NFSU JOURNAL OF CYBER SECURITY & DIGITAL FORENSICS

NFSU – Journal of Cyber Security and Digital Forensics
Volume –3, Issue – 1, June 2024
E – ISSN – 2583-7559

A COMPARATIVE EVALUATION OF MACHINE LEARNING ALGORITHMS FOR NETWORK INTRUSION DETECTION

Kavalla Prasanna

School of Cyber Security & Digital Forensics.
National Forensic Sciences University
Gandhinagar,Gujarat,India
Email address: kavalla.btmtcs2116@nfsu.ac.in

Dr.Ujjaval Patel

School of Cyber Security & Digital Forensics.
National Forensic Sciences University
Gandhinagar,Gujarat,India
Email address: ujjaval.patel@nfsu.ac.in

***Abstract*— Network intrusion detection systems (NIDS) are essential for cyber security since they help find and prevent malicious activities in the network traffic. This research uses a dataset from Kaggle and various machine learning methods to achieve a more effective NIDS. For analyzing dataset, an initial phase of the research a Recursive Feature Elimination (RFE) algorithm was used to identify the most significant features for the Random Forest Classifier. Further, Optuna has been used to find the best model and adjust its settings when utilizing three types of classifiers: Decision Tree, K-Nearest Neighbors (KNN) and Logistic Regression. All these algorithms are used to detect malicious activities for enhancing network security, and improving threat detection strategy. An exhaustive comparative evaluation demonstrates that Decision Tree algorithms outperform other algorithms for the effective network intrusion detection in adaptive manner with much higher accuracy.**

Keywords— Network Intrusion Detection, Machine Learning, Feature Selection, Hyperparameter Optimization, Logistic Regression, K-Nearest Neighbors, Decision Tree, Optuna.

I. INTRODUCTION

In today's world, where everything is becoming more digital, protecting our information and communication is very important. Network Intrusion Detection Systems (NIDS) designed to identify and prevent malicious activities targeting network data. Old ways of detecting these problems, which mostly use known patterns (traditional method), often have trouble finding new or tricky attacks because they depend on already known patterns. This shows that we need smarter and more flexible ways to protect ourselves from changing threats. In recent years, machine learning (ML) has become more common, and it looks like it could be a good way to make our protection better and more adaptable [1-2].

For my research, machine learning algorithms were utilized to train the NIDS. A detailed analysis of the performance of each model will be conducted, specifically focusing on Logistic Regression, K-Nearest Neighbors (KNN), and Decision Trees. These techniques are capable of identifying patterns in historical network traffic data that indicate malicious activities. Moreover, they allow for customization approaches to be taken in addressing evolving threats.

This paper analyses the application of various machine learning algorithms to NIDS and their reference in respect to the means for improving the same system in accuracy and performance [2].

I selected a Kaggle dataset for several significant reasons. First, it boasts a wide variety of attacks and scenarios providing a comprehensive view of the current threat landscape. Such this variety is critical in training models generalizable across various types of intrusion. Second, this dataset comes from real-world network traffic, presenting a more accurate picture of normal user behavior and network conditions. Such authenticity adds extra reliability going further to use such machine learning models applicable to the real-windows space [4].

Kaggle solved the problem of redundancy and duplication found in older datasets like KDD99 and NSL-KDD. This marked an obvious reduction in bias, as classifiers are always trained on entirely different instances. It provides a wider and more relevant attribute set, thus allowing effective feature selection and hyperparameter tuning-the two processes vital for enhancing the model performance in intrusion detection.

1.1 Literature Review

1.1.1 Traditional NIDS Approaches

Traditional network intrusion detection systems primarily rely on signature or rule clashing to differentiate between normal and abnormal behavior in network traffic. Depending on the generation of rules or signatures, these systems can be broadly categorized into two types: misuse detection and anomaly detection. In misuse detection approaches, certain attack scenarios are used to produce a certain set of rules based on known vulnerabilities and exploits. The incoming traffic is then matched against this predetermined rule set. Many of the more conventional systems utilize pattern-matching algorithms to find specific-valued patterns in designated packet fields *[4]*.

Recent advances have emphasized the need for self-adaptive intrusion detection systems that can learn from network traffic data in real-time. For example, the development of hybrid models combining signature-based detection with machine learning allows the system to maintain the reliability of signature matching while also benefiting from the adaptability of machine learning algorithms. Moreover, network security requires emerging zero-trust architectures requiring a change of NIDS methodological approach to

discover anomalies in completely decentralized and dynamic environments[4].

Emerging work suggests incorporating Autonomous Systems It integrates into NIDS, where the system would be actively reasoning about potential threats in real-time and enhance its detection capabilities. By mimicing abilities like pattern recognition and decision-making, such systems could well be the future leap in adaptive and proactive intrusion detection.

1.1.2 Machine Learning in NIDS

In particular, machine learning algorithms are essential because they can dramatically increase the capability for automated detection of intrusions. Unlike signature based techniques in traditional approaches, which are very good at detecting known patterns but struggle to identify new or unknown attacks, ML leverages network data from the past and can serve as a better means of adjusting to movements by attackers. In fact, this ability to change makes them more effective at recognizing not just known but also unknown intrusions.

Logistic Regression, K-Nearest Neighbors, Decision Trees etc are some of the most widely used algorithms in NIDS. Although simple, Logistic Regression may be a great tool to detect unusual behavior in network traffic. Detection of normal activities or malicious behaviour with high accuracy is makes it a great fit for NIDS[16].

Decision Trees are commonly used in detecting anomalies and identifying misuse. They offer a simple and understandable structure that helps to recognize attack patterns from past data. Their ability to handle large amounts of data efficiently and maintain good detection rates makes them a strong choice for network intrusion detection systems (NIDS) [17]. KNN, known for its simplicity and ease of use, can struggle with larger datasets, which might reduce its effectiveness in some situations.

The success of machine learning in NIDS is strongly tied to the quality of the training data. Diverse datasets, like those found on Kaggle, give models real-world traffic examples that include both normal and harmful behaviors, helping them to better handle different types of attacks. Also, fine-tuning these models by choosing the right features and adjusting parameters is crucial for getting accurate and efficient performance in detecting intrusions [16-18].

1.1.3 Hyperparameter Tuning is one of the aspects for optimizing ML models and using toolkits like Optuna for automating searching for the best hyperparameters to improve model performance [5]. Better hyperparameter settings can boost the accuracy and effectiveness of NIDS greatly [8].

1.1.4 Feature selection of ML for NIDS, feature selection is critical to enhance model accuracy and efficiency. Techniques used in feature selection reduce the dimensionality of data by removing irrelevant or redundant features that could degrade the model's performance. Recursive Feature Elimination (RFE) etc used for identifying most significant features for intrusion detection [5][7] Preprocessing steps i.e label encoding, standard scaling etc are also needed to prepare data for ML algorithms to make sure that the models are learning from coherent and normalized data. Recursive Feature Elimination is a popular technique for selecting features by iteratively removing the least important features and assessing the model's performance. However, RFE is only one of the feature selection techniques available, and recent research also presents alternatives, such as mutual information and correlation-based feature selection, among others. Beyond feature selection, hyperparameter optimization is another key area of improvement to enhance NIDS performance. Tools such as Optuna and GridSearchCV automate the search for the best hyperparameters to fine-tune ML models so that they maximize detection accuracy. In combination, feature selection and hyperparameter optimization form a powerful combination in enhancing the capabilities of NIDS to better identify known as well as previously unseen attacks.

1.2 Research Gap

Despite the advancements in applying system learning to NIDS, several gaps continue to be in the present-day studies

1.2.1 Overall Comparison of ML Algorithms

Although several works have separately discussed various ML algorithms, there is a need to compare the diverse set of algorithms under the same framework. Also, many current studies frequently focus on a narrow range of interests and do not provide a comparative evaluation of multiple algorithms on the same platform [10]. In this paper, the performance comparison between Logistic Regression, KNN, and Decision Trees was done under the same experimental environment [9].

1.2.2 Feature selection and hyperparameter optimization

Although numerous works treat feature selection and hyperparameter optimization as different tasks, their integration leads to considerable improvement in the performance of the model. This method ensures that the best features selected simultaneously optimize the model's hyperparameters, making the NIDS more efficient and robut [7]. This paper seeks to embed both strategies in the model training and assessment stages to fill this gap.

1.2.3 Evaluation Metrics and Performance Metrics While accuracy is commonly employed to evaluate machine learning models, additional metrics such as precision and F1-score are crucial for a thorough evaluation of model performance, particularly in the context of intrusion detection [12]. This paper highlights the significance of these metrics and provides an in-depth analysis of various models, emphasizing the importance of cross-validation to ensure a reliable performance evaluation.

II. METHODOLOGY

In this research, the dataset sourced from Kaggle contains a set of network traffic facts with labels indicating whether the activity is an ordinary or an intrusion. Preparing these

statistics for machine gaining knowledge of models concerned several preprocessing steps, which include:

Label Encoding: Converting non-numeric express capabilities into numeric values so that gadget learning algorithms can procedure them correctly.
Standardization: Adjusting the feature values so that they have got a mean of 0 and a general deviation of 1. This is an important step, especially for algorithms like K-Nearest Neighbors and Logistic Regression, as it improves their performance and convergence velocity.
Fig. 1 shows The dataset undergoes preprocessing steps, which encompass label encoding of express capabilities and fashionable scaling of numerical information to normalize the dataset[11].Preprocessing is crucial for improving model accuracy and efficiency by making the data more compatible with the chosen algorithms and preventing potential issues related to feature scaling.

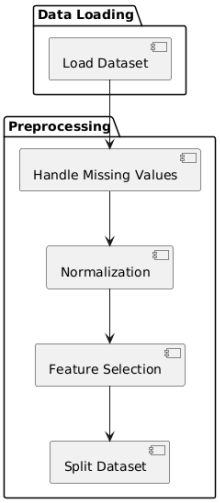

Figure 1 Data Loading and Preprocessing

2.1 Feature Selection Techniques
2.1.1 The Recursive Feature Elimination (RFE) technique is hired to pick out the maximum wide features from the dataset. RFE works with the aid of recursively doing away with the least crucial features and constructing a version with the last ones until the desired variety of features is reached [7],[16]. In this implementation, RFE is used with a Random Forest Classifier to pick the pinnacle 10 capabilities from the training statistics. The Random Forest Classifier ranks the capabilities based on their importance to the version's predictive energy, and RFE iteratively gets rid of the least critical features, refining the selection to hold best those that make a contribution the most to the model's overall performance.

2.1.2 Feature Importance the use of Random Forest:
The Random Forest Classifier is utilized to evaluate function importance directly. This technique entails training a random wooded area model that inherently evaluates the relevance of each function primarily based on how it affects the model's predictions. Features are ranked consistent with their contribution to lowering the impurity or enhancing the version's accuracy [12]. In this implementation, the Random Forest Classifier is included with RFE to identify and retain the maximum informative features, thereby perfecting the model's performance and lowering the chance of overfitting.

Fig.2 illustrates This step reduces dimensionality and improves the overall performance of machine learning fashions by means of fastening at the most informative attributes.

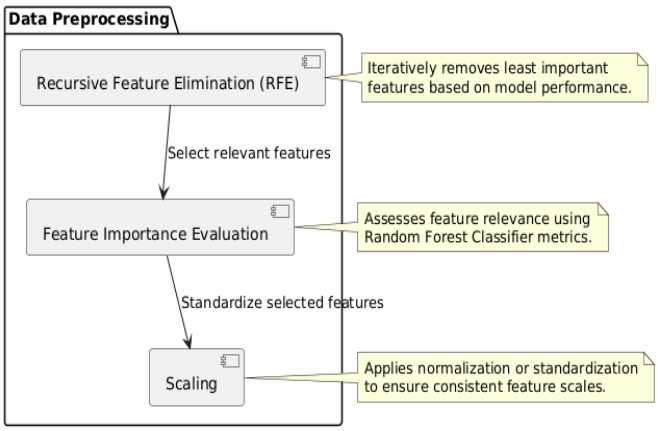

Figure 2 Feature Selection and Scaling

2.2 Machine Learning Algorithms
This challenge involves three systems getting to know algorithms: Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree Classifier. These algorithms have been selected for their effectiveness in class responsibilities and had been carried out to the dataset to expect class labels. The overall performance of each model becomes evaluated using accuracy, precision, and different relevant metrics.
2.2.1 Logistic Regression is a statistical method used for binary class issues. It predicts the opportunity of a target variable belonging to a specific elegance based totally at the enter functions [20]. In this paper, Logistic Regression is used to version the connection among the independent variables and the binary goal variable (regular or anomaly) [16]. The

model turned into training with a maximum generation restriction of 100 to make sure convergence.

Fig. 3 illustrates the methodology used for Logistic Regression.

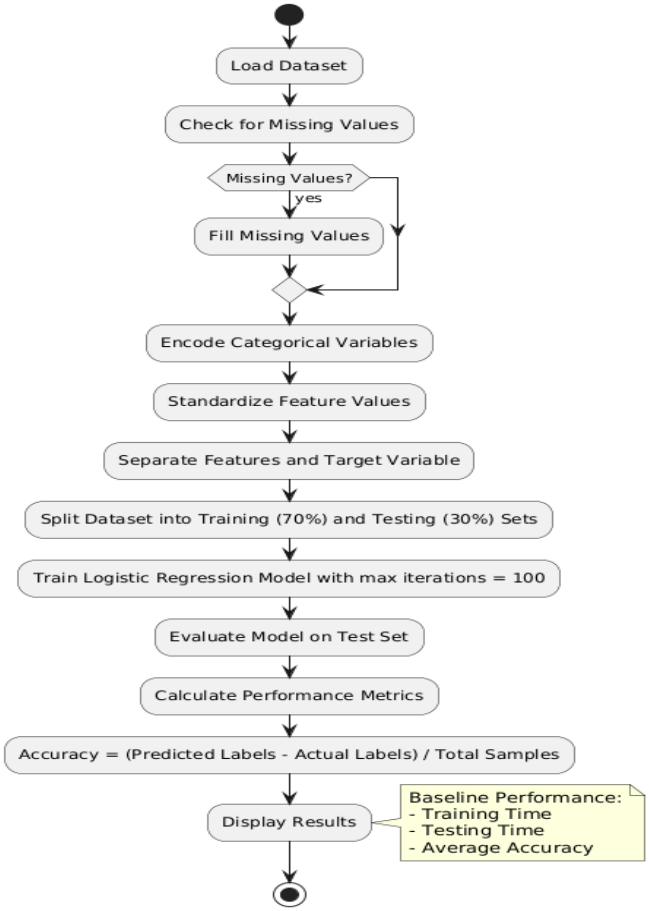

Figure 3 Logistic Regression Process Flowchart

Training and Testing: The dataset is split, with 70% allocated for training and 30% reserved for testing. The training subset was used to maintain the model, while the checking out subset changed into employed to evaluate the version's overall performance [9].
Performance Metrics: The model's accuracy on training and testing records turned into measured. The accuracy became calculated by evaluating the predicted labels with the real labels. These calculations are outlined in the next sections.
Evaluation: Logistic Regression provided a baseline performance and became in particular evaluated for its training time, testing time, and common accuracy.

2.2.2 K-Nearest Neighbors (KNN) is a flexible algorithm applicable to both classification and regression tasks. It works by evaluating the nearest k neighbors of a data point within the feature space and assigning it to the most common class among those neighbors, without relying on any assumptions about the data's underlying distribution [21]. Fig. 4 shows the methodology used for K-Nearest Neighbors.

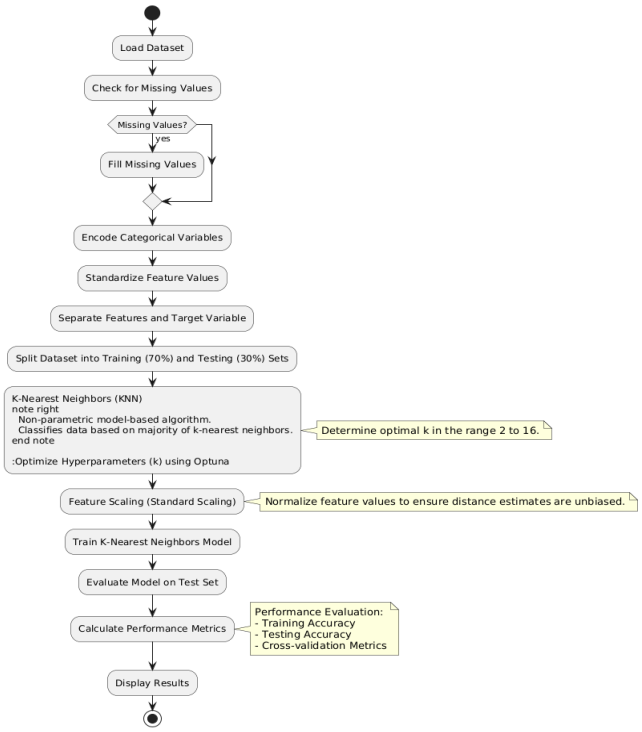

Figure 4 K-Nearest Neighbors Process Flowchart

Parameter optimization: Optuna, a hyperparameter optimization framework, was used to determine the optimal number of neighbors (k). The optimization was performed in 2 to 16 neighbors.
Feature scaling: Standard scaling was used to normalize the data, ensuring that the algorithm's distance estimates were not biased by the scale of the features
Performance evaluation: Performance was evaluated based on training and test accuracy scores. In addition, cross-validation was used to assess accuracy and recall metrics, providing insight into the model's ability to handle imbalanced data. These calculations are outlined in the next section.

2.2.3 Decision Trees Classifier are hierarchical models employed for classification and regression. They operate by dividing the dataset into smaller subsets based on specific feature values. This recursive partitioning process results in a tree-like structure where each node represents a decision point that leads to the prediction of the target variable. By capturing simple decision rules derived from the input features, Decision Trees provide an intuitive and interpretable approach to modeling data

relationships[19].Fig. 5 illustrates the methodology used for **Decision Tree.**

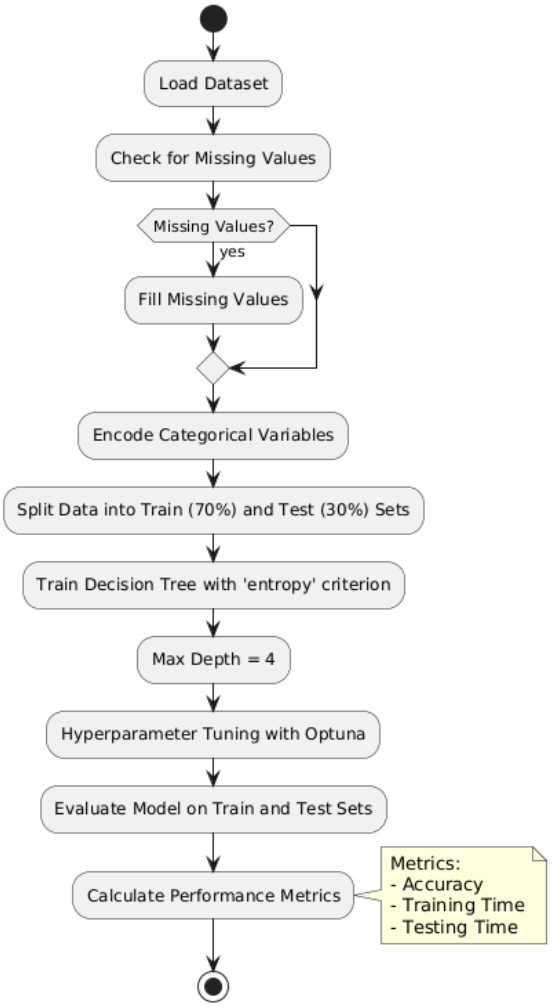

Figure 5 Decision Tree Process Flowchart

Training: A Decision Tree model was trained with the 'entropy' criterion, which measures the impurity of a node. The maximum depth of the tree was set to 4 to prevent overfitting.
Hyperparameter Tuning: Using Optuna, the maximum depth and maximum features parameters were optimized. The optimization was conducted over 30 trials, selecting the parameters that maximized the accuracy.
Evaluation: The Decision Tree model was evaluated on training and testing data, with specific attention given to the training time and testing time. These calculations are outlined in the next section.

Three classifiers are used: Logistic Regression, K-Nearest Neighbors (KNN), and Decision Tree. These models are trained on the preprocessed and reduced dataset. Training includes splitting the data into training and testing sets and fitting the models.

Fig. 6 explains about the evolution process for all the classifiers.

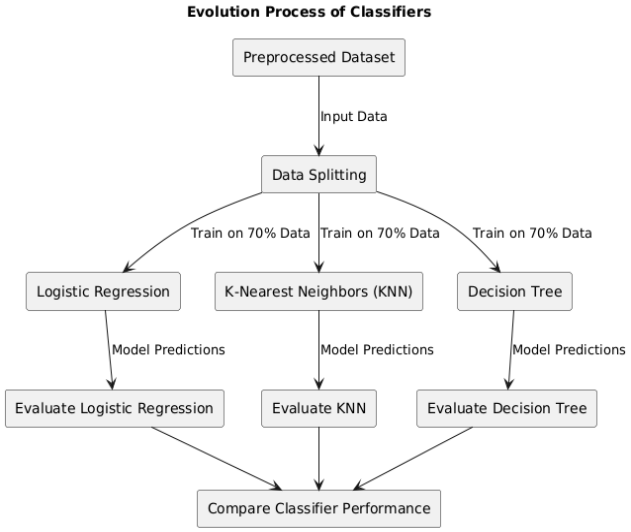

Figure 6 Evolution process

2.3 Hyperparameter optimization algorithm

The effectiveness of machine learning models largely relies on proper tuning of their hyperparameters. In this study, Optuna, a popular library for hyperparameter optimization, was employed to fine-tune the parameters for each algorithm. The key parameters optimized were:

- The number of neighbors (K) for KNN,
- Maximum depth for the Decision Tree, and
- The regularization parameter (C) for Logistic Regression.

The model training involved splitting the dataset into training and testing subsets, wherein 70% of the data were for training and 30% for testing. The optimized parameters were used to train each algorithm and their performance was checked through measures of accuracy, precision, recall, and F1-score. This method can effectively classify network traffic to discriminate between what is normal and what is anomalous.

NFSU JOURNAL OF CYBER SECURITY & DIGITAL FORENSICS

NFSU – Journal of Cyber Security and Digital Forensics
Volume –3, Issue – 1, June 2024
E – ISSN – 2583-7559

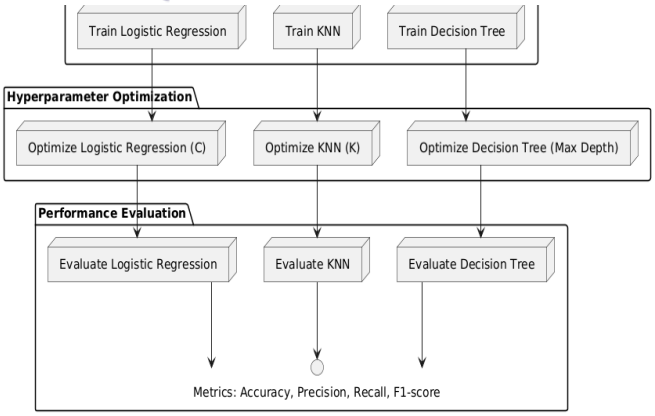

Figure 7 Hyperparameter Optimization

III. RESULTS DISCUSSION

3.1 Data Measurement

The data used in this exercise consisted of 27,192 training samples with 27 attributes. These attributes are carefully chosen to strike a good balance in representing network traffic statistics. On one hand, some attributes are purely statistical metrics, for example, the count of packets and the size in bytes, while others are purely distributional features, for instance, the types of protocols and the states of connections. Every single attribute is critical in pattern discovery, indicative to cohort or anomalous behavior. The 27 features were chosen to strike a balance between having enough but not too much information for accurate predictions and overfitting due to an excessive number of features.

3.2 Feature selection

Recursive feature elimination (RFE) with random forest classification does the evaluation of feature importance systematically through repeated model fitting and removal of the least significant features in each iteration until an optimized feature subset is reached. This acts as a filter for certain noisy data or for overfitting problems, decreasing data dimension. This allows the model to generalize better in predicting because it focuses on the most relevant ones. It ensures the selected features to reflect critical characteristics of network traffic while shutting out irrelevant and redundant data, which may lead to improved accuracy and efficiency in attempting to recognize anomalous behavior in the network.

3.3 Model training and evaluation:

Three classification models were evaluated:

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Tree

The classifiers were trained on the preprocessed dataset, and their performance was measured using both the training and testing data.

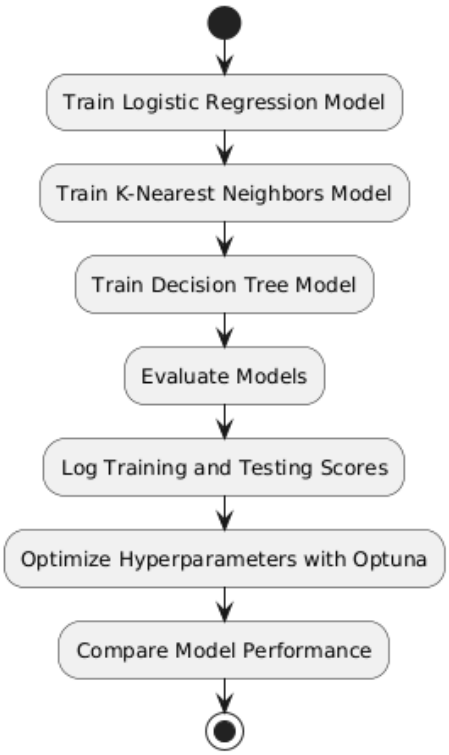

Figure 8 Process Of Project

The results are summarized in Table 1, contains the conclusions, which is an overview of the model. The table displays the efficiency of each classification model, providing a comparison based on their evaluation metrics as follows:

Table 1 Model score

Model	Train Score	Test Score
KNN	0.982817	0.98227
Logistic Regression	0.928887	0.923392
Decision Tree	1	0.994708

The outcomes are given by Table 2, which is the overall summary of the accuracy and precision of each classification model. The table put side by side the performance of the models based on these evaluation metrics, and it allows a clear assessment of their effectiveness.

Table 2 Accuracy, Precision of all models

Model	Accuracy (%)	Precision (%)
KNN	98.45	95
Logistic Regression	94.21	93
Decision Tree	99.33	98.5

3.4 Precision and Recall Analysis

The precision and recall metrics for the models are presented in Table 3, which outlines the performance of each classification model.

Table 3 Precision and recall of models

Model	Metric	Mean(%)	Std Dev (%)
KNN	Precision	98.45	0.48
KNN	Recall	98.24	0.54
Logistic Regression	Precision	91.36	0.56
Logistic Regression	Recall	95.72	0.69
DecisionTreeClassifier	Precision	99.5	0.17
DecisionTreeClassifier	Recall	99.51	0.23

The Decision Tree Classifier had the highest precision and recall rates, demonstrating its superior ability to correctly classify both normal and anomalous connections. KNN also performed exceptionally well, showing its effectiveness in handling this classification problem.

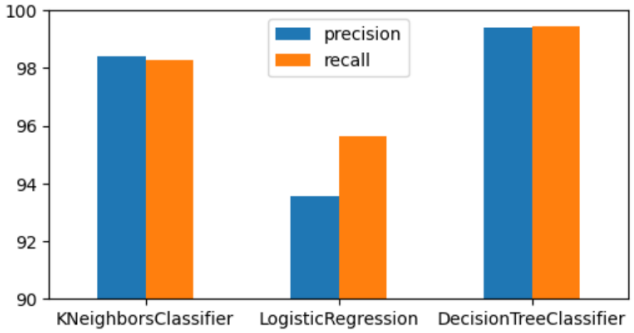

Figure 9 Precision And Recall Of Models

IV. COMPARATIVE ANALYSIS

4.1 Performance Evaluvation

The decision tree model runs better in training and testing aspects than the rest of the classifiers. With a test accuracy of 99.33% and highest precision and recall, one is quite sure that the decision tree becomes a very strong intrusion detection solution. This could be due to its ability to correctly capture the pattern in the data.

4.1.1 K-Nearest Neighbors (KNN)

KNN yielded an impressive performance with 98.45% test accuracy, though it couldn't outperform the Decision Tree in terms of precision and recall. Very high-performance guarantees that this will be a good candidate for intrusion detection at least with well-preprocessed and selected features of the dataset.

4.1.2 Logistic Regression

Logistic Regression, although reliable with a test accuracy of 94.22%, does not quite live up to the KNN or the Decision Tree performances. Its performance is nonetheless commendable, hence, in conjunction with optimized preprocessing and feature selection, it can be a formidable tool for intrusion detection.

V. CONCLUSION

In this paper, a number of machine learning methods have been applied to enhance NIDS performance, given in particular by the performance of three algorithms: Logistic Regression, K-Nearest Neighbors, and Decision Tree classifiers. Methods of preprocessing and feature selection have been performed on the dataset downloaded from Kaggle in order to have better data preparation and optimization before training. Feature selection was done using the Random Forest Classifier and Recursive Feature Elimination to ensure that only the most relevant features were selected.The best performance was obtained by the Decision Tree classifier, with a test accuracy of 99.34%, compared with Logistic Regression, at 94.22%, and KNN at 98.45%. Also, the Decision Tree model had greater precision and recall rates, which showed its strength in classifying normal and anomalous network connections accurately. KNN was very strong in test accuracy with reasonably balanced precision and recall rates.

This research gives great importance to feature selection combined with optimization of hyperparameters to enhance the performance in intrusion detection with the help of machine learning models. Optuna tuned the hyperparameters quite efficiently for better accuracy and efficiency of the models. The results also indicate that research is needed in the development of NIDS using machine-learning methods and to overcome some of the drawbacks of signature-based traditional techniques.

This study is one of several in the quest for a better cybersecurity system and gives a way in which machine learning models in network intrusion detection can be optimized.

REFERENCES

[1] Vinod K. Pachghare, Ph.D , Sharmila Kishor Wagh (2013)."Survey on Intrusion Detection System using Machine Learning Techniques" doi:10.5120/13608-1412.

[2] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," Information Security Journal: A Global Perspective, vol. 25, no. 1-3, pp. 18-31, 2016, doi: 10.1080/19393555.2015.1125974.

[3] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems, and Tools," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303-336, 2014, doi: 10.1109/SURV.2013.052213.00046.

[4] M. H. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Towards developing a systematic approach to generate benchmark datasets for intrusion detection," Computers & Security, vol. 31, no. 3, pp. 357-374, May 2012, doi: 10.1016/j.cose.2011.12.012.

[5] Performance Evaluation of Intrusion Detection System using Selected Features and Machine Learning Classifiers. Baghdad Sci.J [Internet]. 2021 Jun. 20 [cited

2024 Sep. 14];18(2(Suppl.):0884.

[6] Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, Feb. 2016, doi: 10.1109/TETCI.2017.2772792.

[7] Nour Moustafa and J. Slay, A hybrid feature selection for network intrusion detection systems: Central points", 2015. doi: 10.4225/75/57a84d4fbefbb

[8] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges," Computers & Security, vol. 28, no. 1-2, pp. 18-28, 2009, doi: 10.1016/j.cose.2008.08.003.

[9] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," IEEE Symposium on Security and Privacy, pp. 305-316, 2010, doi: 10.1109/SP.2010.25.

[10] W. Stallings, Network Security Essentials: Applications and Standards, 6th ed., Pearson, 2017.

[11] R. Bace and P. Mell, Intrusion Detection Systems, 1st ed., Addison-Wesley, 2001.

[12] Ambikavathi and S. K. Srivatsa, "Predictor Selection and Attack Classification using Random Forest for Intrusion Detection", vol. 79, no. 05, pp. 365–368, 2020.

[13] Abrar, Z. Ayub, F. Masoodi, and A. M. Bamhdi, "A Machine Learning Approach for Intrusion Detection System on NSL-KDD Dataset", Proc-Int.Cong. Smart Electron. Commu.ICOSEC 2020, no. Icosec, pp. 919–924, 2020, doi: 10.1109/ICOSEC49089.2020.921523 2.

[14] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.

[15] Bagdanov, A. D., Frolov, I. O., & Mukhametzyanov, I. R. (2019). Machine Learning for Network Intrusion Detection: A Survey. Computers & Security, 86, 110-128.

[16] Nhu, V.-H., et al. "Shallow Landslide Susceptibility Mapping: A Comparison between Logistic Model Tree, Logistic Regression, Naïve Bayes Tree, Artificial Neural Network, and Support Vector Machine Algorithms." Applied Sciences, 2019.

[17] Mitrokotsa, A., et al. "Artificial Neural Network Techniques in Intrusion Detection." Expert Systems with Applications, 2009.

[18] Ahmad, Z., et al. "Network Intrusion Detection System: A Systematic Study of Machine Learning and Deep Learning Approaches." Transactions on Emerging Telecommunications Technologies, 2021.

[19] Vinod K. Pachghare, Ph.D , Sharmila Kishor Wagh (2013)."Survey on Intrusion Detection System using Machine Learning Techniques" doi:10.5120/13608-1412.

[20] X. Zou, Y. Hu, Z. Tian, and K. Shen, "Logistic Regression Model Optimization and Case Analysis," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, Dalian, China, 2019, pp. 1-5, doi: 10.1109/ICCSNT47585.2019.8962457.

[21] M. Suyal and P. Goyal, "A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms based on Supervised Learning," *International Journal of Engineering Trends and Technology*, vol. 70, no. 7, pp. 43-48, Jul. 2022, doi: 10.14445/22315381/IJETT-V70I7P205.