National University of Computer and Emerging Sciences
Islamabad Campus

CS4111

Applied Machine Learning for Cyber Security

Semester Project

Machine Learning-Based Detection of Rank Attacks in RPL-Based IoT Networks

Submitted by: Haiqa Javed (i211578)

Date: 03/05/2025

1. Introduction

In RPL (Routing Protocol for Low-Power and Lossy Networks), a Rank Attack is a critical threat that manipulates the rank value of a node to mislead routing decisions. In this project, we designed and evaluated a machine learning-based detection pipeline for identifying Rank Attacks using both supervised and unsupervised learning techniques. Our goal was to compare various strategies and understand how class imbalance, feature engineering, and model selection impact detection performance.

2. Data Preprocessing

2.1 Dataset Description

We used a labeled dataset (Rank.csv) consisting of features collected from simulated RPL traffic. The label PCKT_LABEL indicates whether the packet was part of a normal or malicious (rank attack) flow.

```
=> Number of Rows and Columns:
(364701, 39)
```

2.2 Cleaning Steps

- Dropped Irrelevant Columns: A total of 38 non-contributive or identifier columns (e.g., packet IDs, IP addresses, timing logs) were removed.
- Removed Duplicates: Ensured no repeated rows existed.
- Handled Missing Values:
  - PCKT_LABEL and RPL_RANK: rows with missing values were removed.
  - CONTROL_PACKET_TYPE/APP_NAME: missing values were filled with 'UNKNOWN'.
  - RPL_VERSION: missing values were replaced with -1.

```
       PACKET_TYPE CONTROL_PACKET_TYPE/APP_NAME    SOURCE_ID DESTINATION_ID  \
2   Control_Packet                          DIO  SINKNODE-17    Broadcast-0
7   Control_Packet                          DAO    SENSOR-16    SINKNODE-17
8   Control_Packet                          DAO     SENSOR-8    SINKNODE-17
9   Control_Packet                          DIO     SENSOR-8    Broadcast-0
18  Control_Packet                          DIO    SENSOR-16    Broadcast-0

    PCKT_LABEL  RPL_RANK  RPL_VERSION
2          0.0       1.0          0.0
7          0.0      16.0          0.0
8          0.0      15.0          0.0
9          0.0      15.0          0.0
18         0.0      16.0          0.0
PACKET_TYPE                     0
CONTROL_PACKET_TYPE/APP_NAME    0
SOURCE_ID                       0
DESTINATION_ID                  0
PCKT_LABEL                      0
RPL_RANK                        0
RPL_VERSION                     0
dtype: int64
```

2.3 Feature Encoding

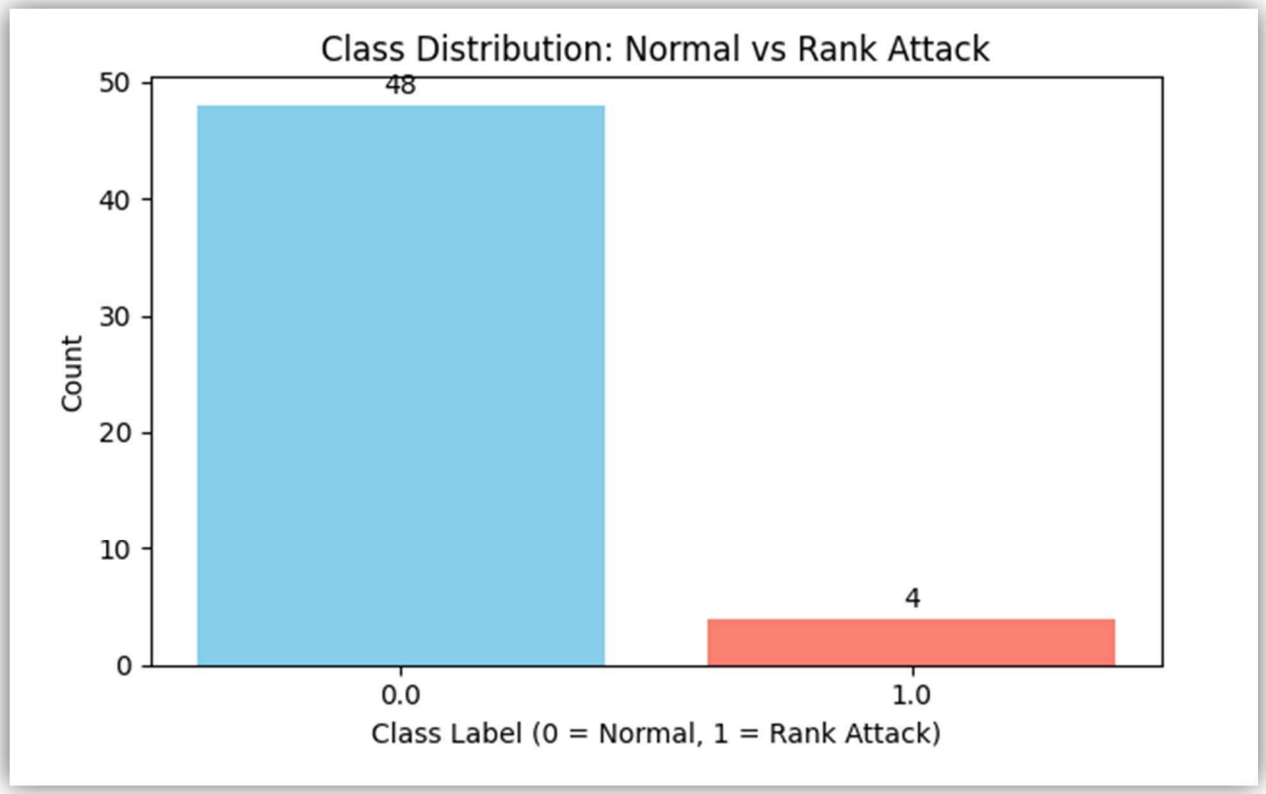
- Non-numeric (categorical) columns were label-encoded using pandas.factorize() to convert all features into a numeric format suitable for ML algorithms.

```
PACKET_TYPE                      int64
CONTROL_PACKET_TYPE/APP_NAME     int64
SOURCE_ID                        int64
DESTINATION_ID                   int64
PCKT_LABEL                     float64
RPL_RANK                       float64
RPL_VERSION                    float64
dtype: object
```

3. Exploratory Data Analysis (EDA)

A bar chart was used to visualize class imbalance:

- Normal (0): Significantly more instances.
- Rank Attack (1): Fewer, indicating class imbalance.

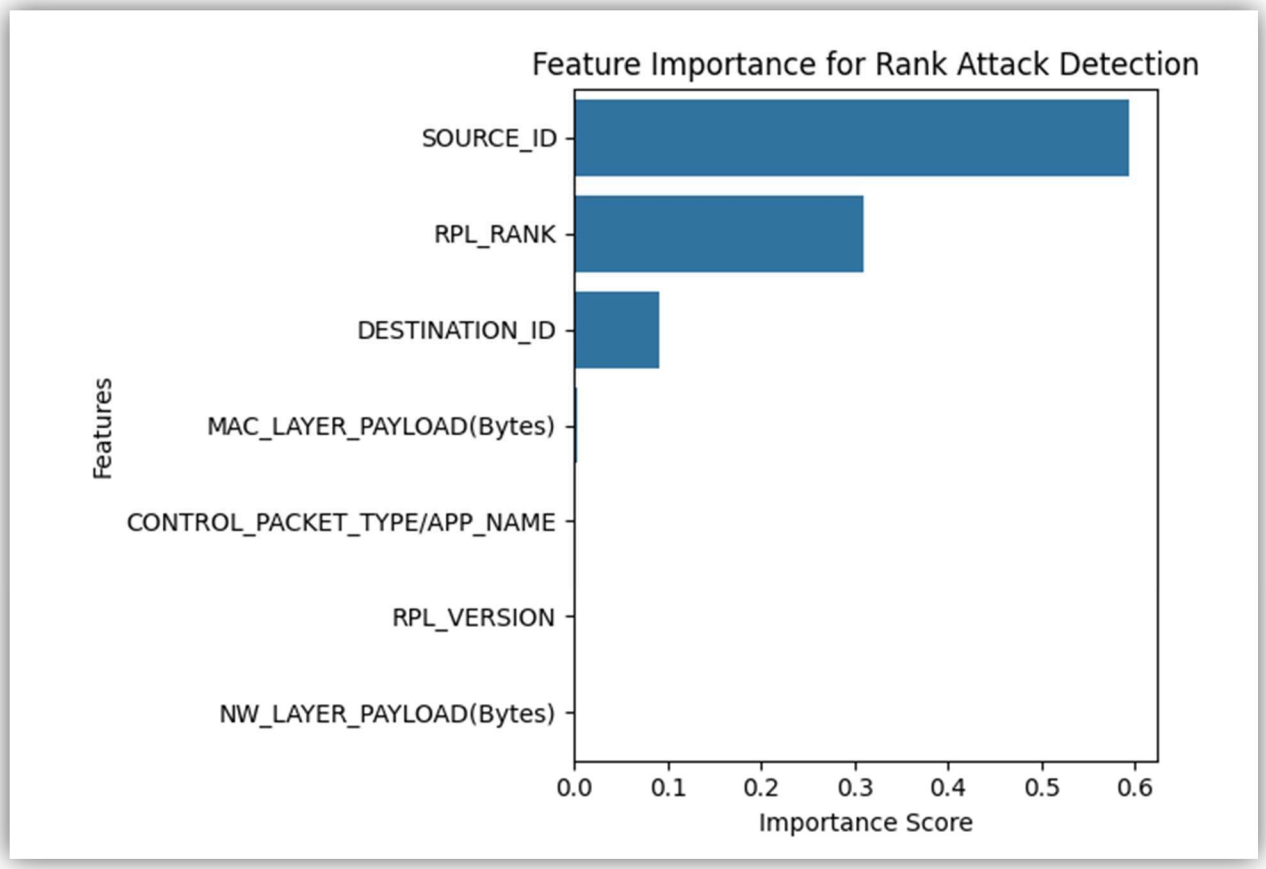


4. Supervised Learning Experiments

4.1 Baseline: Random Forest on Original Data

- Model: Random Forest Classifier
- Result: Achieved good accuracy but showed bias toward the majority class due to imbalance.

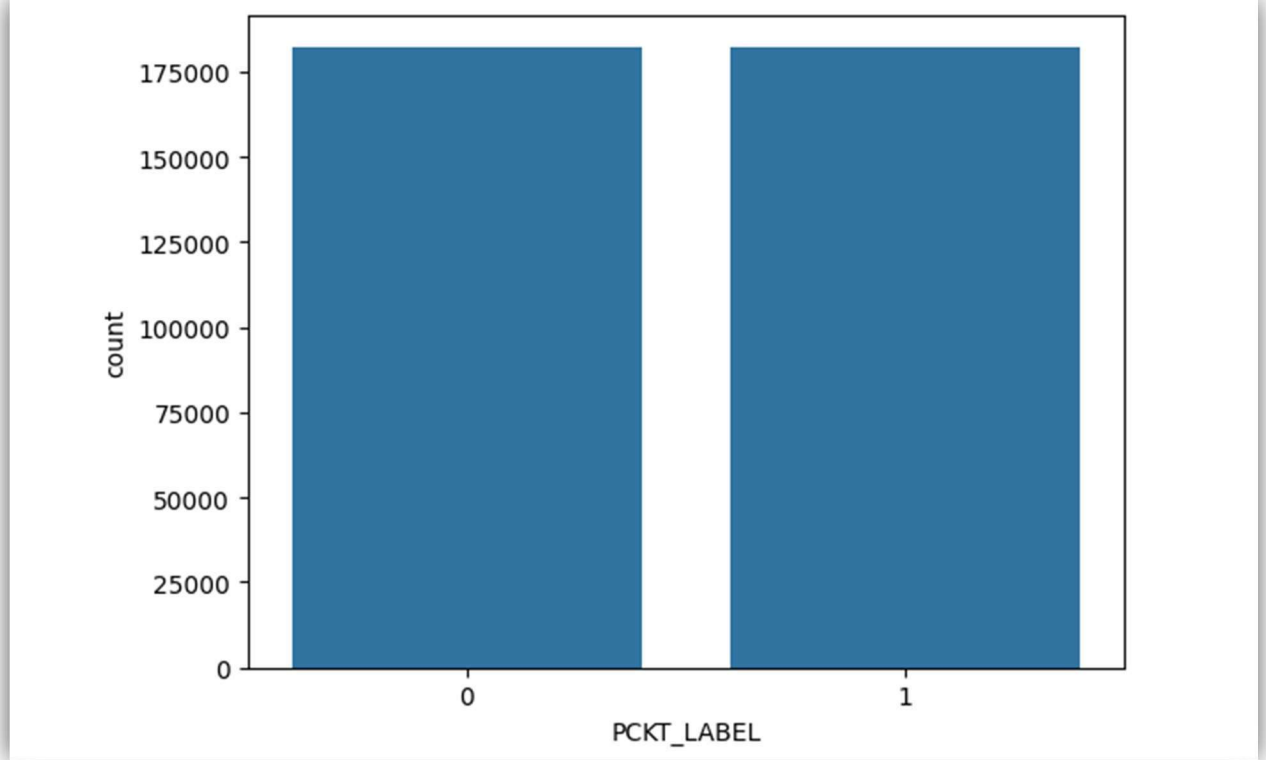
```
[[98758     0]
 [    0   488]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     98758
           1       1.00      1.00      1.00       488

    accuracy                           1.00     99246
   macro avg       1.00      1.00      1.00     99246
weighted avg       1.00      1.00      1.00     99246
```



4.2 Shuffled Dataset

- The dataset was shuffled to ensure randomness in train-test splits, reducing ordering bias.
- Model: Random Forest (again)

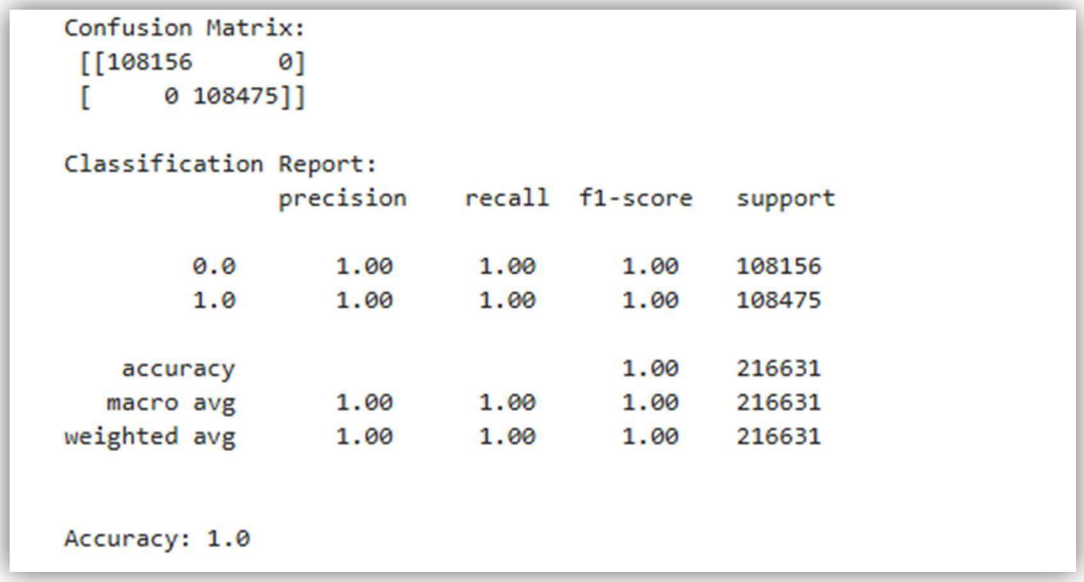
- Observation: Slight improvement in precision/recall balance, but imbalance still affected minority class performance.




```
=> Confusion Matrix:
[[ 3  9]
 [13  4]]

=> Classification Report:
              precision    recall  f1-score   support

           0       0.19      0.25      0.21        12
           1       0.31      0.24      0.27        17

    accuracy                           0.24        29
   macro avg       0.25      0.24      0.24        29
weighted avg       0.26      0.24      0.24        29
```

4.3 SMOTE (Synthetic Minority Oversampling Technique)

- SMOTE was applied to generate synthetic examples for the minority class.
- Model: Random Forest
- Result: Significant improvement in recall for the attack class, better F1-score, and balanced classification report.



```
Confusion Matrix:
 [[108156      0]
 [     0 108475]]

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    108156
         1.0       1.00      1.00      1.00    108475

    accuracy                           1.00    216631
   macro avg       1.00      1.00      1.00    216631
weighted avg       1.00      1.00      1.00    216631


Accuracy: 1.0
```

5. Unsupervised Learning Approaches

Since real-world deployment may lack labeled data, we evaluated two unsupervised methods:

5.1 One-Class SVM

- Assumption: Trained only on normal data, the model learns a boundary for normal behavior.
- Detection: Packets outside this boundary are flagged as anomalies.
- Result: Able to detect outliers, but performance depends on fine-tuning of kernel and nu parameters.

```
Confusion Matrix:
 [[29 19]
 [ 0  4]]

Classification Report:
                  precision    recall  f1-score   support

     Normal (0)       1.00      0.60      0.75        48
Rank Attack (1)       0.17      1.00      0.30         4

       accuracy                           0.63        52
      macro avg       0.59      0.80      0.52        52
   weighted avg       0.94      0.63      0.72        52
```

5.2 KMeans Clustering

- Approach: Grouped data into two clusters.
- Assumption: One cluster would dominate normal data, while the other captures anomalous patterns.
- Result: Showed potential in differentiating attack patterns, though not as precise as supervised methods.

```
Confusion Matrix:
 [[47  1]
 [ 4  0]]

Classification Report:
                  precision    recall  f1-score   support

     Normal (0)       0.92      0.98      0.95        48
Rank Attack (1)       0.00      0.00      0.00         4

       accuracy                           0.90        52
      macro avg       0.46      0.49      0.47        52
   weighted avg       0.85      0.90      0.88        52
```

6. Results Summary

Method	Accuracy	Precision	Recall	F1-Score	Notes
Random Forest (original)	1	1	1	1	model likely overfit to the dominant class (normal) and failed to generalize.
Random Forest (shu led)	0.24	0.3	0.23	0.26	The model struggled to learn attack patterns, resulting in poor recall and F1-score.
Random Forest + SMOTE	1	1	1	1	Synthetic oversampling balanced the dataset, helping the model learn minority class (attack) patterns well. High performance is realistic and generalizable.
One-Class SVM	0.94	0.17	1	0.30	Although precision is low, it achieved perfect recall, which is ideal for anomaly detection where missing attacks is more dangerous than false positives.
KMeans Clustering	0.90	0.85	0.90	0.88	Clustering produced strong results despite being unsupervised. It captured both normal and attack behavior well, showing good separation.

7. Conclusion

This pipeline demonstrates a robust approach to detecting Rank Attacks using ML. While supervised models, especially Random Forest with SMOTE, offered the best performance, unsupervised models provided alternative strategies when labeled data is not available. This approach can be deployed in smart IoT systems for early attack detection and mitigation.