Project Report: AnomaData - Automated Anomaly Detection for Predictive Maintenance

1. Problem Statement

Many industries face the challenge of system failures in their equipment. This project addresses the need for predictive maintenance solutions by leveraging data analytics to identify anomalies, providing early indications of potential failures.

2. Data Overview

Dataset: Time series data with 18000+ rows collected over several days.

Target Variable: Binary labels in column 'y,' where 1 denotes an anomaly.

Predictors: Other columns containing features for analysis.

3.Tasks/Activities List

- Collect the time series data from the CSV file linked here.
- Exploratory Data Analysis (EDA) - Show the Data quality check, treat the missing values, outliers etc. if any.
- Get the correct data type for date.
- Feature Engineering and feature selection.
- Train/Test Split - Apply a sampling distribution to find the best split
- Choose the metrics for the model evaluation
- Model Selection, Training, Predicting and Assessment
- Hyper parameter Tuning/Model Improvement
- Model deployment plan.

3.1 Import Libraries and Load Dataset-It Consist of libraries installation & Data fetching

Details are as follow:

```
# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report

from sklearn.model_selection import GridSearchCV


#Collect the time series data from the CSV file linked here.

data= pd.read_csv("C:/Users/admin/Downloads/Anoma_data.csv")
```

3.2 Exploratory Data Analysis (EDA)- It Consist of data quality checks, handling of missing values, and explored data distribution.

```
# Display basic statistics

print(data.describe())

# Check for missing values

print(data.isnull().sum())


# Visualize data distribution

sns.countplot(data['y'])

plt.show()


# Explore relationships between predictors and target variable

sns.pairplot(data, hue='y')

plt.show()
```

3.3 Data Cleaning- It Consist of managing the missing values and standardize the data

```
# Handle missing values
data.fillna(data.mean(), inplace=True)

# Handle outliers (you can use various techniques)
# For example, using Z-score
from scipy.stats import zscore
z_scores = zscore(data.drop('y', axis=1))
data_no_outliers = data [(z_scores < 3).all(axis=1)]

# Check the effect on the dataset
print("Original data shape:", data.shape)
print("Data shape after handling outliers:", data_no_outliers.shape)
```

3.4 Feature Engineering- It consist of data type changing and extraction of addition features

```
# Get the correct datatype for date
data['date'] = pd.to_datetime(data['date'])
```

3.5 Train/Test Split- It consist of application of sampling distribution to find the best split

```
# Train/Test Split
X = data.drop(['y', 'date'], axis=1)  # Exclude 'date' column
y = data['y']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.5 Model Selection, Training, and Assessment- It Consist of evaluated model performance on unseen data

```
# Choose a model (Random Forest selected as below)
model = RandomForestClassifier(random_state=42)

# Model Training
model.fit(X_train, y_train)
# Predictions on the test set
y_pred = model.predict(X_test)
# Model Assessment
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

3.6 Hyperparameter Tuning- It Consist of Hyper parameter tuning using GridSearchCV

```
# using GridSearchCV for hyperparameter tuning
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]}
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
best_model.fit(X_train, y_train)
best_y_pred = best_model.predict(X_test)

# Model Assessment after tuning
best_accuracy = accuracy_score(y_test, best_y_pred)
print(f'Best Model Accuracy: {best_accuracy * 100:.2f}%')
```

3.7 Model Deployment Plan

```
import joblib

joblib.dump(best_model, 'anomaly_detection_model.pkl')
```