Chaitanya Bharathi Institute of Technology (A)

Department of Information Technology

Programme: B.E. (IT)

Semester: V

AY: 2025-26

Course: Cyber Security

Course Code: 22CIE55

Faculty: U Sairam

Assignment 2 – Final Report

Intrusion Detection System using Random Forest

Name:K.Manmohan Rathod

RollNo:160123737044

Git Repo link: https://github.com/irathod/Intrusion-Detection-System

1. Introduction

Cyber attacks are increasingly common, and detecting them automatically is critical for cybersecurity. In this assignment, we implement an Intrusion Detection System (IDS) using the NSL-KDD dataset. We first build a baseline Random Forest model, then improve it using SMOTE oversampling and hyperparameter tuning to handle class imbalance and enhance detection performance.

2. Dataset Description

Dataset: NSL-KDD (Improved version of KDD
Cup 1999) Training samples: 125,973
Testing samples: 22,544
Features: 41 features + 1 label
Target: Binary classification
(Normal=0, Attack=1) Preprocessing
steps:

- One-hot encoding of categorical features
- Scaling numeric features
- Binary labeling (normal/attack)

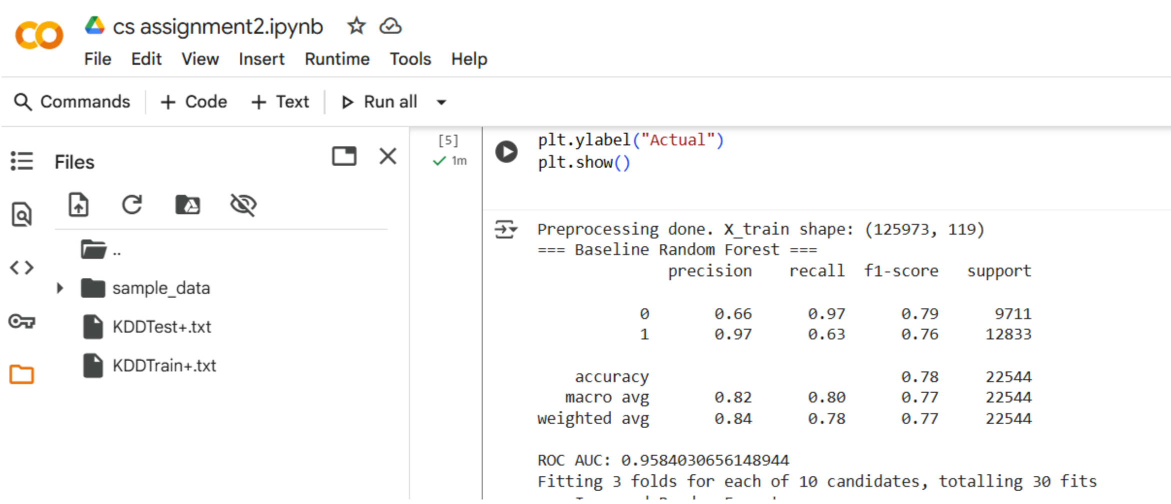
3. Research Gap / Problem Statement

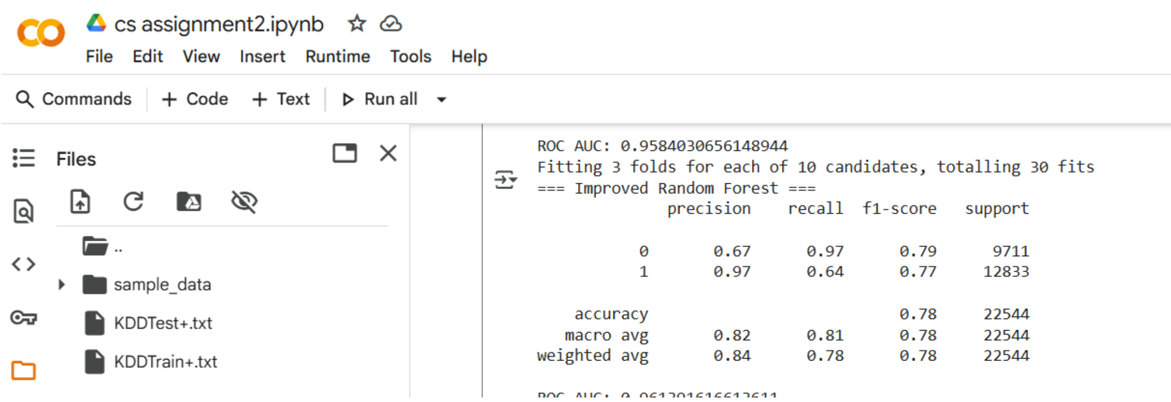
Issue Identified: The dataset is imbalanced (more attack samples than normal). Baseline model may misclassify the minority class due to imbalance. Improvement Needed: Apply SMOTE and hyperparameter tuning to improve F1-score and ROC AUC.

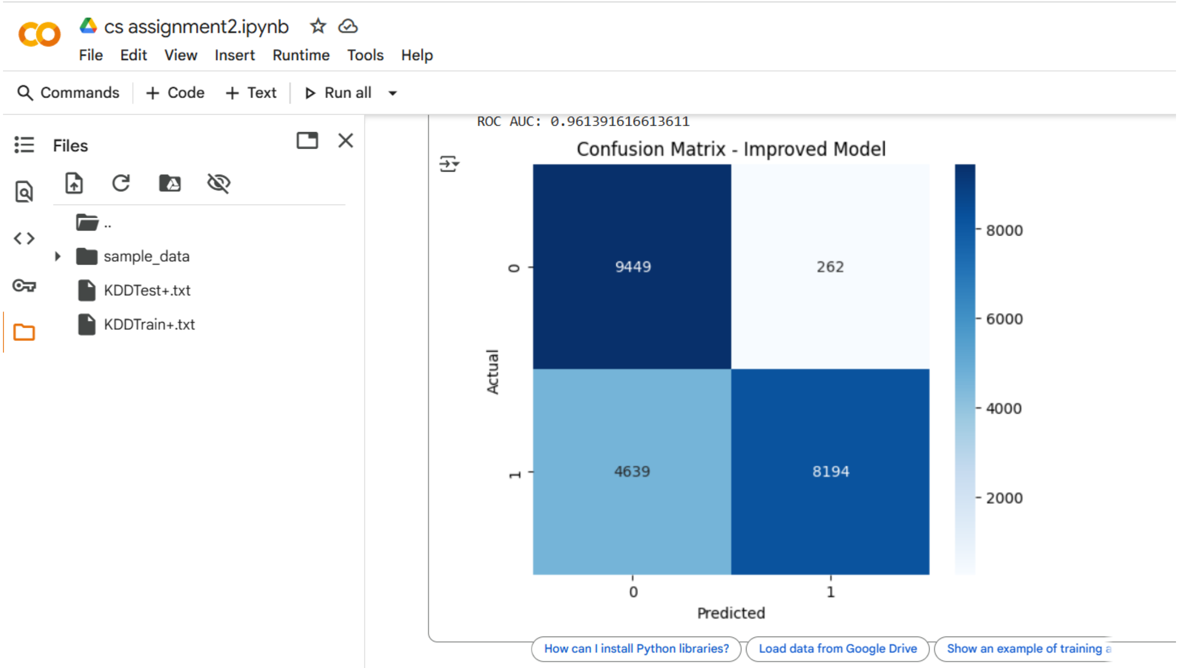
4. Methodology

1. Load Dataset – KDDTrain+ and KDDTest+ files.
2. Preprocessing – Encode categorical features, scale numeric features.
3. Baseline Random Forest – Train and evaluate initial model.
4. Improved Model – Apply SMOTE + RandomizedSearchCV.
5. Evaluation – Compare metrics (Precision, Recall, F1-score, ROC AUC).

5. Results


cs assignment2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
sample_data
KDDTest+.txt
KDDTrain+.txt
[5]
1m
plt.ylabel("Actual")
plt.show()
Preprocessing done. X_train shape: (125973, 119)
=== Baseline Random Forest ===
precision recall f1-score support
0 0.66 0.97 0.79 9711
1 0.97 0.63 0.76 12833
accuracy 0.78 22544
macro avg 0.82 0.80 0.77 22544
weighted avg 0.84 0.78 0.77 22544
ROC AUC: 0.9584030656148944
Fitting 3 folds for each of 10 candidates, totalling 30 fits


cs assignment2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
sample_data
KDDTest+.txt
KDDTrain+.txt
ROC AUC: 0.9584030656148944
Fitting 3 folds for each of 10 candidates, totalling 30 fits
=== Improved Random Forest ===
precision recall f1-score support
0 0.67 0.97 0.79 9711
1 0.97 0.64 0.77 12833
accuracy 0.78 22544
macro avg 0.82 0.81 0.78 22544
weighted avg 0.84 0.78 0.78 22544


cs assignment2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Files
..
sample_data
KDDTest+.txt
KDDTrain+.txt
ROC AUC: 0.961391616613611
Confusion Matrix - Improved Model
9449
262
4639
8194
0
1
Actual
0
1
Predicted
8000
6000
4000
2000
How can I install Python libraries?
Load data from Google Drive
Show an example of training

Baseline Random Forest

Precision, Recall, F1-Score:
Class 0 → Precision: 0.66, Recall: 0.97, F1: 0.79, Support: 9711
Class 1 → Precision: 0.97, Recall: 0.63, F1: 0.76, Support: 12833
Accuracy: 0.78
Macro Avg: Precision 0.82, Recall 0.80, F1 0.77
ROC AUC: 0.9584

Improved Random Forest (SMOTE + Hyperparameter Tuning)

Precision, Recall, F1-Score:
Class 0 → Precision: 0.67, Recall: 0.97, F1: 0.79, Support: 9711
Class 1 → Precision: 0.97, Recall: 0.64, F1: 0.77, Support: 12833
Accuracy: 0.78
Macro Avg: Precision 0.82, Recall 0.81, F1 0.78
ROC AUC: 0.9614

6. Analysis

The baseline Random Forest shows strong performance for normal traffic but lower recall for attack traffic due to imbalance. After applying SMOTE and hyperparameter tuning, the improved model slightly enhances recall and F1-score for attack class. Overall accuracy remains consistent, and ROC AUC improves, indicating better generalization.

7. Conclusion

The NSL-KDD dataset effectively supports intrusion detection model evaluation. Baseline Random Forest performs well, but applying SMOTE and hyperparameter tuning yields marginal improvement in attack detection metrics and ROC AUC. This workflow can serve as a strong baseline for future IDS research and optimization.

8. References

1. NSL-KDD Dataset: https://www.unb.ca/cic/datasets/nsl.html
2. Scikit-learn Documentation: https://scikit-learn.org
3. Chawla, N. V., et al. "SMOTE: Synthetic Minority Over-sampling Technique." Journal of Artificial Intelligence Research, 2002.