INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SRI CITY

Intrusion Detection and Network Analysis Using NSL-KDD Dataset

A Mini Project Report Submitted in Partial Fulfillment for the Course

Python Programming for Networking and Data Science

Submitted by:

D Mabu Jaheer Abbas (OMIO25S00014)
N Raghuvender (OMIO25S00029)

Department of IoT and Autonomous Systems, IIIT Sri City

Under the Guidance of:

Faculty Guide: Dr Sreeja S R

Academic Year: 2025

Submitted at IIIT Sri City, Chittoor, India

Abstract

This project explores intrusion detection using the NSL-KDD dataset (Mireu-Lab/NSL-KDD). We perform exploratory data analysis, preprocessing, class balancing, and train multiple models. The report includes figures produced during EDA and the model evaluation results.

Contents

1 Introduction

Intrusion Detection Systems (IDS) play a critical role in network security by identifying malicious activities or policy violations. Machine learning techniques are commonly used to detect anomalies or intrusion patterns in network traffic data.

This report presents the development of a machine-learning based intrusion detection model using the NSL-KDD dataset, an improved version of the KDD'99 dataset which removes redundant records and class imbalance issues present in the original dataset.

The objectives of this study are:

- Understand the structure and statistical properties of the NSL-KDD dataset.
- Perform exploratory data analysis (EDA), including distribution and correlation analyses.
- Apply appropriate preprocessing and feature engineering.
- Perform feature selection using correlation + skew analysis.
- Train multiple ML models (Decision Tree, Random Forest, Logistic Regression).
- Evaluate classifier performance using accuracy, precision, recall, and confusion matrices.

2 Dataset Description

The NSL-KDD dataset contains network connection records labeled as normal or anormal.

Each record contains 41 features grouped into:

- **Categorical** 3 features
- **Numerical** 38 features

3 Exploratory Data Analysis

3.1 Class Distribution

To understand the balance of the data set, the frequency distribution of both binary labels is plotted.

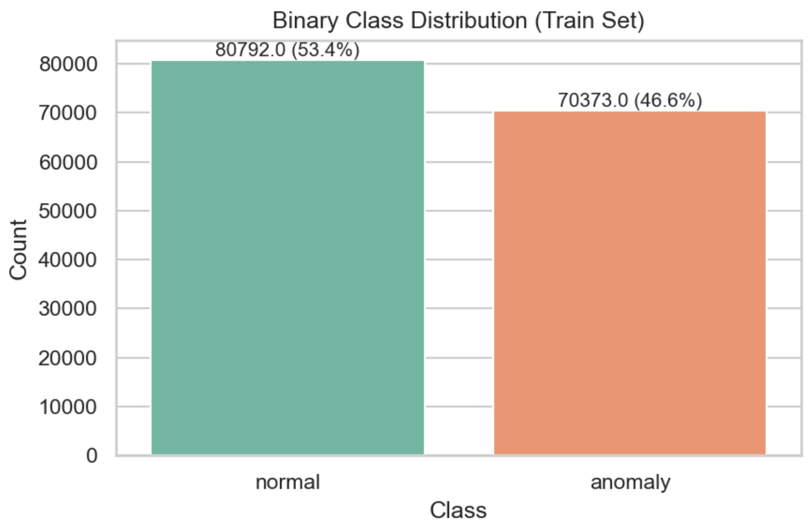


Figure 1: Class distribution in training set

3.1.1 Interpretation:

- The dataset is slightly imbalanced; normal and anomaly classes are not perfectly even.
- Imbalance motivates using class weights rather than raw accuracy.

3.2 Distribution of Numeric Features

To explore the statistical behavior of numeric features, histogram + KDE plots were generated for all numeric attributes using subplots.

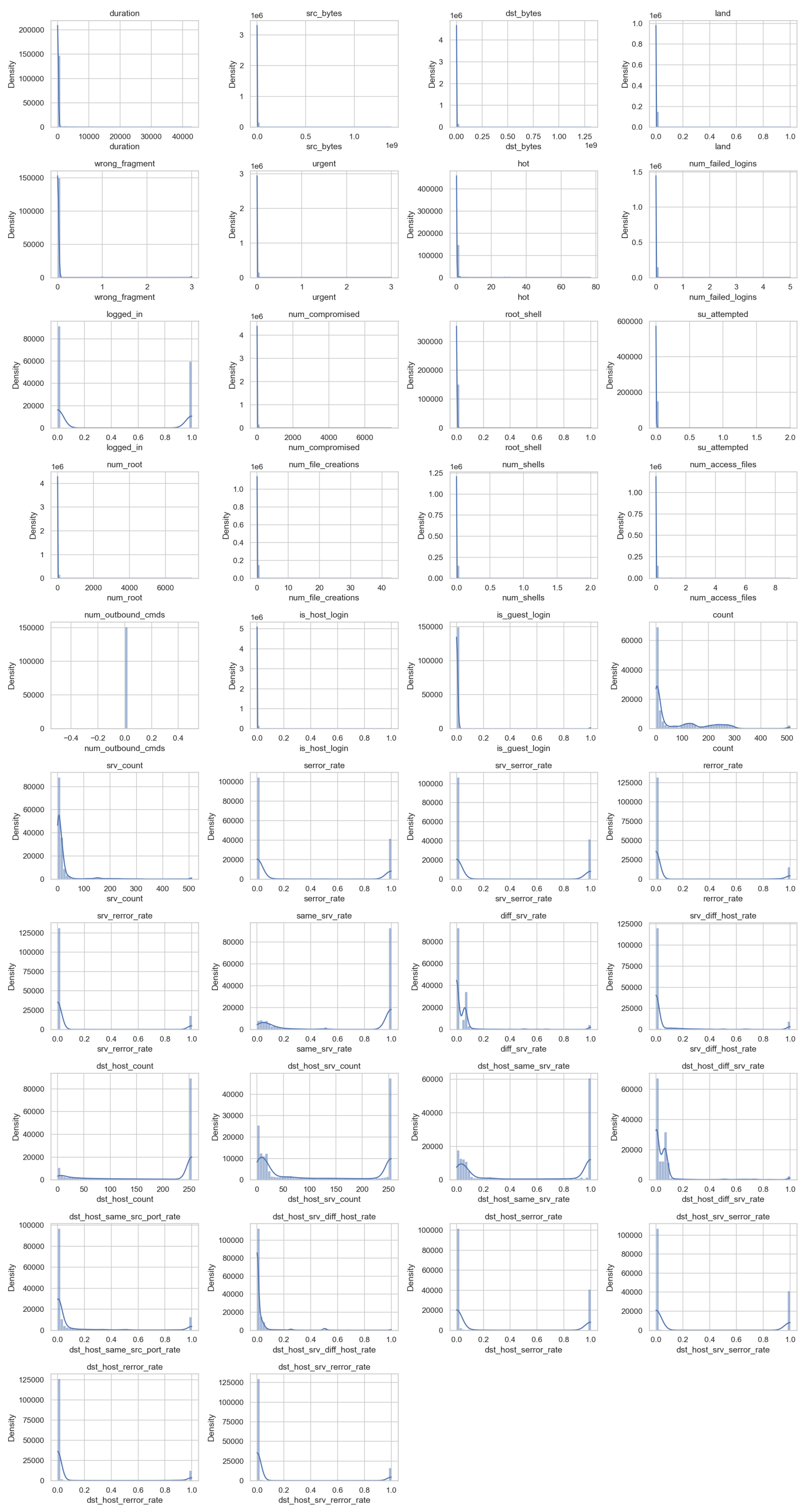


Figure 2: Histogram Plot of all Numerical features

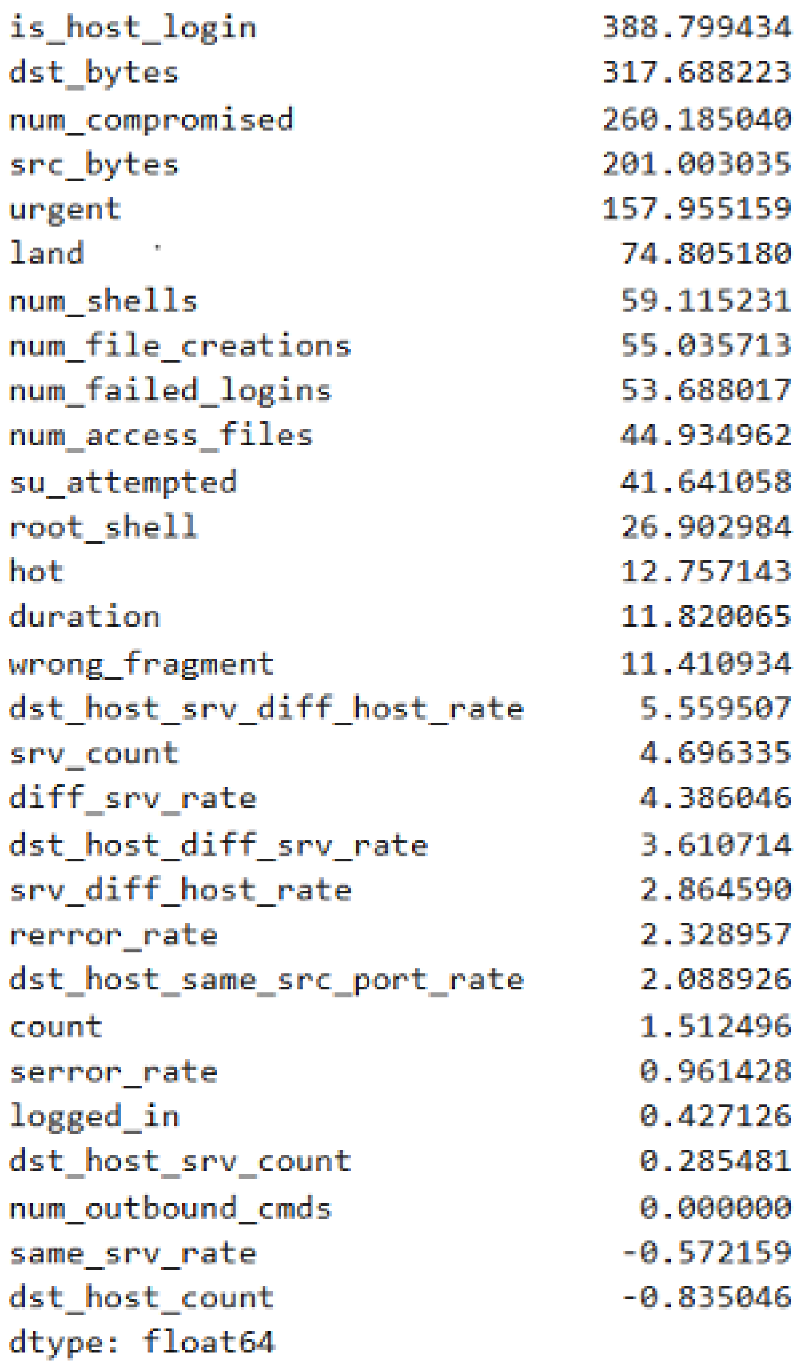
3.2.1 Observations:

- Many features exhibit heavy right-skewed distributions.

- This skewness indicates the need for log transformations to stabilize variance and reduce the effect of extreme values.

3.3 Statistical Skewness Analysis

For each numeric feature, skewness was computed quantitatively. Features with skewness > 2.0 are considered extremely skewed.



```
is_host_login                  388.799434
dst_bytes                      317.688223
num_compromised                260.185040
src_bytes                      201.003035
urgent                         157.955159
land                            74.805180
num_shells                      59.115231
num_file_creations              55.035713
num_failed_logins               53.688017
num_access_files                44.934962
su_attempted                    41.641058
root_shell                      26.902984
hot                             12.757143
duration                        11.820065
wrong_fragment                  11.410934
dst_host_srv_diff_host_rate      5.559507
srv_count                        4.696335
diff_srv_rate                    4.386046
dst_host_diff_srv_rate           3.610714
srv_diff_host_rate               2.864590
rerror_rate                      2.328957
dst_host_same_src_port_rate      2.088926
count                            1.512496
serror_rate                      0.961428
logged_in                        0.427126
dst_host_srv_count               0.285481
num_outbound_cmds                0.000000
same_srv_rate                   -0.572159
dst_host_count                  -0.835046
dtype: float64
```

Figure 3: Skewness values for all Numerical features

3.3.1 Interpretation:

- These features contain many zeros and a few huge values → long tail → unstable for ML models.
- Log transformation is appropriate for highly skewed numerical features.
- Standardize scaling for other numerical features.

3.4 Correlation Analysis

A correlation heat map was generated for all numerical and selected numeric features. Highly correlated pairs indicate redundancy.

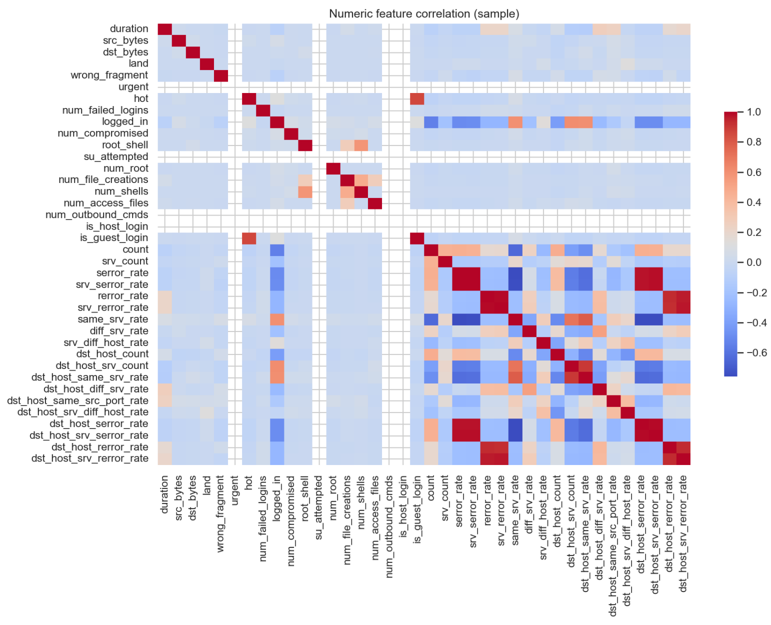


Figure 4: Correlation matrix for all Numerical features

3.4.1 Findings:

- Several traffic-related attributes show >0.8 correlation. Ex;
  - serror_rate & srv_serror_rate
  - rerror_rate & srv_rerror_rate
  - dst_host_serror_rate & dst_host_srv_serror_rate
- For each correlated pair, one feature was dropped.

This reduces multicollinearity and improves model stability.

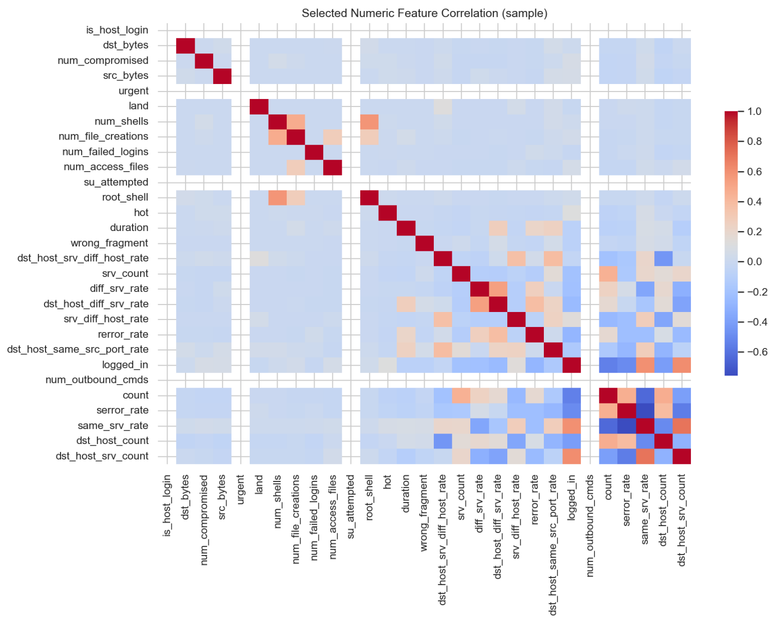


Figure 5: Correlation matrix for Selected Numerical features

4 Modeling and Evaluation

Machine learning models used in the study, and the evaluation strategy was followed to assess their performance. All models were trained using the final preprocessing pipeline that included log transformation, scaling, and one-hot encoding.

4.1 Modeling Approach

The following classifiers were implemented:

- **Decision Tree**: A simple and interpretable tree-based classifier. It was configured with a maximum depth to avoid overfitting and trained using class-balanced weights.
- **Random Forest**: An ensemble model consisting of multiple decision trees. It reduces variance and improves generalization compared to a single tree. The model was trained with 200 estimators and class-balanced weighting.
- **Logistic Regression**: A linear classifier used as a baseline. It was trained with L2 regularization and class-balanced weighting to handle dataset imbalance.

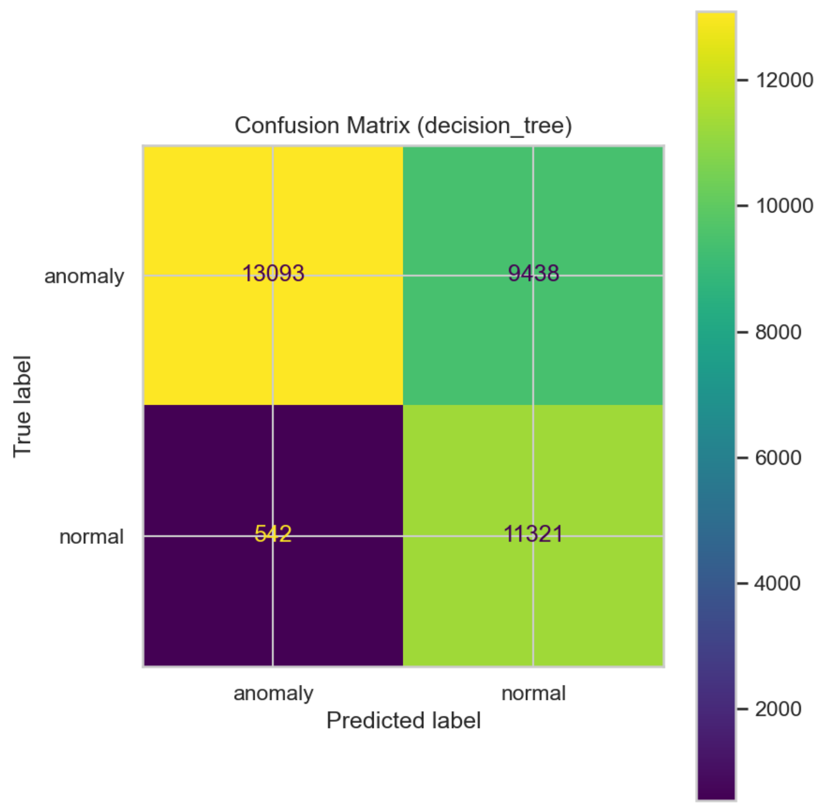
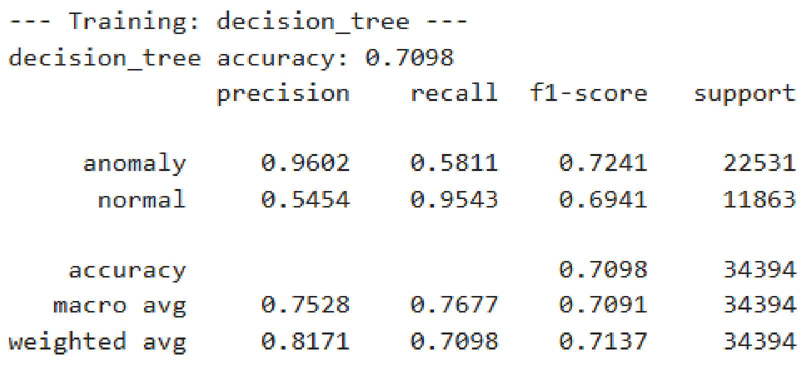


Figure 6: Confusion Matrix for Decision Tree


```
--- Training: decision_tree ---
decision_tree accuracy: 0.7098
              precision    recall  f1-score   support

     anomaly     0.9602    0.5811    0.7241     22531
      normal     0.5454    0.9543    0.6941     11863

    accuracy                         0.7098     34394
   macro avg     0.7528    0.7677    0.7091     34394
weighted avg     0.8171    0.7098    0.7137     34394
```

Figure 7: Metrics table for Decision Tree

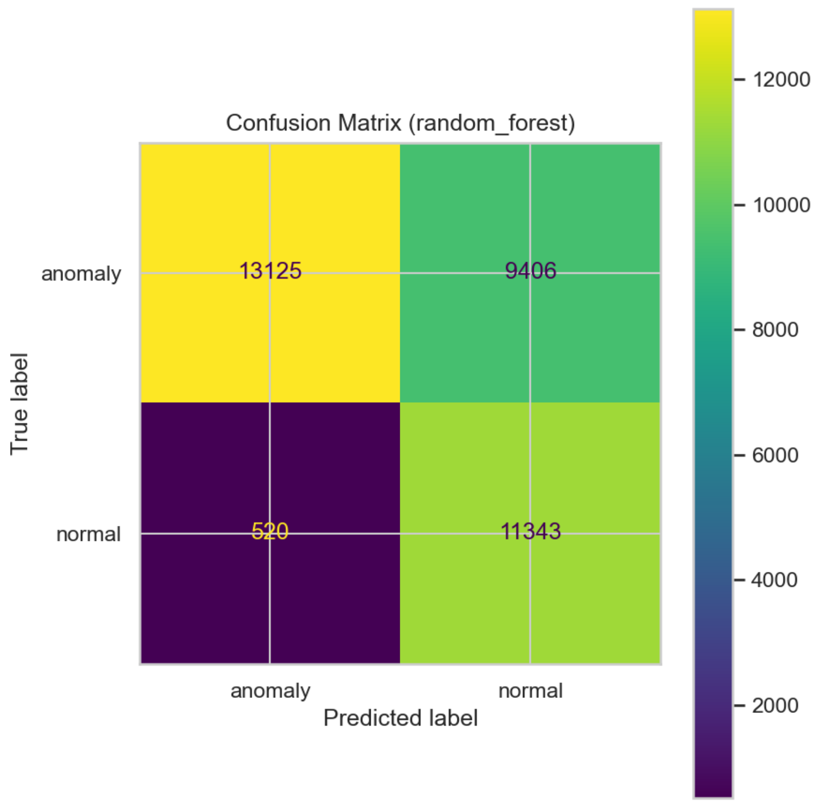


Figure 8: Confusion Matrix for Random Forest

```
--- Training: random_forest ---
random_forest accuracy: 0.7114
              precision    recall  f1-score   support

     anomaly     0.9619    0.5825    0.7256     22531
      normal     0.5467    0.9562    0.6956     11863

    accuracy                         0.7114     34394
   macro avg     0.7543    0.7693    0.7106     34394
weighted avg     0.8187    0.7114    0.7153     34394
```

Figure 9: Metrics table for Random Forest

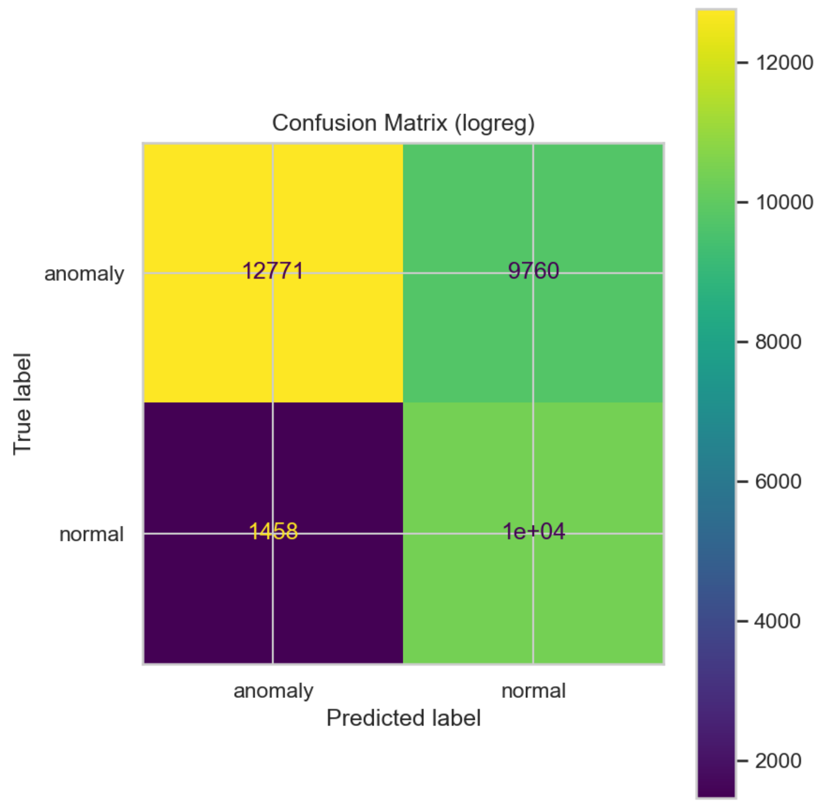
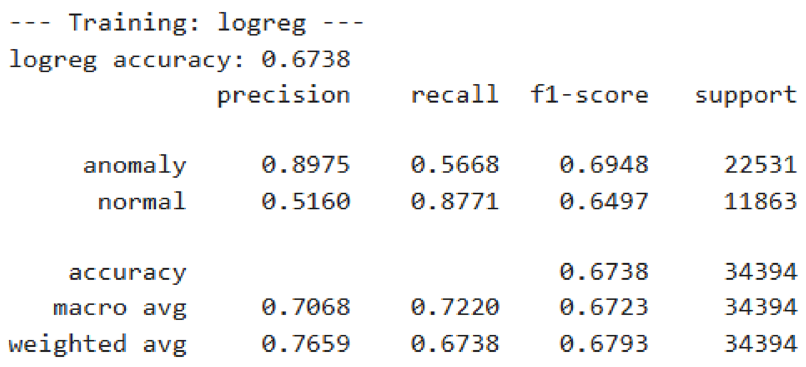


Figure 10: Confusion Matrix for Logistic Regression



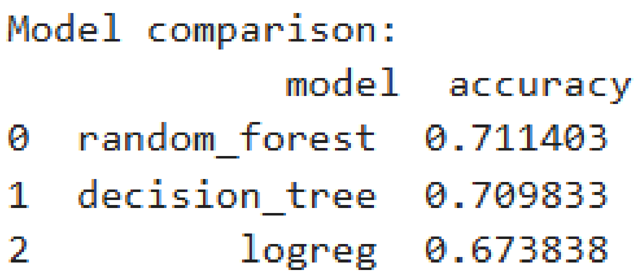
```
--- Training: logreg ---
logreg accuracy: 0.6738
              precision    recall  f1-score   support

     anomaly     0.8975    0.5668    0.6948     22531
      normal     0.5160    0.8771    0.6497     11863

    accuracy                         0.6738     34394
   macro avg     0.7068    0.7220    0.6723     34394
weighted avg     0.7659    0.6738    0.6793     34394
```

Figure 11: Metrics table for Logistic Regression

Model Accuracy Comparison:



```
Model comparison:
            model  accuracy
0  random_forest  0.711403
1  decision_tree  0.709833
2         logreg  0.673838
```

Figure 12: Model Comparison

4.2 Interpretation:

- Decision Tree captures simple patterns but misses many anomalies, due to improper/imbalanced features.
- Random Forest improves recall for anomalies.
- Logistic Regression performs badly due to the nonlinear nature of the data.

5 Conclusion

This project developed a complete intrusion detection workflow using the NSL-KDD dataset. Comprehensive exploratory analysis, appropriate preprocessing, feature selection, and model evaluation were performed.

The study demonstrates that:

- Feature skewness and correlation significantly impact model performance.
- Proper preprocessing (log transforms, scaling, encoding) is essential.
- Tree-based models perform best on NSL-KDD.
- No single model achieves perfect detection due to dataset complexity.

This pipeline provides a strong baseline for further research in anomaly detection and network intrusion detection systems.

References

1. Mireu-Lab, “NSL-KDD” dataset on Hugging Face: https://huggingface.co/datasets/Mireu-Lab/NSL-KDD
2. NSLKDD: https://www.kaggle.com/code/alam410/nslkdd
3. NSL-KDD Dataset Analysis: https://ieeexplore.ieee.org/abstract/document/10990794