Python Project: IoT Sensor Data Analysis and Anomaly Detection using Machine Learning

Project Title: IoT Sensor Data Analysis and Anomaly Detection using Machine Learning

Project Domain / Category: Internet of Things (IoT) / Data Analytics / Machine Learning

Introduction:

This project aims to analyze sensor data from IoT devices and detect anomalies that could indicate unusual patterns, such as equipment malfunction or abnormal environmental conditions. By applying machine learning techniques, we can develop a system that identifies deviations in sensor readings and alerts users to potential issues.

Dataset Suggestions:

- **Intel Berkeley Research Lab Sensor Data:** Intel Lab Data on UCI
- **Smart Home Dataset:** Smart Home Dataset on Kaggle
- **NAB (Numenta Anomaly Benchmark):** NAB on GitHub

These datasets include temperature, humidity, light, and other environmental readings from IoT devices, which are ideal for anomaly detection.

Functional Requirements for the System:

1. **Data Collection and Pre-processing:**
    - **Data Cleaning:** Handle missing values and noise in the sensor data.
    - **Data Aggregation:** Aggregate data over time intervals (e.g., hourly or daily) if needed, to observe patterns.
    - **Normalization and Scaling:** Scale data to a common range to improve model training.
    - **Data Splitting:** Split the data into training, validation, and testing sets for effective model evaluation.
2. **Exploratory Data Analysis (EDA):**
    - Visualize sensor data trends over time using line charts and heatmaps to understand patterns and relationships.
    - Identify natural thresholds or patterns in the data that may indicate normal vs. abnormal states.
3. **Anomaly Detection Model Selection:**
    - **Algorithm Selection:**
        - Test different algorithms such as Isolation Forest, One-Class SVM, Autoencoders (for neural network-based anomaly detection), or K-means clustering.
    - **Threshold Setting:**
        - Define thresholds for normal and anomalous behavior based on statistical measures (e.g., z-score) or based on model output scores.

- **Feature Engineering (if needed):**
  - Derive features like rolling averages, peaks, and valleys from sensor readings to enhance model accuracy.

4. **Anomaly Detection and Alerts:**

   - Use the trained model to detect anomalies in real-time or batch-processed sensor data.
   - If an anomaly is detected, trigger an alert (e.g., email or app notification) to inform users of potential issues.

5. **Evaluation and Model Tuning:**

   - **Evaluation Metrics:**
     - Use metrics like precision, recall, and F1-score to evaluate the anomaly detection model's performance.
   - **Hyperparameter Tuning:**
     - Perform grid search or random search to fine-tune model parameters for improved accuracy.

6. **Visualization and Reporting:**

   - **Anomaly Visualization:**
     - Highlight anomalies on a time-series graph to provide users with visual insights into when and where anomalies occurred.
   - **Performance Report:**
     - Generate a report summarizing the model's performance, including detection accuracy, and anomaly frequency.

7. **Additional Functionalities (Optional):**

   - **Predictive Maintenance:** Use historical data to predict the likelihood of future anomalies, enabling proactive maintenance.
   - **Dashboard:** Create an interactive dashboard using tools like Plotly or Dash to display live sensor data, anomaly status, and analysis metrics.

This project allows you to apply machine learning to IoT sensor data for real-time analysis and anomaly detection. It's a practical project that has real-world applications in industries like manufacturing, smart homes, and environmental monitoring.