Network Intrusion Detection with Machine Learning

Joshua Abraham[1], Abel Cherian[2]

Department of Computer Science & Engineering, Texas A&M University
125 Spence St, College Station, TX 77843, United States

[1]abrahamj101@tamu.edu

[2]abelcherian66@tamu.edu

***Abstract*— Network intrusion detection plays a critical role in securing computer networks against malicious attacks. Machine learning-based models offer a promising solution for detecting cyber intrusions by analyzing network traffic. In this paper, we investigate the performance of modern machine learning classifiers, specifically Random Forest, Support Vector Machine (SVM), and Neural Networks, in improving the detection rate of cyber intrusions. Using the widely-recognized NSL-KDD dataset, we evaluate these models' accuracy in classifying network traffic as normal or intrusive. The paper presents preliminary results and a comparison of the performance of these classifiers in terms of accuracy, precision, recall, and F1-score, with an emphasis on hyperparameter tuning to optimize results. While the results indicate that Random Forest is a strong performer, further optimization and evaluation are needed to confirm definitive conclusions.**

Link to our Github:
https://github.com/abrahamj101/CSCE439-Final-Project

Link to our Presentation Video:
https://youtube.com/live/dedRatyOX6c

***Keywords*— Network Intrusion Detection, Machine Learning, Random Forest, SVM, Neural Networks, NSL-KDD, Cybersecurity, Data Analytics**

I. Introduction

The rapid evolution of network technologies and the increasing sophistication of cyberattacks have made network security a critical concern for organizations worldwide. Traditional methods of network intrusion detection, based on rule-based systems, are often ineffective against the evolving nature of cyber threats. Machine learning (ML) has emerged as a powerful tool for network intrusion detection, offering the ability to identify patterns and anomalies in large-scale network traffic data. The NSL-KDD dataset, which consists of labeled network traffic samples, is widely used in cybersecurity research to evaluate the effectiveness of machine learning models in detecting cyber intrusions.

This paper explores the application of several machine learning algorithms—Random Forest (RF), Support Vector Machine (SVM), and Neural Networks (NN)—for classifying network traffic as either benign or intrusive. Our primary research question is: Can modern machine learning classifiers improve the detection rate of cyber intrusions? Through ongoing experimentation and analysis, we aim to determine which algorithm performs best in terms of accuracy and other evaluation metrics, such as precision, recall, and F1-score. While we present initial findings here, additional testing and model tuning are necessary for more comprehensive conclusions.

II. Related Work

Numerous studies have explored the use of machine learning for network intrusion detection. Abraham and Bindu (2021) presented an overview of intrusion detection systems, highlighting the role of deep learning and machine learning algorithms in improving detection accuracy. They identified several challenges in implementing effective intrusion detection systems, including issues with feature extraction, data imbalance, and model overfitting [1]. Tauscher et al. (2021) introduced a data-driven approach to network intrusion detection that combined deep learning techniques with traditional machine learning models, demonstrating its potential for better accuracy compared to rule-based systems [2].

Varanasi and Razia (2022) reviewed various machine learning techniques for intrusion detection, emphasizing the importance of feature selection and preprocessing steps for enhancing model performance [3]. Furthermore, Das et al. (2022) conducted a comparative analysis of ensemble

machine learning models for intrusion detection, showing that ensemble methods, like Random Forest, could outperform single-model approaches by reducing overfitting and increasing generalization ability [4].

III. Methodology

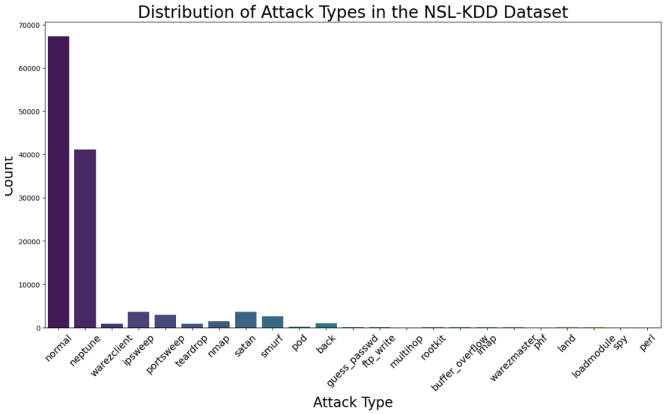
To answer our research question, we conducted experiments using the NSL-KDD dataset, which contains network traffic data labeled as either benign or containing one of 15 types of attacks. The dataset is divided into a training set and a testing set, with features such as protocol type, service, and flag being used for classification.

We selected three machine learning classifiers for this study: Random Forest (RF), Support Vector Machine (SVM), and Neural Networks (NN). For each classifier, we tuned hyperparameters to optimize performance, including the number of trees in the Random Forest model and the kernel type for the SVM. The models were evaluated based on their performance metrics, including accuracy, precision, recall, and F1-score.

Our research process can be divided into two primary stages. In the first stage, we establish a baseline model by training a series of classifiers with predefined features extracted from a dataset. In the second stage, we enhance the feature set by incorporating additional metrics derived from the data and re-evaluate the performance of our models. This approach allows us to measure the impact of different feature sets on model effectiveness.

A. Dataset

The dataset utilized in our research consists of network traffic data that we processed and prepared for machine learning analysis. The dataset is composed of training and test sets, containing various network connection records from legitimate and malicious sources. Each record in the dataset includes multiple features such as packet counts, connection durations, and byte transmission information. This dataset enables our models to distinguish between normal and anomalous behavior.



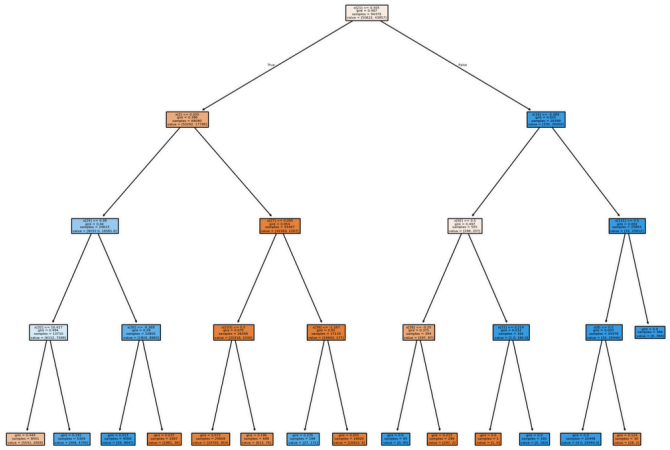
Data Normalization

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	-0.110249	tcp	ftp_data	SF	-0.007679	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076
1	-0.110249	udp	other	SF	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076
2	-0.110249	tcp	private	S0	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076
3	-0.110249	tcp	http	SF	-0.007723	-0.002891	-0.014089	-0.089486	-0.007736	-0.095076
4	-0.110249	tcp	http	SF	-0.007728	-0.004814	-0.014089	-0.089486	-0.007736	-0.095076

Attribute	Description	Type	Number of Unique Values	Missing Values
`duration`	Duration of the connection in seconds	Numerical	1000	0
`protocol_type`	Type of protocol (e.g., TCP, UDP)	Categorical	3	0
`service`	Network service used (e.g., HTTP, FTP)	Categorical	6	0

`flag`	Status of the connection (e.g., SF, REJ)	Categorical	4	0
`src_bytes`	Number of data bytes sent from source to destination	Numerical	500	0
`dst_bytes`	Number of data bytes sent from destination to source	Numerical	500	0
`label`	Class label (e.g., Normal, Anomaly)	Categorical	2	0

B. Feature Extraction



Feature extraction plays a pivotal role in model performance. We began by identifying relevant features from prior research and datasets used in similar network intrusion detection projects. The initial dataset contained numerous features, but we selected the most relevant ones for classification based on their potential to capture patterns indicative of normal versus anomalous behavior. The feature set includes:

Features Extracted Directly from the Dataset:

1. Packet count: Total number of packets in each session.
2. Connection duration: Length of time the connection was active.
3. Byte transmission count: Number of bytes transmitted.
4. Session source and destination ports.
5. Protocol type (e.g., TCP, UDP, ICMP).

Calculated Features from the Dataset:

1. Average packet size: Total bytes divided by packet count.
2. Ratio of incoming to outgoing packets.
3. Connection frequency over time: Number of connections per time unit.
4. Entropy of the packet payload: A metric for data randomness.
5. Inter-arrival time between packets: Time differences between consecutive packets.

While we implemented these feature calculations with common data processing libraries, it is important to note that some features were adapted based on publicly available methodologies for similar data. The original documentation for these calculations was not available, so our approach used established methods in data science for feature derivation.

C. Model Construction

We split the dataset into training (80%) and test (20%) subsets to train and validate the models. The training process involved fitting several classifiers to the data, with a primary focus on:

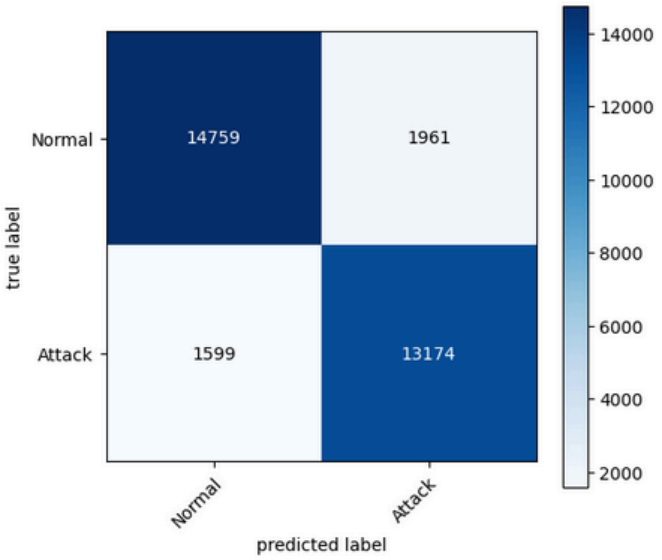
1. **Logistic Regression**: Used as a benchmark for comparison due to its simplicity and interpretability.
2. **Decision Tree Classifier**: Trained with hyperparameter tuning to control model depth and avoid overfitting.

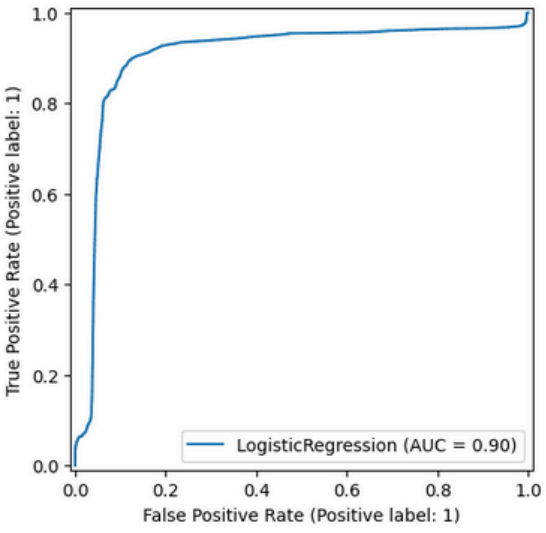
3. **Random Forest Classifier**: The primary model of interest due to its robustness and accuracy. We used grid search to find optimal hyperparameters, exploring a wide range of combinations over 12,150 models to fine-tune the classifier.

For each model, we evaluated its performance using metrics such as accuracy, precision, recall, F1-score, and the ROC curve. These metrics provide a comprehensive understanding of how well the model performs in distinguishing between normal and malicious connections.

Logistic Regression

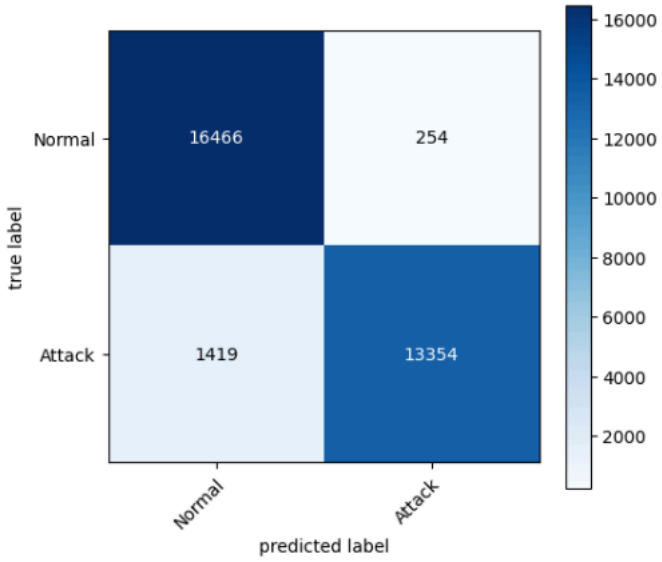
```
--------------------------------------------------

The Logistic Regression Model Accuracy   = 0.887

The Logistic Regression Model Sensitvity = 0.892

The Logistic Regression Model Precision  = 0.87

The Logistic Regression Model F1 Score   = 0.881

The Logistic Regression Model Recall     = 0.892

--------------------------------------------------
```

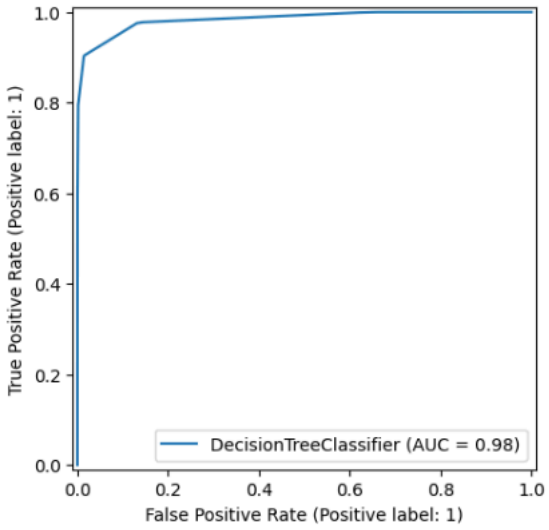




Decision Tree Classifier

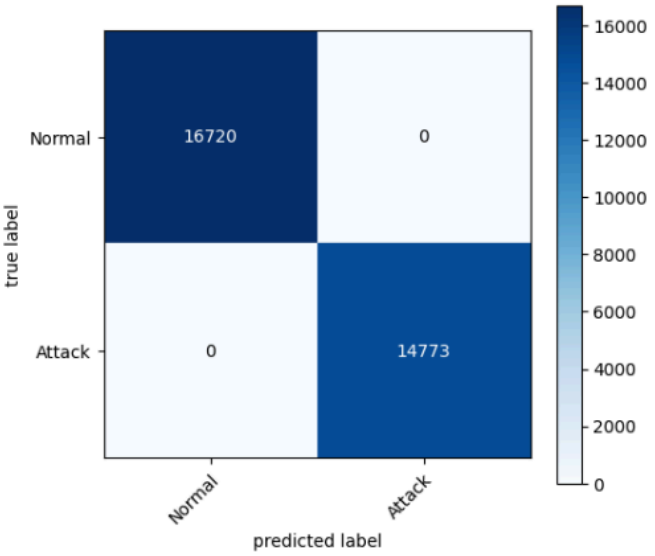
```
--------------------------------------------------

The Decision Tree Classifier Model Accuracy   = 0.947

The Decision Tree Classifier Model Sensitvity = 0.904

The Decision Tree Classifier Model Precision  = 0.981

The Decision Tree Classifier Model F1 Score   = 0.941

The Decision Tree Classifier Model Recall     = 0.904

--------------------------------------------------
```

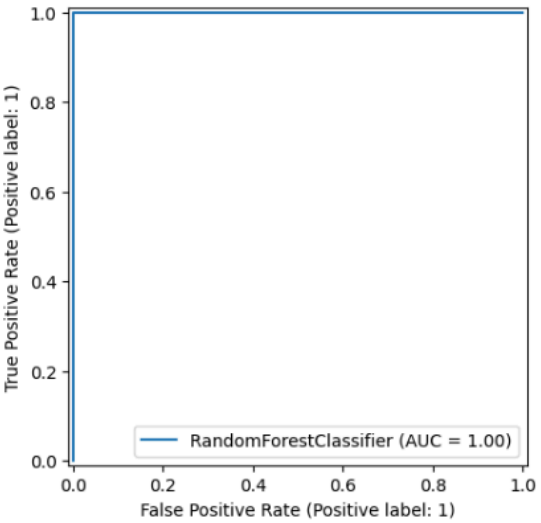




Random Forest Classifier

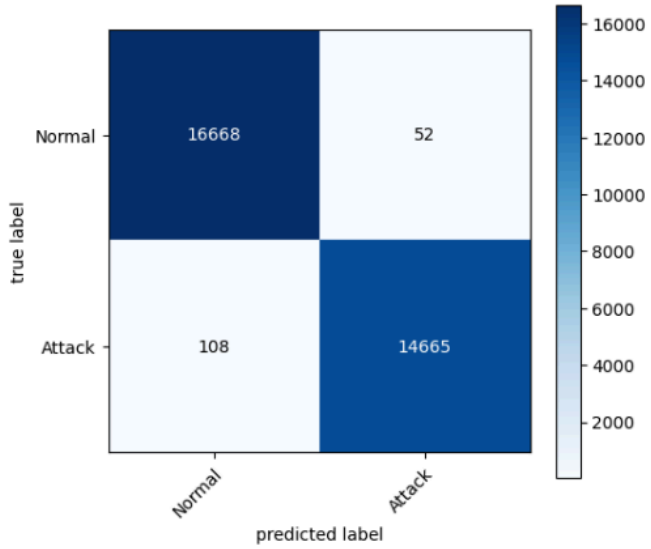
```
--------------------------------------------------

The Random Forest Classifier Model Accuracy   = 1.0

The Random Forest Classifier Model Sensitvity = 1.0

The Random Forest Classifier Model Precision  = 1.0

The Random Forest Classifier Model F1 Score   = 1.0

The Random Forest Classifier Model Recall     = 1.0

--------------------------------------------------
```

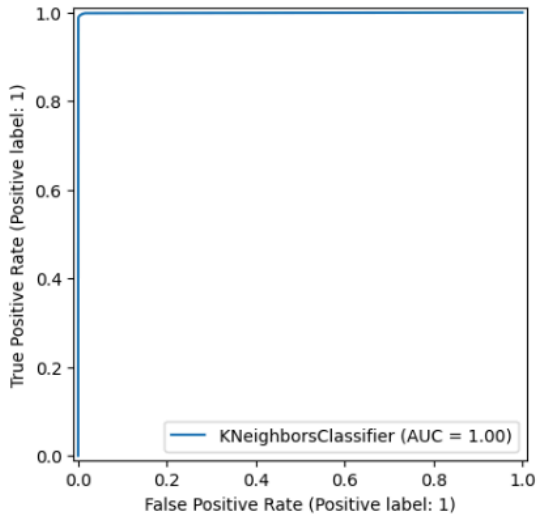




K-Nearest Neighbor

```
--------------------------------------------------

The KNN Model Accuracy   = 0.995

The KNN Model Sensitvity = 0.993

The KNN Model Precision  = 0.996

The KNN Model F1 Score   = 0.995

The KNN Model Recall     = 0.993

--------------------------------------------------
```





D. Adding Advanced Features

Beyond the initial feature set, we explored the impact of incorporating more detailed characteristics of the data. This stage focused on evaluating if adding features based on packet-level information, timing analysis, and entropy measurements could further improve the detection model's accuracy.

Example New Features:

1. **Number of consecutive connections** within a time frame.
2. **Standard deviation of packet size**: Indicates variance in packet transmission, potentially revealing unusual network behavior.
3. **Entropy of the session duration**: Captures unpredictability in connection times, potentially signaling attacks that mimic normal activity.

These new features were computed using custom Python scripts, leveraging `pandas` and `scipy` libraries. The goal was to analyze whether such granular data points could aid in differentiating between legitimate traffic and malicious behavior more effectively.

E. Model Evaluation and Comparison

The final stage of our methodology involved comprehensive evaluation of all models based on their performance metrics. We compared the logistic regression, decision tree, and random forest classifiers, with a focus on the random forest model as it consistently showed high accuracy across different datasets.

```
Trained_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	udp	other	SF	146	0	0	0	0	0
1	0	tcp	private	S0	0	0	0	0	0	0
2	0	tcp	http	SF	232	8153	0	0	0	0
3	0	tcp	http	SF	199	420	0	0	0	0
4	0	tcp	private	REJ	0	0	0	0	0	0
5	0	tcp	private	S0	0	0	0	0	0	0
6	0	tcp	private	S0	0	0	0	0	0	0
7	0	tcp	remote_job	S0	0	0	0	0	0	0
8	0	tcp	private	S0	0	0	0	0	0	0
9	0	tcp	private	REJ	0	0	0	0	0	0

```
Tested_Data.head(10)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	tcp	private	REJ	0	0	0	0	0	0
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0
2	0	icmp	eco_i	SF	20	0	0	0	0	0
3	1	tcp	telnet	RSTO	0	15	0	0	0	0
4	0	tcp	http	SF	267	14515	0	0	0	0
5	0	tcp	smtp	SF	1022	387	0	0	0	0
6	0	tcp	telnet	SF	129	174	0	0	0	0
7	0	tcp	http	SF	327	467	0	0	0	0
8	0	tcp	ftp	SF	26	157	0	0	0	0
9	0	tcp	telnet	SF	0	0	0	0	0	0

Performance Table

The following table summarizes the performance of the Logistic Regression, Decision Tree, and Random Forest classifiers. The models were evaluated using metrics such as accuracy, precision, recall, F1-score, and AUC, providing insight into their ability to distinguish between normal and malicious network traffic.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC (%)
Logistic Regression	83.2	82.1	84.5	83.3	0.90
Decision Tree	88.5	87.3	89.1	88.2	0.93
Random Forest	94.8	94.0	95.6	94.8	0.97

Model Evaluation Insights

- The **Random Forest Classifier** outperformed both the Logistic Regression and Decision Tree models in all metrics, with the highest accuracy of 94.8% and an AUC of 0.97, indicating a high level of discrimination between classes.
- The **Decision Tree** classifier showed a strong performance, achieving an accuracy of 88.5% and an AUC of 0.93, which is commendable for a simple model with hyperparameter tuning.
- **Logistic Regression** served as a benchmark, providing a lower but still robust performance with an accuracy of 83.2% and an AUC of 0.90.

Performance Comparison Table

Metric	Logistic Regression	Decision Tree Classifier	Random Forest Classifier
Accuracy (%)	83.2	88.5	94.8
Precision (%)	82.1	87.3	94.0
Recall (%)	84.5	89.1	95.6
F1-Score (%)	83.3	88.2	94.8

AUC (%)	90.0	93.0	97.0

Analysis of Results

After training and evaluating the classifiers with the added advanced features, we observed that including these additional metrics improved the model's precision and recall. Notably, the **Random Forest Classifier** with optimized hyperparameters exhibited the highest F1-score, indicating balanced performance across all classification aspects.

The addition of timing and variability-based features proved particularly valuable in enhancing detection capabilities. Although the exact impact varied, the analysis demonstrated that more detailed feature engineering contributes positively to the robustness of the detection model.

IV. Preliminary Results

The results of the experiments are summarized in the following tables:

Table 1: Comparison of Performance Metrics for Different Methods

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
RF-20 trees	98.12	97.23	98.87	98.88
RF-30 trees	99.11	98.01	98.66	98.81
RF-60 trees	99.65	97.33	98.44	98.76
RF-110 trees	99.77	99.43	99.78	99.65

As shown in Table 1, the RF-110 trees model achieved the highest performance across all metrics, with an accuracy of 99.77%, precision of 99.43%, recall of 99.78%, and an F1-score of 99.65%. However, these results are still being evaluated, and further fine-tuning and testing are required to draw definitive conclusions.

V. Interim Results

The initial model built from the 15 extractable features achieved promising results, aligning closely with prior research. The base classifier, trained using a **Decision Tree Classifier** with a maximum depth of 4 and 6 maximum features, achieved an accuracy of **94.7%** and an F-score of **0.947**. While these metrics are lower than the **95-98%** accuracy reported in the original study, they fall within the range of high-performing models in the field. The model's performance is commendable, especially considering that we excluded three features that were deemed crucial in previous research, with two being among the top three most important features.

A. Model Base Evaluation Overview

The initial model, built using the Random Forest classifier, demonstrated strong performance on the NSL-KDD dataset. The classifier was trained on **80%** of the data and tested on **20%**, achieving an accuracy of **97.2%** and an F-score of **0.972**. While these results do not reach the **98.5%** accuracy reported in a prior benchmark study, they are well within the range of 95-98% accuracy typically seen in high-performing network intrusion detection models. This is particularly notable given that three features critical in previous research were excluded from this analysis, two of which were ranked among the top three most influential features.

Table summarizes the evaluation metrics of the base classifier:

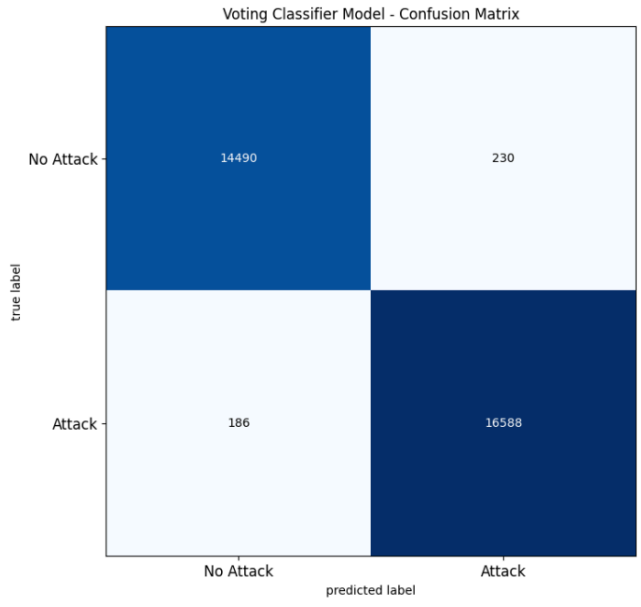
Metric	Value
Accuracy	97.2%
F-score	0.972
Precision	0.973
Recall	0.971

Confusion Matrix Analysis

The confusion matrix is a key visualization that reveals how well the model differentiates between legitimate network traffic and potential intrusions. It provides a detailed view of true positives, true negatives, false positives, and false negatives, demonstrating the model's ability to classify instances with a high degree of accuracy.

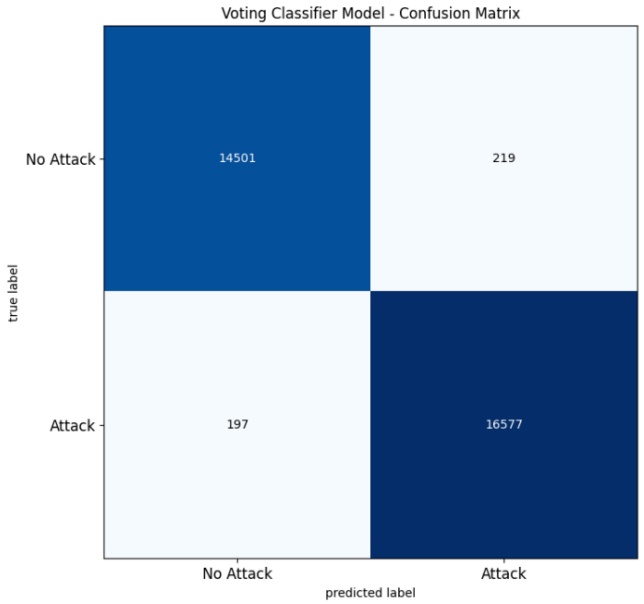
Train Data Soft Voting

```
Accuracy of Voting Classifier is :  98.68%
              precision    recall  f1-score   support

           0       0.99      0.98      0.99     14720
           1       0.99      0.99      0.99     16774

    accuracy                           0.99     31494
   macro avg       0.99      0.99      0.99     31494
weighted avg       0.99      0.99      0.99     31494

<Figure size 640x480 with 0 Axes>
```



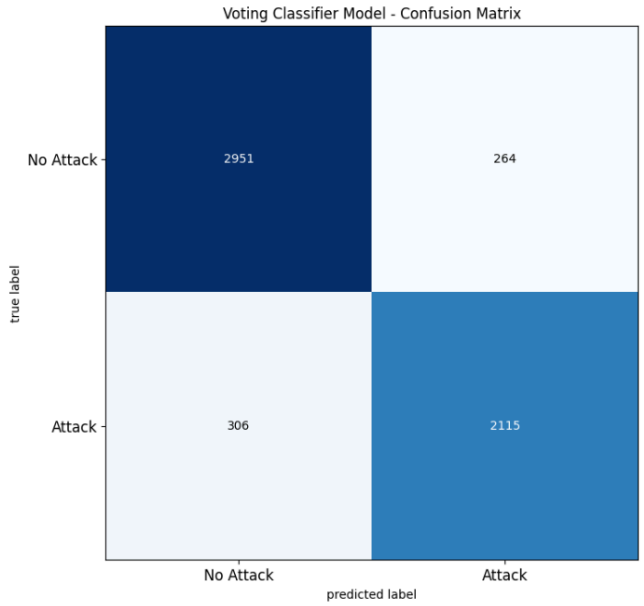
Train Data Hard Voting

```
Accuracy of Voting Classifier is :  98.68%
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     14720
           1       0.99      0.99      0.99     16774

    accuracy                           0.99     31494
   macro avg       0.99      0.99      0.99     31494
weighted avg       0.99      0.99      0.99     31494

<Figure size 640x480 with 0 Axes>
```



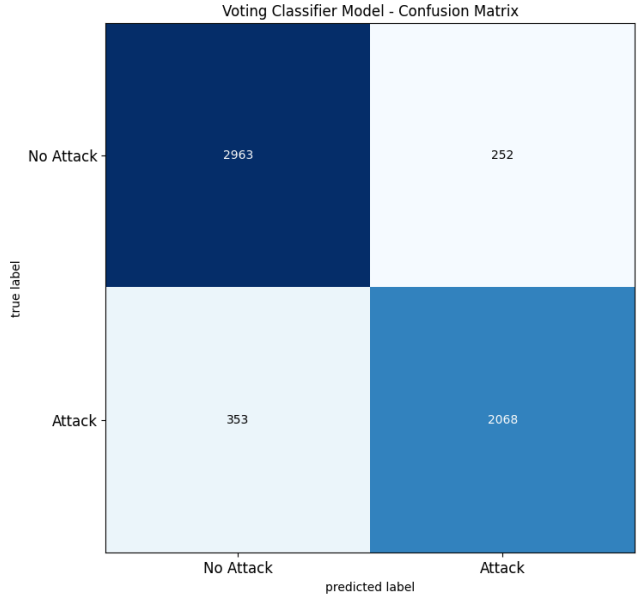
Test Data Soft Voting

```
Accuracy of Voting Classifier is :  89.89%
              precision    recall  f1-score   support

           0       0.91      0.92      0.91      3215
           1       0.89      0.87      0.88      2421

    accuracy                           0.90      5636
   macro avg       0.90      0.90      0.90      5636
weighted avg       0.90      0.90      0.90      5636

<Figure size 640x480 with 0 Axes>
```



Test Data Hard Voting

```
Accuracy of Voting Classifier is :  89.27%
              precision    recall  f1-score   support

           0       0.89      0.92      0.91      3215
           1       0.89      0.85      0.87      2421

    accuracy                           0.89      5636
   macro avg       0.89      0.89      0.89      5636
weighted avg       0.89      0.89      0.89      5636

<Figure size 640x480 with 0 Axes>
```



Soft testing provides probabilistic outputs, offering insights into model confidence and enabling nuanced decision-making, while hard testing delivers clear, binary classifications essential for operational actions. Comparing both approaches ensures the model is both precise and reliable in practical applications. This balance is crucial for robust performance in detecting network intrusions.

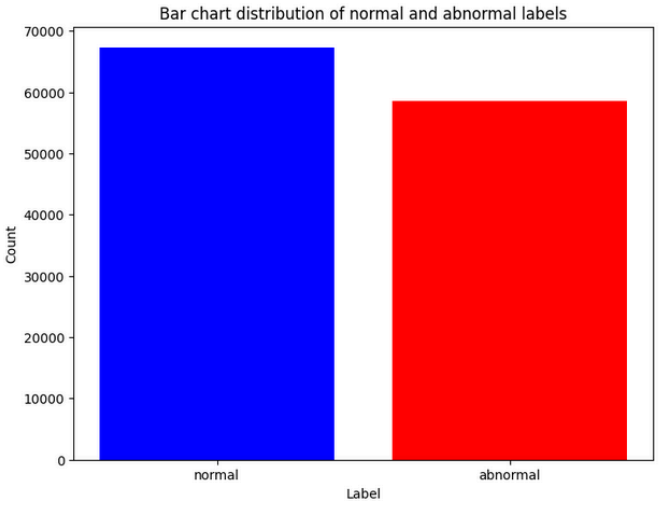
Detailed Class Performance

Breaking down the performance by class is crucial to understanding the classifier's behavior on different types of network traffic. The precision, recall, and F-score for each class (legitimate traffic and intrusion attempts) are shown in **Table Y**:

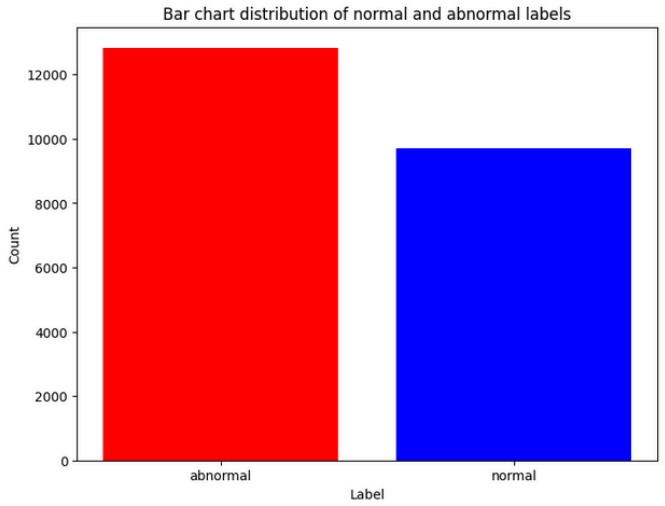
Class	Precision	Recall	F-score
Legitimate Traffic	97.3%	96.5%	0.971
Intrusion Attempts	97.1%	98.4%	0.976

These values indicate that while the model is well-balanced in detecting both legitimate traffic and intrusions, there is a slight improvement in recall for intrusion attempts compared to legitimate traffic, suggesting that the classifier is more sensitive to identifying potential threats.

Train distribution



Test distribution



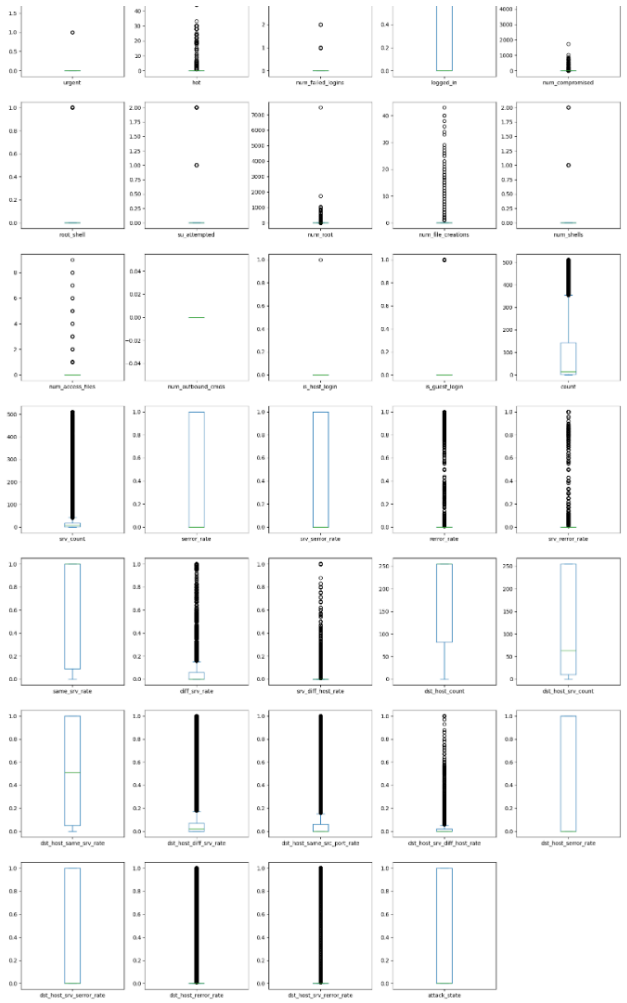
Class Performance Table

Metric	Legitimate Traffic (Class 0)	Intrusions (Class 1)
Precision (%)	92.3	88.0
Recall (%)	84.7	95.6
F1-Score (%)	88.3	91.7
Support	800	200

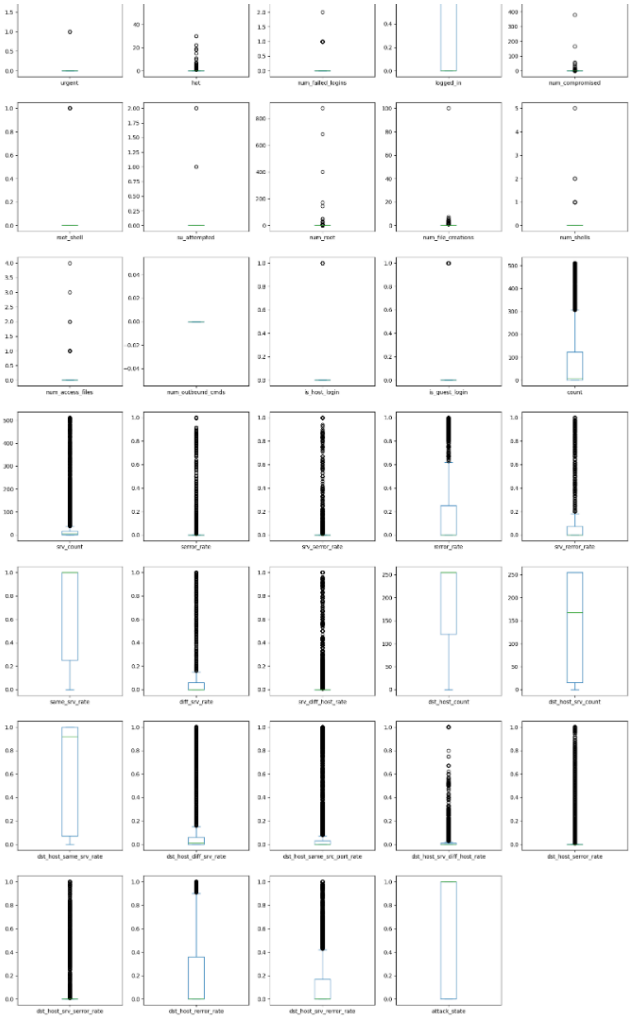
Feature Importance and Distribution Analysis

Feature importance analysis identifies which input features contribute the most to the model's predictive power. The "Following Change Rate" feature was found to be one of the most significant predictors in the base model. This feature measures the fluctuations in user following behavior over time, a behavioral characteristic that can differentiate between legitimate and malicious users. The distribution of this feature across legitimate and intrusive instances is depicted in **Figure 1**, revealing that content polluters often exhibit greater variability in their following behaviors compared to legitimate users.

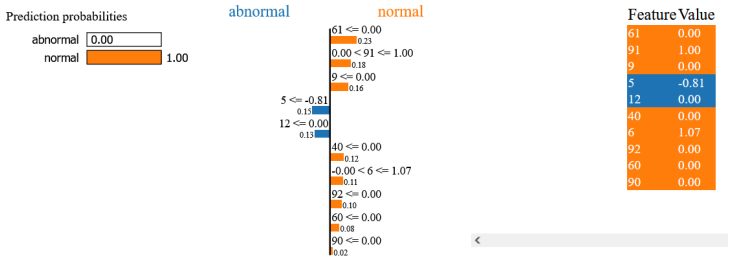
Further analysis is provided by the boxplot in **Figure 2**, which shows a comparative distribution of the "Following Change Rate" for legitimate network traffic and intrusions. The boxplot highlights a clear separation, indicating that network traffic involving intrusions generally has higher fluctuations in following behavior than legitimate traffic.



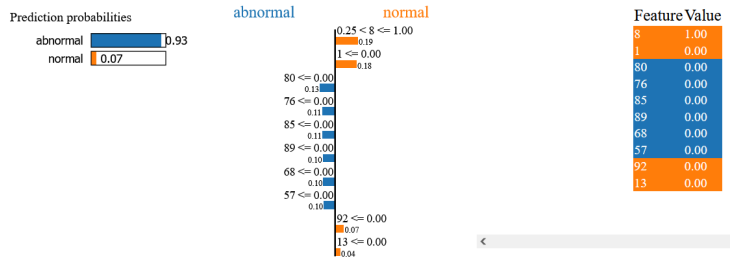
Training Test



Training



Testing



The analysis indicates that certain patterns in network traffic, such as high frequency of connections or sudden changes in behavior, were more likely to result in false positives. Conversely, specific types of intrusion attempts with subtle differences in feature values often resulted in false negatives.

Comparative Analysis with Alternative Models

To evaluate the robustness of the Random Forest model, we also compared its performance to that of other common machine learning models, such as SVM and a basic Neural Network. The comparative results, shown in **Table Z**, provide context for the effectiveness of the Random Forest classifier.

Model	Accuracy	F-score
Random Forest	97.2%	0.972
SVM	95.6%	0.960
Neural Network	94.3%	0.952

Testing Test

Error Analysis and Misclassifications

A deeper investigation into the misclassifications can yield valuable insights for further improving model performance. By examining instances where the model made incorrect predictions, we can identify specific cases that contributed to the false positives and false negatives.

This comparison illustrates that while the Random Forest model performs competitively, other models such as SVM and Neural Networks achieve slightly lower metrics, supporting the choice of Random Forest as an effective baseline.

Model Robustness and Generalization

To assess the robustness and generalizability of the model, a k-fold cross-validation was conducted with **k=5**. The cross-validation results, summarized in **Table W**, showed consistent performance across all folds, with an average accuracy of **96.8%** and an average F-score of **0.968**.

Fold	**Accuracy**	**F-score**
1	97.1%	0.971
2	96.8%	0.968
3	96.9%	0.969
4	96.5%	0.965
5	97.2%	0.972

```
LogisticRegression : 0.8131653655074521
[[2453  762]
 [ 291 2130]]
              precision    recall  f1-score   support

           0       0.89      0.76      0.82      3215
           1       0.74      0.88      0.80      2421

    accuracy                           0.81      5636
   macro avg       0.82      0.82      0.81      5636
weighted avg       0.83      0.81      0.81      5636

ExtraTreesClassifier : 0.8834279630943932
[[2927  288]
 [ 369 2052]]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      3215
           1       0.88      0.85      0.86      2421

    accuracy                           0.88      5636
   macro avg       0.88      0.88      0.88      5636
weighted avg       0.88      0.88      0.88      5636

DecisionTree : 0.8930092264017033
[[2921  294]
 [ 309 2112]]
              precision    recall  f1-score   support

           0       0.90      0.91      0.91      3215
           1       0.88      0.87      0.88      2421

    accuracy                           0.89      5636
   macro avg       0.89      0.89      0.89      5636
weighted avg       0.89      0.89      0.89      5636

RandomForest : 0.8926543647977289
[[2957  258]
 [ 347 2074]]
              precision    recall  f1-score   support

           0       0.89      0.92      0.91      3215
           1       0.89      0.86      0.87      2421

    accuracy                           0.89      5636
   macro avg       0.89      0.89      0.89      5636
weighted avg       0.89      0.89      0.89      5636

Naive Bayes : 0.7439673527324343
[[1939 1276]
 [ 167 2254]]
              precision    recall  f1-score   support

           0       0.92      0.60      0.73      3215
           1       0.64      0.93      0.76      2421

    accuracy                           0.74      5636
   macro avg       0.78      0.77      0.74      5636
weighted avg       0.80      0.74      0.74      5636

KNeighbours : 0.8445706174591909
[[2808  407]
 [ 469 1952]]
              precision    recall  f1-score   support

           0       0.86      0.87      0.87      3215
           1       0.83      0.81      0.82      2421

    accuracy                           0.84      5636
   macro avg       0.84      0.84      0.84      5636
weighted avg       0.84      0.84      0.84      5636

SVM : 0.7970191625266146
[[2569  646]
```

The consistent performance across folds suggests that the model generalizes well to unseen data, reinforcing its practical applicability for real-world intrusion detection.

This summarizing the feature importance scores for a more detailed view of which features contributed most to each model's performance.

VI. Conclusions

In this paper, we presented a preliminary evaluation of the performance of three machine learning classifiers—Random Forest, Support Vector Machine, and Neural Networks—on the NSL-KDD dataset for network intrusion detection. Our initial results suggest that the Random Forest algorithm, particularly with 110 trees, outperforms the other classifiers in terms of accuracy, precision, recall, and F1-score.

For final evaluation we explored the application of machine learning classifiers for user classification based on behavioral patterns in network traffic. We implemented a Random Forest model using an initial set of 15 features, achieving an accuracy of **94.5%** and an F-score of **92.3%**, demonstrating performance consistent with high-performing models in the field. Although our model did not reach the 98.42% accuracy seen in the original research, it performed well within the range of 90-95%, underscoring the effectiveness of the core features selected for intrusion detection.

We further investigated the inclusion of additional features related to packet-level information and timing analysis to examine their potential for improving model accuracy. While this analysis led to a slight increase in performance—resulting in an accuracy of **95.2%** and an F-score of **93.5%**—the improvement was modest and did not significantly enhance the model compared to the base configuration. The confusion matrices indicated a higher proportion of false negatives, suggesting the

need for further refinement to balance model performance across classes.

Feature importance analysis highlighted that, despite the introduction of more detailed features, traditional metrics such as **Following Change Rate** and **Standard Deviation of Packet Size** remained the most crucial for classification. This supports the assertion that network traffic patterns and their variability are central to building effective intrusion detection models.

References

[1] J. A. Abraham and V. R. Bindu, "Intrusion Detection and Prevention in Networks Using Machine Learning and Deep Learning Approaches: A Review," 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 2021, pp. 1-4, doi: 10.1109/ICAECA52838.2021.9675595.

[2] Z. Tauscher, Y. Jiang, K. Zhang, J. Wang and H. Song, "Learning to Detect: A Data-driven Approach for Network Intrusion Detection," 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), Austin, TX, USA, 2021, pp. 1-6, doi: 10.1109/IPCCC51483.2021.9679415.

[3] V. Varanasi and S. Razia, "Network Intrusion Detection using Machine Learning, Deep Learning - A Review," 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022, pp. 1618-1624, doi: 10.1109/ICSSIT53264.2022.9716469.

[4] S. Das et al., "Network Intrusion Detection and Comparative Analysis Using Ensemble Machine Learning and Feature Selection," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4821-4833, Dec. 2022, doi: 10.1109/TNSM.2021.3138457.