NSL-KDD DATASET
INTRUSTION DETECTION

Enbarasan . D
24bai017

What is NSL-KDD Dataset?

The NSL-KDD dataset is an improved version of the older KDD’99 dataset, created for network intrusion detection research.

It contains records of network connections, each labeled as normal or an attack (like DoS, Probe, R2L, or U2R).

Each record has 41 features, such as duration, protocol type, source bytes, and service type, which describe how a computer connects over a network.

It’s used to train and test machine learning models to detect abnormal or malicious activities.

Why We Use It

We use the NSL-KDD dataset because:

- It’s a standard benchmark for intrusion detection systems (IDS).
- It removes redundant data from the old KDD dataset, making results more reliable.
- It helps evaluate how well models can detect cyberattacks automatically.

Models Used

Three classification models are used here to predict whether a connection is normal or an attack:

Logistic Regression:

- A simple model that finds the best line (or boundary) to separate normal and attack data.
- Works well when the relationship between features and output is roughly linear.

Decision Tree

- A tree-based model that splits the data based on decision rules (like "if bytes > 1000 → attack").
- Easy to interpret and handles both numeric and categorical data.

Random Forest

- An ensemble model made of many decision trees.
- Combines their outputs to give better accuracy and reduce overfitting.

- Import Libraries: Used for data handling (pandas, numpy), visualization (matplotlib, seaborn), and model building (sklearn).

- Load Dataset

```
df = pd.read_csv("/content/nsl_kdd_dataset.csv")
```

- Label Conversion

```
df["label"] = df["label"].apply(lambda x: 0 if x == "normal" else 1)
```

Converts “normal” to 0 and “attack” to 1 for binary classification.

Data Visualization

- Histograms show the distribution of numeric features.
- Scatter plots visualize the separation between attack and normal data.

Split Features and Labels

- X contains all input features.
- y contains output labels (0 or 1).

- Training and Evaluation

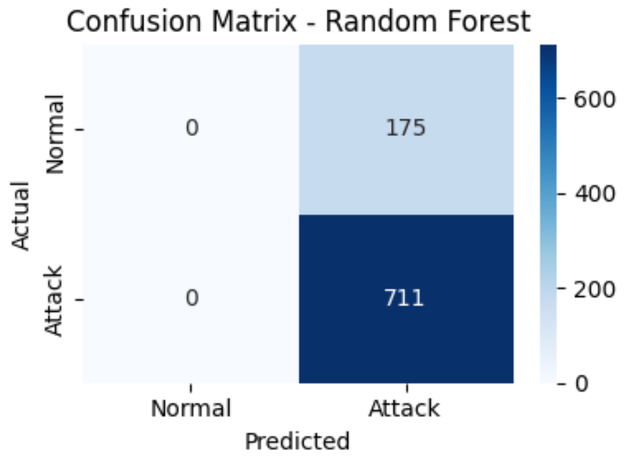
Each model is trained using a pipeline (preprocessing + classifier).

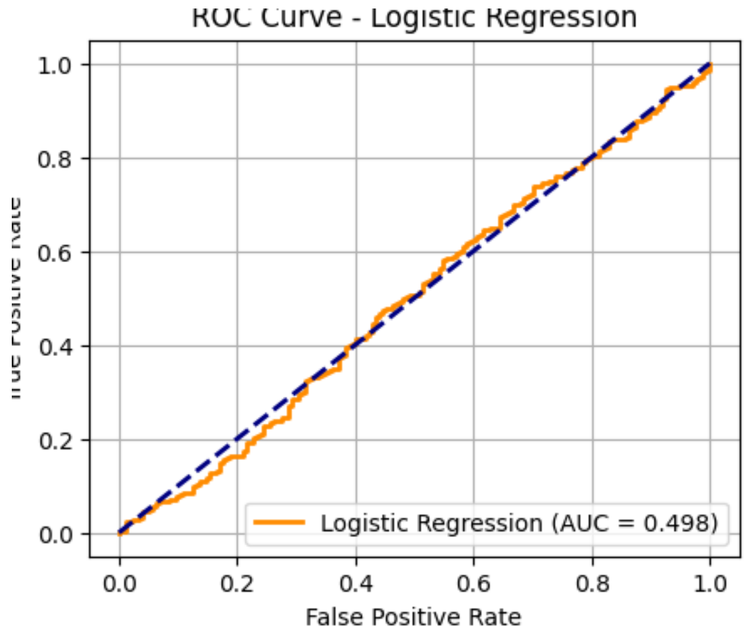
Model performance is measured by:

- Accuracy (overall correctness)
- Precision (how many predicted attacks are actually attacks)
- Recall (how many real attacks were detected)
- ROC-AUC (model’s overall classification ability)
- Confusion Matrix shows counts of correct/incorrect predictions.

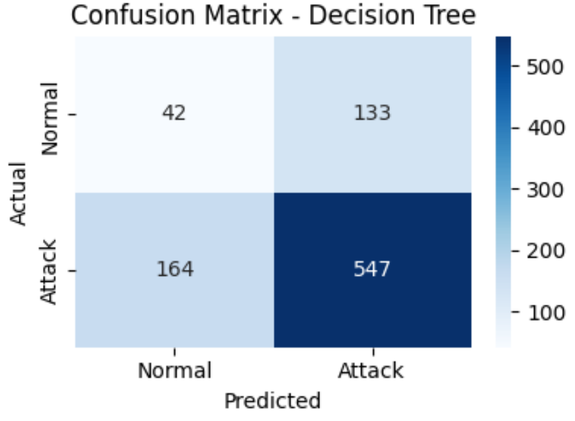
OUTPUT:

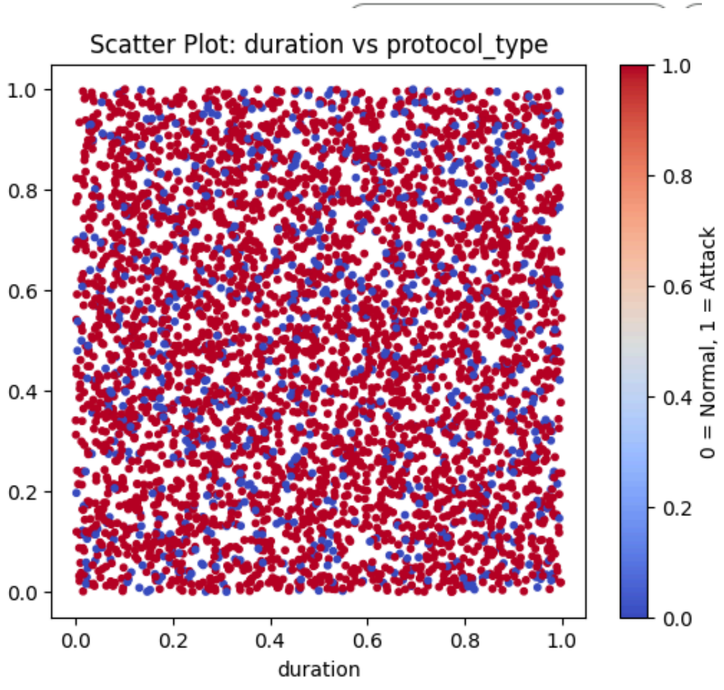
```
TRAINING MODEL: Random Forest
========================================

✓ Results for Random Forest:
Accuracy : 0.8024830699774267
Precision: 0.8024830699774267
Recall   : 1.0
ROC-AUC  : 0.5033674904560981
```



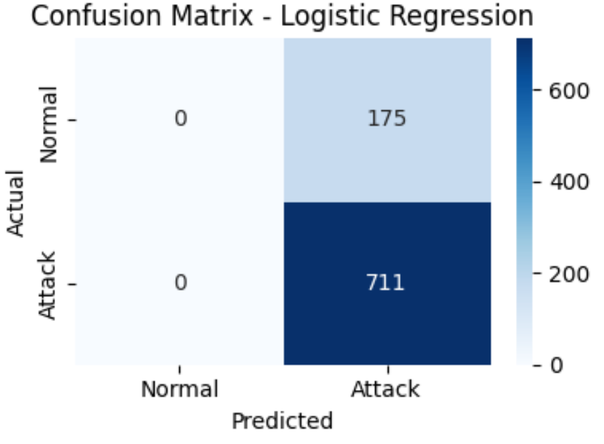



```
----------------------------------------
 TRAINING MODEL: Decision Tree
========================================

✓ Results for Decision Tree:
Accuracy : 0.664785553047404
Precision: 0.8044117647058824
Recall   : 0.7693389592123769
ROC-AUC  : 0.5046694796061885
```






```
TRAINING MODEL: Logistic Regression
========================================

✓ Results for Logistic Regression:
Accuracy : 0.8024830699774267
Precision: 0.8024830699774267
Recall   : 1.0
ROC-AUC  : 0.49840466144263607
```



THANK YOU