Automated Anomaly Detection for Predictive Maintenance

Capstone Project Presentation

Project Overview

Objective:

Predict anomalies in machine sensor data to prevent unplanned downtime and enhance maintenance planning.

Approach:

Building a machine learning model that analyzes time-series sensor data.

Use of Flask for deploying the anomaly detection system.

Dataset Overview

Dataset:

Sensor data including temperature, pressure, RPM, etc.

Collected over several months with a large number of records.

Key Features:

Time-based, multi-featured dataset with anomalies occurring at irregular intervals.

Data Preprocessing

Steps:

Missing Values: Imputed using forward fill techniques.

Outlier Detection: Handled extreme outliers using statistical thresholds.

Feature Engineering: Rolling averages and moving standard deviations to detect trends.

Model Selection

Considered Algorithms:

Isolation Forest: Suitable for unsupervised anomaly detection.

Autoencoders: Deep learning technique to reconstruct input and detect anomalies based on reconstruction error.

ARIMA: Time-series forecasting and anomaly detection.

Final Choice:

Isolation Forest due to efficiency and ease of interpretation.

Model Training

Training Setup:

- 70% of data used for training, 30% for testing.
- Hyperparameter tuning done using GridSearch.

Validation:

- 5-fold cross-validation to avoid overfitting.

Flask Deployment

Deployment via Flask:

- **Flask API** used to serve the trained model.
- Deployed on a server for real-time anomaly detection.

How it Works:

- Sensor data is sent to the Flask API.
- The model processes the data and returns whether an anomaly is detected.

Performance Metrics

Metrics Used:

- Precision, Recall, F1-Score for anomaly detection performance.

Results:

- **Precision**: 85%
- **Recall**: 78%
- **F1-Score**: 81%

Results and Visualization

Anomaly Detection Visualization:

- Graph showing sensor data with anomalies detected in real-time.

Impact:

- Reduced unplanned machine downtime and improved maintenance scheduling.

Conclusion

Summary:

- Successfully implemented an anomaly detection system using Isolation Forest and Flask API for real-time deployment.

Future Work:

- Extend the system to handle more complex sensor networks and larger data volumes.