de clasificación indica que la precisión correspondiente a la clase 0 es del 100%, de la clase 1 del 98%, de la clase 2 el 99% y de la clase 3, 98%. La precisión ponderada es del 99 % , al igual que el promedio ponderado de precisión, f1-score y recall. Con estos resultados, se puede afirmar que este modelo es excelente a la hora de clasificar ataques de cualquiera de las cuatro clases.

5.2.7 RED NEURONAL

Una red neuronal artificial es un modelo que trata de asemejar el funcionamiento del cerebro humano en el procesamiento de la información. Es una herramienta potente y muy útil en una gran cantidad de campos al tener la capacidad de "aprendizaje".

Entre las diversas ventajas que presentan las redes neuronales se encuentran la capacidad de tolerancia a fallos, el aprendizaje adaptativo y la capacidad de procesar grandes cantidades de datos. Son capaces de hacer generalizaciones a partir de patrones y sacar conclusiones.

Las redes neuronales se componen de capas, formadas por un conjunto de neuronas. La primera es la capa de entrada, que recibe los datos que va a procesar la red. Las capas intermedias se denominan capas ocultas, que ajustan los pesos de las entradas durante el entrenamiento de la red. Por último, la capa de salida genera la predicción final. Existen distintos tipos de redes como, por ejemplo: monocapa, multicapa, convolucionales o recurrentes, entre otras.

Para transmitir la información por las conexiones a las siguientes capas de la red se utilizan funciones de activación. Estas funciones determinan si la información debe continuar a la siguiente neurona. Entre las más comunes se encuentran la función sigmoidal, función ReLU (Rectified Linear Unit), softmax o función escalón.

En la siguiente figura [22] se muestra un ejemplo de red neuronal artificial con 4 capas de entrada y 2 capas de salida.

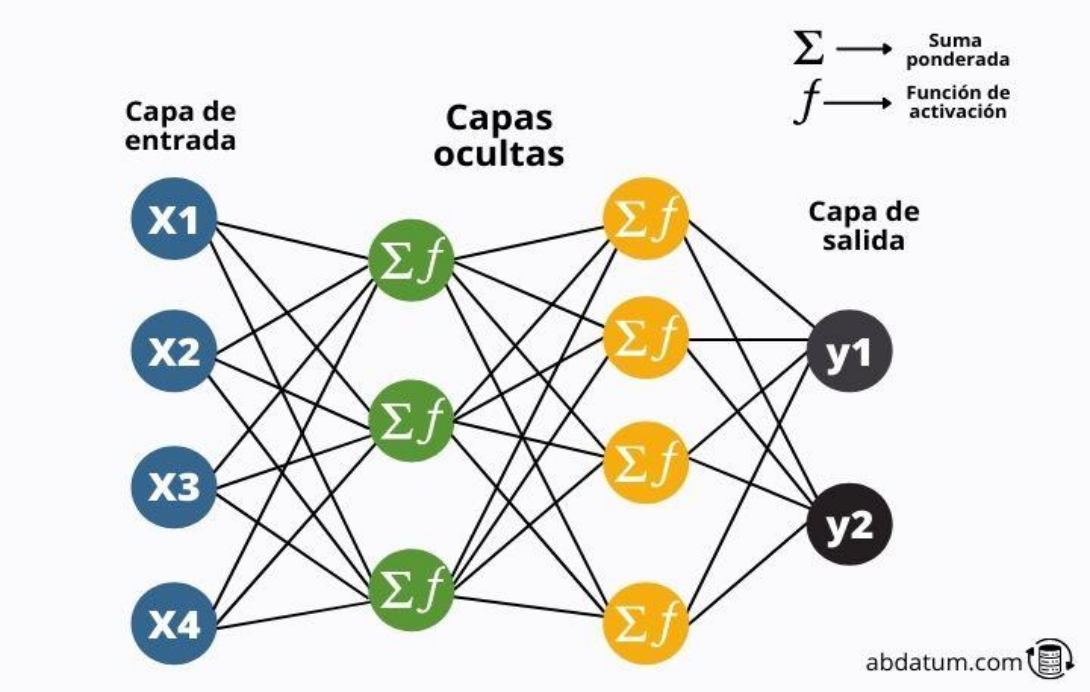


Ilustración 49 Red Neuronal artificial

Se entrenaron varios modelos de red neuronal con la biblioteca TensorFlow. Lo primero que se realizó fue convertir las etiquetas categóricas en matrices binarias mediante una técnica denominada "one-hot encoding", necesaria para el correcto procesamiento en la red neuronal.

Se utiliza la función de activación "relu" tanto para la capa de entrada como para las ocultas. La capa de salida consta de 4 neuronas y utiliza la función de activación "softmax", usada en clasificación multiclase.

La función de pérdida que se eligió fue 'categorical_crossentropy', y el optimizador escogido fue 'adam', en el cual se puede ir ajustando el learning rate o tasa de aprendizaje que consiste en la proporción de cambio con el que se actualizan los pesos del modelo en cada iteración en el proceso de entrenamiento.

Se hizo uso de la técnica Early Stopping para evitar el sobreajuste. De esta forma, se detiene el entrenamiento del modelo si hay un incremento en el valor del error de validación. Es

decir, si la función de pérdida no mejora tras un número de épocas, especificado en el parámetro "patience", se parará el entrenamiento.

Para evaluar el modelo se utilizó la función de Tensorflow "argmax" en las clases predichas y en las clases reales. Esta función devuelve los índices de los valores máximos de un tensor.

5.2.7.1 Resultados ajustes de los hiperparámetros

Para el entrenamiento de modelo, se especifican además de los datos de entrenamiento, hiperparámetros como el número de épocas, tamaño del batch y valor de proporción de datos validación deseado. El número de épocas representa el número de veces que la red itera cada ejemplo de datos de entrenamiento. El valor batch (lote) es la cantidad de datos que tiene cada iteración de una época. Es común elegir un valor de potencia de 2. Se seleccionó un valor de 64. Se especificó un valor de conjunto de validación del 20% de los datos de entrenamiento.

Para obtener la combinación mejor de los parámetros, se probaron varios valores de número de neuronas de capa de entrada y capas ocultas, learning rate , patience y número de épocas. Se muestra a continuación:

capa entrada	capa1	capa2	capa3	capa4	capa5	epocas	learning rate	early stopping	precision	tiempo
128	64	64	-	-	-	70	0.00001	7	86.272673	1.824026
128	128	64	64	-	-	70	0.00001	7	86.249613	1.440698
128	128	64	64	64	-	70	0.00001	7	90.550626	1.291270
128	128	64	64	64	64	70	0.00001	7	86.247878	0.700405
256	128	128	64	64	-	70	0.00001	7	86.315178	1.475562
128	64	64	-	-	-	80	0.00010	7	85.748808	0.719487
128	128	64	64	-	-	80	0.00010	7	85.500646	0.653844
128	128	64	64	64	-	80	0.00010	7	85.759940	1.418140
128	128	64	64	64	64	80	0.00010	7	86.024583	0.788458
256	128	128	64	64	-	80	0.00010	7	86.055305	0.742643
128	64	64	-	-	-	90	0.00001	7	90.362462	3.269170
128	128	64	64	-	-	90	0.00001	7	86.193157	2.031957
128	128	64	64	64	-	90	0.00001	7	86.340478	2.250675
128	128	64	64	64	64	90	0.00001	7	86.118701	2.532087
256	128	128	64	64	-	90	0.00001	7	85.755603	4.357112
128	64	64	-	-	-	100	0.00001	7	86.044968	3.881170
128	128	64	64	-	-	100	0.00001	7	86.261035	3.992420
128	128	64	64	64	-	100	0.00001	7	85.786614	2.785371
128	128	64	64	64	64	100	0.00001	7	86.248095	4.278605
256	128	128	64	64	-	100	0.00001	7	85.770133	11.061167
128	64	64	-	-	-	90	0.00001	8	86.209566	2.082074
128	128	64	64	-	-	90	0.00001	8	86.331731	2.344297
128	128	64	64	64	-	90	0.00001	8	85.775410	1.950561
128	128	64	64	64	64	90	0.00001	8	89.845537	1.135840
256	128	128	64	64	-	90	0.00001	8	90.324945	1.484811

Ilustración 50 Resultados evaluación hiperparámetros Red neuronal

Como se observa en la tabla, la precisión mayor se obtiene con 128 neuronas de entrada, 4 capas ocultas de 128, 64, 64 y 64 neuronas respectivamente, un learning rate de 0.00001, 70 épocas y un valor de 7 de patience (Early Stopping).

5.2.7.2 Métricas de evaluación del modelo y gráficas

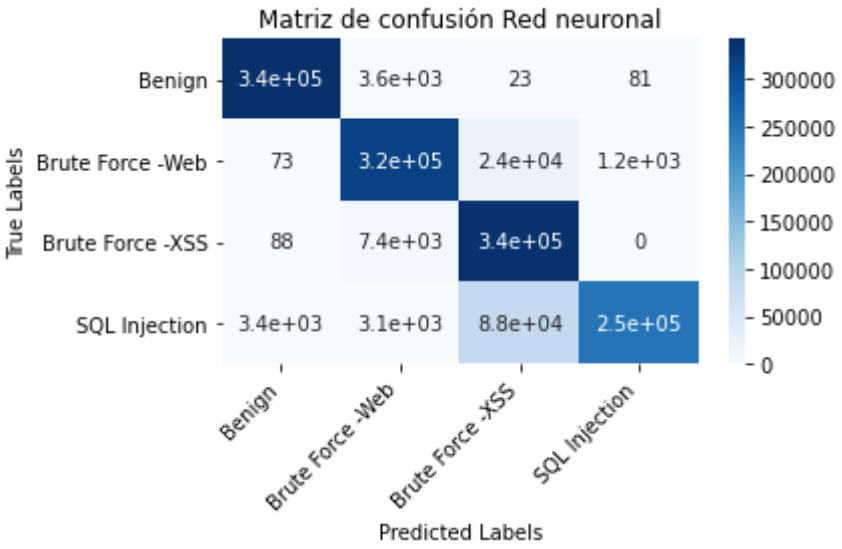


Ilustración 51 Matriz de confusión Red neuronal

	precision	recall	f1-score	support
0	0.99	0.99	0.99	345764
1	0.96	0.93	0.94	345302
2	0.75	0.98	0.85	346631
3	1.00	0.73	0.84	345675
accuracy			0.91	1383372
macro avg	0.92	0.91	0.91	1383372
weighted avg	0.92	0.91	0.91	1383372

Ilustración 52 Classification report Red neuronal

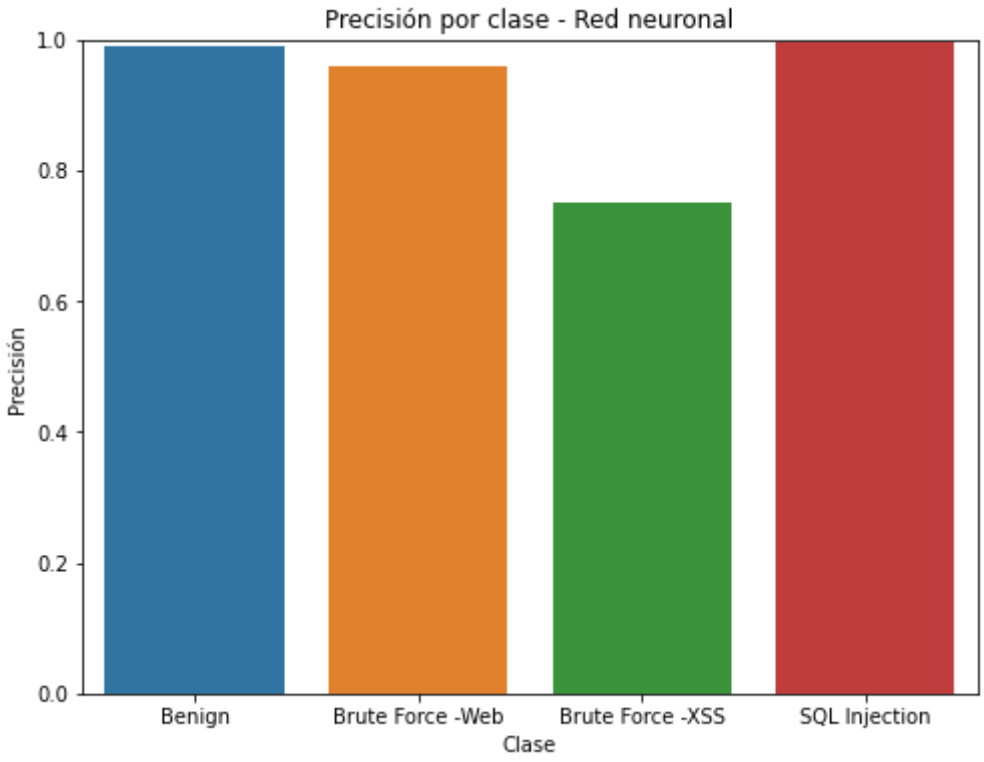


Ilustración 53 Precisión por clase Red neuronal

En la matriz de confusión se observa que la red ha predicho correctamente aproximadamente 340000 muestras de la clase 'Bening', 320000 de la clase 'Brute-Force -Web' , 340000 de la clase 'Brute-Force -XSS' y 250000 de la clase 'SQL Injection'. El reporte de clasificación indica que la precisión correspondiente a la clase 0 es del 99%, de la clase 1 del 96%, de la clase 2 el 75% y de la clase 3, 100%. La precisión ponderada es del 92 % , al igual que el promedio ponderado de precisión. F1-score y recall es del 91%. Con estos resultados, se puede afirmar que la red neuronal muestra una buena capacidad para clasificar la mayoría de las clases, a excepción de la clase 2 correspondiente a ataque de fuerza bruta -XSS, cuya precisión es menor que las demás.

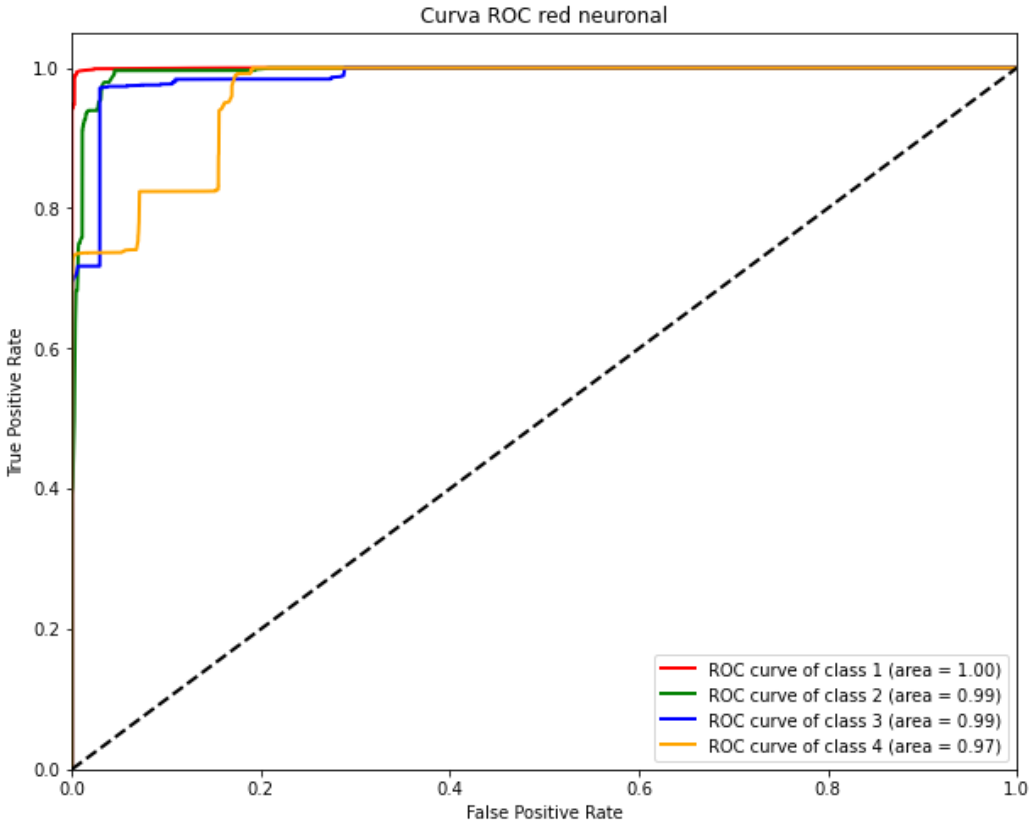


Ilustración 54 Curva ROC Red neuronal

Se obtuvo una gráfica que representa la evolución de las pérdidas (loss) y la precisión (accuracy) de la red durante las épocas de su entrenamiento.

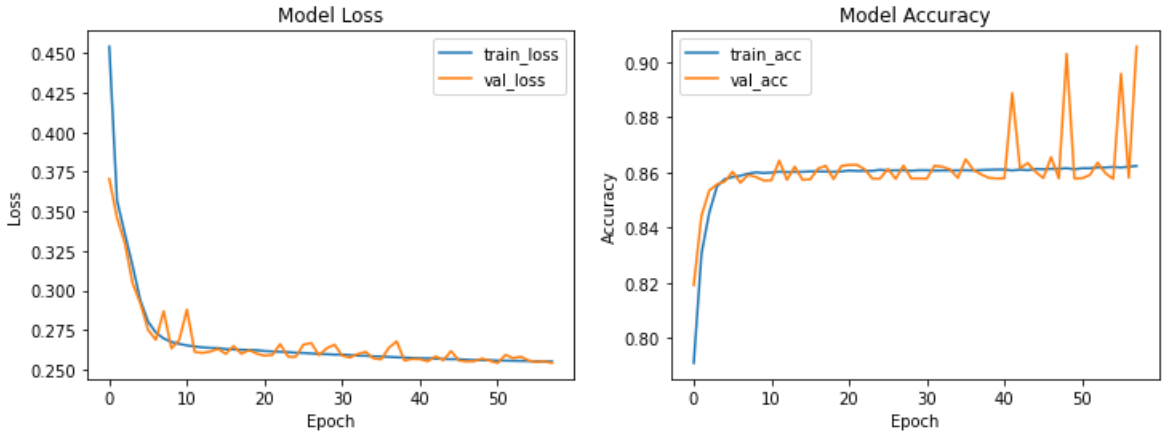


Ilustración 55 Model Loss, Model Accuracy Red neuronal

Los valores que se muestran en la gráfica son los siguientes:

- train_loss: Indica las pérdidas que se obtienen durante el proceso del entrenamiento.
- val_loss: Representa el validation loss, es decir, el valor de la función coste en el conjunto de validación.
- train_acc: Indica la precisión que se obtiene durante el entrenamiento.
- val_acc: Representa el validation accuracy, la precisión que se va obteniendo sobre el conjunto de validación.

Como puede observarse en la ilustración 50, las pérdidas van reduciéndose de manera significativa cuanto más aumenta el número de épocas. Esto es debido al ajuste de los pesos durante el proceso de entrenamiento. Aunque el número de épocas definido era 70, en la gráfica se muestra que el entrenamiento se paró en aproximadamente en 55, debido al Early Stopping utilizado. También se visualiza que la precisión va aumentando en las primeras 5 épocas y se estabiliza.

Capítulo 6. ANÁLISIS DE RESULTADOS

A la vista de los resultados obtenidos (Ilustración 51), se determinó que los modelos con mayor precisión son el Árbol de decisión, Random forest y KNN.

Modelo	Precisión	Tiempo (horas)
Regresión Logística	83.135122	0.062919
SGD	80.662324	0.007545
Árbol decisión	97.813603	0.018899
Random Forest	97.299555	0.012715
KNN	98.970487	1.649002
Red neuronal	90.550625	1.291269

Ilustración 56 Resultados precisión-tiempo modelos

Con los modelos entrenados y testeados, se hizo una combinación de sus predicciones para obtener una única predicción. El objetivo era observar si de esta manera, mejoraría la precisión general del modelo. Al combinar las predicciones, las fortalezas y debilidades de cada modelo se pueden compensar para obtener un resultado más preciso. A esta estrategia se la denomina Ensemble.

Las distintas formas de construir un ensemble son con Votación por mayoría, Bagging, Stacking o Boosting. Para este trabajo, se ha escogido Votación por mayoría. Un diagrama de este tipo de ensemble se muestra a continuación [23].

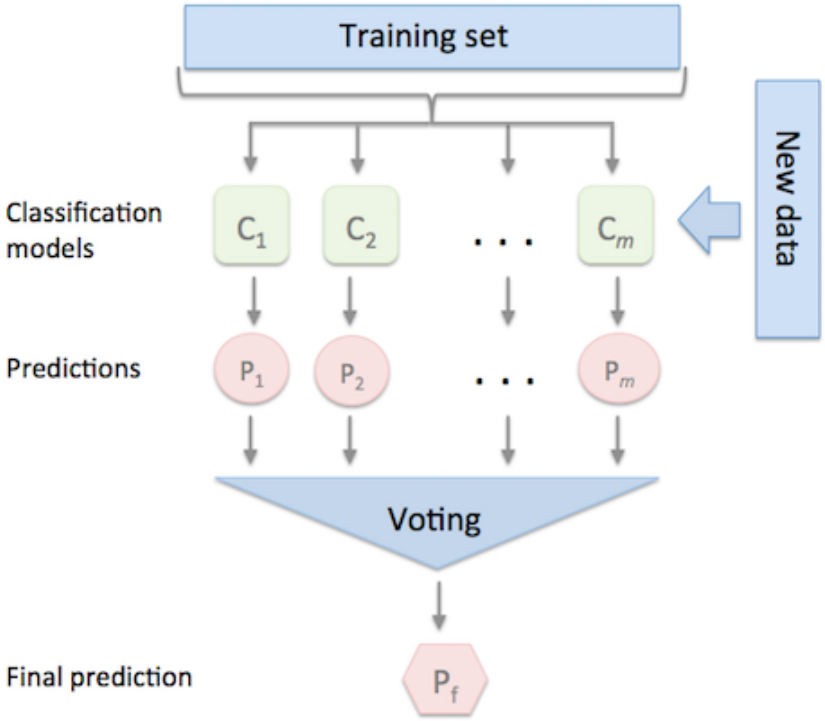


Ilustración 57 Ensemble por votación

Para ello, se creó un nuevo dataframe que contenía las predicciones de los modelos con mejor precisión. Se colocaron en las tres primeras columnas del dataframe. La cuarta columna se rellenó con la clase mayoritaria para cada fila, es decir, la clase más comúnmente predicha por los modelos. Se compararon posteriormente estos resultados de predicciones con el conjunto y_test original para evaluar la precisión global del modelo construido. La precisión y el tiempo de ejecución para construir este ensemble se muestran a continuación.

precisión	tiempo (horas)
98.455947	0.190284

Ilustración 58 Resultados Ensemble

Se ha conseguido de esta forma un modelo más robusto en términos de precisión global para la detección de ataques que el Árbol de decisión y Random Forest. Sin embargo, el algoritmo de KNN continúa siendo el modelo con mayor precisión.

A continuación, se analizará en más detalle la efectividad del ensemble con las métricas utilizadas en el análisis del rendimiento de los otros modelos empleados.

6.1.1.1 Métricas de evaluación y gráficas Ensemble

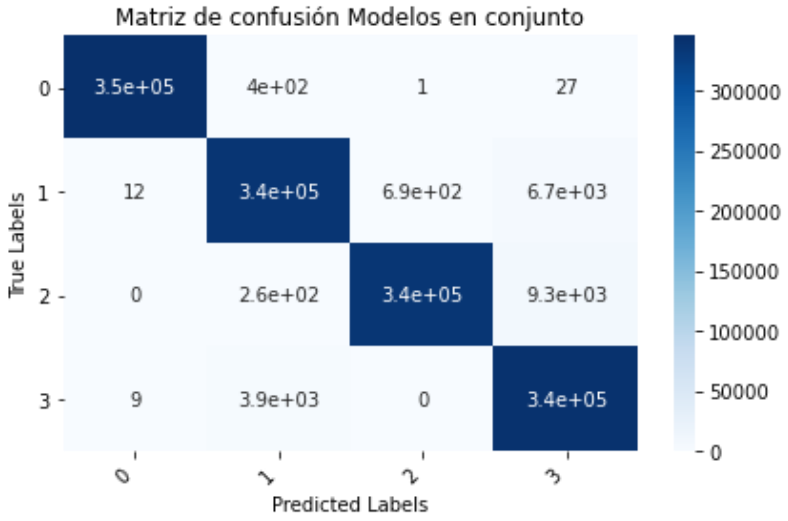


Ilustración 59 Matriz de confusión Ensemble

Informe de clasificación:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	345764
1	0.99	0.98	0.98	345302
2	1.00	0.97	0.98	346631
3	0.96	0.99	0.97	345675
accuracy			0.98	1383372
macro avg	0.98	0.98	0.98	1383372
weighted avg	0.98	0.98	0.98	1383372

Ilustración 60 Informe de clasificación Ensemble

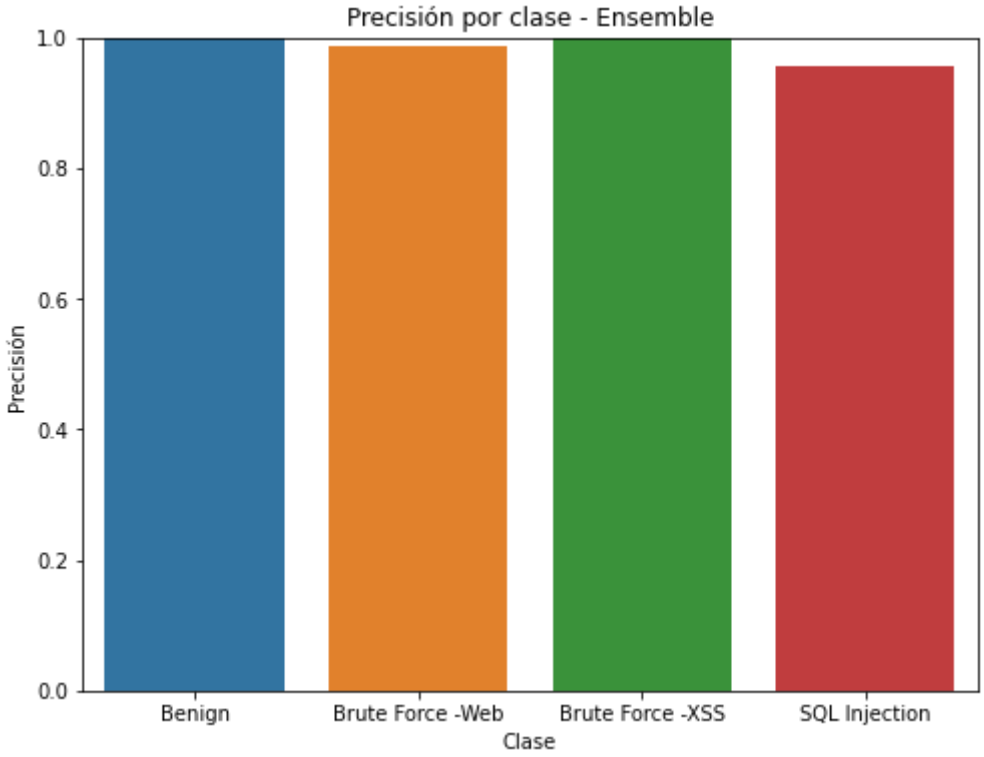


Ilustración 61 Precisión por clase Ensemble

En la matriz de confusión se observa que el modelo en conjunto ha predicho correctamente 345335 muestras de la clase 0, 337926 de la clase 1, 337053 de la clase 2 y 341722 de la clase 3. El reporte de clasificación indica que la precisión correspondiente a la clase 0 es del 100%, de la clase 1 del 99%, de la clase 2 el 100% y de la clase 3, 96%. La precisión ponderada es del 98 % , al igual que el promedio ponderado de precisión, f1-score y recall. Son resultados que demuestran la eficacia de realizar un ensemble al obtener las accuracies por clase más altas de todos los modelos entrenados, a excepción de KNN.

Para comprobar de forma más detallada la mejora proporcionada por el ensemble se muestra a continuación una gráfica de comparación de la precisión por clases entre los tres modelos que conforman el ensemble y la precisión por clases del propio ensemble, junto a una tabla con las precisiones medias obtenidas y si se ha conseguido una mejora con el ensemble.

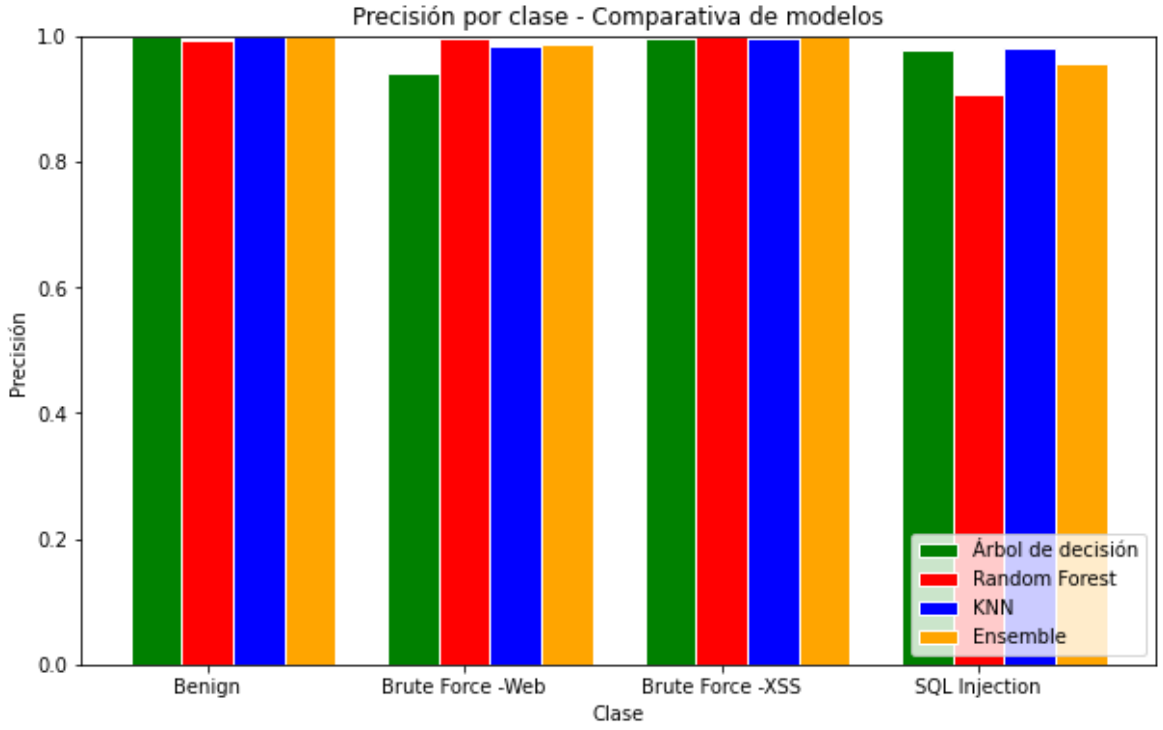


Ilustración 62 Precisión por clase comparativa modelos y Ensemble

Tipo de Clase	Precisión Árbol decisión	Precisión Random Forest	Precisión KNN	Precisión Media	Precisión Ensemble	¿Conseguida mejora con Ensemble?
Bening	99.989286	99.137701	99.995657	99.707548	99.994209	Sí
Brute Force - Web	94.138296	99.661078	98.413504	97.404293	98.595344	Sí
Brute force - XSS	99.707707	99.871478	99.489164	99.689450	99.761145	Sí
SQL injection	97.660119	90.651863	97.984080	95.432021	95.600507	Sí

Ilustración 63 Comparación precisiones por clases modelos y ensemble

Los resultados obtenidos revelan que todos los ataques son clasificados con precisiones mayores al 95% de media y con el ensemble con más del 98.6%. Los ataques mejor clasificados son la clase benigna y el ataque fuerza bruta - XSS. La clase SQL Injection tiene una precisión media ligeramente inferior a las demás, aunque sigue siendo buena, al igual que la de ataque tipo fuerza bruta -Web. Se ha demostrado que el ensemble proporciona una mejora a la hora de detectar todos los ataques analizados en este proyecto al superar la precisión media por clases del conjunto de los algoritmos.

Adicionalmente, se graficó la precisión (representado con la línea roja) del ensemble, con la individual de cada modelo utilizado en este trabajo y el tiempo de ejecución de cada uno para observar sus diferencias, mostrado en las barras. Se muestra en la siguiente ilustración:

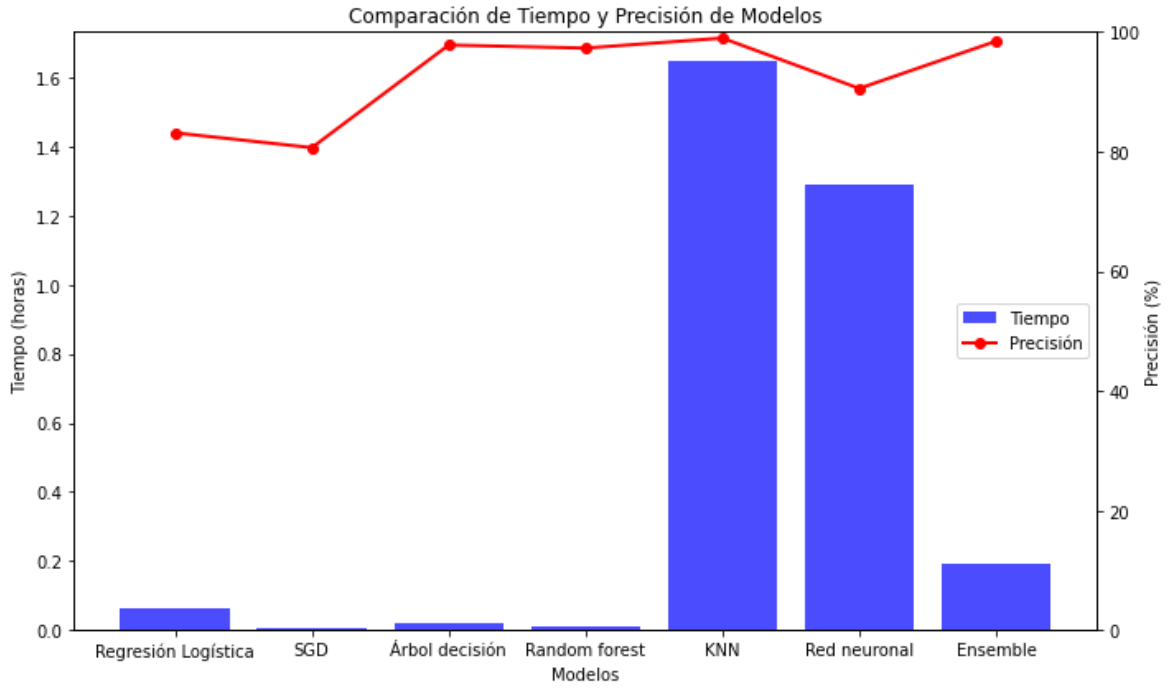


Ilustración 64 Comparación tiempo-precisión modelos

Como puede observarse en la gráfica, el algoritmo KNN es el que más tiempo de ejecución tiene de todos los algoritmos propuestos. Esto es debido a que, para predecir cada punto de datos, el modelo utiliza el dataset en completo para calcular la distancia entre ese punto y el todos los puntos del conjunto de entrenamiento. Esto hace que consuma una gran cantidad de recursos de memoria y CPU. En cambio, el resto de los algoritmos a excepción de la red neuronal tardan mucho menos en comparación al no ser modelos que realizan el cálculo de distancias. La red neuronal muestra un elevado tiempo de proceso de entrenamiento al haber definido 70 épocas en su entreno y debido también a su proceso de actualización de pesos. Sin embargo, el ensemble no presenta un tiempo grande de ejecución ya que no consiste en un proceso de entrenamiento, si no en un cálculo de la clase mayoritaria.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

Se ha cumplido con todos los objetivos propuestos en este trabajo.

- En primer lugar, se ha obtenido un dataset de una variedad de datos de cuatro tipos de ataques en el tráfico de redes. Se ha procedido a un filtrado de los datos, la utilización de la técnica de balanceo de SMOTE para igualar las muestras de las clases y la separación en un conjunto de train, que se utilizara en el entrenamiento del modelo, y un conjunto de prueba para testearlo y obtener las predicciones.
- Con los datos procesados se han construido, entrenado y testeado diversos modelos de ML y DL mediante el ajuste y optimización de sus hiperparámetros. Además, se han aplicado distintas métricas de evaluación del rendimiento a estos algoritmos como la matriz de confusión, curva ROC o precisión por clases.
- El modelo de Regresión Logística muestra una precisión del 83,135122%. En general, el rendimiento es bueno. No obstante, la clase Brute Force – XSS tiene el valor más bajo de precisión de todos los modelos, siendo un 66%.
- Con el modelo SGD se ha obtenido una precisión del 80,662323%. Presenta altas tasas de precisión en la clasificación de tres de los ataques. Sin embargo, es el modelo con el valor más bajo en la clasificación de alguna de las cuatro clases, siendo un 59% correspondiendo al ataque SQL injection.
- El Árbol de decisión muestra una precisión del 97.813603% con 12 niveles de profundidad. Su capacidad de clasificación es excelente.
- Con el algoritmo de Random Forest se obtiene una precisión del 97,299555%. Al igual que el algoritmo SGD, el ataque SQL injection es la clase con peor precisión.
- KNN es el algoritmo con mayor precisión: 98,97047%. Todos los ataques son clasificados con precisiones mayores al 98%.
- La red neuronal presenta una precisión de clasificación de 90,550625%. Con precisiones mayores a 95% en tres de las clases, se concluye que su rendimiento es

positivo. De forma parecida a Regresión logística, la clase que la red clasifica de forma menos precisa es Brute Force -SXX con un 75% de precisión.

- La clase que los modelos clasifican con mayor exactitud es la clase Benigno con una media de 99.707548%. SQL injection, con una precisión media de 95,600507% es el ataque con la clasificación más débil.
- Asimismo, se ha propuesto un entorno de colaboración conjunta de los modelos con mejor desempeño en la clasificación de los cuatro tipos de ataques, y se ha demostrado que es más eficaz en la tarea de clasificación que dos de los tres algoritmos de los cuales se conformaba al obtenerse una precisión de 98.455947%. Se concluye, de esta forma, que el algoritmo KNN es el más preciso al clasificar las cuatro clases.

7.2 TRABAJOS FUTUROS

Para futuros proyectos en esta área se podría:

- Explorar más modelos de ML y DL. En este trabajo se han construido seis modelos, pero podría ampliarse a otros algoritmos como SVM (Máquinas de vectores de soporte), Gradient Boosting , Redes neuronales convolucionales o Redes neuronales recurrentes. Utilizar un número mayor de modelos para la tarea de clasificación conllevaría varios beneficios como considerar otros enfoques para clasificar los ataques o emplear técnicas más avanzadas y de esta forma, encontrar el algoritmo que mejor se adapte a los datos.
- Realizar un estudio más exhaustivo en el ajuste de los hiperparámetros. Aunque en este trabajo se ha realizado una optimización de los hiperparámetros para cada modelo, hay diversas técnicas que se emplean para este proceso como optimización bayesiana, o búsqueda por cuadrícula y aleatorizada para encontrar la combinación más adecuada en cada caso.
- Utilizar otros tipos de ensemble. En este trabajo se ha utilizado Votación por mayoría, pero podrían investigarse otros tipos como Bagging, Boosting y Stacking para compararlos y obtener el método con más eficacia para los datos específicos.

- Búsqueda más amplia de datos de ataques a sistemas de información. Emplear datos más recientes de ataques en sistemas de información y aplicarlos a los algoritmos ya construidos y los futuros desarrollados para observar el rendimiento.
- Añadir más tipos de ataques. En este trabajo se han clasificado cuatro ataques, pero se podrían estudiar otros como, por ejemplo, el ataque de denegación de servicio (DoS), Spoofing, Man in the Middle o el ataque Remote to local (R2L). Existen una gran variedad de amenazas contra los sistemas informáticos y la aparición de nuevos ataques hace necesario su estudio y su prevención.
- Despliegue de los modelos en AWS (Amazon Web Service). Con la gran multitud de servicios que ofrece esta plataforma, se podría acelerar el proceso de entrenamiento de los modelos con el uso de herramientas como son la ejecución en paralelo o instancias de GPU.

Capítulo 8. BIBLIOGRAFÍA

[1] «Intrusion detection system,» <Packt>, [En línea]. Available: https://subscription.packtpub.com/book/cloud-and-networking/9781787128873/2/ch02lvl1sec10/intrusion-detection-system. [Último acceso: 13 Mayo 2023].

[2] J. L. Alonso, «¿Qué es TensorFlow y para qué sirve?,» 16 Junio 2022. [En línea]. Available: https://www.incentro.com/es-ES/blog/que-es-tensorflow. [Último acceso: 09 Noviembre 2022].

[3] D. G. Ionos, «Jupyter Notebook: documentos web para análisis de datos, código en vivo y mucho más,» 28 Febrero 2019. [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/. [Último acceso: 09 Noviembre 2022].

[4] D. G. F. Ramírez, «SDN con Sistema de Detección de Amenazas,» 2021. [En línea]. Available: https://repositorio.uniandes.edu.co/bitstream/handle/1992/55580/26181.pdf?sequence=1&isAllowed=y. [Último acceso: 13 Mayo 2023].

[5] T. H. N. S. Y. C. Marwa Baich, «Machine Learning for IoT based networks intrusion detection: a comparative study,» Elsevier B.V, 2022. [En línea]. Available: https://doi.org/10.1016/j.procs.2022.12.076. [Último acceso: 13 Mayo 2023].

[6] J. A. a. K. Alsubhi, «Internet of Things Cyber Attacks Detection using Machine Learning,» International Journal of Advanced Computer Science and

Applications(IJACSA), 10(12), 2019. [En línea]. Available: http://dx.doi.org/10.14569/IJACSA.2019.0101280. [Último acceso: 13 mayo 2023].

[7] I. D. Dorvigny Dorvigny, E. D. Barrera Pérez y L. L. Rodríguez Vallejo, «Técnicas de Inteligencia Artificial para Sistemas de Detección de Intrusiones,» UCIENCIA, 2021. [En línea]. Available: https://repositorio.uci.cu/jspui/bitstream/123456789/9679/1/UCIENCIA_2021_paper_224.pdf. [Último acceso: 11 Noviembre 2022].

[8] Rambasnet, « Colorado-Mesa-University-Cybersecurity /DeepLearning-IDS,» 15 Diciembre 2019. [En línea]. Available: https://github.com/Colorado-Mesa-University-Cybersecurity/DeepLearning-IDS. [Último acceso: 17 Octubre 2022].

[9] N. A. a. L. Q. Kumar A, «IoT Network Attack Detection using Supervised Machine Learning,» International Journal of Artificial Intelligence and Expert Systems, 10(2): 18-32., 2021. [En línea]. Available: https://shsu-ir.tdl.org/handle/20.500.11875/3245. [Último acceso: 19 mayo 2023].

[10] M. A.-k. E. A.-H. Ghazi Al-Naymat, «Exploiting SNMP-MIB Data to Detect Network Anomalies using Machine Learning Techniques,» arxiv.org, 2018. [En línea]. Available: https://arxiv.org/ftp/arxiv/papers/1809/1809.02611.pdf. [Último acceso: 19 Mayo 2023].

[11] A. M. M. D. V. S. S. Saikat Das*, «DDoS Intrusion Detection through Machine Learning,» Department of Computer Science The University of Memphis, 2019. [En línea]. Available: https://drsaikatdas.com/papers/ddos_ensemble.pdf. [Último acceso: 14 Mayo 2023].

[12] N. S. M. &. N. Thockchom, «U. A novel ensemble learning-based model for network intrusion detection,» Complex Intell. Syst, 3 Abril 2023. [En línea]. Available: https://doi.org/10.1007/s40747-023-01013-7. [Último acceso: 14 Mayo 2023].

[13] EBF, «Ventajas y desventajas de las metodologías Agile (ágiles),» EBF, 11 septiembre 2019. [En línea]. Available: https://ebf.com.es/blog/ventajas-y-desventajas-de-las-metodologias-agiles-y-su-aplicacion-en-el-trabajo/. [Último acceso: 24 mayo 2023].

[14] «CSE-CIC-IDS2018 on AWS,» Canadian Institue for Cybersecurity, [En línea]. Available: https://www.unb.ca/cic/datasets/ids-2018.html. [Último acceso: 8 Noviembre 2022].

[15] «A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018),» aws, [En línea]. Available: https://registry.opendata.aws/cse-cic-ids2018/. [Último acceso: 8 Noviembre 2022].

[16] jdvelasq, GitHub, [En línea]. Available: https://jdvelasq.github.io/courses/notebooks/sklearn_model_selection_and_evaluation/2-13_roc_curve.html. [Último acceso: 4 Mayo 2023].

[17] M. Sotaquirá, «¿Qué es la Regresión Multiclase?,» codificandobits, 20 Agosto 2018. [En línea]. Available: https://www.codificandobits.com/blog/regresion-multiclase/. [Último acceso: 25 Abril 2023].

[18] S. Mehta, «All you need to know about log loss in machine learning,» aim, 7 Julio 2022. [En línea]. Available: https://analyticsindiamag.com/all-you-need-to-know-about-log-loss-in-machine-learning/. [Último acceso: 13 Mayo 2023].

[19] C. GOYAL, «Bagging- 25 Questions to Test Your Skills on Random Forest Algorithm,» Analystics Vidhya, 24 junio 2022. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/05/bagging-25-questions-to-test-your-skills-on-random-forest-algorithm/. [Último acceso: 20 marzo 2023].

[20] «Entropy and Information Gain to Build Decision Trees in Machine Learning,» Section, 3 Julio 2021. [En línea]. Available: https://www.section.io/engineering-education/entropy-information-gain-machine-learning/. [Último acceso: 2023 Abril 2023].

[21] «JavaTpoint,» JavaTpoint, [En línea]. Available: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. [Último acceso: 20 marzo 2023].

[22] R. Cañadas, «Redes neuronales artificiales,» Abdatum, 4 noviembre 2021. [En línea]. Available: https://abdatum.com/tecnologia/redes-neuronales-artificiales. [Último acceso: 20 marzo 2023].

[23] S. Raschka, «mlxtend,» [En línea]. Available: https://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/. [Último acceso: 9 Mayo 2023].

[24] N. Unidas, «Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/infrastructure/. [Último acceso: 6 Noviembre 2022].

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Entre los diversos objetivos de la ONU llamados ODSs, este proyecto se alinea principalmente con el siguiente objetivo:

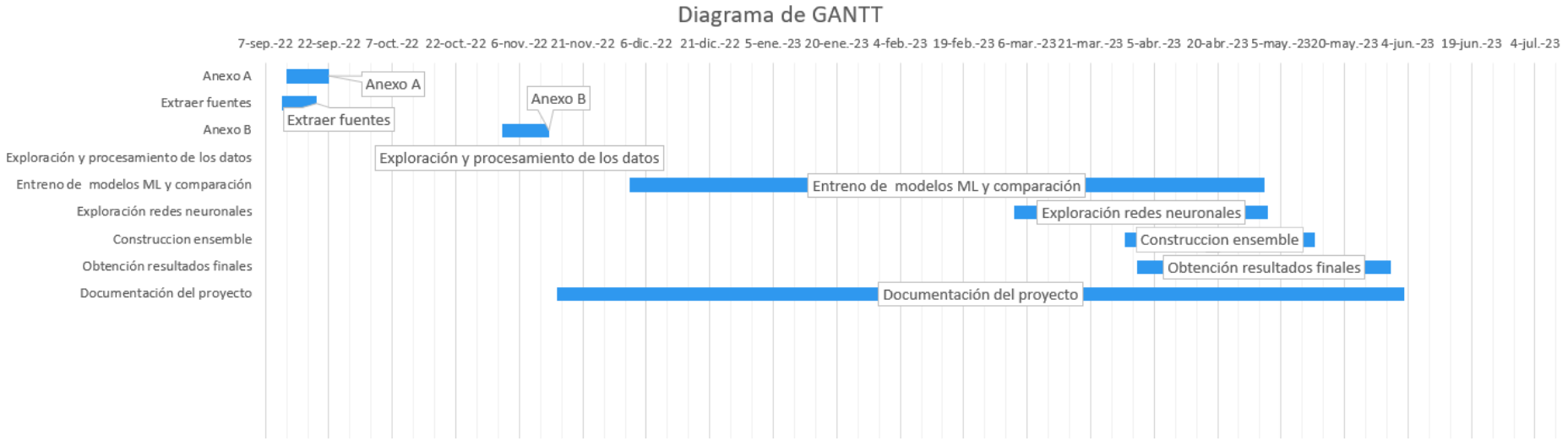
Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.

Ilustración 65 Objetivo 9

La industrialización inclusiva y sostenible, junto con la innovación y la infraestructura, pueden dar rienda suelta a las fuerzas económicas dinámicas y competitivas que generan el empleo y los ingresos. Estas desempeñan un papel clave a la hora de introducir y promover nuevas tecnologías, facilitar el comercio internacional y permitir el uso eficiente de los recursos. [24]

La ciberseguridad es actualmente una de las áreas más importantes y con más evolución de las infraestructuras. Contar con herramientas de prevención y protección con altas precisiones frente a ataques maliciosos es un requerimiento fundamental para las empresas y los gobiernos. Este proyecto busca analizar el rendimiento de varios modelos para la clasificación de varios ataques tanto benignos como maliciosos, y de esta forma, aportar más robustez a los sistemas de prevención de intrusiones. Por ello, se concluye que este trabajo cumple el objetivo número 9.

ANEXO II: DIAGRAMA DE GANTT



ANEXO III: CÓDIGO EMPLEADO

El enlace al repositorio de GitHub donde se encuentra el código de Python empleado en este trabajo es el siguiente:

https://github.com/MiriamCol/TFG-Miriam-Colino-Ruiperez-

El archivo de Jupyter Notebook TFG_MiriamColinoRuiperez.ipynb contiene todo el código. Su estructura es la siguiente:

- Filtrado de datos
- SMOTE y separación en conjunto train y conjunto test
- SGD
  - Entrenamiento y obtención de predicciones SGD
  - Matriz de confusión e informe de clasificación SGD
  - Precisión por clase SGD
  - Curva ROC SGD
  - Tabla con hiperparámetros SGD
- Regresión Logística
  - Entrenamiento y obtención de predicciones Regresión Logística
  - Matriz de confusión e informe de clasificación Regresión Logística
  - Precisión por clase Regresión Logística
  - Curva ROC Regresión Logística
  - Tabla con hiperparámetros Regresión Logística
- Árbol de decisión
  - Entrenamiento y obtención de predicciones Árbol de decisión
  - Matriz de confusión e informe de clasificación Árbol de decisión
  - Precisión por clase Árbol de decisión
  - Curva ROC Árbol de decisión

- Tabla importancia características Árbol de decisión
- Árbol en texto
- Árbol 12 niveles a color y número de hojas totales
- Gráfico variano max_depth Árbol de decisión
- Tabla hiperparámetros Árbol de decisión

- Random Forest
  - Entrenamiento y obtención de predicciones Random Forest
  - Matriz de confusión e informe de clasificación Random Forest
  - Curva ROC Random Forest
  - Precisión por clase Random Forest
  - Tabla importancia características Random Forest
  - Gráfico variando varios n_estimators Random Forest
  - Tabla con hiperparámetros Random Forest

- KNN
  - Entrenamiento y obtención de predicciones KNN
  - Matriz de confusión e informe de clasificación KNN
  - Precisión por clase KNN
  - Tabla variando varios K en KNN
  - Tabla con hiperparámetros KNN
- Ensemble
  - Construcción del Ensemble formado por Árbol de decisión, Random Forest y KNN
  - Matriz de confusión e informe de clasificación Ensemble
  - Precisión por clase Ensemble
  - Ficha Ensemble
  - Gráfico de comparación por precisión por clases de los modelos que conforman el Ensemble + Ensemble
  - Tabla precisiones medias por ataque (clase)

- Red neuronal
    - Entrenamiento y obtención de predicciones Red neuronal
    - Matriz de confusión e informe de clasificación Red neuronal
    - Curva ROC Red neuronal
    - Precisión por clase Red neuronal
    - Gráfica losses-accuracy Red neuronal
    - Tabla hiperparámetros Red neuronal

- Análisis de resultados
    - Tabla precisiones de todos los modelos
    - Gráfica tiempo-precisión de todos los modelos