Lab: Building and Evaluating Decision Trees for Intrusion Detection Using NSL-KDD and IoT-Detection Datasets

Lab Overview

In this lab, you will learn how to implement and evaluate decision trees for classification tasks. The application will be on two real-world datasets: NSL-KDD for intrusion detection and IoT-Detection for identifying malicious IoT behavior. You will explore how decision trees work, from data preprocessing to model training, tuning, and evaluation. Additionally, you will examine feature importance and experiment with techniques to prevent overfitting, such as tree pruning.

By the end of the lab, you should be able to:

1. Preprocess data from different sources (NSL-KDD and IoT-Detection).
2. Build a decision tree classifier and fine-tune its hyperparameters.
3. Evaluate the model's performance using key metrics like precision, recall, and F1-score.
4. Visualize and interpret the importance of features in decision-making.
5. Understand and apply pruning to improve generalization.

1. Dataset Overview

- **NSL-KDD**: Discuss how this dataset is used for intrusion detection, highlighting different features related to network traffic.
- **IoT-Detection Dataset**: Highlight how this dataset is used for detecting malicious behaviors in IoT environments.

2. Preprocessing

- **NSL-KDD**: Convert categorical features into numerical ones (if applicable). You might also need to normalize certain features.
- **IoT-Detection**: Handle missing values, categorical encoding, normalization, etc.

3. Building a Decision Tree Classifier

- Using scikit-learn:

1. Load the dataset.
2. Perform a train-test split (e.g., 80% training and 20% testing).
3. Train a DecisionTreeClassifier on the training data.
4. Tune the decision tree by varying parameters such as maximum depth, minimum samples per leaf, and splitting criterion (entropy vs Gini).

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
# Load data (Assume X and y are preprocessed feature matrix and labels)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize and train the model
clf = DecisionTreeClassifier(criterion='gini', max_depth=10, random_state=42)
clf.fit(X_train, y_train)
# Make predictions
y_pred = clf.predict(X_test)
# Evaluate the model
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

4. Evaluating the Model

- Discuss the evaluation metrics, especially precision, recall, F1-score, and confusion matrix.
- experiment with different parameters to observe overfitting/underfitting.

5. Feature Importance

- extract feature importance from the trained decision tree:

```
import matplotlib.pyplot as plt
import pandas as pd
feature_importance = clf.feature_importances_
feature_names = X.columns  # Assuming you have a DataFrame
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':
feature_importance})
importance_df.sort_values(by='Importance', ascending=False, inplace=True)
plt.barh(importance_df['Feature'], importance_df['Importance'])
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Decision Tree')
plt.show()
```