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 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

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

# Load the dataset

Anoma_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(Anoma_data.describe())


# Check for missing values

print(Anoma_data.isnull().sum())


# Visualize data distribution

sns.countplot(Anoma_data ['y'])

plt.show()


# Explore relationships between predictors and target variable

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

plt.show()
```

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

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

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

# Check the effect on the dataset
print("Original data shape:", Anoma_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

```
# Convert date column to datetime type
Anoma_data ['date'] = pd.to_datetime(Anoma_data ['date'])

# Extract features from date
Anoma_data ['day'] = Anoma_data ['date'].dt.day
Anoma_data ['month'] = Anoma_data ['date'].dt.month
Anoma_data ['year'] = Anoma_data ['date'].dt.year

# Drop unnecessary columns
Anoma_data.drop(['date'], axis=1, inplace=True)
```

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

```
# Split data into features and target variable

X = Anoma_data.drop('y', axis=1)

y = Anoma_data ['y']


# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)6. Model
Selection, Training, and Assessment
```

3.6 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)


# Train the model

model.fit(X_train, y_train)


# Make predictions

y_pred = model.predict(X_test)


# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)


# Classification Report

print("Classification Report:\n", classification_report(y_test, y_pred))


# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(conf_matrix, annot=True, fmt='d')

plt.show()
```

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

```
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Retrain the model with the best parameters
best_model = grid_search.best_estimator_
best_model.fit(X_train, y_train)
```

3.8 Model Validation

```
# Validate the model
y_val_pred = best_model.predict(X_test)

# Evaluation
val_accuracy = accuracy_score(y_test, y_val_pred)
print("Validation Accuracy:", val_accuracy)
```

3.9 Model Deployment Plan

```
# Save the model using joblib

joblib.dump(best_model, 'predictive_maintenance_model.joblib')
```

3.10 Success Metrics

```
# Evaluate the model on test data

y_test_pred = best_model.predict(X_test)

test_accuracy = accuracy_score(y_test, y_test_pred)

print("Test Accuracy:", test_accuracy)


# Hyperparameter tuning details

print("Best Hyperparameters:", best_params)


# Model Validation on Unseen Data

y_val_pred = best_model.predict(X_test)

val_accuracy = accuracy_score(y_test, y_val_pred)

print("Validation Accuracy:", val_accuracy)
```