Python Project: Network Intrusion Detection System using Machine Learning

Project Title: Network Intrusion Detection System using Machine Learning

Project Domain / Category: Computer Networks / Cybersecurity / Machine Learning

Introduction:

The objective of this project is to build a Network Intrusion Detection System (NIDS) that uses machine learning to detect and classify network intrusions. This system can help in identifying malicious activities in network traffic by analyzing patterns and behaviors. By using a dataset containing both normal and abnormal network traffic, we aim to develop a model that can classify traffic as either benign or malicious.

Dataset Suggestions:

- **KDD Cup 99 Dataset:** KDD Cup 99 on Kaggle
- **NSL-KDD Dataset:** NSL-KDD on Canadian Institute for Cybersecurity
- **CICIDS2017 Dataset:** CICIDS2017 on Canadian Institute for Cybersecurity

These datasets contain records of network activities and labels identifying if they represent normal or malicious traffic.

Functional Requirements for the System:

1. **Data Collection and Pre-processing:**
    - **Data Cleaning:** Handle missing values, duplicates, and irrelevant features in the dataset.
    - **Feature Engineering:** Convert categorical data to numerical formats (e.g., one-hot encoding for protocol types).
    - **Data Normalization:** Normalize numerical values to a common scale for consistent model training.
    - **Data Splitting:** Divide the dataset into training, validation, and testing sets to ensure accurate model evaluation.
2. **Model Selection and Training:**
    - **Algorithm Selection:**
        - Experiment with algorithms such as Decision Trees, Random Forest, Logistic Regression, Naïve Bayes, and Support Vector Machine (SVM).
        - Evaluate the effectiveness of each algorithm on detecting intrusions based on accuracy, precision, and recall.
    - **Feature Selection:**
        - Use methods like Recursive Feature Elimination (RFE) or feature importance from Random Forest to select the most impactful features, improving the model's performance.
    - **Model Training:**

- Train the selected model(s) on the pre-processed training data using Scikit-Learn or TensorFlow.
- Implement cross-validation to fine-tune the hyperparameters for better results.

3. **Intrusion Detection and Classification:**
    - **Real-Time Detection (Optional):**
        - Use the trained model to classify incoming network traffic in real-time, if feasible.
    - **Batch Processing:**
        - For datasets that are not real-time, classify network traffic records as "normal" or "intrusion."
    - **Confusion Matrix:**
        - Generate a confusion matrix to evaluate model performance. Measure metrics like accuracy, precision, recall, and F1-score to assess the detection capabilities.
4. **Result Display and Analysis:**
    - **Intrusion Class Display:**
        - After detecting an intrusion, display the type of intrusion (e.g., DoS, Probe, U2R, R2L for the KDD dataset) and any related details.
    - **Performance Metrics Display:**
        - Display performance metrics (e.g., accuracy, precision, recall, F1-score) for the model, helping the user understand the reliability of the detection.
5. **Accuracy and Model Comparison:**
    - Compare the accuracy and other performance metrics of different models to identify the best-performing algorithm for network intrusion detection.

This project provides a hands-on opportunity to apply machine learning in the field of cybersecurity, particularly for detecting and classifying malicious network traffic.