JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

NOIDA, SECTOR 62

MINOR PROJECT - 2

Performance Comparison of Traditional Machine Learning and Ensemble Models in Network Intrusion Detection

B.TECH SEMESTER – VI

SUBMITTED BY : -

NAME	ENROLLMENT
RAVI RAUSHAN	22103183
VIYOM SHUKLA	22803030
ABHIJEET KUMAR	22803029

SUPERVISION OF : - Dr. MEENAL JAIN (DEPT. OF COMPUTER SCIENCE & ENGINEERING)

SUBMITTED TO : -

NAME	DEPARTMENT
Dr. AMANPREET KAUR	DEPARTMENT OF CSE
Dr. ALKA SINGHAL	DEPARTMENT OF CSE

Summary Report for Minor Project

1. Motivation Behind the Project

The motivation behind this project is to address the pressing challenges in network intrusion detection by systematically comparing the effectiveness of traditional machine learning models and ensemble learning techniques. By partitioning the dataset into four distinct binary classification problems-Normal vs U2R, Normal vs R2L, Normal vs Probe, and Normal vs DoS-the project aims to develop specialized detection strategies tailored to the unique characteristics of each attack type. Additionally, the use of Conditional GANs (CGANs) for synthetic data generation is motivated by the need to overcome class imbalance, which is a common issue in intrusion detection datasets and can significantly affect model performance. The application of data preprocessing steps and feature selection through Recursive Feature Elimination (RFE) is intended to enhance model accuracy and efficiency by ensuring that only the most relevant features are used. Finally, by evaluating and comparing the performance of both traditional and ensemble models using standard metrics, the project seeks to identify the most effective and robust approach for detecting various types of network intrusions, ultimately contributing to the development of more secure and reliable network systems.

2. Type of Project

The project falls under the category of:
b. Development cum research project

Explanation:

Development Aspect

1. The project involves creating a system that can detect suspicious activities on computer networks
2. It follows a step-by-step approach:
    - Planning: Figuring out what the system needs to do
    - Analysis: Understanding how to solve the problem
    - Design: Creating blueprints for the solution
    - Implementation: Building the actual system
    - Testing: Making sure it works correctly
    - Evaluation: Checking if it meets standards
    - Deployment: Installing it and training users
3. The project uses free, open-source tools
4. These tools are adapted to fit specific security needs
5. Thorough testing ensures the system is reliable and easy to use

Research Aspect

1. The project compares different methods for detecting network intrusions
2. It reviews existing security systems to learn from them
3. It explores various computer learning techniques:
    - Basic methods like Decision Trees and SVM
    - More advanced methods like Random Forest
    - Combined approaches called "ensemble techniques"
4. The research uses standard datasets to test how well each method works
5. It tackles common challenges like:
    - Unbalanced data (having too few examples of certain attacks)
    - Choosing the most important data features
    - Finding which methods work best for different types of attacks
6. The goal is to improve network security technology

2. i) New Technologies, Tools, and Software Learned:

Programming Languages Used

- Python was the main language for the entire project
- Used for data processing, building detection models, and creating visualizations

Tools and Libraries Used

- NumPy: Handled numerical calculations and array operations
- Pandas: Managed data preprocessing and manipulation tasks
- Matplotlib/Seaborn: Created graphs and visualizations to analyze the data
- Scikit-learn: Implemented various machine learning models for detecting intrusions
- Google Colab/VSCode: Provided development environments (cloud and local)

Key Learnings

- Gained practical experience with the complete machine learning workflow:
    - Data cleaning and preparation
    - Feature engineering and selection
    - Model training and optimization
    - Performance evaluation
- Developed data visualization skills to identify patterns in network traffic
- Learned how to properly evaluate security models using metrics

- Improved Python coding skills, particularly for data science applications
- Became proficient with open-source libraries that streamline security development

3. ii) Critical Analysis of Research Papers and Gaps:

1. **Old and Limited Datasets:**
   Even though researchers now use better datasets like CIC-IDS2017 and UNSW-NB15 instead of the outdated KDDCup 99, there's still a big problem: we don't have new, regularly updated datasets that reflect the latest cyber threats.
2. **Limited Types of Models Used:**
   Research has improved from basic machine learning to advanced methods like deep learning. But most studies still depend on supervised learning, which needs labeled data — and getting those labels takes a lot of time and effort.
3. **Accuracy Isn't Enough:**
   Many studies only talk about accuracy, which can be misleading when the data is imbalanced (e.g., too many normal cases, not enough attack cases). Some papers don't even include all the important evaluation metrics like precision, recall, or F1-score.
4. **Can the Models Handle New Attacks?**
   Only one study (by Sohail et al., 2023) talks about testing the model's ability to catch new types of attacks it hasn't seen before. This is a major issue — most research doesn't check if the models can detect unknown or novel threats.
5. **Not Much Innovation in Model Design:**
   While researchers use ensemble methods (like combining multiple models), very few explore creative or new approaches like generative models in cybersecurity.
6. **Big Missing Piece — Use of CGANs:**
   No research has yet used Conditional Generative Adversarial Networks (CGANs), which could:
   - Create fake (synthetic) but realistic network traffic to help train models.
   - Solve the problem of class imbalance (too few attack samples).
   - Generate rare or new attack examples.
   - Make models stronger against trick attacks (adversarial threats).
7. **Not Using Ideas from Other Fields:**
   Some work, like Peihong Wu's breast cancer study using ensemble methods, shows how techniques from one field could help in others. But cybersecurity researchers haven't fully taken advantage of such cross-domain knowledge.
8. **Real-Time Detection Is Missing:**
   Most research doesn't focus on how fast or efficient the models are. In the real world, systems need to detect threats instantly — especially in networks with huge amounts of data.

There's a big opportunity to improve intrusion detection systems using CGANs. These models can help create realistic attack data, solve data imbalance issues, and make detection systems smarter and stronger against new cyber threats.

3. Design of the Project

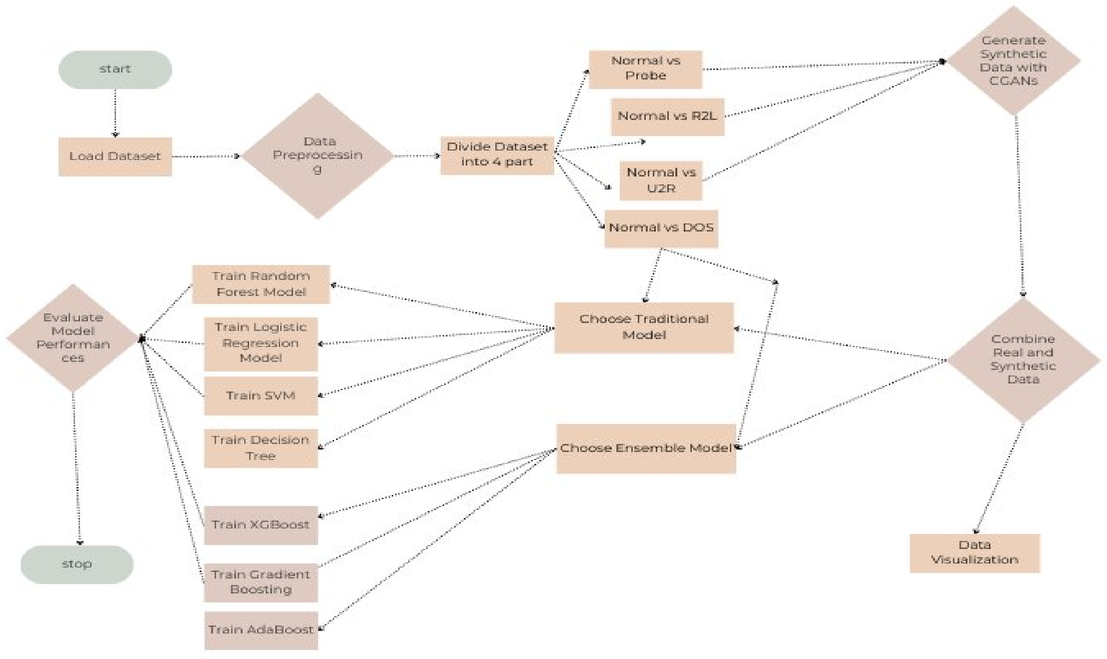

Figure: FlowChart

5. Features Built and Programming Language Used

Features Built & Languages Used

Programming Languages

- **Python**: Primary language used throughout the project implementation

Features Built

1. Data Processing Module

- Built robust data preprocessing pipeline for network traffic data
- Implemented feature extraction from raw packet captures
- Created data normalization and scaling functions for model compatibility
- Developed methods to handle missing values and outliers in network data

2. Model Implementation

- Implemented multiple machine learning algorithms:
    - Traditional models (Random Forest, SVM, Decision Trees)
    - Ensemble methods (XGBoost, Stacking, Bagging)

  Developed a CGAN (Conditional Generative Adversarial Network) module to:
    - Generate synthetic network attack data
    - Address class imbalance issues
    - Create realistic examples of rare attack patterns

3. Evaluation Framework

- Built comprehensive model evaluation system with multiple metrics
- Implemented cross-validation procedures for reliable performance assessment
- Created comparison tools for benchmarking different detection approaches
- Developed specific tests for generalization to novel attacks

4. Documentation System

- Built comprehensive documentation using Jupyter notebooks
- Implemented automated report generation for detected threats
- Created training materials for system users

6. Proposed Methodology

1. **Dataset Handling**
    - Used NSL-KDD dataset with 113,270 network traffic records
    - Addressed class imbalance problem (DoS classes had thousands of samples, while Probe,R2L and U2R had very few)
2. **Data Preprocessing**
    - Cleaned missing values using mean replacement
    - Converted categorical features to numerical using one-hot encoding
    - Split data into four binary classification tasks (Normal vs. DoS, Normal vs. Probe, Normal vs. R2L, Normal vs. U2R)
3. **Feature Selection**
    - Used Recursive Feature Elimination (RFE) to select the most important features
4. **Data Augmentation**
    - Implemented Conditional Generative Adversarial Network (CGAN)

- Generated synthetic samples for minority attack classes
- Balanced the dataset to improve detection of rare attacks

5. **Model Building**
    - Implemented traditional machine learning models:
        - Random Forest
        - Logistic Regression
        - Support Vector Machine (SVM)
        - Decision Tree
    - Implemented ensemble learning models:
        - Adaptive Boosting (AdaBoost)
        - Gradient Boosting Machine (GBM)
        - Extreme Gradient Boosting (XGBoost)
6. **Performance Evaluation**
    - Measured model performance using:
        - Accuracy
        - Precision
        - Recall
        - F1-Score
7. **Final Stacking Model**
    - Combined the strengths of all models (traditional and ensemble)
    - Created a more robust and accurate detection system

7. Algorithm of the Workflow

Input Output	NSL-KDD Dataset D Final Classifier C
Step1	Preprocess the dataset
Step 2	Split data into four binary classification tasks
Step 3	Balance the dataset using Conditional GAN (CGAN) 1.Initialize the parameters of Generator G and Discriminator D 2.Repeat for number of training iterations: a. Sample a minibatch of noise vectors z~p_z(z) b. Sample a minibatch of class labels y c. Generate fake samples $\tilde{x}$=G(z|y) d. Sample a minibatch of real data x~p_{data}(x|y) e. Update Discriminator D by maximizing: $\log D(x\|y) + \log(1 - D(G(z\|y)\|y))$ f. Update Generator G by minimizing:

	$\log(1 - D(G(z\|y)\|y))$ 3. Until the generator produces high-quality samples that can fool the discriminator 4. Output the trained Generator G to generate synthetic samples for minority classes.
Step 4	Perform RFE on Dataset
Step 5	Train Base Classifiers For each task t=1 to T: Train Traditional base classifiers c_t: • Random Forest • Logistic Regression • Decision Tree • SVM Train Ensemble base classifiers c_t: • AdaBoost • XGBoost • GBM

8. Division of Work Amongst Students

The project was a collaborative effort with clear division of responsibilities among the team members:

Abhijeet Kumar: Data Prep

- Split dataset into 4 attack types (U2R, R2L, Probe, DoS).
- Handle missing values, label encode, and one-hot encode.

Viyom Shukla: Modeling

- Generate synthetic data using CGAN and combine with real data.
- Use RFE to select best features.
- Train traditional models (e.g., SVM, Decision Tree).
- Train ensemble models (AdaBoost, GBM, XGBoost).

Ravi Raushan Vishwakarma,: Evaluation & Report

- Evaluate models using accuracy, precision, recall, F1-score.
- Compare traditional vs ensemble results.
- Write report and create final presentation.

9. Results

- For **DOS attacks**, Logistic Regression (0.8833) and AdaBoost (0.8741) had the highest accuracy among their respective groups.
- For **Probe attacks**, AdaBoost (0.8946) and Gradient Boosting (0.8941) outperformed traditional models, with SVM (0.8909) being the best among traditional models.
- For **R2L attacks**, XGBoost (0.7800) slightly outperformed all other models, but overall accuracy was lower for this attack type.

- For **U2R attacks**, all models performed very well, with SVM (0.9954) and Decision Tree/Random Forest/Gradient Boosting (0.9950) achieving the highest scores.
- Overall, ensemble models showed a slight advantage for Probe and R2L attacks, while traditional models performed best for DOS and U2R attacks.

10. Conclusion

In this project, we compared how well traditional machine learning models (like SVM and Logistic Regression) and ensemble models (like AdaBoost and Gradient Boosting) detect network attacks. We first cleaned and prepared the data, then created extra (synthetic) data to balance the classes, and selected the best features. After testing, we found that ensemble models performed slightly better in spotting some attacks like *Probe* and *R2L*. However, traditional models still worked very well for detecting *U2R* and *DOS* attacks. Overall, using a mix of different models can help improve the accuracy and reliability of network security systems.

Future Directions:

- **Real-Time Use**: Test how fast and efficient these models are when used in live networks.
- **Smarter Feature Selection**: Use advanced methods to automatically choose the best features for the model.
- **Testing on Different Datasets**: Make sure the model works well not just on one dataset, but on other real-world network data too.
- **Learning from New Attacks**: Build models that keep learning and adapting as new types of attacks appear.