Volume: 13 Issue: VII Month of publication: July 2025
DOI: https://doi.org/10.22214/ijraset.2025.73398

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VII July 2025- Available at www.ijraset.com

Intrusion Detection System for Network Attacks

Rakesh Das[1], Gudikandhula Narasimha Rao[2]
[1]*Department Of Information Technology & Computer Applications, Andhra University College of Engineering(A), Andhra University, Visakhapatnam, Andhra Pradesh - 530003*
[2] *Professor Department of CS&SE, Andhra University College of Engineering(A), Andhra University, Visakhapatnam, Andhra Pradesh - 530003*

Abstract: Network security has become a top priority for both businesses and individuals in recent years because of the rise in cyber threats. Finding unauthorized or harmful activities on a network is one of the biggest problems. Traditional Intrusion Detection Systems (IDS) often miss new or changing attack patterns because they use fixed rules or known attack signatures. Machine learning techniques are being used more and more to build intelligent and adaptive IDS models to get around this problem. Using the NSL-KDD dataset, we built an intrusion detection system based on machine learning for this Research. Because of its better structure and balanced records, the NSL-KDD dataset is a better version of the original KDD'99 dataset and is often used to compare IDS research. The Research has several steps, such as preparing the data, encoding the features, doing exploratory data analysis, and putting classification algorithms into action. We trained and tested two models, Logistic Regression and XGBoost, using metrics like accuracy, precision, recall, F1-score, and ROC-AUC. XGBoost. The system does a good job of telling the difference between normal and harmful network connections, and it can be improved even more for use in real time. This study shows that machine learning can help make networks more secure and lays the groundwork for creating scalable, smart, and real-time intrusion detection systems.

I. INTRODUCTION

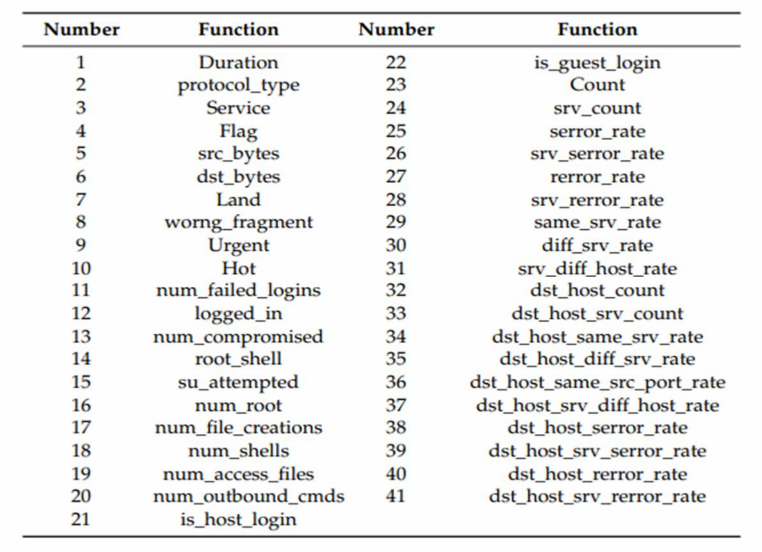
With the fast growth of the internet and digital tools, communication through networks has become a key part of how industries, governments, and people around the world operate. However, this progress has also brought a lot of cyber dangers and harmful activities, like data leaks and denial-of-service (DoS) attacks. Because of this, making network environments secure is now a top priority. Among the security tools available, Intrusion Detection Systems (IDS) are important for finding and dealing with unwanted access or strange behaviour on a network. Most traditional IDS systems work by using signature-based methods, which means they look for known attack patterns and rules.

These systems are good at finding threats they've seen before, but they can't detect new threats, like zero-day attacks or advanced malware that doesn't match known patterns. Also, these systems need regular updates by hand and often give false warnings or miss real threats. These problems have led experts to look into machine learning (ML) for IDS, which can learn from past data and adapt to find both known and unknown threats. In this research, I developed a machine learning-based IDS model that was trained and tested using the NSL-KDD dataset.

This dataset is an improved version of the KDD Cup 1999 dataset and fixes problems like repeated records and an uneven mix of normal and attack data, making it more reliable for training and testing machine learning models. The NSL-KDD dataset includes various types of network traffic, which are labelled as either normal or one of several attack types, such as DoS, Probe, R2L, and U2R.The aim of this study was to create a strong system that can tell the difference between normal and harmful traffic. The work involved preparing the data, turning categorical information into numbers, exploring the data, and applying classification methods. Specifically, I used and compared two supervised learning models: Logistic Regression, which is a common starting point for classification, and XGBoost, a powerful gradient boosting model known for its accuracy and speed. To measure how well the models worked, I used standard classification metrics like accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC). The results showed that XGBoost performed better than Logistic Regression, with higher detection rates and the ability to find important features that help in accurate classification. This research shows that using machine learning can significantly improve the accuracy and smartness of intrusion detection systems. The findings from this study support the development of better and more flexible network security solutions and set the stage for future work on real-time IDS, deep learning models, and handling large amounts of data.

II. DATASET DESCRIPTION

The dataset used in this research is the NSL-KDD dataset, which is a commonly used standard for testing how well intrusion detection systems (IDS) work. It is an improved version of the original KDD Cup 1999 dataset, which had some problems like too many repeated and similar records that sometimes caused unfair or biased results. The NSL-KDD dataset fixes these issues by removing duplicates and making the data more balanced, which helps create a fairer and more accurate way to train and test models.
This dataset includes records of network connections that are labelled, with each record having 41 features that describe different aspects of the traffic, such as basic connection details, content features, and statistical information about the traffic.
The last column in each record shows the label, indicating whether the connection is normal or an attack. The attacks are grouped into four main types: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). These categories show the different ways attackers might try to harm systems, scan networks, get in without permission, or take on more control.
The dataset is split into two parts: one for training and one for testing, called KDDTrain+ and KDDTest+. Each part has thousands of records that include both normal and harmful traffic. During the preprocessing stage of this Research, we converted categorical data such as protocol type, service, and flag into numbers using encoding methods, and we scaled numerical values to help the machine learning models work better with the data.



Number	Function	Number	Function
1	Duration	22	is_guest_login
2	protocol_type	23	Count
3	Service	24	srv_count
4	Flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	Land	28	srv_rerror_rate
8	worng_fragment	29	same_srv_rate
9	Urgent	30	diff_srv_rate
10	Hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login		

Table-2.1: NSL-KDD dataset features

III. ARCHITECTURE AND METHODOLOGY

The design of this Intrusion Detection System (IDS) relies on a machine learning process that uses the NSL-KDD dataset. This dataset includes labelled network traffic, covering both normal activity and different types of attacks. The process starts with data preparation, which involves converting categorical data like protocol type, service, and flag into a format that can be used by machine learning models, as well as scaling numerical features to a standard range. Once the data is ready, feature selection is carried out using XGBoost to determine which features are most important. Then, several machine learning models, such as Logistic Regression, Decision Tree, and XGBoost, are trained and tested on the cleaned data. The performance of these models is assessed using metrics like accuracy, precision, recall, and F1-score. XGBoost performs the best because it offers high accuracy, handles feature well, and reduces overfitting. This organized method helps in achieving efficient and dependable intrusion detection.

A. System Architecture Overview

The system is built in four main steps: Data Acquisition, Preprocessing, Model Training & Evaluation, and Intrusion Detection. The NSL-KDD dataset is used as the input, which includes records that are labelled as either normal or showing signs of an attack.
In the Preprocessing stage, features that are in categories are converted into numbers, and numerical data is adjusted so it's ready for the models. Then, different machine learning models like Logistic Regression, Decision Tree, and XGBoost are trained and tested. The importance of each feature is also figured out using XGBoost.
Once the models are trained, they are used in the Intrusion Detection stage to determine if new network traffic is normal or harmful. The system is made to handle large amounts of data, work accurately, and effectively spot various kinds of cyber threats.

B. UML Design Overview

The UML (Unified Modeling Language) design for this Research shows how the system is built and how data moves through different parts of the Intrusion Detection System. It includes three main diagrams: the Use Case Diagram, the Class Diagram, and the Activity Diagram. These diagrams explain how the user, the system, and the machine learning models work together.

The Use Case Diagram shows how the user, such as an analyst or administrator, interacts with the system. It includes important actions like uploading a dataset, preparing the data, training a model, checking how well it works, and identifying intrusions. This diagram explains the main functions the system needs to perform.

The Class Diagram shows the system's internal parts. It includes classes such as Dataset Loader, Preprocessor, Model Trainer, and Evaluator. Each class has specific information, like input data or the type of model used, and actions, such as train_model (), evaluate (), and predict (). These classes work together to make the intrusion detection system function properly

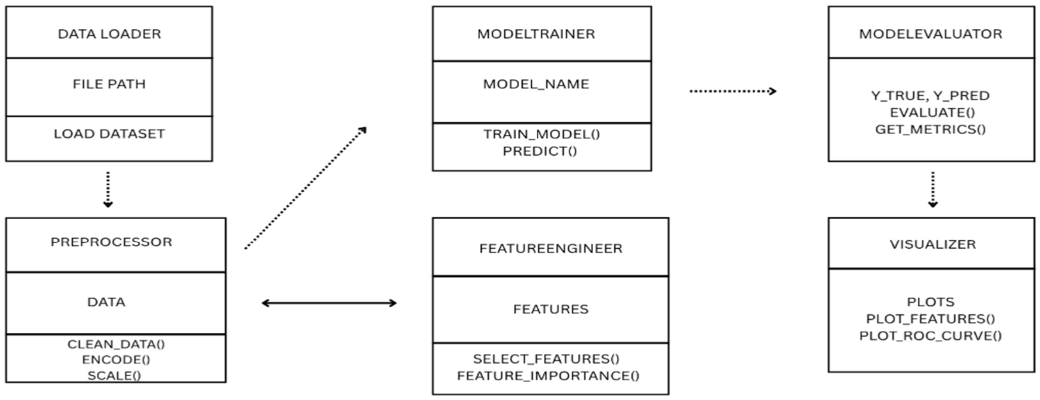


Figure-3.1. Class Diagram for data engineering

The Activity Diagram shows how control moves throughout the intrusion detection process. It starts with data being input, then moves on to preprocessing, choosing important features, training the model, and testing it. Finally, it ends with classifying the i ntrusion. This diagram clearly shows each step of the IDS, from the original data to predicting threats.

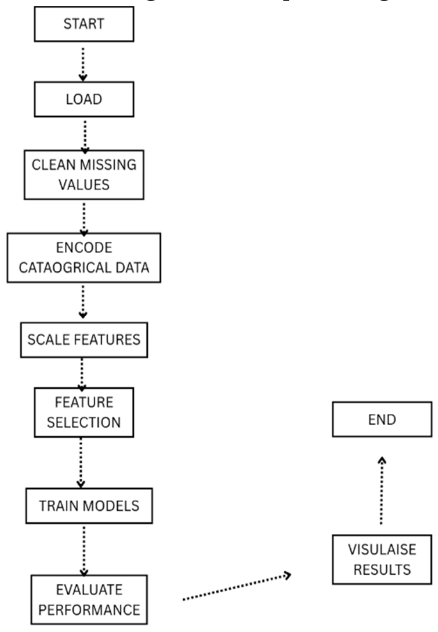


Figure-3.2. Activity Diagram for Model making

C. Data Preprocessing

In this research, data preprocessing was important to get the NSL-KDD dataset ready for training the model. Features like protocol_type, service, and flag were turned into numbers using label encoding so they could be used by machine learning models. Then, numerical features were scaled using Min-Max Normalization to make sure all the data was on the same scale.

Unnecessary or repeated features were removed by looking at which ones were most important using XGBoost. This helped make the model more accurate and simpler. After cleaning and preparing the data, it was split into training and testing parts so the models could be tested properly. To avoid the model being biased toward the most common type of attack, class imbalance was handled. These steps made sure the input data was of good quality, which helped in detecting intrusions accurately.

D. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is important for understanding the structure, patterns, and connections in a dataset before using any machine learning models. In this Research, EDA was done on the NSL-KDD dataset to learn more about how the features are distributed, how balanced the attack and normal classes are, and how different variables relate to each other.

At first, the dataset was looked at to see how the classes were distributed. It was found that the data is not balanced some attack types, like 'DoS' and 'Probe', appear more often than others, such as 'R2L' or 'U2R'. Bar charts and count plots were used to show how many samples there were in each class, which helped in understanding the need for methods to handle the imbalance. Then, the categorical features like protocol_type, service, and flag were analyzed to see how often they occurred and how they connected to different types of intrusions.

Heatmaps and correlation matrices were used to look at how numerical features relate to each other, making it clear which variables were closely connected and which might be removed or combined. Another key part of EDA was looking at the distribution of continuous features. DE plots and histograms were used to see how the features are spread out, find any outliers, and compare how different attributes behave in normal versus attack situations.

The insights from EDA helped guide data cleaning, choosing important features, and planning the model training process. It provided a better understanding of the dataset, which is essential for creating a trustworthy and accurate intrusion detection system.

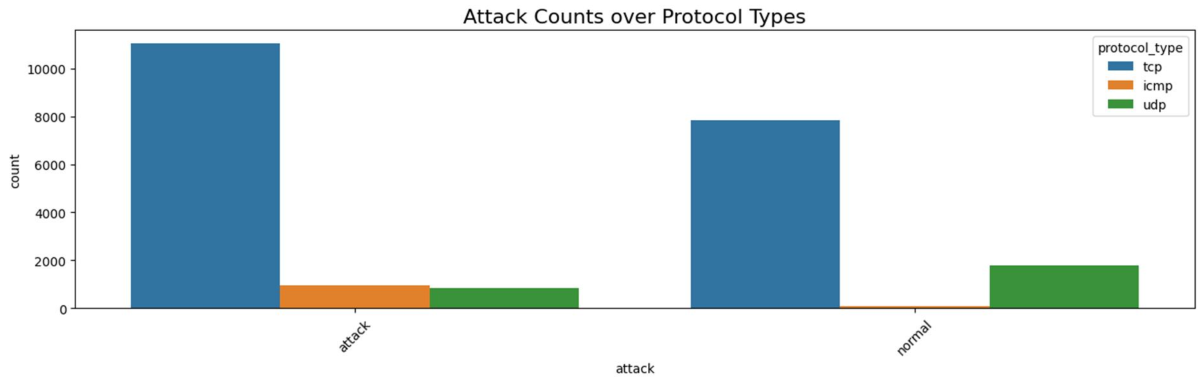


Figure 3.4. Protocol types vs Attacks

E. Feature Selection and Transformation

In this Research, feature selection was done to find the most important attributes from the NSL-KDD dataset, which originally has 41 features. Not every feature is equally useful for classifying network traffic, so reducing the number of features helps make the model more efficient and accurate. We used XGBoost's feature importance score to rank and select the most significant features based on how much they affect the model's performance. After choosing the important features, we transformed the data to make it suitable for machine learning models.

Categorical features like protocol_type, service, and flag were encoded using Label Encoding, and in some cases, One-Hot Encoding. These transformations help the models understand the data correctly and handle categorical values properly. Additionally, numerical features were normalized using Min-Max Scaling to bring all values into a consistent range between 0 and 1.

This step ensures that features with larger ranges don't have too much influence during training. Overall, feature selection and transformation helped create a more accurate, faster, and reliable intrusion detection system.

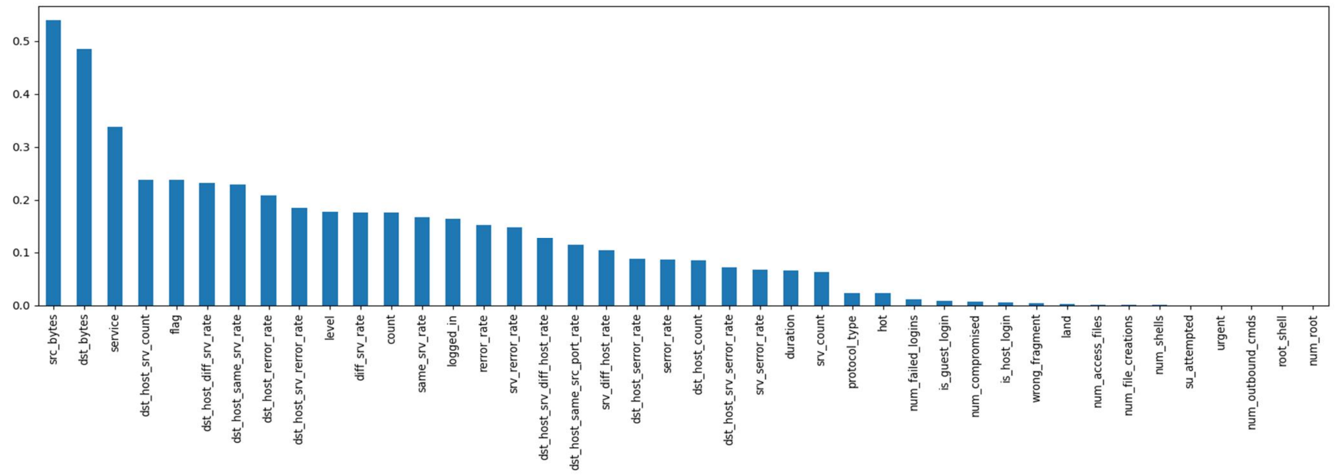


Figure-3.5: Feature Selection For training data set

F. Machine Learning Models Used

In this Research, several supervised machine learning algorithms were used and tested to detect network intrusions using the NSL-KDD dataset. The models were chosen to determine if a network connection is normal or an attack, and in some cases, to identify the type of attack. Each model was selected based on its strengths in classification tasks, and they were evaluated using performance metrics like accuracy, precision, recall, and F1-score. The first model used was Logistic Regression, a linear classifier known for being fast and easy to use. It served as a starting point for comparison and performed well on binary classification tasks. In this Research, it showed good results when the dataset was properly cleaned and encoded, showing how well it can identify linear patterns in the data. The most successful model was XGBoost, which is a strong and efficient algorithm that uses many small models, called decision trees, in sequence to reduce errors. XGBoost achieved the highest accuracy among the models tested. It also helped in understanding which features were most important, allowing us to focus on the key ones and improve the model's performance while reducing overfitting. XGBoost was especially useful for this intrusion detection task because it can handle messy data and a large number of features. Another model used was the Decision Tree Classifier. While not as strong as XGBoost, it was transparent in how it made decisions and worked well with both types of data—categorical and numerical. However, it had some issues with overfitting the training data, which showed why using more advanced methods like XGBoost can be beneficial.

By comparing these models, the Research showed that using ensemble methods, especially XGBoost, gives better performance and reliability in detecting network intrusions. The detailed results from the experiments confirmed that XGBoost performed better than the other models in both simple and complex classification tasks.

G. Evaluation Metrics and Performance Analysis

To evaluate how well the machine learning models worked in this Intrusion Detection System (IDS) Research, several important measures were used: accuracy, precision, recall, and F1-score. These measures help understand how effectively the models can tell the difference between normal and abnormal network activity, especially when the dataset has an uneven distribution of normal and attack data. Accuracy tells us how often the model makes the right prediction out of all predictions it makes. It gives a general sense of how well the model performs, but it might not be the best measure when the data is not balanced, which is often the case in intrusion detection.

Precision shows how many of the predicted attacks are actually real attacks. It tells us how accurate the model is when it flags something as an attack. Recall shows how many of the actual attacks the model correctly identifies. It tells us how good the model is at catching all the attacks. The F1-score is a way to balance precision and recall. It helps when the cost of missing an attack (a false negative) is different from the cost of wrongly flagging a normal activity as an attack (a false positive).

In this Research, XGBoost performed better than other models in most of these measures. It had the highest accuracy and F1-score, showing that it's very effective at detecting different types of attacks while keeping false alarms low. Logistic Regression, which is simpler, also gave good results and served as a baseline model for comparison. The Decision Tree Classifier was easy to understand but tended to overfit the data. These results confirm that XGBoost is a good choice for intrusion detection because it handles data well, is strong at classification, and is reliable.

IV. EXPERIMENT AND RESULTS

In this research, I used several machine learning models to identify intrusions using the NSL-KDD dataset, which is a commonly used benchmark for intrusion detection systems. The models were trained to classify network traffic as either normal or an attack, and their performance was assessed based on how well they could classify these types of traffic. In this study, I used logistic regression and xGBoost to compare the differences between the two algorithms. The results clearly explain how each algorithm performs and are easy to understand.

The evaluation metrics used in this study include Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.

These metrics were selected to give a balanced assessment of each model's ability to detect both normal and malicious network traffic.

The results are based on the classification report, which includes the following metrics: accuracy, precision, recall, F1-score, and support.

These values show the performance of both algorithms used in the research. Accuracy measures how many of the total predictions were correct. Precision refers to the proportion of detected attacks that are actually real attacks, which helps in reducing false alarms. Recall is the proportion of actual attacks that were correctly identified, helping to reduce missed attacks. The F1-score is a balance between precision and recall. ROC-AUC measures the overall ability of the model to distinguish between different classes; a higher value closer to 1. 0 indicates better performance.

A. *Logistic Regression Results*

Test Set

	Precision	Recall	F1-Score	Support
Class 0	0.87	0.94	0.90	1252
Class 1	0.91	0.82	0.87	1003
Accuracy			0.89	2255
Macro Avg	0.89	0.88	0.88	2255
Weighted Avg	0.89	0.89	0.89	2255

Table 4.1: Logistic Regression Test Set Results

Train Set

	Precision	Recall	F1-Score	Support
Class 0	0.88	0.93	0.91	11580
Class 1	0.91	0.84	0.87	8708
Accuracy	—	—	0.89	20288
Macro Avg	0.89	0.89	0.89	20288
Weighted Avg	0.89	0.89	0.89	20288

Table 4.2: Logistic Regression Train Set Results

B. *Train Set Performance*

On the training data, the model achieved an accuracy of 0.89. For Class 0, the F1-score was 0. 91, and for Class 1, it was 0. 87. These results are similar to what was seen on the test set, which suggests the model isn't overfitting and is performing well on new, unseen data. The recall and precision values are fairly balanced, with a macro and weighted average of 0. 89 across all metrics.

C. *Test Set Performance*

The Logistic Regression model achieved an overall accuracy of 0.

89 on the test set, meaning it correctly classified 89% of the samples. The F1-score was 0. 90 for Class 0 and 0. 87 for Class 1, showing that the model did slightly better at identifying benign traffic than malicious activity. The precision for Class 1 was higher (0. 91) compared to Class 0 (0. 87), which means the model was more confident in predicting attacks. However, the recall for Class 1 was relatively low at 0. 82, suggesting that some attacks were missed.

Logistic Regression performed reasonably well on the majority classes normal and attack which are labeled as Class 0 and Class 1 in the metrics above.

However, it had difficulty identifying less common attack types such as U2R and R2L. Most of the misclassifications involved rare attacks being incorrectly predicted as normal or as more common attacks.
In order to understand the other algorithm as discussed, we went through the xGBoost gradient algorithm.
This result was achieved after using the tuning algorithm to get the best results so far.
Below are the details discussed regarding the boost train and test performance.

D. XGBoost: Results

Test Set

Class	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.98	1252
1	0.98	0.99	0.98	1003
Accuracy	—	—	0.98	2255
Macro Avg	0.98	0.98	0.98	2255
Weighted Avg	0.98	0.98	0.98	2255

Table 4.3: XGBoost Test Set Results

Train Set

Class	Precision	Recall	F1-Score	Support
0	0.99	0.99	0.99	11580
1	0.99	0.99	0.99	8708
Accuracy	—	—	0.99	20288
Macro Avg	0.99	0.99	0.99	20288
Weighted Avg	0.99	0.99	0.99	20288

Table 4.4: XGBoost Train Set Results

E. Train Set Performance

The model shows very strong results when tested on the training data, with an overall accuracy of 0.99. It performs equally well for both classes, achieving precision, recall, and F1-score of 0. 99 in each case. This shows that the model is very accurate on the training set, but it also brings up a small concern about possible overfitting, which is mentioned in the conclusion.

F. Test Set Performance

The XGBoost classifier performed very well on the test data, achieving an accuracy of 0.98. Both classes benign (Class 0) and malicious (Class 1) were identified with high precision and recall. For Class 0, the model had a precision of 0. 99 and a recall of 0. 98, while for Class 1, it had a precision of 0. 98 and a recall of 0. 99. The F1-score was the same for both classes at 0. 98, showing that the model has a good balance in its predictions, with few false positives and false negatives.

V. CONCLUSION

In this research, I created a machine learning-based Intrusion Detection System (IDS) using the NSL-KDD dataset, which is a widely used benchmark for network intrusion detection research. The main goal was to classify network traffic as either normal or an attack, helping in early detection and prevention of cyber threats.
After cleaning the data, encoding it, and visualizing it, I trained several models such as Logistic Regression and XGBoost.
The dataset was divided into training and testing parts, and the models were evaluated using standard classification metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
Among the models tested, XGBoost performed the best with the following results:
- Accuracy: 0.98%
- Precision: 0.98%
- Recall: 0.99%
- F1-score: 0.87%

These results show that XGBoost is very effective at detecting both normal and malicious traffic, and it outperforms traditional models like Logistic Regression.
One of the main advantages of XGBoost is its ability to work with high-dimensional and imbalanced data, along with built-in regularization to prevent overfitting.
Additionally, Kernel Density Estimation (KDE) was used during the exploratory data analysis to understand how the features were distributed and to spot differences between normal and attack traffic.
This helped in choosing and creating meaningful features, which improved the model's ability to tell the two categories apart.
The entire process, from data preparation, feature creation, model training, and evaluation, was done efficiently using Python and libraries like Pandas, Scikit-learn, Matplotlib, and XGBoost.
In conclusion, this Research shows how powerful machine learning, especially techniques like XGBoost, can be in building accurate and scalable intrusion detection systems.
These results suggest that such models can be integrated into real-world cybersecurity systems for real-time monitoring and threat prevention.

VI. FUTURE SCOPE

This research focuses on finding and using the most important features from the NSL-KDD dataset to make machine learning models better at detecting network intrusions. The dataset originally has 41 features that describe different parts of network traffic, like the type of protocol, the service being used, flags, source and destination bytes, and various behavior patterns. Together, these features show what normal and harmful activities look like in a network.
To make the models work better, we used feature selection methods to reduce the number of features, remove unnecessary or irrelevant data, and keep only the most important ones that help in accurate classification.
This not only made the models run faster and use less computer power but also improved how well they can detect intrusions by reducing noise in the data. We also applied transformations like one-hot encoding and normalization to make the features more suitable for the learning algorithms. The work done on feature engineering and selection is very important for building a strong and scalable intrusion detection system, as it directly affects how well the models can learn and apply what they've learned to new data.

REFERENCES

[1] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA) (pp. 1-6). IEEE. https://doi.org/10.1109/CISDA.2009.5356528
[2] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM. https://doi.org/10.1145/2939672.2939785
[3] Scikit-learn Developers. (2023). *Scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/
[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
[5] NSL-KDD Dataset. (2009). A Refined KDD99 Dataset for Intrusion Detection Research. Canadian Institute for Cybersecurity. https://www.unb.ca/cic/datasets/nsl.html
[6] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.
[7] Zhang, J., & Zulkernine, M. (2006). Anomaly based network intrusion detection with unsupervised outlier detection. In Proceedings of the IEEE International Conference on Communications (Vol. 5, pp. 2388-2393). IEEE.
[8] Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. Journal of Computational Science, 25, 152-160. https://doi.org/10.1016/j.jocs.2017.03.006
[9] XGBoost Documentation. (2024). XGBoost Python Package. https://xgboost.readthedocs.io/
[10] Rao, Gudikandhula Narasimha, et al. "Fire detection in kambalakonda reserved forest, visakhapatnam, Andhra pradesh, India: An internet of things approach." Materials Today: Proceedings 5.1 (2018): 1162-1168.
[11] ovith, A. Arokiaraj, et al. "DNA Computing with Water Strider Based Vector Quantization for Data Storage Systems." Computers, Materials & Continua 74.3 (2023).
[12] Rao, Gudikandhula Narasimha, et al. "Geospatial Study on Forest Fire Disasters–A GIS Approach." Ecological Engineering & Environmental Technology 24 (2023)