Anomaly Detection for Predictive Maintenance - Report

1. Introduction

In this project, we aim to build a machine learning model to predict anomalies in time series data to help identify potential machine breakdowns. The dataset contains various time series features and the target variable indicates whether an anomaly is present.

1.1 Objective

- Predict machine anomalies based on sensor data.
- Achieve >75% accuracy on the test set.

2. Data Overview

2.1 Dataset

- **Number of observations**: 18,000+
- **Features**: 60 time-series features (`x1`, `x2`, ..., `x60`) and 1 time-related feature (`time`).
- **Target variable**:
    - `y`: 1 (anomaly) or 0 (no anomaly).
    - Additional column `y.1` for validation.

2.2 Feature Description

- `time`: Timestamp of the observation.
- `x1`, `x2`, `x3`, ... , `x60`: Sensor readings from different aspects of the machine.
- `y`: Binary target indicating anomaly (1) or no anomaly (0).

2.3 Data Preprocessing

Steps performed:

- **Checked for missing values**: There were no missing values in the dataset.

- **Dropped irrelevant columns**: Removed `y.1`, as it appeared to be a duplicate target column.
- **Data Types**: Ensured that all feature columns were in numerical format, converted any categorical or datetime features if present.

3. Exploratory Data Analysis (EDA)

3.1 Correlation Matrix

- Generated a correlation heatmap to examine relationships between features. Some features exhibited moderate correlations, while most were independent of one another.

3.2 Distribution of Target Variable

- The dataset is **highly imbalanced** with fewer instances of anomalies (`y = 1`), indicating the need for handling imbalanced data during modeling.

4. Model Selection

4.1 RandomForest Classifier

- Selected Random Forest due to its robustness for handling tabular data with many features and its ability to handle imbalanced datasets.

4.2 Train-Test Split

- Split the data into training and testing sets (80-20 split).

5. Model Training

- **Model Used**: RandomForestClassifier
- **Training Data**: 80% of the dataset.
- **Random State**: 42 to ensure reproducibility.

6. Model Evaluation

6.1 Initial Results

- **Accuracy**: 100% (This is likely due to overfitting or data leakage and needs further investigation).

6.2 Next Steps

- Perform cross-validation to validate the model's performance on different subsets of data.
- Address any overfitting or imbalance issues by applying methods such as:
    - Cross-validation.
    - Resampling (SMOTE, etc.) to balance the target classes.
    - Hyperparameter tuning to reduce overfitting.

7. Conclusion

- A Random Forest model was trained on the sensor data and achieved an accuracy of 100%. However, further analysis is needed to confirm whether this performance is valid or due to data leakage or overfitting.
- Next steps will include addressing these issues and evaluating the model using cross-validation and other metrics like precision, recall, and F1-score.