IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Network Intrusion Detection Using Supervised Machine Learning Technique With Feature Selection

[1]C.R.MURALI, [2]Dr. G. Mohan Ram
[1]M.Tech. Scholar, [2]Associate Pofessor
[1]School Of Engineering,Malla Reddy University,
[2]School Of Engineering,Malla Reddy University

ABSTRACT

Intrusion detection systems (IDS) are required to protect the modern communication networks. The main aim of such systems was pattern recognition, signature analysis and rule violation detection. The past few years have been promising in the field of network intrusion detection due to the implementation of Machine Learning and Deep Learning methods. These techniques are able to distinguish normal and abnormal trends. This paper checks Network Intrusion Detection Systems (NIDS) with the NSL-KDD benchmark data set using different forms of machine learning algorithms such as Support Vector Machines (SVM), Random Forest (RF), Decision Tree and Logistic Regression among others. We go into depth on how those methods worked, and how their findings surpassed those of earlier studies.

Chapter 1 INTRODUCTION

The installation of a firewall is therefore a typical security technique. In order to decide whether or not to let packets through, a packet filter (firewall) looks at fields in the header of the packet, including ICMP, TCP, IP, and UDP. This process was covered earlier in the section. While packet filters are useful, Deep Packet Inspection (DPI) is necessary for detecting many different kinds of attacks. An intrusion prevention device may be useful if it undertakes deep packet inspections in addition to analyzing the headers of all packets passing through it, unlike a packet filter. When an Intrusion Prevention System (IPS) device identifies a suspicious packet or sequence of packets, it drops them to stop the unauthorized users from gaining access to the company's network. If a device can let packets to travel through it on their way to the company network, but then either logs them or notifies the network administrator, it is using an intrusion detection system. We will examine intrusion detection in more detail in this part. When it comes to protecting networks and hosts from potential threats, intrusion detection systems are invaluable tools. Protecting a computer system or network against unauthorized access while keeping sensitive information private and accessible is the major goal of intrusion detection

systems. The intrusion detection system will kick in if it finds that an attack has happened and the firewall wasn't able to block it. A firewall is the primary line of defense against unauthorized access. Using the Intrusion Detection System, on the other hand, presupposes that an assault would doom the firewall to nothing. Depending on the platform they monitor or the method they use to detect unusual behavior, Intrusion Detection Systems may be categorized in several ways.

1.1 Functional Significance of Machine Learning

Critical parts of cyber security infrastructure, network intrusion detection systems (NIDS) use machine learning to analyze network traffic for indicators of harmful activity or policy breaches. The rule-based and signature-based techniques used by traditional NIDS work well against known threats but fail miserably when faced with new or complex threats. Machine learning (ML) is a game-changer in this regard, providing intrusion detection systems with adaptive and intelligent capabilities that make them more efficient and effective. Through the use of machine learning, NIDS is able to autonomously learn from past data patterns of typical and unusual network activity. Unlike signature-based methods that require prior knowledge of an attack, ML models can detect deviations from established baselines— also known as anomaly detection. In order to detect zero-day attacks or other unknown dangers, this capacity is crucial.

1.2 Integrating Network Intrusion Detection Systems with Machine Learning Models

ML algorithms, especially those optimized for streaming data (e.g., online learning), can process large volumes of network traffic in real time. This is crucial in preventing or mitigating damage from attacks as they happen. Real-time capabilities enable swift alerting and response, which is a critical requirement in dynamic network environments. A significant drawback of conventional NIDS is the frequent occurrence of false positives, in which harmless actions are mistakenly identified as harmful ones. Machine learning models can also be trained to better distinguish between legal and malicious behavior with respect to improve reliability of warnings and reduce noise. This can be particularly true of managed learning such as Random Forests or Support Vector Machines (SVM). The ML-based-systems are capable of developing as time goes, learning afresh after every piece of data. This versatility is essential in cybersecurity where the strategies used by the attackers change very often. Based on unsupervised learning strategies such as dimensionality reduction (e.g. PCA, t- SNE) and clustering (e.g. K-means, DBSCAN), behavioral baselines can be learned dynamically without labeling.

1.3 Proposed ML-Random Forest and ML-Decision Tree Frameworks

A significant drawback of conventional NIDS is the frequent occurrence of false positives, in which harmless actions are mistakenly identified as harmful ones. By training machine learning models to differentiate between legitimate and malicious activity, noise can be reduced and alert reliability improved. This is especially true of supervised learning techniques such as Support Vector Machines (SVM), Random Forest (RF), Decision Tree and Logistic Regression algorithms. The proposed study introduces several hybrid models that integrate especially in classification principles with machine learning structures:

This research used a variety of supervised machine learning methods, including Support Vector

Machines (SVMs), Random Forest (RFs), Decision Trees, and Logistic Regressions, to assess NIDS's performance on the NSL-KDD dataset. The normal data collection, preprocessing, feature extraction, classification, modelfitting and performance matrices are some of the common processes involved in any machine learning classification. Compared to other models our Random Forest and Decision Tree models scored the optimal accuracy mark.

1.4 Addressing Model Complexity and Training Challenges

Intrusion detection in computer networks is a critical task for maintaining cybersecurity. Machine Learning (ML)-based Network Intrusion Detection Systems (NIDS) offer significant promise, but their development involves several key challenges related to model complexity and training. Below is an in-depth analysis of these challenges and strategies to address them.

a) **High Dimensionality of Network Data:** Numerous characteristics, such as packet size, protocol type, and time-based statistics, are often included in data on network traffic.Because of the "curse of dimensionality," which occurs when dimensionality is high, learning becomes inefficient and models are more likely to overfit.

b) **Model Interpretability vs. Performance:** Deep learning models (e.g., CNNs, RNNs) offer high detection accuracy but are black boxes. Security professionals prefer interpretable models for understanding and auditing.

c) **Computational Overhead:** It takes a lot of processing power to train and infer from complex models in real time, such as deep learning or ensemble learning.

d) **Labeling and Ground Truth Challenges:** Labeling network traffic data requires expert knowledge and is resource-intensive. Lack of high-quality labeled datasets limits supervised learning.

e) **Noise and Redundancy in Network Data:** Real-world traffic contains noise and redundant features, reducing model accuracy.

1.5 Handling Challenges in ML-Random Forest and ML-Decision Tree Models

Machine learning methods like Decision Trees (DT) and Random Forests (RF) are widely used for network intrusion detection because they are easy to understand and perform well on structured data. These algorithms aim to tackle these difficulties. There are a number of technological hurdles to overcome before they can be successfully used to an intrusion detection system (IDS) in the real world, especially when employing benchmark datasets such as NSL-KDD. A comprehensive approach to understanding and resolving these issues is provided below. By eliminating duplicate entries and adjusting for class imbalance, the NSL- KDD dataset outperforms its predecessor, the KDD Cup 1999. Training and evaluating ML- based IDS makes extensive use of it.

The data collection includes:

- 41 features per record (e.g., protocol_type, src_bytes, dst_bytes).
- Classes: normal and intrusion categories.

a) **Class Imbalance:** Ordinary or intrusive records make up the majority. Due to the rarity of samples, detection rates for rare attack types such as U2R and R2L are low.

b) **Overfitting in Decision Trees:** Deep, complicated trees that remember the training data are often produced by Decision Trees, particularly when pruning is not performed.

c) **High Feature Correlation and Redundancy:** NSL-KDD has several features that are irrelevant or highly correlated, which can mislead tree-based splits.

d) **Long Training Time in Random Forests:** RF trains multiple trees (often hundreds), which increases training time and memory usage, especially on large datasets.

e) **Interpretability for Decision Making:** Although DTs are interpretable, Random Forests are not easily explainable due to ensemble complexity. Random Forests can be computationally expensive during prediction if not optimized.

1.6 Advantages of the Proposed Framework

The use of ML models for real-time network intrusion detection has several benefits over the more conventional methods that rely on rules or signatures. Listed below are the main advantages:

a) **Real-Time Threat Detection:** Once trained, machine learning models can quickly analyze and categorize network data. Suitable for detecting malicious activities as they happen (e.g., denial-of-service attacks, port scans), allowing prompt response.

b) **Adaptability to New/Unknown Attacks:** Machine learning methods, in contrast to rule-based systems, are able to identify zero-day threats by analyzing trends and outliers. Unsupervised and anomaly detection models are especially useful in identifying deviations from normal behavior.

c) **Automation and Reduced Human Dependency:** ML models reduce the need for constant manual updates and tuning of rules. They can automatically learn from new data, minimizing human oversight in threat detection processes.

d) **Efficient Handling of Large-Scale Network Traffic:** ML models can be deployed in distributed environments and scale easily with big data infrastructure. Capable of monitoring high-volume traffic in cloud, enterprise, or IoT environments.

e) **Pattern Recognition beyond Human Capability:** ML models (especially deep learning) can capture complex patterns in multidimensional data that traditional methods miss. Enables early-stage detection of stealthy or multi-stage attacks.

f) **Continuous Learning and Improvement:** Using incremental learning approaches, models may be retrained or updated to represent the changing behavior of networks. Helps in mitigating issues

like concept drift in real-world network environments.

g) **Improved Accuracy over Static Rule Sets: ML** algorithms (e.g., Random Forest, XGBoost, SVM) often achieve higher precision and recall compared to static rules.

Ensemble and hybrid models can further enhance detection performance.

h) **Reduction in False Positives and Negatives:** ML models can be fine-tuned to balance false alarms and missed detections using advanced optimization and evaluation techniques. This leads to more actionable alerts and reduced alert fatigue for security teams.

1.7 Application Areas and Future Potential

Network Intrusion Detection Systems (NIDS) using Machine Learning (ML) have become a critical component in modern cyber security infrastructure. Detailed descriptions of their current and prospective uses are provided below:

Application Areas of Network Intrusion Detection Using ML.

a) Enterprise and Cloud Security: Detects internal and external attacks such as DDoS, data exfiltration, and lateral movement. Monitors cloud traffic patterns (e.g., AWS, Azure) for anomalies and zero-day exploits.

b) Internet of Things (IoT) Environments: Identifies abnormal behavior in connected devices, which often lack robust built-in security. Machine learning models trained on network traffic can flag compromised IoT nodes.

c) Industrial Control Systems (ICS) & SCADA: Protects critical infrastructure like energy grids and water systems from cyber-attacks. Learns normal command patterns to detect protocol violations and unauthorized access.

d) Smart Cities and Autonomous Systems: Enhances security in smart traffic management, surveillance, and vehicle-to-infrastructure communications. Real-time intrusion detection in connected transport systems.

e) Mobile and Wireless Networks: Identifies rogue access points, spoofed MAC addresses, and wireless eavesdropping. Useful in securing 5G and edge networks where traditional security tools may fall short.

f) Financial and E-commerce Platforms: Detects fraudulent behavior, phishing attempts, and botnet activities in transactional networks. ML models can flag patterns deviating from normal financial behavior.

g) Government, Military, and National Defense Networks: Monitors secure communication channels for advanced persistent threats (APT). Provides threat intelligence through behavioral analysis and correlation of attack signatures.

Chapter 2 LITERATURE REVIEW

Cloud-Network-Signatures was trained and evaluated with the use of four ML algorithms: Support Vector Machine (SVM), Logistics Regression, Decision Tree, and Random Forest (RF). An essential part of getting the data ready to feed into the algorithm is pre-processing it. The purpose of data preparation is to ensure that IDS receives correct data by removing any ambiguity from the dataset. Feature selection and normalization are brought together by it. Nominal values are assigned to several symbolic characteristics in the dataset, including protocol types and flags. We used Support Vector Machines (SVMs), Random Forest (RFs), Decision Tree and Logistic Regression algorithms for each feature selection approach to assess NIDS's performance on the Cloud-Network-Signatures dataset in this research.

2.1 Integration of ML Networks with Ensemble Principles

A Network Intrusion Detection System (NIDS) can combine Machine Learning Networks with Ensemble Principles to enhance detection rates, and reduce false positives and to deal with emerging threats. Here's a structured explanation of how this integration works and how to implement it in a project or research paper:

1. **Ensemble Learning:** Ensemble Learning combines multiple classifiers, in order to develop a more powerful model of prediction. Accuracy, reduction of bias and variance, and enhancement of generalizability are all improved.
2. **Stacking:** Combine base model outputs using a meta-learner (e.g., Logistic Regression)
3. **Voting:** Use hard or soft voting to aggregate predictions.
4. **Boosting:** Use sequential models where each focuses on correcting previous errors.
5. **Detection Engine:** Run live or simulated network traffic through the trained ensemble model. Traffic is classified as Normal or Intrusion (and possibly the type of attack). Message trigger alerts/logs on detection.

2.2 Iterative Weak Supervision in Machine Learning

Implementing Iterative Weak Supervision (IWS) for Network Intrusion Detection Systems (NIDS) using the NSL-KDD dataset is a practical way to improve detection performance when high-quality labeled data is scarce. Weak Supervision means Instead of relying solely on manually labeled data, weak supervision uses noisy or imprecise sources of labeling such as: Heuristics or domain rules, Pre-trained models, Crowd-sourced or incomplete annotations. To apply Iterative Weak Supervision (IWS), you start with noisy labels from weak supervision. Use a model to learn from these and refine the labels in an iterative fashion. Each iteration improves the quality of pseudo-labels and, consequently, the model. Here’s how you can structure your project or implementation:

1. **Preprocess NSL-KDD:** normalize features, one-hot encode categorical variables (like protocol_type, service, flag). Split into train, validation, and test sets.
2. **Create Weak Labeling Functions:** Define simple rule-based or heuristic labeling functions (LFs). For example: If src_bytes == 0 and dst_bytes > 0 → likely DoS. If protocol_type == "icmp" and wrong_fragment > 0 → suspicious. Each function votes on whether a sample is malicious or normal. These are weak, noisy, and may conflict.
3. **Apply a Label Aggregation Method:** Use a label model like Snorkel's LabelModel to combine weak labels. It resolves conflicts and estimates true labels probabilistically.
4. **Train a Discriminative Model:** Use aggregated labels to train a classifier like: Logistic Regression / Random Forest (for interpretability). Deep Neural Network (for higher performance). Loss is based on probabilistic (soft) labels rather than hard labels.
5. **Iterative Refinement:** Relabel the unlabeled or poorly labeled data using the learned model. Feed these new labels into the next iteration of training. Repeat 2–5 times, each time using a better model to create better pseudo-labels.
6. **Evaluation:** Evaluate on the ground truth test set of NSL-KDD. the metrics to study are Precision, Accuracy, Recall, F1-Score, and AUC-ROC.

2.3 Boosting-Inspired Architectures in Machine Learning

There has been a dramatic improvement in NIDS thanks to the development of ML and deep learning algorithms. Intrusion detection systems that utilize machine learning seek to automatically teach themselves to identify a harmless network data pattern versus a harmful pattern. Below are some prominent machine learning-inspired architectures and concepts used in network intrusion detection:

1. **Traditional Machine Learning Models**: Those are baseline and are commonly applied even nowadays when working with structured datasets such as NSL-KDD or CIC-IDS. k-NN, Decision Trees (DT), random forest (RF), Support Vector Machines (SVM), Naive Bayes (NB), and Gradient Boosting (XGBoost, LightGBM) are some examples of machine learning models.
2. **Deep Learning Architectures:** include the following: ANN, CNN, RNN, auto encoders, and DBN, all of which are used extensively and serve as basic networks for artificial neural networks.
3. **Hybrid and Ensemble Architectures:** CNN and LSTM work together to extract spatial characteristics and temporal relationships, respectively. Autoencoder + SVM: AE for feature compression, SVM for anomaly detection. Random Forest + Deep Neural Network: Feature selection via RF, classification via DNN.
4. **Graph-Based Learning (GNNs):** Models network traffic as graphs where nodes represent entities (IP addresses, ports). Graph Neural Networks (GNNs) learn relationships and detect suspicious connections.
5. **Transformer-Based Architectures:** Attention mechanisms simulate traffic patterns with long-term interdependence. Recently applied to detect advanced persistent threats (APTs) and stealthy

attacks. For example: Using BERT-style embeddings of network flows for classification.

6. **Federated and Distributed Learning:** Train models across distributed edge devices (routers, switches) without centralizing sensitive data. Useful in privacy-sensitive or large-scale environments (e.g., IoT networks)
7. **Reinforcement Learning for Adaptive NIDS:** Agent learns to respond to threats over time by receiving feedback. Used for dynamic policy adjustment or threat mitigation.
8. **Generative Models (GANs):** Used for synthetic data generation to augment training datasets. Can also detect anomalies by learning data distributions.

2.4 Our Approach: Unified Boosting for CNN and ML

Unified boosting of CNN and ML combines the deep learning power of CNNs and the generalization and interpretability of ML models, enhanced by a boosting strategy to improve intrusion detection in modern networks. Attacks in network traffic may be detected using Network Intrusion Detection Systems (NIDS). When working with geographical data, CNNs really shine at feature extraction and pattern identification, but when it comes to classification, ML algorithms like Random Forests, XGBoost, or SVMs usually do a fantastic job.

- Use Auto encoders for anomaly detection before classification
- Implement attention mechanisms in CNN to prioritize important features
- Use feature selection methods like Recursive Feature Elimination (RFE)
- Apply explain ability tools like SHAP or LIME to interpret ML classifier outputs

2.5 Summary of Key Research Challenges

A brief overview of the main research obstacles in developing and implementing NIDSs using ML models is provided here:

- **High Dimensionality and Feature Selection:** There are usually a lot of superfluous or useless details in data about network traffic. Selecting the most informative features is crucial for improving detection accuracy and reducing computational costs.
- **Data Imbalance**: Real-world intrusion datasets are heavily skewed, with far more normal traffic than attack instances. This imbalance leads to biased models that perform poorly in detecting rare but critical intrusions.
- **Lack of Quality and Realistic Datasets:** Many widely used datasets (e.g., KDDCup99, NSL-KDD) are outdated or do not reflect modern threats. There are a lot of privacy and security considerations when it comes to collecting actual, tagged, and current data on network traffic.
- **Evasion and Adversarial Attacks:** Attackers can craft inputs to fool machine learning models (e.g., adversarial examples). Making models robust against such evasion techniques is an active area of research.

- **Concept Drift:** Network behaviors and attack techniques evolve over time. ML models must adapt to this "concept drift" to remain effective, requiring continuous learning mechanisms.
- **Real-Time Detection and Scalability**: ML models are in need of large quantities of data processed in almost close time with low lag time. Achieving high performance without sacrificing detection accuracy is difficult.
- **Interpretability and Explainability:** Black boxes are a common trait among ML models, particularly those that use deep learning. Security professionals need understandable explanations for alerts to trust and act on them.
- **False Positives and Alert Fatigue:** Overwhelmed analysts may find the IDS useless if its false positive rate is high. Balancing sensitivity and specificity is a key tuning problem.

Each of these problems highlights a potential avenue for further investigation into how to strengthen intrusion detection systems that rely on machine learning. Let me know if you'd like elaboration on any of these points or need references for a report or thesis.

2.6 Our Strategy to Address the Gaps

Addressing the gaps in detecting network intrusions using machine learning (ML) involves identifying current limitations in research and practice, and proposing viable solutions or future research directions. Below is a structured overview of key gaps and how they might be addressed:

1. **Data Imbalance (Skewed Datasets)**
   - Intrusion datasets often contain far more normal traffic than malicious activity.
   - ML models trained on such imbalanced data may ignore rare, but critical, attacks.

   Solution:
   - Use data resampling techniques like Oversampling (e.g., SMOTE) minority classes and Undersampling majority classes.
   - Apply anomaly detection or cost-sensitive learning approaches.
   - Explore generative models (e.g., GANs) to synthesize realistic attack samples.
2. **Lack of Realistic, Up-to-Date Datasets:** Popular datasets such as KDDCup99 and NSL-KDD do not reflect the variety of recent attacks and are thus obsolete.

   Solution:
   - Develop and publish modern, labeled intrusion datasets (e.g., CICIDS2017, TON_IoT, CSE-CIC-IDS2018).
   - Encourage collaborative dataset sharing between industry and academia under privacy-preserving conditions.
   - Explore federated learning and synthetic data generation to bypass data access constraints.
3. **Vulnerability to Adversarial Attacks:** The use of adversarial inputs allows attackers to circumvent detection systems.

   Solution:

- Implement adversarial training to harden models against evasion.
- Use ensemble and randomized models to reduce the success of targeted attacks.
- Research robust ML algorithms specifically for security applications.

4. **Real-Time Performance and Scalability:** Some ML models are too heavy to process high-speed network data in real time.

   Solution:

   - Optimize model architectures and use lightweight algorithms.
   - Frameworks for stream processing, such as Apache Flink and Kafka, should be used.
   - Make use of distributed computing platforms and hardware acceleration tools (such as GPUs and TPUs).

5. **High False Positive Rates:** Excessive false alerts reduce the trustworthiness and usefulness of the system.

   Solution:

   - Employ ensemble techniques to boost prediction accuracy.
   - Use feedback loops and human-in-the-loop learning to refine model behavior.
   - Incorporate context-aware analysis (e.g., user behavior, system logs).

6. **Integration with Traditional Security Systems:** ML-based IDS often exist in silos, without proper integration into SOCs (Security Operations Centers).

   Solution:

   - Design ML solutions as modular APIs or plug-ins that can work with SIEM tools.
   - Provide human-readable outputs and alert prioritization.
   - Ensure interoperability through standard communication protocols.

7. **Privacy and Ethical Challenges:** Data collection and analysis of network traffic poses potential privacy risks to users.

   Solution:

   - Use methods that protect users' privacy, such federated learning and differential privacy.
   - Follow legal frameworks (e.g., GDPR, HIPAA) and anonymize sensitive data.
   - Incorporate ethical risk assessments during model design and deployment.

Addressing these gaps requires a multi-disciplinary approach involving machine learning, cyber security, network engineering, and ethics. Researchers and developers must balance accuracy, performance, and trust while staying responsive to emerging threats.

Chapter3

DESIGN ANALYSIS AND IMPLEMENTATION

This article presented the network intrusion detector system based on machine learning. Assessing how well the suggested detection system performs on the NSLKDD dataset using a variety of machine learning methods. When it came to forecasting the malicious packets, the findings showed that decision tree and Random Forest algorithms outperformed the other models. This was particularly true

when looking at accuracy, recall, and the Mathews correlation coefficient. In addition, cutting-edge intrusion detection systems were beaten by the RF classifier. Despite its flaws, the NSL-KDD dataset does not represent actual assaults as it has uneven classes and the malicious traffic it records is artificial. The classifiers have shown promise and can identify intrusions into networks.

3.1 Design Strategy and Model Construction

Here, an end-to-end tutorial on how to develop a machine learning based network intrusion detection system (NIDS) model exists. It will aim at creating a system to identify malicious activity in the network through supervised learning by means of labeled data. The design strategy involves defining the system's components, data flow, and selection of appropriate ML algorithms. Our objectives are Classify network traffic as normal or intrusion, Support real-time detection, Minimize false positives and false negatives. Our system Design strategy includes data sourcing, processing, and model pipeline. Feature selection, data preparation, training, and assessment are the four main steps in building a model. Begin with classical models and optimize with cross-validation or ensemble techniques. Deploy with an interface for live detection or alerting.

3.2 Machine Learning Architecture for Network Intrusion Detection System

An ML-NIDS is a network intrusion detection system that depends on the concept of machine learning to detect suspicious or malicious activity present in a network through automation. The architecture involves a pipeline that processes raw network data and classifies it as normal or intrusive using intelligent learning algorithms. Here is a typical architecture breakdown:

Architectural Components:

a) **Dataset Collection:** Original KDD dataset is available at the Canadian Institute for Cyber Security and NSL-KDD is a compressed version of it. It has the same characteristics employed in KDD. Every entry has one class attribute and forty one characteristics. To each connection an incursion is attributed or a normal connection. The 39 attacks which constitute NSL-KDD are put into two categories, namely normal and intrusive. We had 995 examples as our training data. Thereafter, 249 cases were employed to test and evaluate models trained. The remaining part of the data was then utilized in validation practice.

b) **Data Preprocessing:** An essential part of getting the data ready to feed into the algorithm is pre-processing it. The purpose of data preparation is to ensure that IDS receives correct data by removing any ambiguity from the dataset. Feature selection and normalization are brought together by it. Nominal values are assigned to several symbolic characteristics in the dataset, including protocol types and flags. If you want your dataset to work better, you need to transform these values to numbers. Dubious datasets in bin 10 have been used to change binary classification issues into intrusion or normal classification problems.

c) **Feature Extraction:** When working on the creation of a machine learning model, it is important to select the training and testing sets of data, one that will allow the viewer to understand the difference between the two. In this work, there were the two halves of the dataset that were utilized: some data was retained to be used as the training data and the other half of data was

utilized as the test data. Our next step was to rebrand each assault label so it could be classified as either "ordinary" or "intrusion" traffic. The following methods from the field of feature engineering are employed.

- Categorical variable one-hot encoding.
- Making numerical characteristics more consistent or normal.
- Dimensionality reduction (PCA, t-SNE) if necessary.

d) **Model Selection & Training:** Several supervised machine learning techniques, such as Support Vector Machines (SVMs), Random Forests (RFs), Decision Trees, or Logistic Regressions, were applied to this research as a strategy to evaluate the effectiveness of NIDS on the NSL-KDD dataset.

Support Vector Machines (SVM):

One of the most widespread supervised machine learning algorithms that might be applied in the classification tasks is Support Vector Machines (SVM), and one of their applications is Network Intrusion Detection Systems (NIDS). Support Vector Machines (SVMs) have shown superior efficacy in identifying and categorizing network intrusions when used on datasets such as NSL-KDD, a benchmark for assessing intrusion detection models. This approach using SVM with RBF kernel and balanced class weights is excellent for the NSL-KDD dataset and evaluating its performance.

- kernel='rbf': Utilizes the Radial Basis Function (Gaussian) kernel, which is effective for non-linear classification.
- class_weight='balanced': Automatically modifies weights inversely relative to class frequencies, beneficial for unbalanced datasets such as NSL-KDD.
- probability=True: Enables probability estimates (useful for ROC/AUC or ensemble methods).
- cls.fit(): Uses the data you provided to train the SVM.
- This appears to be a custom function (prediction), which likely performs cls.predict(X_test). Make sure this function returns a list or array of predicted labels.

Random Forest (RF): A decision-tree-based ensemble method, Random Forest is used in machine learning. In training, it builds a forest of decision trees and then, when asked to classify anything, returns the result with the highest percentage of correct answers. Its success with NSL-KDD highlights its suitability for both research and practical deployment in cyber security. For Model Initialization, we are creating a Random Forest classifier:

- n_estimators=10: Uses 10 decision trees in the ensemble. (Can be increased for better performance.

- criterion="entropy": Uses information gain to split nodes (as opposed to “gini”).
- RFT.fit(X_train, y_train) means Trains (fits) the model on training data. X_train: Features (input) for training and y_train: Labels (target variable) for training.

Decision Tree (DT): When the Decision Tree method is trained on benchmark data including NSL-KDD the network intrusion detection systems (NIDS) which the Decision Tree method becomes part of it. Decision trees are easy to understand and observe. Security analysts can trace the logic used to classify an event as intrusion or normal. This is especially important in security domains where transparency is key.

- Makes a Decision Tree model using the DecisionTreeClassifier function.
- Using the training data (features X_train and labels y_train) the model is trained with DT.fit(X_train, y_train).
- Using the training model, DT.predict(X_test) may anticipate labels (such as attack kinds) for the test characteristics. test X.

Logistic Regression (LR): In the first phases of developing an intrusion detection system using the NSL-KDD dataset, Logistic Regression is a crucial method. Its strength lies in simplicity, speed, and interpretability. While it may not outperform more sophisticated models in complex scenarios, it remains a crucial tool for understanding attack patterns and benchmarking model performanceA logical match would be Logistic Regression, which is fundamentally intended to forecast the likelihood of a binary occurrence. Logistic Regression is computationally lightweight and trains quickly, even on large datasets like NSL-KDD. Apt for resource- constrained detection systems operating in real-time or near-real-time.

- LogisticRegression with a random state of 0 is applied to the data. With this code, the Logistic Regression model is instantiated.
- Results can be reliably reproduced using random state.
- As a training set, X_train, and labels, y_train, are used in the LR.fit(X_train, y_train) statement to train the model.
- The expression prediction(X_test, LR) invokes a new function called prediction, which utilizes the trained model (LR) to make predictions on the test data (X_test).

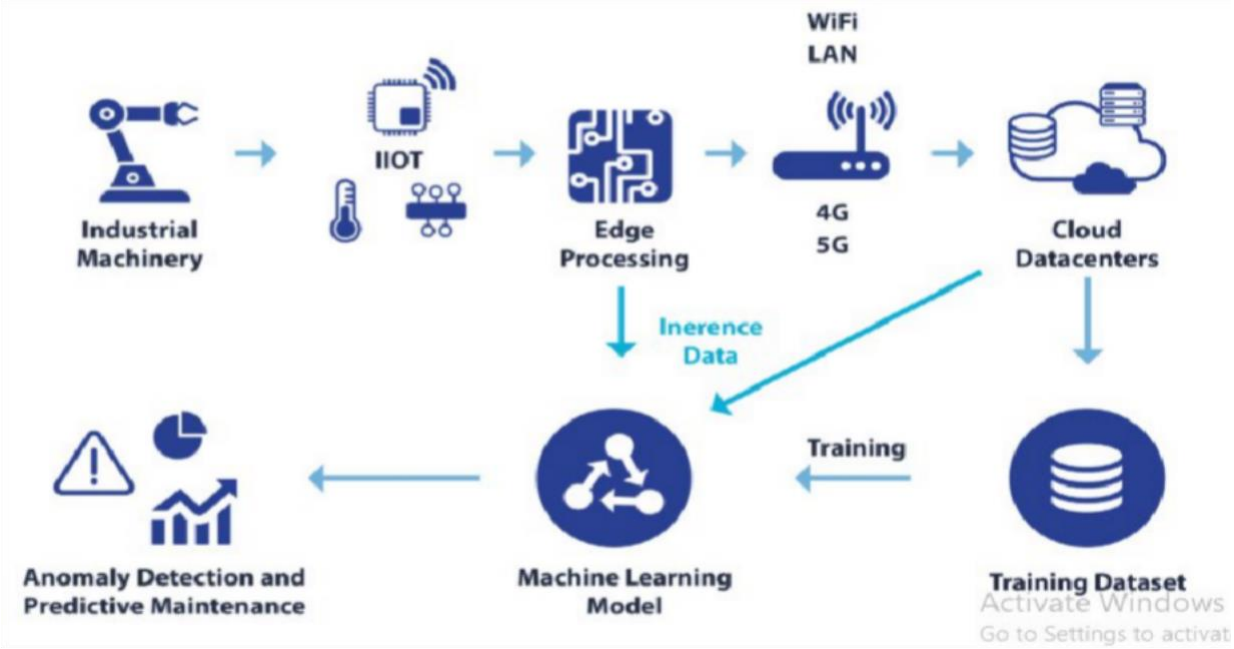

Fig 3.1 Network Intrusion Detection system using Machine Learning Stage

3.3 Performance Evaluation Metrics for Machine Learning Models

The proposed Support Vector Machines (SVM), Random Forest (RF), Decision Tree, and Logistic Regression structures were assessed with the help of different evaluation matrices. These measures helped in giving an excellent understanding of the models, their reliability and the behavior in classification. These metrics go beyond overall accuracy, focusing on how well individual classes are predicted and how errors are distributed across categories.

Confusion Matrix: The confusion matrix presents a class-wise summary of prediction outcomes, showing how many instances of each class were correctly or incorrectly labeled. The number of samples predicted for a class compared to the actual class is quantified by each cell in the matrix, enabling identification of class-specific misclassifications and patterns of confusion across similar categories.

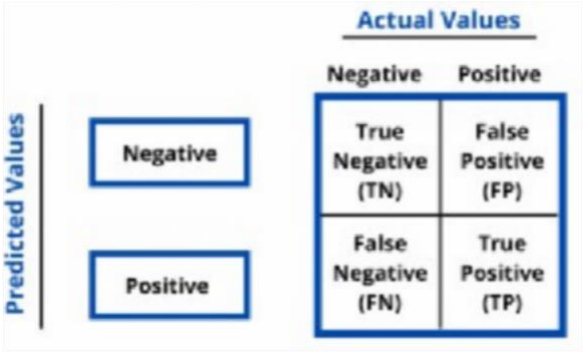

Fig 3.2 Confusion Matrix Architecture

Accuracy: To measure accuracy, we divide the number of inputs by the number of right predictions. While commonly used, it may not reflect true performance in imbalanced datasets. However, since the dataset used for this research, the NSL-KDD contains uniformly distributed classes, accuracy serves as a reliable primary indicator:

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

Classification Report: A classification report is regularly used to measure the effectiveness of a deep learning model on the performance of its classification work. As a rule, it will include the following metrics for every dataset class or label:

Precision: The accuracy rate of a classification is measured by its precision, which is the percentage of correctly classified predictions. The subject of how many samples really belonged to class X out of all the ones projected as such is addressed. This metrics especially useful when minimizing false positives is crucial, such as in applications requiring high confidence predictions:

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TruePositives}+\text{FalsePositives}}$$

Recall: The recall metric measures how well a model can identify all instances of a certain class. How many real samples from class Y were properly identified by the model? Higher recall indicates better sensitivity and is important when missing instances of a class is more costly than over-predicting it.

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives}+\text{FalseNegatives}}$$

F1-measure: The F1-score provides good compromise between accuracy and recall. It is particularly useful when neither metric can be optimized independently without affecting the other. By combining both into a harmonic mean, the F1-score reflects a more balanced view of a model's predictive strength, especially in multi-class tasks like NSL- KDD Dataset:

$$\text{F1-Score} = 2x\frac{\text{PrecisionxRecall}}{\text{Precision}+\text{Recall}}$$

Support: The number of examples of every type in the test data is referred to as support.Although it is not a performance statistic, it describes the dataset's dominating and underrepresented classes and helps understand accuracy, recall, and F1-score values.

Prediction Results

SVM Accuracy

Accuracy : 48.59437751004016

Classification Report:

	precision	recall	f1-score	support
0.0	0.49	1.00	0.65	121
1.0	0.00	0.00	0.00	128
accuracy			0.49	249
macro avg	0.24	0.50	0.33	249
weighted avg	0.24	0.49	0.32	249

Confusion Matrix:

[[121 0]
[128 0]]

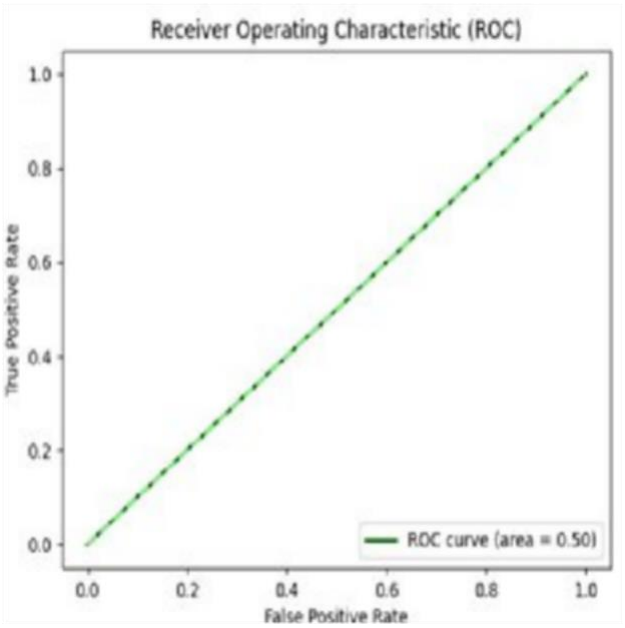

Fig 3.3 SVM Performance Metrics with ROC Curve Graph

LogisticRegression Accuracy

Accuracy : 87.55020080321285

Classification Report:

	precision	recall	f1-score	support
0.0	0.87	0.88	0.87	121
1.0	0.88	0.88	0.88	128
accuracy			0.88	249
macro avg	0.88	0.88	0.88	249
weighted avg	0.88	0.88	0.88	249

Confusion Matrix:

[[106 15]
[16 112]]

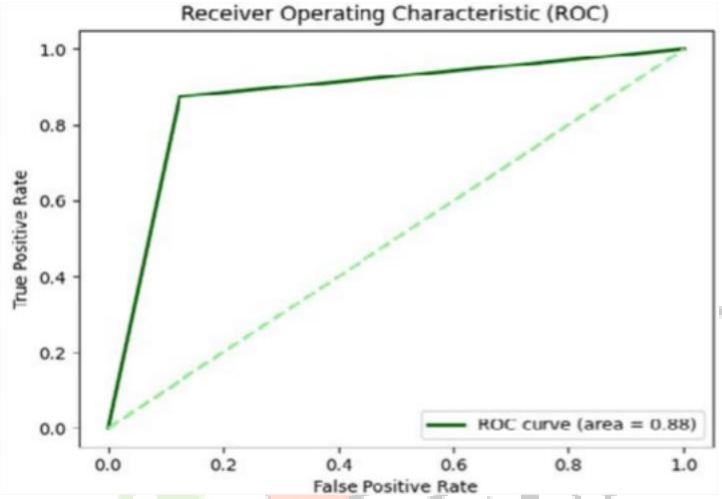

Fig 3.4 Logistic Regression Performance Metrics with ROC Curve Graph

DecisionTreeClassifier Accuracy

Accuracy : 95.98393574297188

Classification Report:

	precision	recall	f1-score	support
0.0	0.97	0.94	0.96	121
1.0	0.95	0.98	0.96	128
accuracy			0.96	249
macro avg	0.96	0.96	0.96	249
weighted avg	0.96	0.96	0.96	249

Confusion Matrix:

[[114 7]
[3 125]]

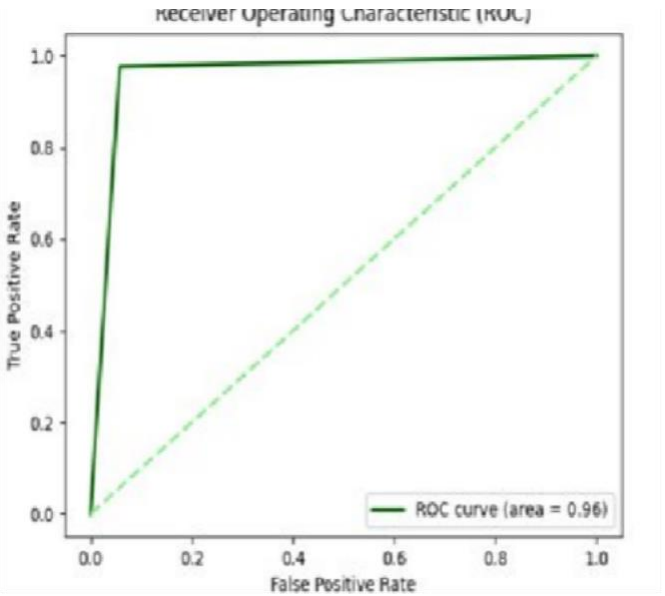

Fig 3.5 Decision Tree Performance Metrics with ROC Curve Graph

RandomForestClassifier

Accuracy : 98.39357429718876

Classification Report:

	precision	recall	f1-score	support
0.0	0.98	0.99	0.98	121
1.0	0.99	0.98	0.98	128
accuracy			0.98	249
macro avg	0.98	0.98	0.98	249
weighted avg	0.98	0.98	0.98	249

Confusion Matrix:
[[120 1]
[3 125]]

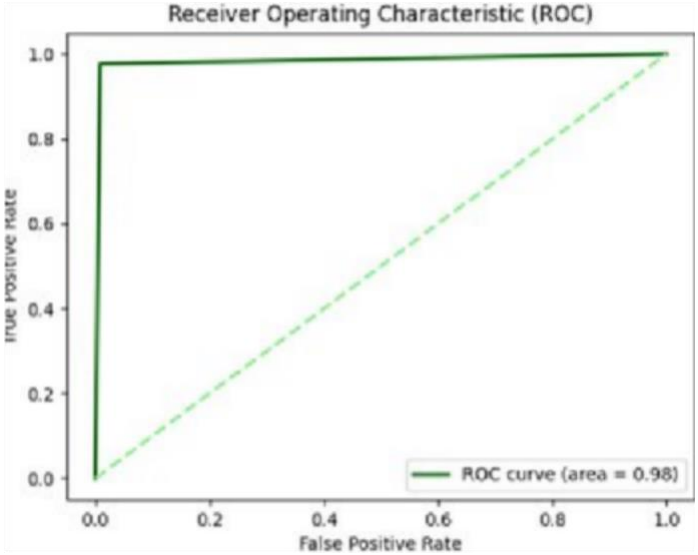

Fig 3.6 Random Forest Performance Metrics with ROC Curve Graph

3.3 Training and Validation Monitoring of Machine Learning Models

It is essential to train and validate machine learning models before constructing an effective Intrusion Detection System (IDS) using the NSL-KDD dataset. In order for the model to effectively identify intrusions, it must first learn patterns from the network's history traffic and then generalize well to unknown data. The following components were used for monitoring:

- **Training Accuracy and Loss:** Follows the performance of the model in training on the data over time.
- **Validation Accuracy and Loss:** Assesses generalization performance on unseen data.
- **Performance Curves:** Epoch-wise plots of accuracy and loss reveal learning patterns and convergence behavior. A steady increase in accuracy and decrease in loss is an indicator of effective training.
- **Early Stopping Criteria:** A system was put in place to stop training when validation performance stopped getting better. This would avoid calculations that weren't essential and help with over fitting..

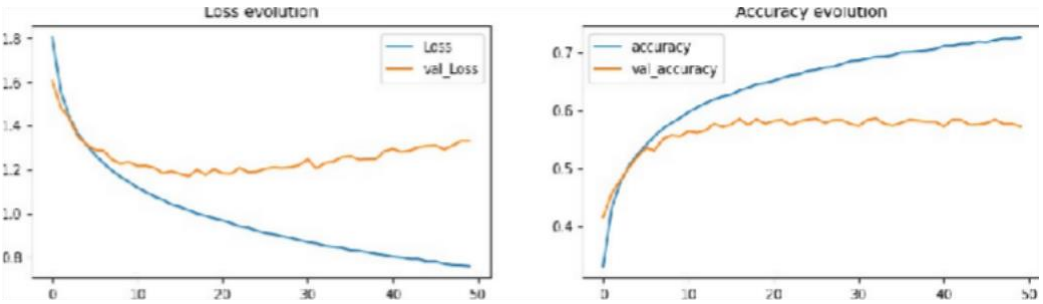

Fig 3.7 Loss Evolution and Accuracy Evolution

3.4 Calculate ROC curve for each Machine Learning Models

Process In a multi-class classification problem, you can compute and plot a Receiver Operating Characteristic (ROC) curve of each of the classes using roc_curve class and the Scikit-learn library. In the case of binary classification, the ROC curve can be used to take a look at the performance of the

model at different thresholds. However, for multi-class problems, you can calculate the ROC curve for each class by considering each class as a "positive" class and the rest of the classes as "negative" (One-vs-Rest approach). Here is how you can compute and plot the ROC curve for each class using Scikit-learn:

Steps:

- **Train the Model:** A trained classifier, such as an SVM, RF, decision tree, or logistic regression method, is required..
- **Obtain Class Probabilities:** Use predict_proba() to get the predicted probabilities for each class.
- **Calculate the ROC Curve:** For multi-class classification, you need to use the One-vs-Rest method. This requires constructing the ROC curve for each class as if it were a positive class and every other class a negative class.
- **Plot the ROC Curve:** Display the ROC curve for every category with the Area under the Curve (AUC), a typical measure for ROC curve assessment.

3.5 Detailed Explanation of Training Classification Using Machine Learning Models

3.5.1 Dataset Preprocessing

The data preparation phase when it comes to creating an effective machine learning- based network intrusion detection system (NIDS) is one of the most significant ones. Prior to putting it into a machine learning model, it cleans, transforms, and properly structures raw network data. Data Cleaning Goal is Remove or correct erroneous, missing, duplicate, or irrelevant data entries using pre-processing Techniques like Remove rows with missing values or fill them using imputation techniques (mean, median, etc.). Filter out outliers or noise (e.g., corrupted packets or malformed headers). Remove redundant features or non- informative attributes (e.g., packet ID, timestamps if not needed). After data pre-processing stage, Divide the preprocessed data into: Training Set: About seventy to eighty percent of the time, yeah. Set for Test: 20-30% to measure how well the model works. Validation Set (optional): For hyper parameter tuning.

3.5.2 Model Architecture

This was a project that employed a diverse series of supervised machine learning techniques, such as Support Vector Machines (SVMs), Random Forests (RFs), Decision Trees, and Logistic Regressions to measure the performance of NIDS on dataset NSL-KDD. Classification in machine learning typically consists of the following five stages: 1. Gathering relevant data: Information unique to the study is gathered and saved in the memory. Step two involves data preparation, and it involves putting the obtained data in an organized manner that the machine learning algorithm will understand it. With the feature extraction, the information of use is extracted out of the data and the irrelevant information is washed away. Third, during training, the data obtained after preprocessing is fed into the machine learning system. The training phase is finished and a model is formed at this step. Algorithms like Decision Tree, Logistic Regression, Support Vector Machines (SVM), and Random Forest (RF) are under consideration. 4. Testing: A model is generated by transferring the dataset from the training phase to the chosen machine learning method. This step may be repeated once or more than once. Step five,

deployment, involves determining the most appropriate model for classification or prediction in light of the current situation. Extensive model comparisons are conducted before the selection is decided.

3.5.3 Training Process

In order to train ML models for an NIDS, there are a number of critical processes, beginning with data preparation and ending with assessment. Below is a structured breakdown of the training process: Use standard NIDS datasets like NSL-KDD (improved version of KDD'99). These datasets contain features extracted from network traffic along with labels: normal or intrusion type. Data cleaning process is used to handle missing values or corrupted records. Label Encoding is used to convert attack types to binary (e.g., Normal = 0, Attack = 1) or multi-class. Feature Selection/Extraction is used to remove redundant or non-informative features by using PCA or feature importance ranking (e.g., mutual information, chi-square) techniques. Use MinMaxScaler or StandardScaler for ML models sensitive to feature scale. Train using your chosen algorithm after Split dataset: 80% training / 20% test. Optionally use cross-validation (k-fold, e.g., k=5). Use the training data to fit the model. The next step is to determine an accuracy score by using suitable metrics like as Accuracy, Precision, Recall, F1- score, Confusion Matrix, and ROC-AUC (for binary classification) to evaluate the effectiveness of each method as a whole.

Input Design

The user interacts with network intrusion detection systems via the input design. In order to guarantee that the data used for processing is in a useable format, it is necessary to establish standards and processes for data preparation. This can be achieved through direct user input or automated data extraction from existing sources. The focus of input design is to minimize input errors, reduce complexity, and maintain security while ensuring user- friendliness. Key aspects of the input design include:

Here are a few sample records from the NSL KDD Dataset that have request signatures; the author has utilized these to perform experiments. The 'dataset' folder has the same dataset that I have used. Whether it's a typical request signature or an attack signature, the last value carries the corresponding class label. The signature values are stored in the first two records. "Neptune" is an aggressive name in the second record. In a similar vein, the collection contains approximately 30 distinct attack names. The PREPROCESSING Concept will be used to eliminate values in string format from the records in the dataset that are not necessary for prediction. These values include tcp and ftp data. Since the algorithm would struggle to recognize all attack names in string format, we must instead assign a numerical value to each attack. Following the completion of the aforementioned PREPROCESS stages, a new file named "clean.txt" will be created for the purpose of generating the training model. I am giving each assault a numerical ID in the line below. "intrusion":1, "normal": 0. The lines above show that for all assaults, normal has the identifier 0 and anomaly has the identifier 1. Please perform the two instructions below before executing the code.

Screen shots

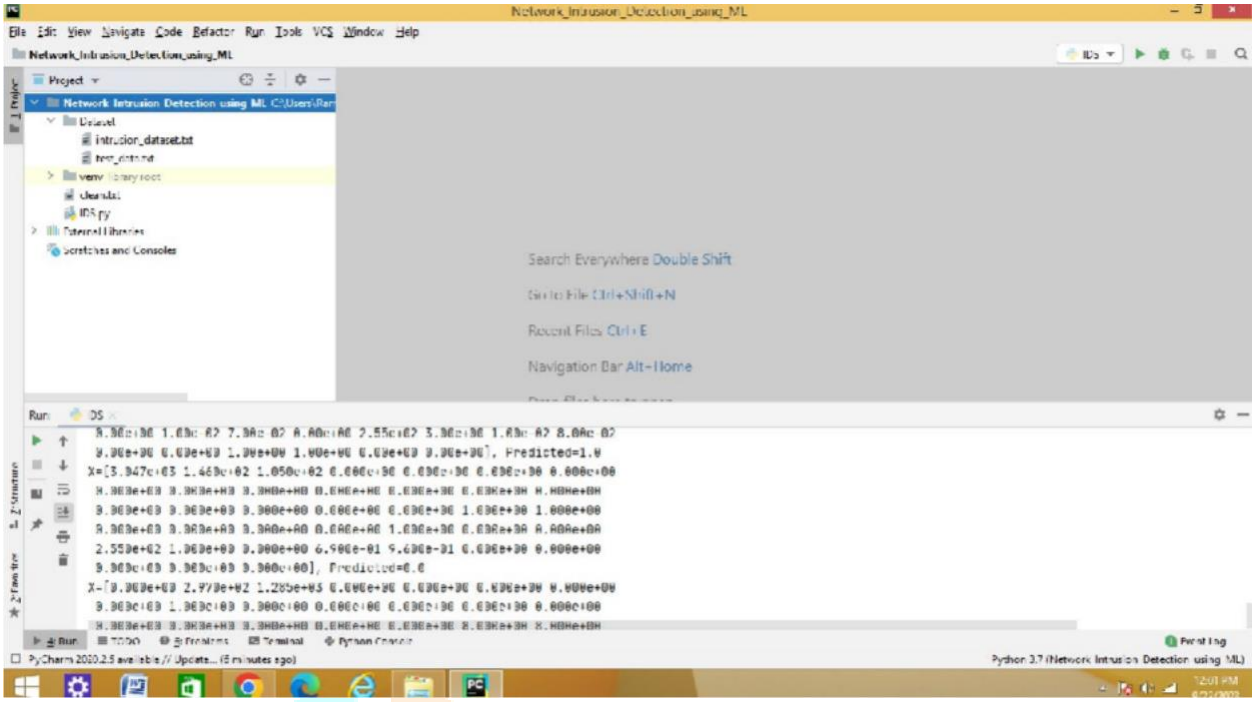

Fig 3.8 Run the Application in PyCharm Community Edition

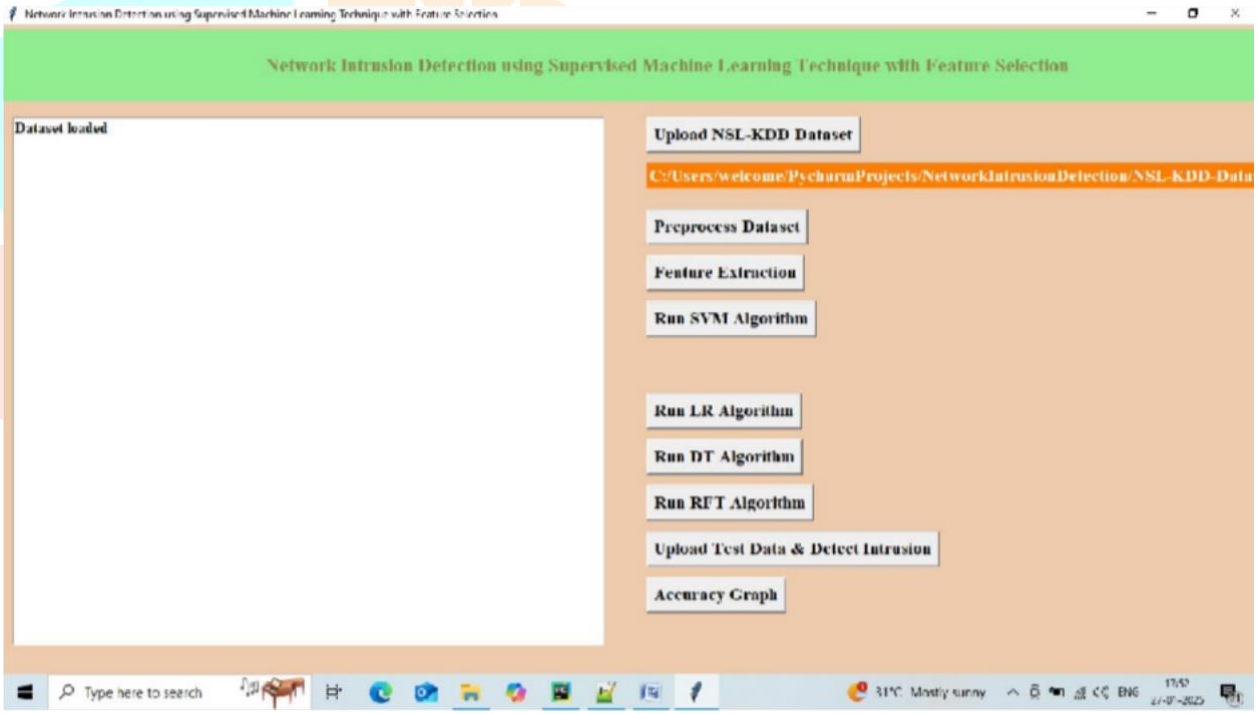

Fig 3.9 Upload NSL-KDD Dataset

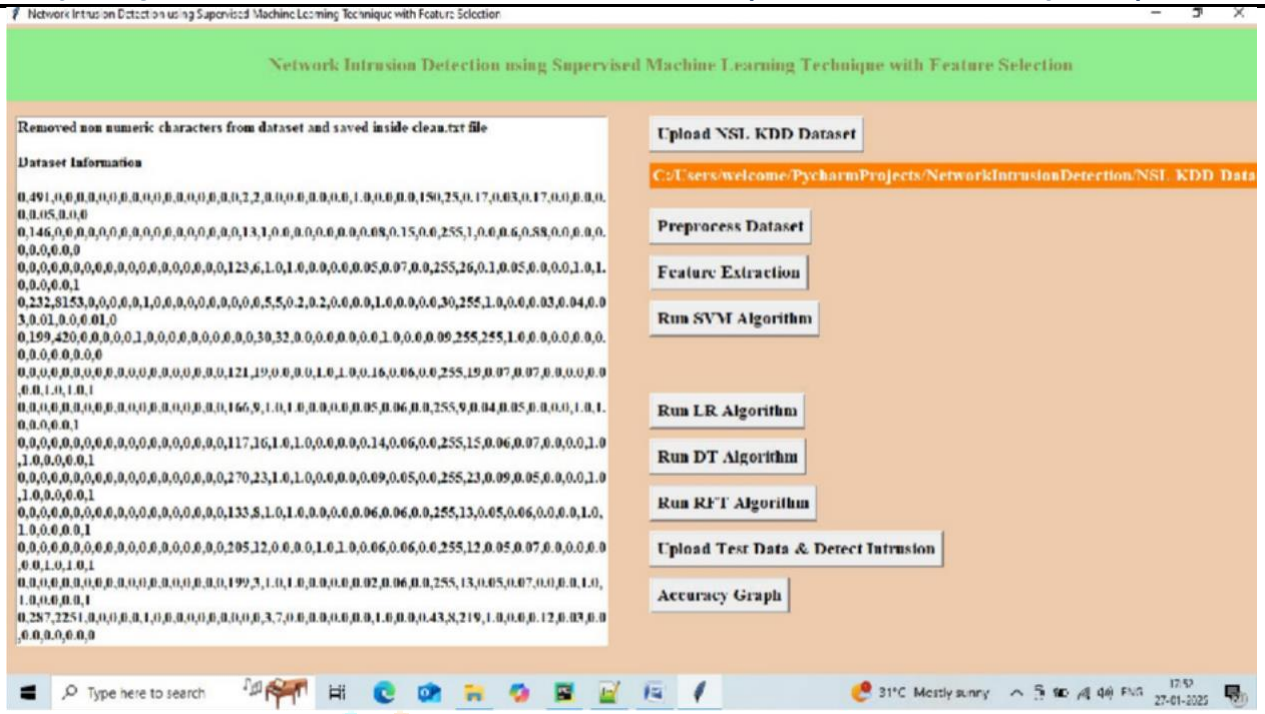

Fig 3.10 Dataset Pre-processing

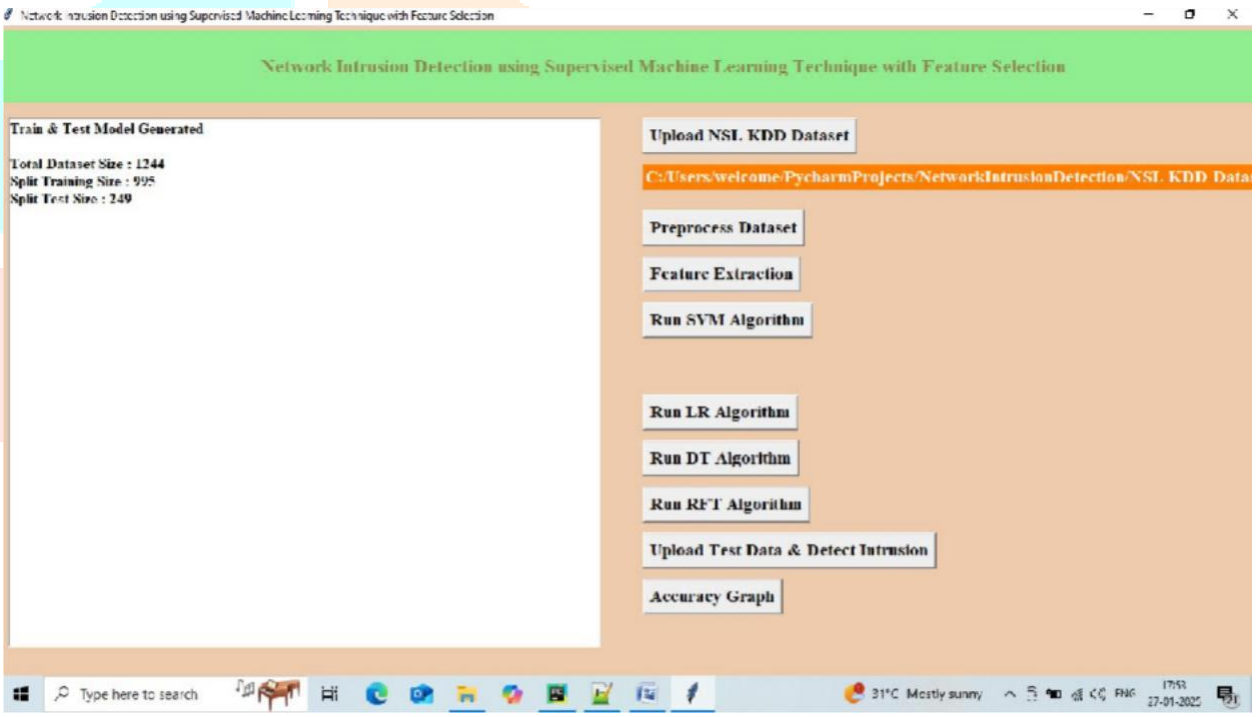

Fig 3.11 Feature Extraction/Classification

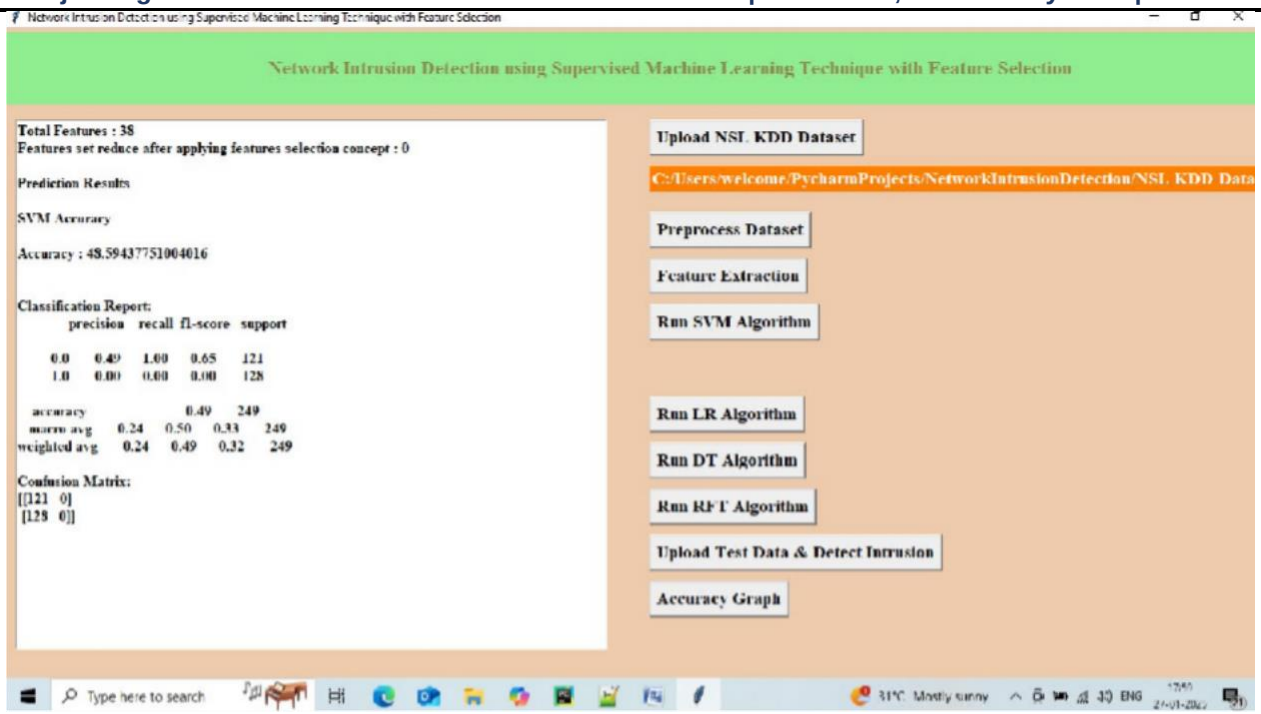

Fig 3.12 Building the SVM Model and calculating performance matrices

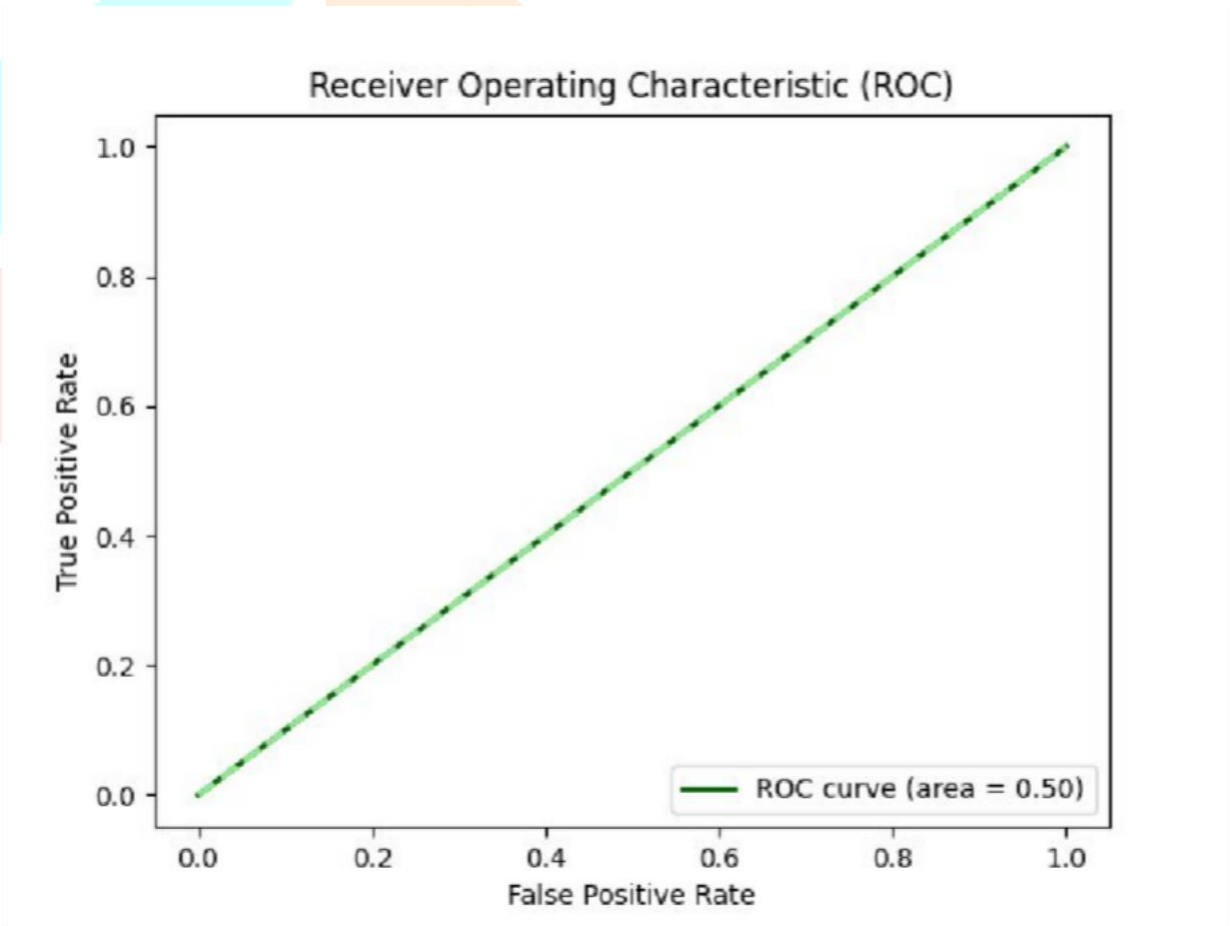

Fig 3.13 ROC Curve Graph for Support Vector Machine Model

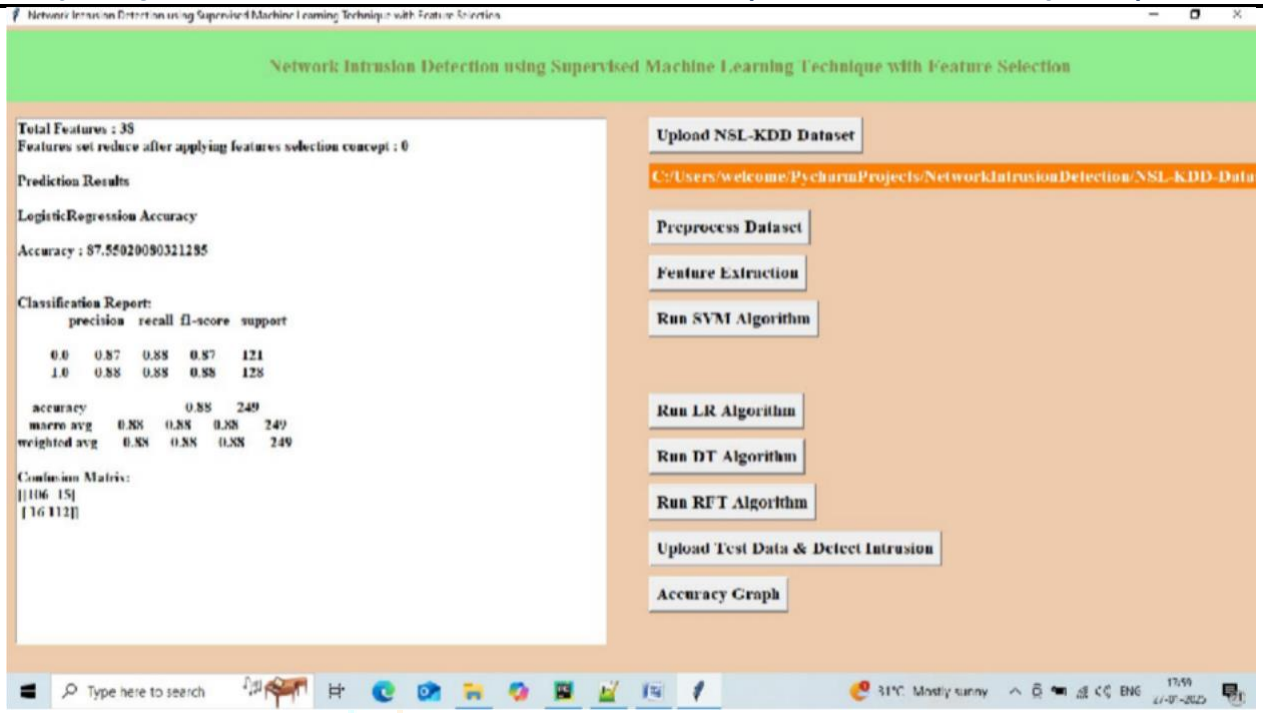

Fig 3.14 Building the LR Model and calculating performance matrices

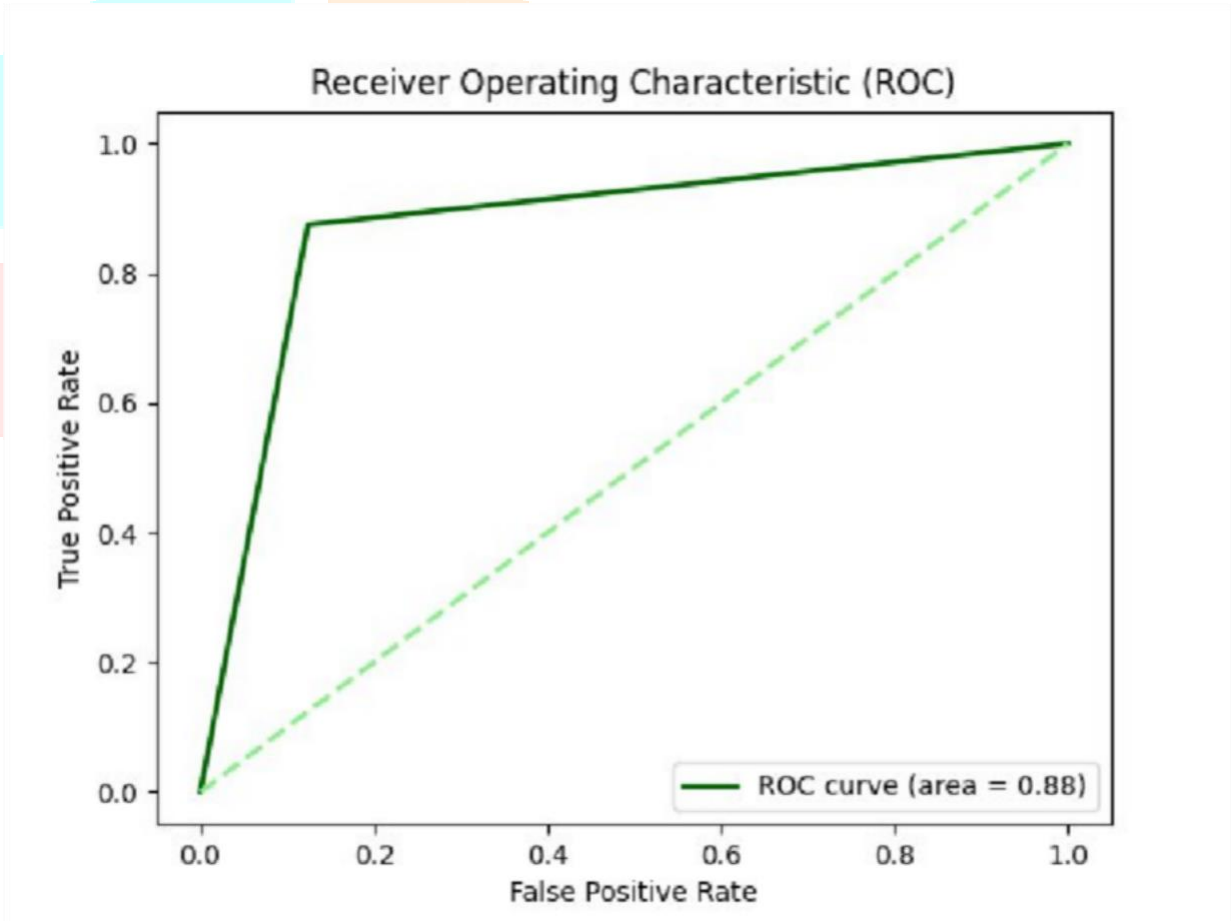

Fig 3.15 ROC Curve Graph for Logistic Regression Model

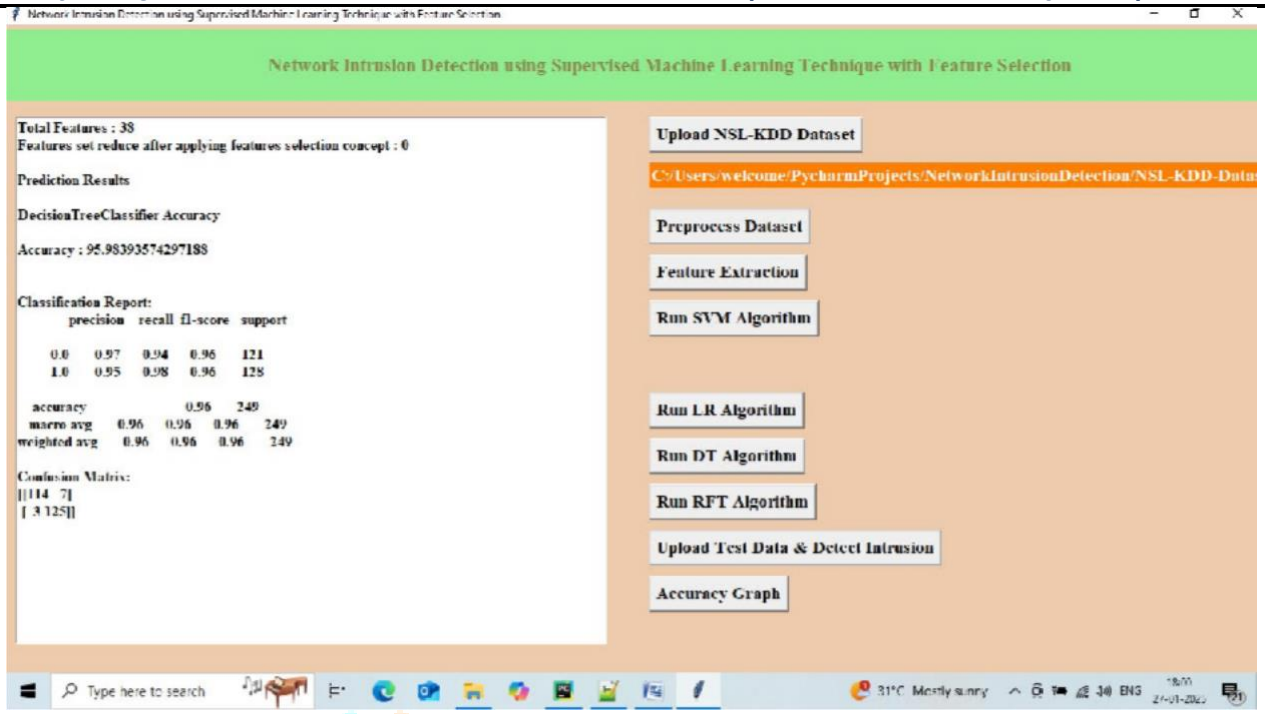

Fig 3.16 Building the DT Model and calculating performance matrices

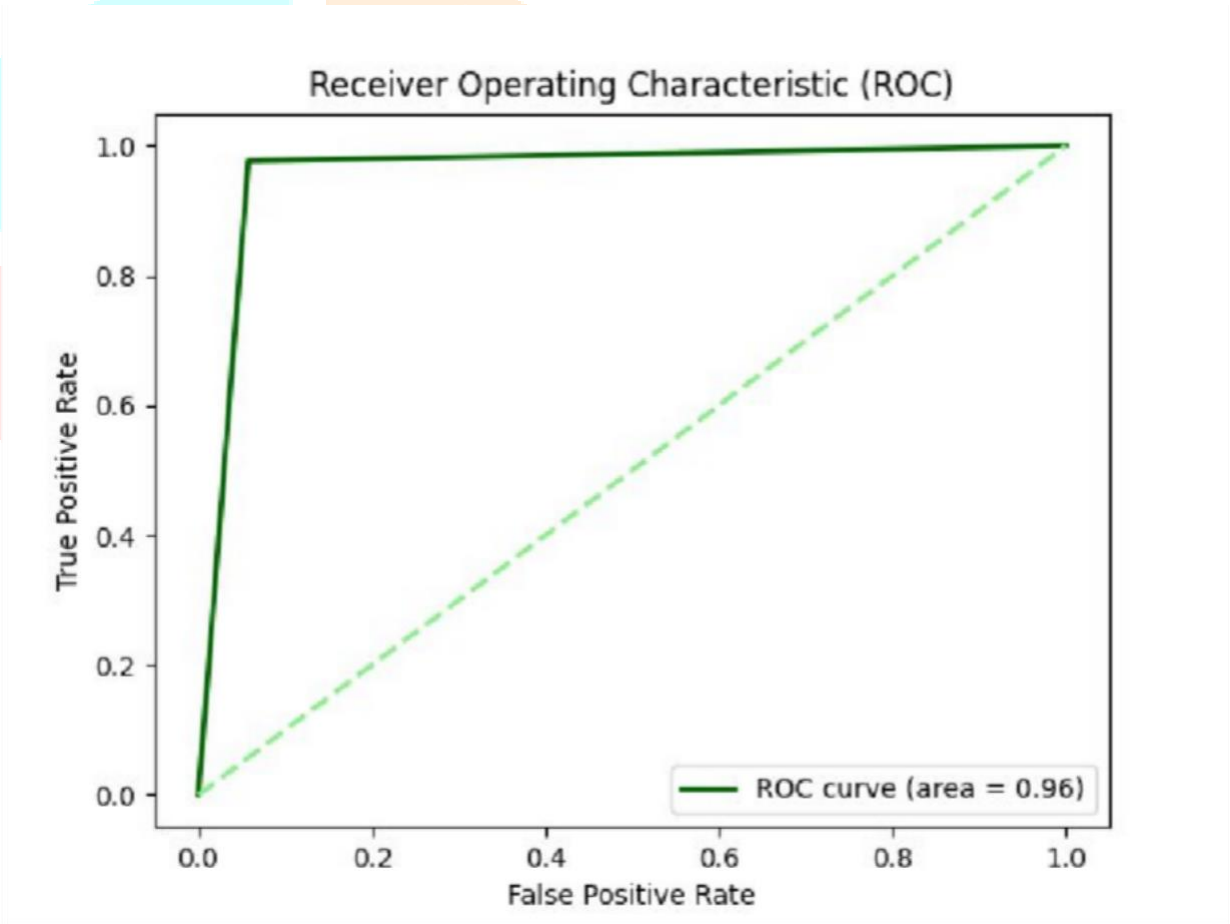

Fig 3.17 ROC Curve Graph for Decision Tree Model

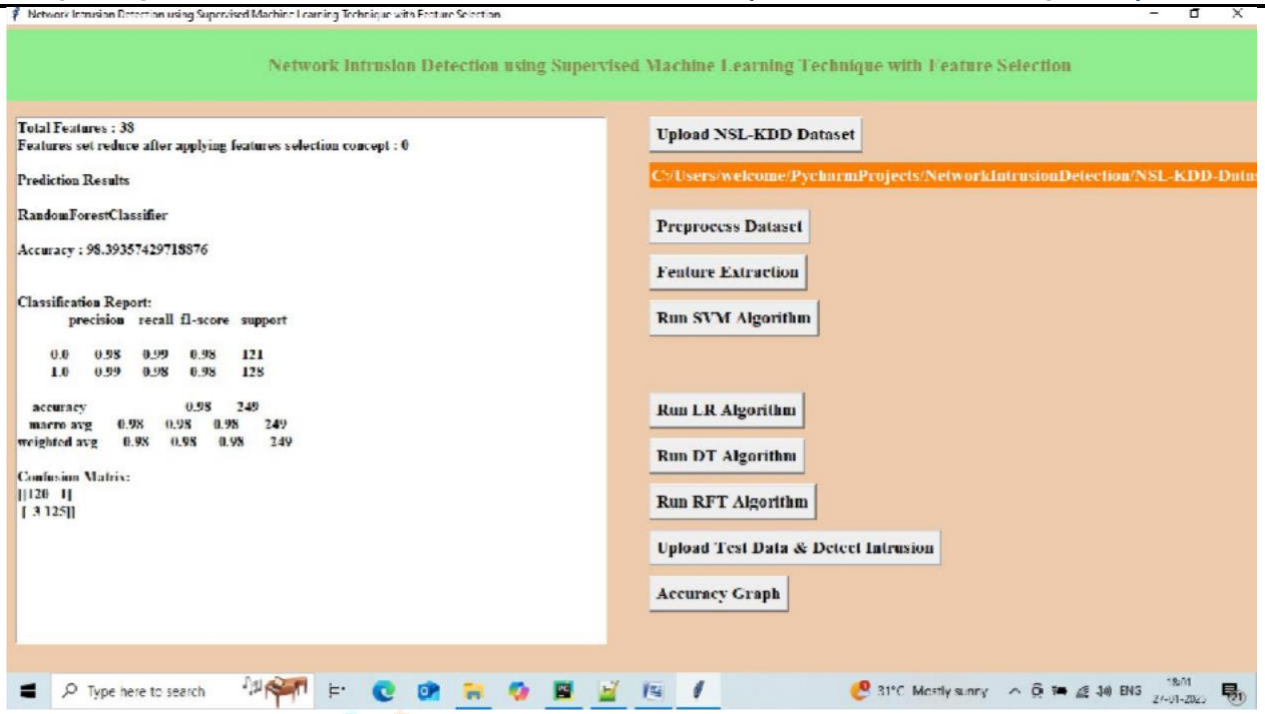

Fig 3.18 Building the RF Model and calculating performance matrices

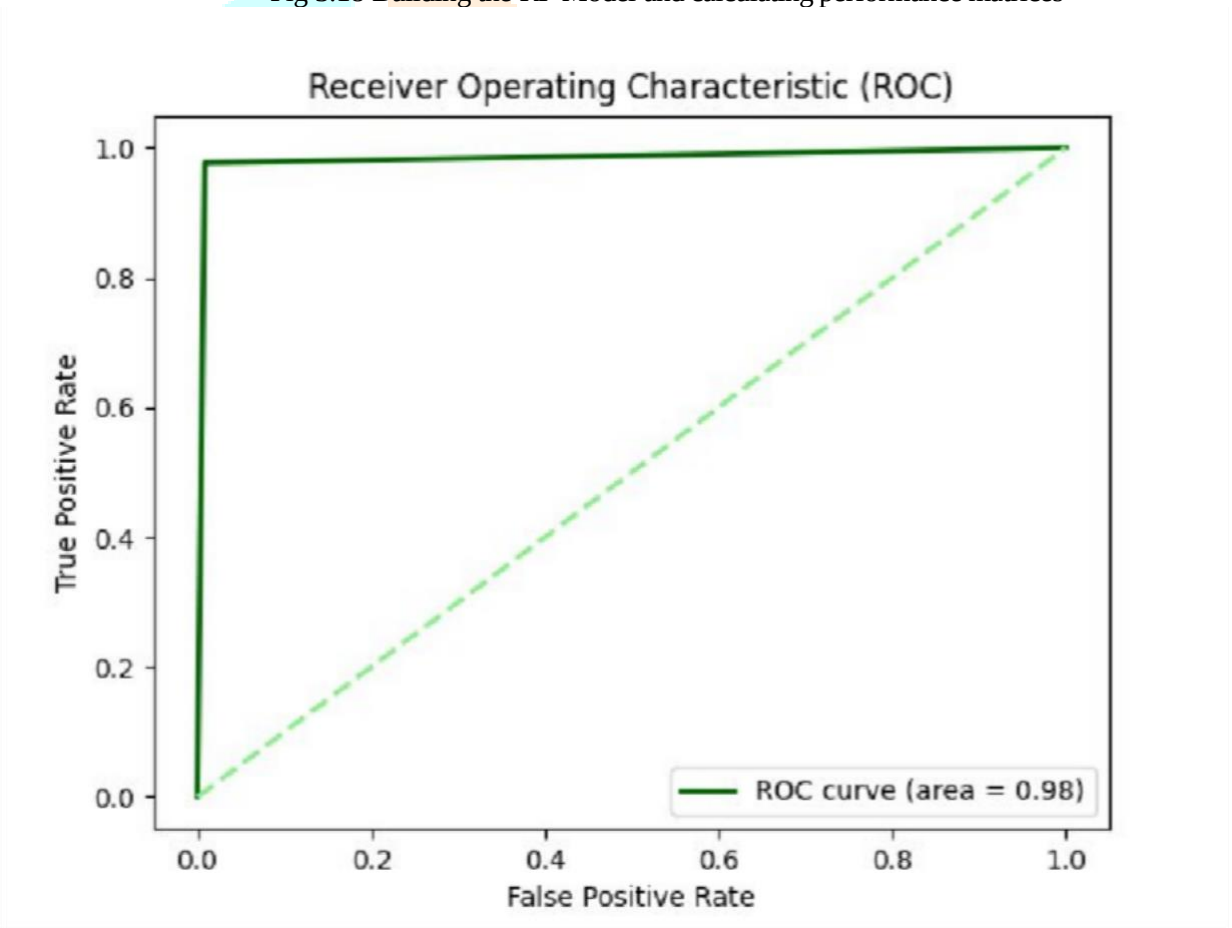

Fig 3.19 ROC Curve Graph for Random Forest Model

Output Design

When the result is both clear and fits the needs of the end user, we may say that it is of high quality. The outputs of a system are the means via which its users and other systems get the results of its processing. Output design determines how data will be displaced to be immediately used as well as being used as hard copy of output. The user uses it as the main and most important information source. The system (the fact that it can help users in their decision- making process) is supplemented by intelligent and effective output design.

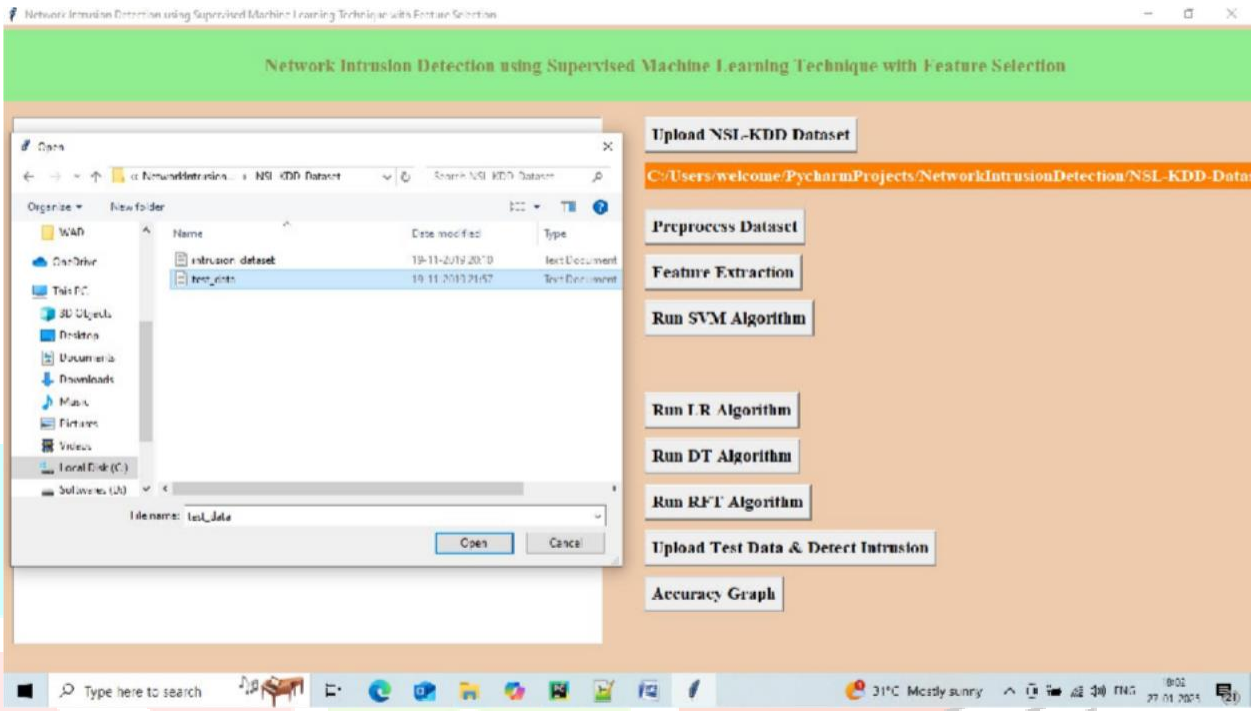

Fig 3.20 Upload Test Dataset to detect the network intrusions

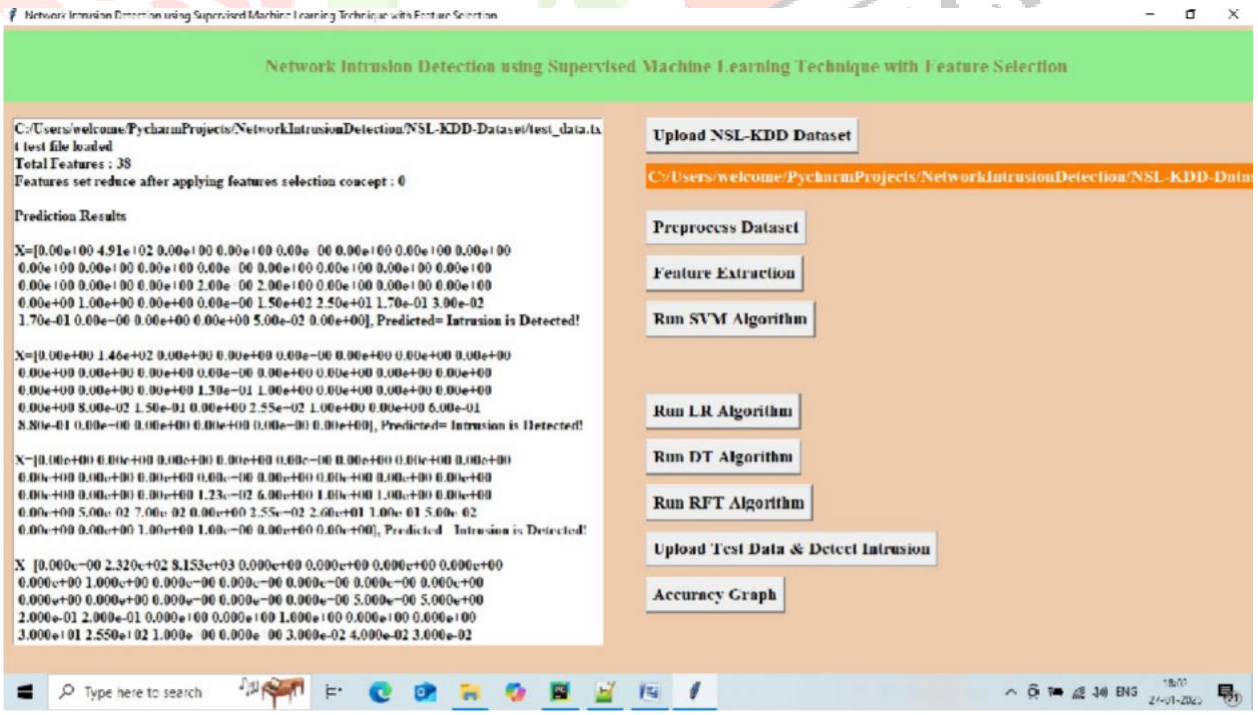

Fig 3.21 View Network Intrusion Prediction Results

3.5.4 Evaluation Metrics and Analysis

The model evaluation is carried out on a test dataset. The performance of its classification skills can be properly examined through the prism of recall, accuracy, precision, and F1-score metrics. For detailed insights, a **confusion matrix** is visualized, revealing class- specific performance and highlighting instances of class confusion.

Results from the NSL-KDD dataset show that machine learning architectures outperform standard recurrent and deep neural networks. In complicated picture classification settings, the search itectures' capacity to match the loss function's gradient or the pseudo-residuals of prior models is useful. The iterative addition of dense layers, coupled with frozen weights from previous iterations, reduces over fitting and fine-tunes the model effectively.

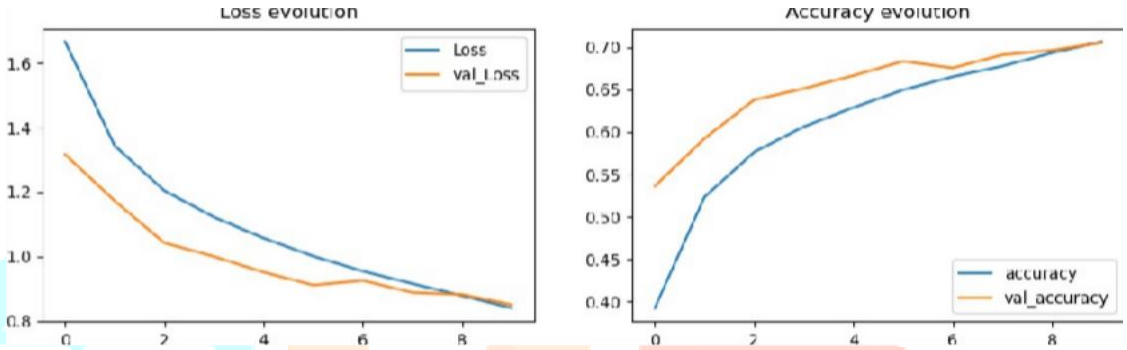

Fig 3.22 Output Evaluation Metrics

By combining thought full input and output design, machine learning model systems can be optimized for seamless operation, improved performance, and enhanced user satisfaction. This approach ensures that the models are effective in handling complex image classification and other challenging tasks.

3.5.5 Comparative Analysis of SVM, LR, DT, RF

Despite promising results, challenges emerged during training and evaluation. Over fitting, even with drop out regularization, remained a concern, indicating the potential need for additional techniques like weight regularization or early stopping. The NSL-KDD dataset used was small, which is a source of limitation to the full generalization of the model to unseen data. Class-specific performance analysis revealed that certain classes achieved higher accuracy than others. This discrepancy high lights the difficulty of distinguishing visually similar categories, suggesting the need for further refinement in feature extraction or augmentation strategies tailored to specific classes.

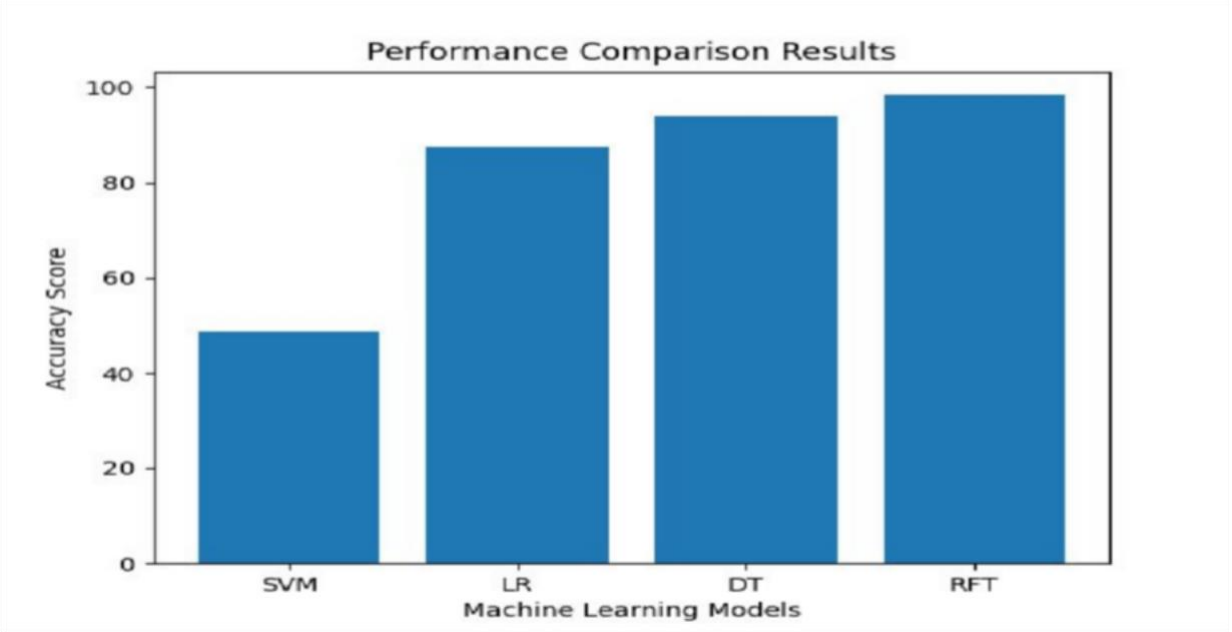

Fig 3.23 Performance Comparison Results

3.5.6 Future Directions

Where Might ML-Reliant NIDSs Go From Here? System updates for Network Intrusion Detection Systems (NIDS) are necessary due to the ever-changing nature of cyber security threats.. Here are some promising future directions in the field of ML-based NIDS:

1. **Temporal and Spatial Learning:** Utilize LSTM, GRU, and Transformer-based architectures to capture temporal patterns in network flows (especially useful for detecting slow or persistent attacks).
2. **Graph Neural Networks (GNN):** Model network behavior as a graph and use GNNs to detect anomalies in node (host) or edge (connection) activity.
3. **Federated Learning:** Enable collaborative learning across multiple organizations without sharing raw data—ideal for privacy-preserving intrusion detection.
4. Apply knowledge gained from one domain (e.g., enterprise networks) to another (e.g., IoT networks), improving detection with limited new data.
5. Combine signature-based methods with anomaly-based ML models to detect both known and unknown attacks.

Chapter 4

CONCLUSION AND FUTURE SCOPE

Conclusion

This piece presented a network intrusion detection system that involves the machine learning theory. Determining the quality of the proposed detection system working on the NSLKDD dataset with the help of various machine learning techniques. In predictions of the malicious packets, the results indicated that the decision tree and Random Forest algorithms were better than the other performed models. This was particularly true when looking at accuracy, recall, and the Mathews correlation coefficient. In addition, cutting-edge intrusion detection systems were beaten by the RF classifier. Despite its flaws, the NSL-KDD dataset does not represent actual assaults as it has uneven classes and the malicious traffic it records is artificial. The classifiers have shown promise and can identify intrusions into networks.

4.1 Future Research Directions

Improving features is an essential part of creating effective NIDSs that rely on machine learning. In order to increase the accuracy of detection of known and unknown incursion with the use of the model, the quality and representation of the utilized information should be improved. To improve our system's accuracy, we want to expand the amount of test data in the future. We are also looking on ways to enhance the accuracy of IDS by combining the RST technique with genetic algorithms. The current setup only shows the log data without doing any analysis or knowledge extraction from the log records. Data mining tools may be integrated into the system to sift through log data for useful information that might improve decision-making. At this time, the system can only identify certain types of assaults. Intelligence may be added to this so it can learn new intrusion patterns and analyze increasing traffic on its own.

REFERENCES

[1]. Cisco Annual Internet Report (2018–2023) White Paper. (2022, January 23). Cisco. https://www.cisco.com/c/en/us/solutions/collateral/executiveperspectives/annual-internet- report/white-paperc11–741490.html

[2. Dyn Analysis Summary of Friday October 21 Attack (2022, February 20). https://web.archive.org/web/20200620203923

[3]. Dartigue, C., Jang, H.I., Zeng, W. A new data-mining based approach for network intrusion detection. In Seventh Annual Communication Networks and Services Research Conference. 2009; 372–377.

[4]. García-Teodoro, P., Díaz-Verdejo, J., MaciáFernández, G., Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security. 2009; 28(1–2); 18–28.

[5]. Cisco. What Is Network Security? (2022, February,8). Cisco. https://www.cisco.com/c/en/us/products/security/what-is-network-security.html

[6]. Kurose, J.F., Ross, K.W. Computer Networking: A Top-Down Approach (6th Edition). Pearson,

2012.

[7]. Tanenbaum, A., Wetherall, D. Computer Networks (5th Edition). Pearson, 2010.

[8]. Fernandes, G., Rodrigues, J.J.P.C., Carvalho, L.F., Al-Muhtadi, J.F., Proença, M.L. A comprehensive survey on network anomaly detection. Telecommunication Systems. 2018; 70(3): 447–489.

[9]. Othman, S.M. Alsohybe, N.T., Ba-Alwi, F.M., Zahary, A.T. Survey on intrusion detection system types. 2018; 7(4): 444–463.

[10]. Pal Singh, A., Deep Singh, M. Analysis of HostBased and Network-Based Intrusion Detection System. International Journal of Computer Network and Information Security, 2014; 6(8): 41–47.

[11]. Ferrag, M.A. Maglaras, L. Moschoyiannis, S., Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications. 2020; 50.

[12]. Boutaba, R. Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., Caicedo, O.M. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications. 2018; 9(1).

[13]. Buczak, A.L., Guven, E.A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials. 2016; 18(2): 1153–1176.

[14]. Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., Faruki, P. Network Intrusion Detection for IoT Security Based on Learning Techniques. IEEE Communications Surveys & Tutorials. 2019; 21(3): 2671–2701.

[15]. Berman, D., Buczak, A., Chavis, J., Corbett, C. A Survey of Deep Learning Methods for Cyber Security. Information. 2019; 10(4): 122.

[16]. Mahdavifar, S., Ghorbani, A.A. Application of deep learning to cybersecurity: A survey. Neurocomputing. 2019; 347: 149–176.

[17]. Ahmed, M., Naser Mahmood, A., Hu, J. A survey of network anomaly detection techniques. Journal of Network and Computer Applications. 2016; 60: 19–31.

[18]. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A. A survey of network- based intrusion detection data sets. Computers & Security. 2019; 86: 147–167.

[19]. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K. Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials. 2014; 16(1): 303–336.

[20]. Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C. Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access. 2018; 6: 35365–35381.

[21]. UNB (2021, Novamber 15). https://www.unb.ca/ cic/datasets/nsl.html

[22]. Chumachenko, K. Machine learning methods for malware detection and classification., 2017.

[23]. Zou, J., Han, Y., So, S.S. Overview of artificial neural networks. Methods in molecular biology (Clifton, N.J.). 2008; 458: 15–23.

[24]. Dong, B., Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In 2016 8th IEEE International Conference on Communication Software and

Networks (ICCSN). 2016; 581–585.

[25]. Mahesh, B. Machine Learning Algorithms – A Review. International Journal of Science and Research (IJSR). 2020; 381–386.

[26]. Farnaaz, N., Jabbar, M.A. Random forest modeling for network intrusion detection system. Procedia Computer Science. 2016; 89: 213–217.

[27]. Bhumgara, A., Pitale, A. Detection of Network Intrusions using Hybrid Intelligent Systems. 1st International Conference on Advances in Information Technology (ICAIT). 2019; 500–506.

[28]. Kumar, K., Batth, J.S. Network Intrusion Detection with Feature Selection Techniques using Machine-Learning Algorithms. International Journal of Computer Applications. 2016; 150(12): 1–13.

[29]. Dhanabal, L., Shantharajah, S.P. A study on NSLKDD dataset for intrusion detection system based on classification algorithms. International journal of advanced research in computer and communication engineering. 2015; 4(6): 446–452.