NATURALISTA CAMPANO
ISSN: 1827-7160
Volume 28 Issue 1, 2024

Network Traffic Analysis of Anomaly Detected Attacks Using Random Forest Algorithm in Cloud Environment

D. Sakthivel[1*], B. Radha[2]

[1*]Computer Science, Sree Saraswathi Thyagaraja College, Bharathiar University, Tamilnadu, 641046, India
[2]Computer Science, Sri Krishna Arts and Science College, Bharathiar University, Tamilnadu, 641046, India

Abstract: This research aims to predict the total number of network connections or events that occur within a specific time period, number of different services or protocols used in the network connections, length of time for each network connection or event, frequency or rate of occurrence of a particular event or behavior within a given time period in a cloud environment. These are all the fundamental metrics used in network traffic analysis to quantify various aspects of network behavior and performance. By monitoring and analyzing these metrics, network administrators and security analysts can gain valuable insights into the nature of network traffic, detect anomalies or security threats, and optimize network performance and resource utilization. This research provides insights into effectively predicting network security logs using Random Forest with selected features from the real -time NSL Dataset.

Keywords: cloud environment; network security; anomaly attacks; traffic analysis, random forest;

1. Introduction

The rapid proliferation of cloud computing is changing business, enabling easier IT management and improving resources. However, in addition to the benefits, the general location of cloud services also creates significant security problems. The most important thing is to detect and mitigate network failures that could indicate a potential cyber threat or crime.

While cloud adoption simplifies server management and reduces costs, it also presents disadvantages, especially in many virtualized environments. Attacks that exploit these vulnerabilities can compromise the integrity, availability, and confidentiality of data. Special consideration is given to insider threats affecting virtual machines (VMs). Malware can spread between virtual machines, increasing the risk of distributed denial of service (DDoS) attacks.

Detecting and mitigating insider threats in the cloud requires careful monitoring of virtual machines to detect and eliminate malicious behavior. Traditional intrusion detection systems (IDS) have difficulty adapting to changing weather conditions. Additionally, behavioral IDS technology faces issues with configuration information and training in virtualized environments.

To solve these problems, complex IDS designed specifically for the cloud are crucial. Cloud-based IDS systems leverage data collected from the cloud environment to better classify and respond to attacks. While the virtual machine monitor and the hypervisor monitor the communication between the virtual machine, the vulnerability detection system of the hypervisor layer detects suspicious vulnerabilities. Leveraging strategic software computing techniques can improve the visibility of cloud operations and provide good security even in its native state.

Traffic analysis plays a crucial role in identifying and understanding patterns of network traffic, including both normal and abnormal behavior. Anomaly detection techniques are employed to detect deviations from expected traffic patterns, which may signify malicious activities such as unauthorized access, data exfiltration, or denial-of-service attacks.

In this context, the use of machine learning algorithms, particularly the Random Forest algorithm, has emerged as a promising approach for anomaly detection in cloud environments. Random Forest is an ensemble learning

method that leverages the power of decision trees to detect patterns and anomalies in complex datasets.
The count, srv_count, duration, and rate in network security logs using the Random Forest algorithm with selected features (same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_same_srv_rate, dst_host_diff_srv_rate, and dst_host_same_src_port_rate) from the NSL Dataset. The process involves data preparation, including loading the dataset, handling missing values, and splitting it into training and testing sets. Separate Random Forest models are initialized for each target variable, and trained using the training data. During training, the algorithm constructs multiple decision trees based on subsets of the training data and features. Model predictions are made on the testing data, with numerical values predicted for count, srv_count, and duration, and class labels predicted for rate. The performance of each model is evaluated using appropriate metrics, and feature importance are analyzed to understand the impact of features on predictions. Optional steps include hyper parameter tuning and deployment of trained models for making predictions on new data.

The NSL Dataset, a widely used benchmark dataset in the field of network security, provides a rich source of network traffic data collected from a variety of simulated attacks in a cloud environment. Leveraging this dataset, we aim to conduct a comprehensive analysis of anomaly detection techniques using the Random Forest algorithm in a cloud environment.
This research aims to:

- Explore the characteristics of network traffic in a cloud environment using the NSL Dataset.
- Investigate the efficacy of Random Forest algorithm in detecting anomalous network traffic such as number of network connections or events that occur within a specific time period, number of different services or protocols used in the network connections, length of time for each network connection or event, frequency or rate of occurrence of a particular event or behavior within a given time period that indicative of potential attacks.
- Evaluate the performance of the Random Forest model in terms of R-squared (R2) score, accuracy, precision, recall, and F1-score for various types of attacks.
- Provide insights into the strengths and limitations of using Random Forest for anomaly detection in cloud environments based on the analysis of the NSL Dataset.

By conducting this research, we aim to contribute to the advancement of anomaly detection techniques in cloud security and provide valuable insights for enhancing the security posture of cloud environments against potential cyber threats.

2. Related Works

The potential benefits and enhancement of services have made cloud computing an attractive field in the current era. Cloud computing has various benefits that have grabbed a focal point among researchers. Cloud service providers pose numerous security challenges and are highly susceptible to attacks. In the context of cloud computing, the anomalies and insider attacks will deactivate the service providers, which results in the malfunctioning of the entire system. Traditional defense systems in the network are not efficient in handling insider attacks and intrusion. In this work, the anomaly identification technique is developed to identify the attack incidence, and the proposed approach uses the fuzzy min-max neural network (FMM-NN). The classification accuracy is enhanced by the effective identification of features using a neural network. The performance investigation and outcome of the FMM-NN identifies and classifies the real-time attacks in the cloud environment with high identification accuracy.

Anomaly represents deviation from the normal behavior of an event. Detection of anomaly provides means to take appropriate countermeasures in various domains. Examples include detection of fraudulent transactions in banking or financial domain, detection of cyber-attacks in networking environments, detection of abnormal behavior of vital signs of patients in healthcare domain. Also, detection of anomalies with respect to time of arrival of data is crucial in deciding the accomplishment of successful countermeasures. Selection of suitable algorithms or methods for detection of anomalies is also equally important for successful detection of anomalies. In this paper it is proposed to compare the performance of two different algorithms, namely, Isolation Forest (unsupervised) and Random Forest (supervised) by varying the operating parameters of the algorithms. Experiment is carried out using a benchmark dataset that belongs to the healthcare domain. The data is preprocessed for missing values and then detection accuracy of the algorithm is analyzed with respect to the number of records. Results are discussed.

Isolation forest or"iForest" is an intuitive and widely used algorithm for anomaly detection that follows a simple yet effective idea: in a given data distribution, if a threshold (split point) is selected uniformly at random within the range of some variable and data points are divided according to whether they are greater or smaller than this threshold, outlier points are more likely to end up alone or in the smaller partition. The original procedure

suggested the choice of variable to split and split point within a variable to be done uniformly at random at each step, but this paper shows that ”clustered” diverse outliers - oftentimes a more interesting class of outliers than others - can be more easily identified by applying a non-uniformly-random choice of variables and/or thresholds. Different split guiding criteria are compared and some are found to result in significantly better outlier discrimination for certain classes of outliers.

Intrusion detection systems are the foundation of network security (IDS). In order to detect intrusions, IDSs keep a check on the system's activity and behavior. Various IDS models, such as misuse detection and anomaly detection, can be used to identify attacks at all levels. For both known and undiscovered attacks, anomaly detection has a high rate of false positives, but misuse detection has a high rate of detection accuracy for only known assaults. This r presents an intrusion detection system that uses machine learning to solve the shortcomings of current approaches. This research proposes the Hybrid IDS, which employs various supervised machine learning techniques in which Grid search hyper-parameter optimization for Binary and Multiclass classification systems with univariate feature selection is used. Random forest and the multi-layer perceptron neural network algorithm are utilized in supervised machine learning methods. UNSW-NB15, a dataset developed in 2015, is used to evaluate the suggested model's performance. Dataset splitting, data preprocessing, feature extraction and selection, and model training, hyper-parameter tuning, and classification are the four steps of the proposed hybrid intrusion detection algorithm. In terms of intrusion detection, the results obtained show that the suggested model is successful, and it is able to increase accuracy and minimize FAR. In addition, the time it takes to process a request is very minimal.

3. Adaptive Model for Traffic Analysis of Real-Time Attacks

In this section, the proposed traffic analysis of anomaly attacks is discussed, and initially, the method is pre-processed. This data is transmitted to the feature selection phase, and the FMM-NN technique classifies the selected features. The process of classification is detailed in this section.

a. Pre-Processing

Initially, the dataset is prepared for classification by altering the symbols numerically by utilizing representation. In protocol features, the names of the protocols {TCP, UDP, ICM} are converted as numerical values {1, 2, 3}. After the process of representation, the data is normalized as follows,

$$f = \frac{f - minm}{maxim - minm}$$

where the feature in the dataset is indicated as f, the minimum value is shown as minm, and the maximum value is shown as maxim. The process of normalization utilizes the values that lie between (0, 1).

b. Feature Selection

This CNN algorithm uses the NSL-KDD dataset for feature selection for logging into cloud environments. It iterates through all 41 features, training a CNN model for each combination. The performance of the training model is evaluated and the best combination is selected based on the highest test result. This approach improves detection accuracy by focusing on key features, thus improving the overall effectiveness of intrusion detection. By leveraging CNN, the algorithm can effectively manage feature selection and ensure efficient use of computing resources. It provides more efficient access to the cloud environment by simplifying the process of determining key names. The NSL-KDD dataset serves as the basis for training and evaluation of CNN models, providing a reliable basis for feature selection. The methodology of this algorithm will help systematically search for feature combinations to optimize the performance of intrusion detection systems in real cloud environments. In general, it helps strengthen network security measures by increasing the accuracy and efficiency of access detection mechanisms.

Algorithm 1. CNN Procedure

```
# Algorithm 2: CNN Feature election Procedure

# Input: All possible combinations of features, Training Data

def CNN_Feature_Selection(All_Feature_Combinations, Training_Data):
CNN_Storage = {}  # Dictionary to store CNN models and their performance

for combination in All_Feature_Combinations:
# Train CNN model with the current feature combination
CNN_Model = train_CNN(Training_Data[combination])

# Store the CNN model along with its performance metrics
CNN_Storage[combination] = evaluate_CNN_Model(CNN_Model, Validation Data)

# Find the combination with the highest test accuracy
best_combination = max(CNN_Storage, key=CNN_Storage. Get)

# Return the best feature combination and its corresponding CNN model
return best_combination, CNN_Storage[best_combination]
```

3.3 Traffic Analysis

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the average prediction of the individual trees for regression tasks. The key idea behind Random Forest is to introduce randomness in the tree-building process by selecting random subsets of features and samples for each tree, which helps improve the model's robustness and generalization. Decision trees are built recursively by partitioning the feature space into smaller regions based on the values of input features. At each node of the tree, a split is chosen to maximize the reduction in variance (for regression) or increase in information gain (for classification). The splitting criterion can be based on various algorithms such as CART (Classification and Regression Trees), which uses measures like Gini impurity or mean squared error to evaluate splits.

Gini impurity and entropy are commonly used measures of impurity or disorder in decision tree algorithms, including Random Forest Classifier, during model training.

Gini Impurity:

- Gini impurity is a measure of how often a randomly chosen element from a dataset would be incorrectly classified if it were randomly labeled according to the distribution of labels in the subset.
- Mathematically, for a dataset with K classes, the Gini impurity is calculated as:

$$\text{Gini} = 1 - \sum_{i=1}^{K} (p_i)^2$$

where p_i is the probability of choosing a sample of class i in the dataset.

Entropy:

- Entropy is a measure of impurity or disorder in a dataset. It is calculated based on the probability distribution of classes in the dataset.
- Mathematically, for a dataset with K classes, the entropy is calculated as:

$$\text{Entropy} = -\sum_{i=1}^{K} p_i \log_2(p_i)$$

where p_i is the probability of choosing a sample of class i in the dataset.

- Entropy is higher when the dataset contains a mix of classes, and it decreases as the dataset becomes more homogeneous.

For each prediction task (Count, srv_count, duration, or rates), there are two Random Forest model used:

- For regression tasks (Count, srv_count, and duration), use a Random Forest regressor.
- For classification tasks (rates), use a Random Forest classifier.

3.3.1 Random Forest regressor

Random Forest Regressor is capable of capturing complex non-linear relationships between input features (such as same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_same_srv_rate, dst_host_diff_srv_rate, and dst_host_same_src_port_rate) and the target variables (counts, srv_count, and duration). This is crucial as network data often exhibits non-linear patterns that traditional linear regression models may struggle to capture. To build a Random Forest Regressor, multiple decision trees are trained independently on random subsets of the training data (bootstrapped samples) and random subsets of features. During training, each tree learns to predict the target variable (counts, srv_count, and duration) based on the selected numerical features. The predictions from all trees are aggregated to obtain the final prediction. For regression tasks, this typically involves averaging the predictions of individual trees.

Features:

- same_srv_rate: Rate of connections to the same service
- diff_srv_rate: Rate of connections to different services
- srv_diff_host_rate: Rate of connections to different hosts for the same service
- dst_host_same_srv_rate: Rate of connections to the same service for the destination host
- dst_host_diff_srv_rate: Rate of connections to different services for the destination host
- dst_host_same_src_port_rate: Rate of connections from the same source port to the destination host

Let's denote the dataset as

$$D = \{ (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N) \}$$

where x_i represents the input features (same_srv_rate, diff_srv_rate, etc.) and y_i represents the target variable (count, srv_count, or duration).

Random Forest Regressor constructs M decision trees T_m, where m=1, 2... M.

Each decision tree Tm is built using a random subset of the training data (bootstrapped sample) and a random subset of features. The decision tree Tm is trained recursively by partitioning the feature space into regions Rm. At each node of the tree, a split is chosen to minimize the mean squared error (MSE) of the target variable within each region Rm.

The predictions of individual trees Tm are aggregated to obtain the final prediction for a given input xi:

$$\widehat{y_i} = \frac{1}{M} \sum_{m=1}^{M} T_m(x_i)$$

where $\widehat{y_i}$ is the predicted value for the target variable based on the Random Forest Regressor model.

3.3.2 Random Forest Classifier

Random Forest Classifier is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and robustness. Theoretically, the purpose of using Random Forest Classifier in this context is to leverage the collective decision-making of multiple trees to effectively classify rates of attacks based on the provided features. By aggregating predictions from individual decision trees, Random Forest Classifier can capture complex relationships and patterns in the data, leading to more accurate and reliable predictions. In the term "rates of attacks" refers to the frequency or proportion of malicious or unauthorized activities detected within a network or system. These activities would include various types of cyber-attacks, such as intrusion attempts, denial-of-service (DoS) attacks, malware infections, or any other unauthorized access or exploitation of system vulnerabilities. The "rates of attacks" represent the occurrence or frequency of such malicious activities within a certain period or context, and they are typically measured as a numerical value or percentage. These rates are often used as indicators of the security posture of a network or system, with higher rates indicating a higher level of threat or vulnerability. The goal is to predict these rates of attacks based on certain features extracted from the NSL Dataset. These features include same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_same_srv_rate, dst_host_diff_srv_rate, and dst_host_same_src_port_rate, which are numerical values representing different aspects of network traffic and communication patterns. This information can then be used for security monitoring, threat detection, and decision-making in network security operations.

Random Forest Classifier combines the predictions of multiple decision trees, each trained on a random subset of the training data and features.

Let's denote the dataset as

$$D = \{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$$

Where x_i represents the input features and y_i represents the corresponding rates of attacks.
Random Forest Classifier constructs M decision trees T_m, where m=1, 2... M.
Each decision tree Tm is trained using a random subset of the training data (bootstrapped sample) and a random subset of features.
The class label (rate of attacks) predicted by each tree Tm is aggregated using majority voting to obtain the final prediction for an input xi. Random Forest Classifier to predict rates of attacks based on the selected features from the NSL Dataset is to leverage the collective decision-making of multiple trees to improve prediction accuracy and robustness, capturing complex relationships and patterns in the data for more reliable predictions.

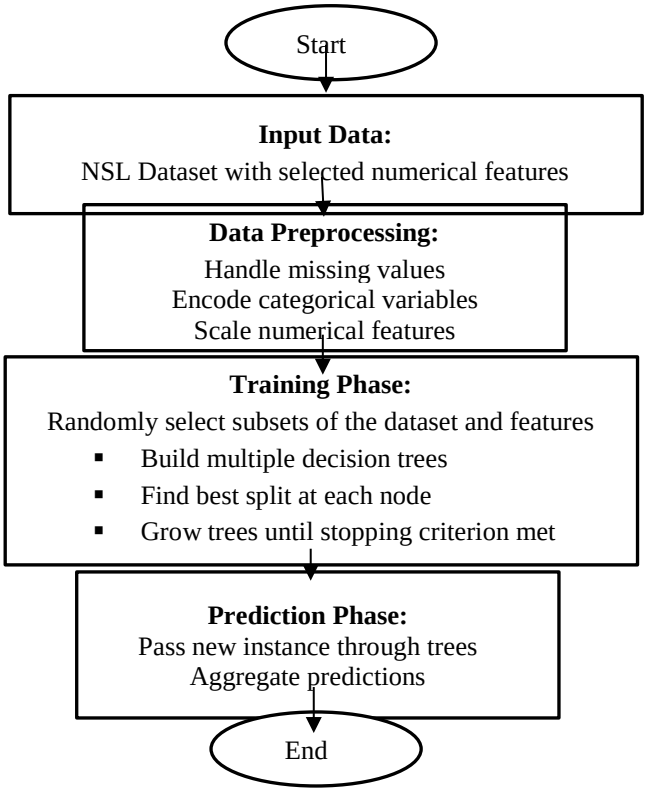


Figure 1. The overall outline of Random Forest Algorithm

4. Result and Discussion

The cloud simulation for the identification of intrusion detection is identified from the offline dataset. The experiment is accomplished with Python, and its machine learning package scikit-learn and other packages such as Pandas and Numpy, NSL-KDD dataset. The Random Forest Algorithm predicts the number of network connections ('counts' and 'srv_count') and the duration of connections and classifies the frequency or rate of occurrence of a particular event or behavior within a given time period. In the research 80% for Training and 20% for testing NSL-KDD Dataset were used.

4.1 Evaluation metrics to assess the performance of the Random Forest Regression
The evaluation metrics in assessing the performance of the Random Forest Regression (or any regression model) is to quantify how well the model is able to predict the target variable (dependent variable) based on the input features (independent variables). These metrics provide insights into the accuracy, precision, and overall effectiveness of the model in making predictions.

4.1.1 R-squared (R^2) score

The R-squared (R^2) score is a commonly used metric for evaluating the performance of regression models, including the Random Forest Regressor algorithm. It measures the proportion of the variance in the dependent variable (target) that is explained by the independent variables (features) in the model. R-squared score is a value between 0 and 1, where:

- 0 indicates that the model does not explain any of the variance in the target variable.
- 1 indicates that the model explains all of the variance in the target variable.

The formula for calculating the R-squared score is:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

where,

- SS_{res} is the sum of squared residuals, which is the sum of the squared differences between the actual values (y_i) and the predicted values ($\hat{y_i}$)
- SS_{tot} is the total sum of squares, which is the sum of the squared differences between the actual values (y_i) and the mean of the actual values($\bar{y}$).

The model predicts the value 0.63 for the dataset and it is a highly predicted value. The R-squared (R^2) score ranges between 0 and 1, where 1 indicates a perfect fit, and values closer to 1 indicate better model performance in terms of predicting the target variable "rate"

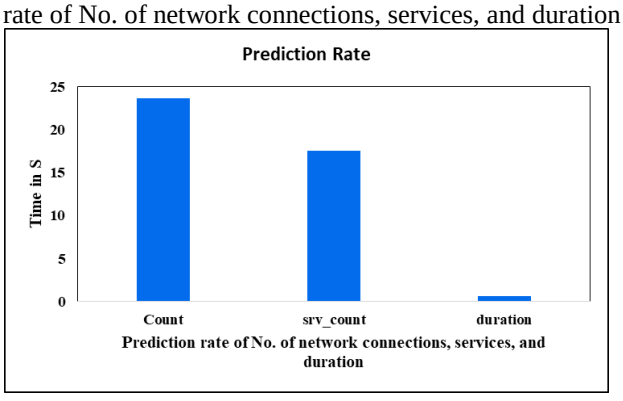
4.1.2 Prediction Rate

The Random Forest Classifier algorithm predicts the total number of network connections or events (count) that occur within a specific time period, number of different services or protocols (src_count) used in the network connections, length of time (duration) for each network connection or event per second. From the observation of numerical outcome in Table 1 and Figure 1, the Random Forest Classifier acquires the highest prediction rate in seconds

Table 1: Prediction rate of No. of network connections, services, and duration of each connection

Prediction	**Time (S)**
Count	23.7
srv_count	17.57
duration	0.59

Figure 1: Prediction rate of No. of network connections, services, and duration of each connection



4.2 Evaluation metrics to assess the performance of the Random Forest Classification

4.2.1 Accuracy

Accuracy is a commonly used metric to measure the performance of the Random Forest Classifier. It represents the proportion of correctly classified instances out of all instances in the dataset. The accuracy score is calculated using the following formula:

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}$$

where,

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

In Random Forest Classifier for predicting rates using features from the NSL Dataset, accuracy measures how often the classifier correctly predicts the rate category (e.g., whether an instance corresponds to a specific rate level) based on the selected features. The accuracy of classification is 0.97 of rate of occurrence of DoS attacks such as back, neptune, teardrop, smurf.

4.2.2 Precision

Precision is a metric used to evaluate the performance of a Random Forest Classifier. It measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives + false positives). Precision is particularly useful when the cost of false positives is high. Precision is calculated using the following formula:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

In the Random Forest Classifier for predicting rates using features from the NSL Dataset, precision measures how accurately the classifier identifies instances belonging to a specific rate category (e.g., whether an instance is correctly classified as a specific rate level) among all instances predicted to belong to that category.

4.2.3 Recall

Recall, also known as sensitivity or true positive rate, is a metric used to evaluate the performance of a Random Forest Classifier. Recall measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances in the dataset. Recall is particularly useful when it is important to capture all positive instances, even at the cost of more false positives. Recall is calculated using the following formula:

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

In a Random Forest Classifier for predicting rates using features from the NSL Dataset, recall measures how effectively the classifier identifies instances belonging to a specific rate category (e.g., whether an instance is correctly classified as a specific rate level) out of all instances that actually belong to that category.

4.2.4 F1 Score or F – Measure

The F1 Score is a metric used to evaluate the performance of a classification model, including the Random Forest Classifier. It is the harmonic mean of precision and recall and provides a balance between the two metrics. The F1 Score is particularly useful when there is an uneven class distribution (class imbalance) in the dataset.

$$\text{F1 Score} = \frac{2 * (\text{Precison} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

The F1 Score is calculated using the following formula:
In the context of evaluating a Random Forest Classifier for predicting rates using features from the NSL Dataset:

- Precision represents the proportion of correctly predicted positive instances out of all instances predicted as positive.
- Recall represents the proportion of correctly predicted positive instances out of all actual positive instances.

From the observation of numerical outcome in Table 2, Table 3, Figure 2 and Figure 3, the Random Forest Classifier acquires the highest precision, recall, and f-measure of classification of rate of Occurrence of normal and DoS types of attacks.

Table 2: Classification of Rate of Occurrence of Attacks

Attack Type	Precision	Recall	F-Measure
normal	0.98	0.99	0.98
back	0.96	0.91	0.92
neptune	0.96	0.91	0.92
teardrop	0.96	0.91	0.92
back	0.96	0.91	0.92

Table 3: Classification of Rate of Occurrence of Attacks f-measure information

Attack Type	TP	FP	TN	FN
normal	323	1	421	5
back	135	1	611	7
neptune	142	1	602	5
teardrop	86	2	656	6
back	49	1	697	3

Figure 2: Classification of Rate of Occurrence of Attacks

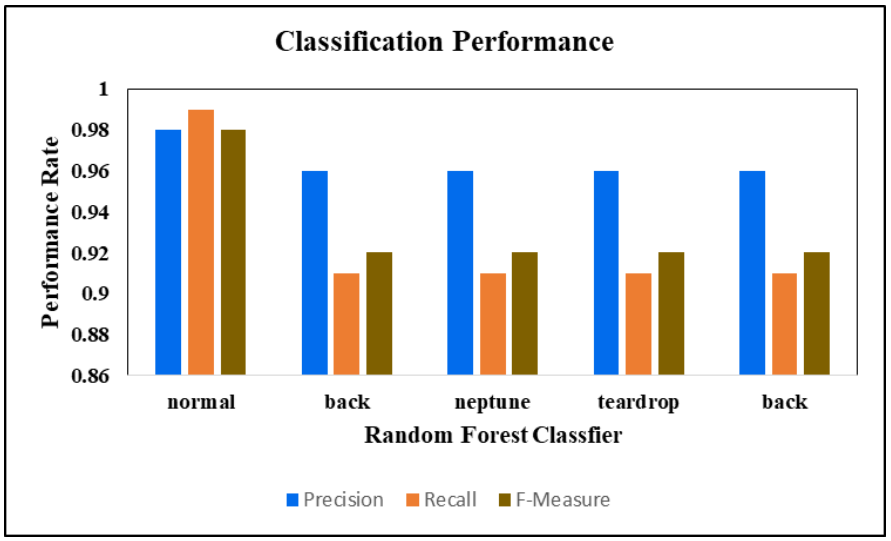
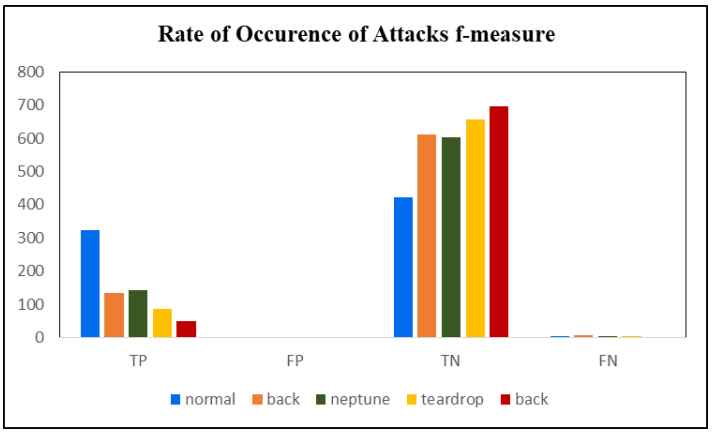


Figure 3: Classification of Rate of Occurrence of Attacks f-measure information



5. Conclusion

The research successfully predicts various metrics related to network traffic, including counts, service counts, duration, and rates of occurrence of different types of attacks. The paper evaluates the performance of the Random Forest model by using various metrics and provides insights into the predictive capabilities of the models for different types of attacks and network traffic patterns. The Random Forest algorithm exhibits high prediction rates and accuracy for predicting network traffic metrics and identifying different types of attacks. It effectively captures complex non-linear relationships between input features and target variables, making it suitable for anomaly detection in cloud environments. In future , augmenting the NSL Dataset with additional real-world data or generating synthetic data using AI based advanced techniques could help in training more robust models and improving their generalization capabilities and developing mechanisms for real-time monitoring and detection of anomalies in cloud environments could be valuable. Integrating the anomaly detection system with cloud infrastructure monitoring tools and security incident response mechanisms can enhance the overall security posture.

6. References

1. D.Sakthivel, Dr.B.Radha, “Adaptive Model to Detect Anomaly and Real-Time Attacks in Cloud Environment Using Data Mining Algorithm”, International Journal of Performability Engineering (IJPE), 2021, Vol. 17, Issue (10).
2. D.Sakthivel, Dr.B.Radha, “A Study on Security Issues and Challenges in Cloud IaaS,” International Journal for Research in Applied Science & Engineering Technology (IJRASET), 6(3), 909-914, November 2018.
3. Chellammal Suriyanarayanan, and Saranya Kunasekaran. “Anomaly Detection Using Machine Learning Techniques”. Malaya Journal of Matematik, vol. 8, no. 04, Oct. 2020, pp. 2144-8.
4. David Cortes, “Revisiting randomized choices isolation forests”, https://arxiv.org/abs/2110.13402, December 7,202.
5. Priya R. Maidamwar, Dr. Mahip M. Bartere, Dr. Prasad P. Lokulwar, “Classification of Hybrid Intrusion Detection System Using Supervised Machine Learning with Hyper-Parameter Optimization”, JOURNAL OF ALGEBRAIC STATISTICS Volume 13, No.3, 2022, p.1532-1550.
6. D.Sakthivel, Dr.B.Radha, “Detection of Signature Based Attacks in Cloud Infrastructure using Support Vector Machine,” https://link.springer.com/chapter/10.1007/978-981-16-7018-3_38, First Online: 03 March 2022
7. D. Sakthivel, Dr.B. Radha, “Novel Study on Machine Learning Algorithms for Cloud Security”, JOURNAL OF CRITICAL REVIEWS, VOL 7, ISSUE 10, 2020.
8. D. Sakthivel, Dr.B. Radha, “SNORT: Network and Host Monitoring Intrusion Detection System”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 6 Issue X, Oct 2018
9. Nada Aboueata, Sara Alrasbi, Aiman Erbad” Supervised Machine Learning Techniques for Efficient Network Intrusion Detection”, IEEE, 2019
10. Almasoudy, F. H., Al-Yaseen, W. L., N Idrees, A. K, “Differential evolution wrapper feature selection for intrusion detection system,” Procedia Computer Science, 167, 1230-1239, 2020.

11. Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J, "DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system," Security and Communication Networks, 2020.
12. Simpson, P. K., "Fuzzy Min—MaX Neural Networks—Part 1: Classification," IEEE Trans. on Neural Networks, 3(5), 776-786, 1992.
13. Kazi Abu Taher, Billal Mohammed YasinJisan, Md. Mahbubur Rahman "Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection ",International Conference on Robotics, Electrical and Signal Processing Techniques, 2019.
14. Monica Gadre, Shruti Chincholkar, "Data Security in Cloud Security Attacks and Preventive Measures", International Journal for Research in Applied Science & Engineering Technology, March 2018.
15. S. Adepu and A. P. Mathur, ``Distributed attack detection in a water treatment plant: Method and case study," IEEE Trans. Depend- able Secure Compute., vol. 16, no. 1, pp. 1_14, Jan./Feb.2018.