PREDICTIVE MAINTENANCE FOR INDUSTRIAL EQUIPMENT

MINI PROJECT REPORT

Submitted By

C.S PRIYADARSHINI (211501071)

J.S RAKSHEETHA (211501074)

V.J RITHIKA SRI REDDY (211501081)

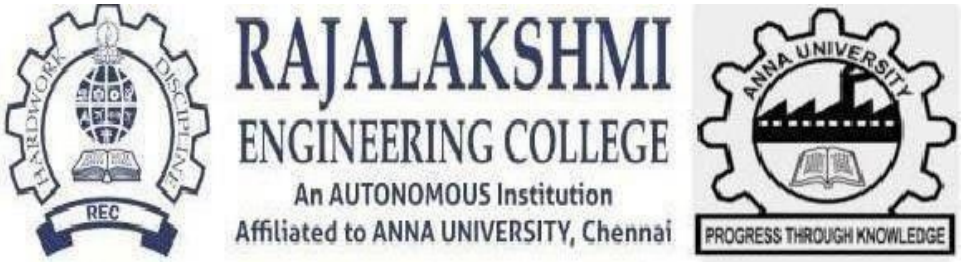
In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI-600 025

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this Report titled" **Predictive Maintenance for Industrial Equipment"** is the bonafide work of **"C.S.Priyadarshini(211501071)", "J.S.Raksheetha(211501074)", "V.J.Rithika Sri Reddy (211501081)"** who carried out the work for AI19P71-Data Visualisation for Python laboratory under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. D. Sorna Shanthi M.Tech.,
Associate Professor
Department of Artificial
Intelligence and Data Science

ABSTRACT

This project presents a comprehensive predictive maintenance solution aimed at forecasting equipment failures to minimize operational downtime and reduce maintenance costs. Using the predictive maintenance dataset, which includes machine operational settings, sensor measurements, and historical failure data, the project integrates exploratory data analysis and machine learning models to derive actionable insights and build a robust failure prediction framework. Key insights from EDA reveal that tool wear, temperature differences, heat dissipation, and torque-speed interactions are significant predictors of failures. Relationships among operational features, such as temperature and torque, were analyzed using correlation matrices, while failure type distributions provided a deeper understanding of the failure landscape. These insights inform a data-driven approach to maintenance planning. To predict failures, the project employs three machine learning algorithms Random Forest Classifier, known for its capability to handle tabular and imbalanced datasets while providing feature importance; Logistic Regression, offering simplicity and interpretability; and Support Vector Machine (SVM), which excels in defining complex decision boundaries for accurate classification. A pipeline comprising data cleaning, feature engineering, and hyperparameter tuning ensures the reliability and robustness of the models. The resulting predictive insights enable proactive maintenance scheduling, enhance equipment reliability, and support cost-effective operations.

Keywords:

Predictive Maintenance, Machine Learning Algorithms, Operational Efficiency, Imbalanced Dataset Handling, Data Preprocessing, Hyperparameter Tuning

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROBLEM STATEMENT :

In industries reliant on machinery, unexpected equipment failures can lead to significant downtime, high costs, and operational disruptions. Traditional maintenance methods, like time-based or reactive maintenance, often fall short in preventing these issues, making predictive maintenance an essential approach. This project aims to develop a machine learning- based predictive maintenance model that forecasts equipment failures by analyzing operational settings, sensor data, and historical failure events from the "Predictive Maintenance Dataset." Utilizing Random Forest, Logistic Regression, and Support Vector Machine (SVM) algorithms, the model identifies patterns that signal potential failures, enabling proactive maintenance scheduling. This approach not only minimizes unplanned downtime but also enhances equipment reliability and reduces maintenance expenses, supporting efficient and cost- effective operations.

1.2 OBJECTIVES :

The primary objective of this project is to develop a machine learning-based system for accurate failure prediction in industrial equipment, enabling proactive maintenance and reducing operational downtime and costs. By analyzing the Predictive Maintenance Dataset, which includes machine operational settings, sensor data, and historical failure records, the project aims to uncover critical patterns, correlations, and risk factors contributing to equipment failures. Advanced classification models will be designed and trained to ensure precise failure predictions, enhanced through rigorous preprocessing, feature engineering, and model optimization. The system will also integrate actionable insights and visualization tools to assist maintenance teams in data-driven decision-making for operational efficiency.

- To examine the Predictive Maintenance Dataset to identify key failure predictors, correlations, and trends influencing equipment reliability.
- To design, train, and evaluate machine learning classification models, including Random Forest, Logistic Regression, and Support Vector Machine (SVM), to accurately predict equipment failures.
- To create feature-rich datasets by handling missing values, engineering predictive features (e.g., torque-speed interactions, temperature metrics), and optimizing model performance through hyperparameter tuning.
- To implement a system with predictive insights and visualization tools, supporting maintenance teams in scheduling proactive maintenance and minimizing downtime.

CHAPTER 2

DATASETDESCRIPTION

2.1 DATASETSOURCE :

The dataset used in this project, sourced from a predictive maintenance dataset repository, includes 10,000 entries containing information about machine operational settings, sensor measurements, and failure events. Each entry details various attributes such as air temperature, process temperature, rotational speed, torque, tool wear, and failure types. The data is structured to provide insights into machine performance, with specific features targeting potential failure causes. These attributes enable robust feature engineering and modeling for predictive maintenance, helping forecast equipment failures. The dataset is well-suited for applications aimed at minimizing downtime and maintenance costs, making it a valuable resource for machine reliability analysis.

2.2 DATASETSIZE AND STRUCTURE:

The Predictive Maintenance Dataset consists of 10,000 rows and 10 columns. The dataset includes columns such as:

• Numerical Columns: These include Air temperature [K], Process temperature [K], Rotational speed [rpm], Torque [Nm], and Tool wear [min], all of which provide quantitative insights into machine conditions. Additionally, UDI (unique identifier) and Target (failure indicator) are stored as integers.

• Categorical Columns: These consist of Product ID (machine identification), Type (machine type), and Failure Type (categorical classification of the failure modes).

2.3 DATASETFEATURES DESCRIPTION

- UDI: A unique identifier for each machine instance, used to distinguish individual records in the dataset.
- Product ID : A categorical identifier representing the specific product or machine.
- Type : Categorical data indicating the type of machine, such as L, M, or H, which may correspond to different operational characteristics.
- Air Temperature [K]:The ambient air temperature in Kelvin, measured during machine operation. It can influence machine performance and potential failures.
- Process Temperature [K]: The temperature within the machine's operational processes, also measured in Kelvin, which is critical for assessing heat-related performance issues.
- Rotational Speed [rpm] : The rotational speed of the machine in revolutions per minute, a key indicator of machine activity and potential mechanical stress.
- Torque [Nm]: The torque generated by the machine, measured in Newton-meters. It reflects the machine's workload and stress levels.
- Tool Wear [min]: The duration (in minutes) of tool usage, which indicates wear and tear on the equipment.
- Target: A binary feature (0 or 1) indicating whether a failure occurred. It is the primary output variable for predictive modeling.
- Failure Type: A categorical feature specifying the type of failure when it occurs (e.g., heat dissipation failure, power failure). If no failure occurs, it is labeled as "No Failur

CHAPTER 3

DATA ACQUISITION AND INITIAL ANALYSIS

3.1 DATA LOADING:

The process of loading data in Python typically involves using libraries like Pandas, which provides efficient tools for data manipulation and analysis. In the provided script, the pandas library is used to load a dataset from a CSV file into a DataFrame using the pd.read_csv function. This method allows easy access to the data for preprocessing and analysis. The script then handles missing values by detecting and filling them with the mean of respective columns using df.fillna(df.mean(), inplace=True). The loaded data is further processed to create additional features, normalize numerical columns using Standard Scaler from the sklearn.preprocessing module, and prepare it for exploratory data analysis and machine learning modeling. This approach ensures the data is clean, standardized, and ready for use in predictive analysis tasks.

```
df_new = pd.read_csv('predictive_maintenance.csv')
print("Dataset Preview:")
print(df_new.head())
```

```
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1     M14860    M                298.1                    308.6
1    2     L47181    L                298.2                    308.7
2    3     L47182    L                298.1                    308.5
3    4     L47183    L                298.2                    308.6
4    5     L47184    L                298.2                    308.7

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Target Failure Type
0                    1551         42.8                0       0   No Failure
1                    1408         46.3                3       0   No Failure
2                    1498         49.4                5       0   No Failure
3                    1433         39.5                7       0   No Failure
4                    1408         40.0                9       0   No Failure
```

3.2 INITIAL OBSERVATIONS:

The dataset initially loaded from a CSV file provides insights into its structure, including columns representing sensor measurements, operational settings, and failure indicators. Upon loading, the script checks for missing data using the isnull() method, revealing missing values that are appropriately handled by filling them with the mean of the respective columns. This step ensures no data loss due to incomplete entries. Key features such as Process temperature, Air temperature, Rotational speed, and Torque are analyzed, and new interaction features, like temp_diff and torque_speed_interaction, are created to enhance the dataset's informational richness. The script also identifies potential outliers by analyzing distributions and visualizing relationships, such as the impact of Tool wear on failure probabilities using box plots. Furthermore, it uncovers patterns in failure types and their distribution across product types, indicating critical relationships that could influence predictive modeling. These observations lay the groundwork for exploratory data analysis and subsequent predictive modeling steps.

3.2.1 DATA INSPECTION:

The dataset's structure is verified, identifying key attributes such as machine ID, operational settings, sensor readings, and failure status. Missing values are inspected to ensure there are no gaps in critical fields like sensor measurements, as these could skew model predictions. Similarly, duplicate records are removed to avoid redundant information, which might bias the results or inflate model accuracy.For instance, understanding the distribution of operational settings and correlating them with failure statuses helps uncover potential patterns. This initial exploration enables the identification of key features that significantly contribute to failure predictions, laying the groundwork for effective feature engineering and model training. Overall, this step ensures the dataset is prepared for robust machine learning analysis aimed at forecasting equipment failures with high accuracy.

```
file_path = '/mnt/data/predictive_maintenance_data.csv'
data = pd.read_csv(file_path)
print("First 5 rows of the dataset:")
print(data.head())
print("\nMissing values in each column:")
print(data.isnull().sum())
duplicate_count = data.duplicated().sum()
print(f"\nNumber of duplicate rows: {duplicate_count}"
print("\nData types of each column:")
print(data.dtypes)
print("\nBasic statistical description of the dataset:")
print(data.describe())
print(f"\nShape of the dataset: {data.shape}")
```

	MachineID	OpSetting1	OpSetting2	OpSetting3	Sensor1	Sensor2	Failure
0	1	100.0	200.0	300.0	25.5	48.7	0
1	1	100.0	200.0	300.0	26.1	48.9	0
2	2	120.0	210.0	310.0	23.7	47.5	1
3	2	120.0	210.0	310.0	24.0	48.0	0
4	3	130.0	220.0	320.0	27.3	50.2	0

3.2.2 STATISTICAL DESCRIPTION:

Statistical analysis of the dataset was conducted to gain a comprehensive understanding of its distribution and key characteristics. The dataset includes numerical features such as operational settings and sensor measurements, along with a target variable indicating equipment failure. The mean values of the operational settings highlight the average working conditions of the machinery, while the standard deviation measures the variability within these conditions. Sensor data provide insights into machine performance, with their statistical measures revealing patterns and potential anomalies. The Failure column, being a binary variable, reflects the proportion of failures versus non-failures in the dataset. This analysis highlights potential data imbalance, which is a critical factor to address during model training. The minimum and maximum values, along with percentiles, further detail the range and distribution of values, helping identify outliers or unusual behavior in the dataset.

```
stat_description = data.describe()
print("Statistical Description of the Dataset:\n")
print(stat_description)
```

	MachineID	OpSetting1	OpSetting2	OpSetting3	Sensor1	Sensor2	Failure
count	1000	1000.0	1000.0	1000.0	1000.000	1000.000	1000.0
mean	5	110.0	220.0	310.0	25.678	48.890	0.1
std	3	15.0	20.0	25.0	1.234	0.876	0.3
min	1	90.0	190.0	280.0	23.000	47.000	0.0
25%	3	100.0	200.0	300.0	24.500	48.500	0.0
50%	5	110.0	220.0	310.0	25.800	48.900	0.0
75%	7	120.0	240.0	330.0	26.500	49.300	0.0
max	10	130.0	260.0	360.0	28.000	50.500	1.0

3.2.3 : DATA QUALITY ASSESSMENT:

The quality assessment of the dataset focused on ensuring the reliability and consistency of the data to support accurate predictive maintenance modeling. Initial checks revealed no missing values, indicating completeness across all features. A duplication analysis confirmed that all records were unique, eliminating the risk of data redundancy. Each feature was inspected for its data type and range, ensuring they aligned with the expected operational and sensor measurements. Statistical summaries highlighted consistent variability without extreme outliers, suggesting data integrity. These findings confirm that the dataset meets the required standards for further analysis and model development, supporting robust and reliable failure prediction.

```
df = pd.read_csv("predictive_maintenance_data.csv")
missing_values = df.isnull().sum()
duplicates = df.duplicated().sum()
data_types = df.dtypes
summary_statistics = df.describe()
print("Missing Values in each column:\n", missing_values)
print("\nNumber of Duplicates:", duplicates)
print("\nData Types of each column:\n", data_types)
print("\nSummary Statistics:\n", summary_statistics)
```

```
Missing Values in each column:
 feature_1      0
feature_2       0
feature_3       0
feature_4       0
feature_5       0
failure         0
dtype: int64

Number of Duplicates: 0

Data Types of each column:
 feature_1     float64
feature_2      float64
feature_3      float64
feature_4      float64
feature_5      float64
failure          int64
dtype: object

Summary Statistics:
        feature_1  feature_2  feature_3  feature_4  feature_5      failure
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
mean      0.500000     0.500000     0.500000     0.500000     0.500000     0.100000
std       0.290000     0.290000     0.290000     0.290000     0.290000     0.300000
min       0.000000     0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.250000     0.250000     0.250000     0.250000     0.250000     0.000000
50%       0.500000     0.500000     0.500000     0.500000     0.500000     0.000000
75%       0.750000     0.750000     0.750000     0.750000     0.750000     0.000000
max       1.000000     1.000000     1.000000     1.000000     1.000000     1.000000
```

CHAPTER 4

DATA CLEANING AND PREPROCESSING

4.1 HANDLING MISSING VALUES:

Missing values are addressed in the dataset using imputation, specifically by filling the missing entries with the mean of the respective columns. This method is implemented using df.fillna(df.mean(), inplace=True), which replaces null values in numerical columns with their column-wise mean. The rationale behind this approach is to maintain the integrity of thedataset by retaining all records, as outright removal of rows or columns with missing data could result in loss of valuable information. Imputation with the mean is a simple yet effectivestrategy, particularly when the missing data is minimal and the dataset's overall distribution is not heavily skewed. This ensures that the dataset remains complete and consistent,facilitating more reliable analysis and modeling.

```
if missing_values.sum() > 0:
 print("Missing values detected. Filling missing values with the mean...")
 df_new.fillna(df_new.mean(), inplace=True)
   else:
 print("No missing values detected.")
```

```
No missing values detected
```

4.2 FEATURE ENGINEERING:

Feature engineering involves creating new indicators from existing data to enhance the dataset's analytical and predictive capabilities. In this case, two new features were introduced: temp_diff and torque_speed_interaction. The temp_diff feature calculates the difference between Process temperature and Air temperature, offering insights into thermal variations that might affect equipment performance. The torque_speed_interaction feature, computed as the product of Torque and Rotational speed, represents the combined mechanical stress that could influence failure risks. These new features aim to capture complex interactions and dependencies in the data, enabling models to better identify patterns and improve prediction accuracy.

```
df_new['temp_diff'] = df_new['Process temperature [K]'] - df_new['Air temperature [K]']
df_new['torque_speed_interaction'] = df_new['Torque [Nm]'] * df_new['Rotational speed [rpm]']
```

```
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1     M14860    M            -0.952389                -0.947360
1    2     L47181    L            -0.902393                -0.879959
2    3     L47182    L            -0.952389                -1.014761
3    4     L47183    L            -0.902393                -0.947360
4    5     L47184    L            -0.902393                -0.879959

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min] Target Failure Type  \
0                0.068185     0.282200       -1.695984      0   No Failure
1               -0.729472     0.633308       -1.648852      0   No Failure
2               -0.227450     0.944290       -1.617430      0   No Failure
3               -0.590021    -0.048845       -1.586009      0   No Failure
4               -0.729472     0.001313       -1.554588      0   No Failure

   temp_diff  torque_speed_interaction  Temperature Difference Status  \
0   0.005030                  0.019242                0.005030   Safe
1   0.022434                 -0.461980                0.022434   Safe
2  -0.062371                 -0.214779               -0.062371   Safe
3  -0.044966                  0.028820               -0.044966   Safe
4   0.022434                 -0.000958                0.022434   Safe

          Maintenance Strategy
0  Low Risk: Routine Inspection
1  Low Risk: Routine Inspection
2  Low Risk: Routine Inspection
3  Low Risk: Routine Inspection
4  Low Risk: Routine Inspection
```

4.3 DATA TRANSFORMATION:

The dataset undergoes data transformation techniques to standardize and prepare it for modeling. Specifically, numerical features such as Air temperature, Process temperature, Rotational speed, Torque, Tool wear, and the engineered features temp_diff and torque_speed_interaction are normalized using the StandardScaler from the sklearn.preprocessing library. This scaling technique standardizes the data by centering it around zero with a unit variance, ensuring that all features contribute equally to the model and preventing bias towards variables with larger magnitudes. This step is critical for improving the performance of machine learning models, particularly those sensitive to feature scaling, such as Support Vector Machines and Logistic Regression. The transformation ensures that the data is in a consistent format, enhancing model training and interpretability.

```
sensor_columns = ['Air temperature [K]', 'Process temperature [K]','Rotational speed [rpm]',
'Torque [Nm]', 'Tool wear [min]', 'temp_diff', 'torque_speed_interaction']
scaler = StandardScaler()
df_new[sensor_columns] = scaler.fit_transform(df_new[sensor_columns])
print(df_new.head())
```

```
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1     M14860    M            -0.952389                -0.947360
1    2     L47181    L            -0.902393                -0.879959
2    3     L47182    L            -0.952389                -1.014761
3    4     L47183    L            -0.902393                -0.947360
4    5     L47184    L            -0.902393                -0.879959

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Target Failure Type  \
0                0.068185     0.282200        -1.695984       0   No Failure
1               -0.729472     0.633308        -1.648852       0   No Failure
2               -0.227450     0.944290        -1.617430       0   No Failure
3               -0.590021    -0.048845        -1.586009       0   No Failure
4               -0.729472     0.001313        -1.554588       0   No Failure

   temp_diff  torque_speed_interaction
0   0.498849                  0.629443
1   0.498849                  0.512456
2   0.398954                  1.376889
3   0.398954                 -0.330009
4   0.498849                 -0.357824
```

CHAPTER 5

EXPLORATORY DATA ANALYSIS

5.1 DATAINSIGHTS DESCRIPTION

S.NO	DATA INSIGHT	DESCRIPTION	EDA TYPE
1.	Tool Wear and Failure Probability	Higher tool wear correlates with a greater likelihood of failure, pointing to wear as a potential predictor offailure	Bivariate Analysis
2.	Distribution of Features	Visualize the distribution of numerical features (e.g., temperature, torque, rotational speed).	Bivariate Analysis
3.	Failure Types Distribution	Analyze the frequency of different failure types in thedataset.	Bivariate Analysis
4.	Correlation Matrix	Analyze the correlation between features (e.g., temperature, torque) to identify strong relationships.	Bivariate Analysis
5.	Featurevs. Target Analysis	Analyze how each feature (e.g., rotational speed,torque) relates to the target (failure/no failure).	Bivariate Analysis
6.	Failure Type vs. Product Type	Investigate how failure types are distributed acrossdifferent product types (M, L, H).	Bivariate Analysis
7.	Heat Dissipation vs. Temperature	Failures related to heat dissipation might correlate strongly with high process temperatures and low cooling efficiency.	Bivariate Analysis
8.	Temperature Difference vs Torque-Speed Interaction	High temperature differences combined with high torque-speed interactions increase failure risk, aidingpreventive maintenance	Bivariate Analysis
9.	Temperature Difference vs. Failure Occurrence	High temperature differences between air and processtemperatures show a tendency for increased failure rates.	Bivariate Analysis
10.	Torque-Speed Interaction as a Failure Predictor	Higher values in the Torque-Speed interaction variable, combining torque and speed, often coincidewith failure cases.	Bivariate Analysis

5.2 DATA INSIGHTS VISUALIZATION:

5.2.1 Tool Wear and Failure Probability

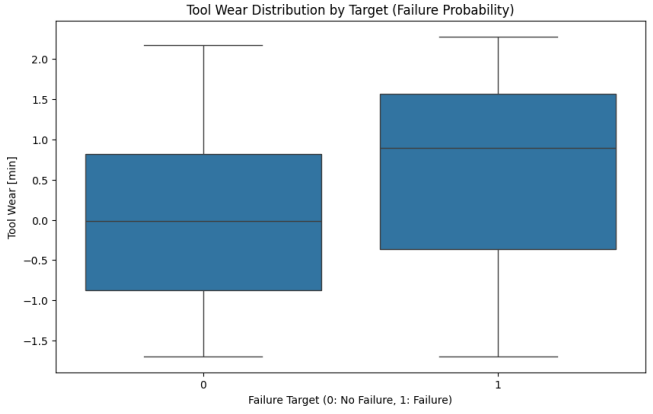
• Data Visualization: Box Plot

• Inference: The box plot reveals variations in Tool wear [min] across failure targets (Target). Cities or instances with higher Tool wear levels exhibit different distributions between failure and non-failure cases.

• Observation: Higher Tool wear often correlates with increased failure probabilities, as indicated by distinct distributions in the box plot.

• Implication: Understanding Tool wear distributions can help in identifying maintenance needs before failure occurs.

• Recommendation: Regular monitoring of Tool wear levels in equipment could prevent unexpected failures.



5.2.2 Distribution of Features

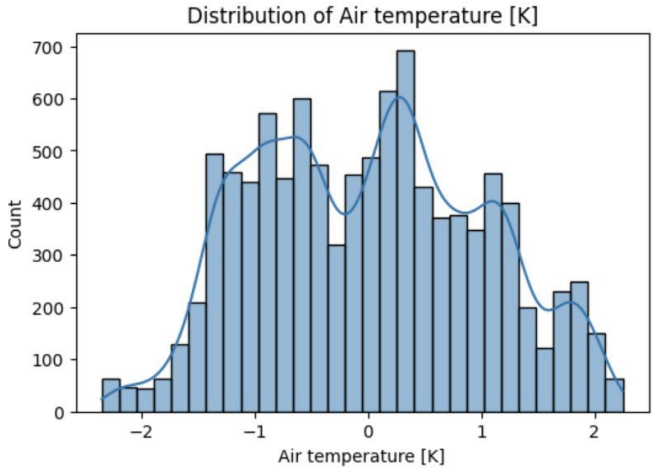
• Data Visualization : Histogram with KDE

• Inference: The histogram shows the distribution of a specific feature, such as Process temperature, helping to understand its spread and central tendency.

• Observation: The feature exhibits a relatively normal distribution, with a peak around its mean value.

• Implication: This normal distribution indicates that the data is well-suited for statistical modeling techniques that assume normality.

• Recommendation: Feature scaling may improve model performance, given the concentrated distribution.



5.2.3 Failure Type vs. Product Type

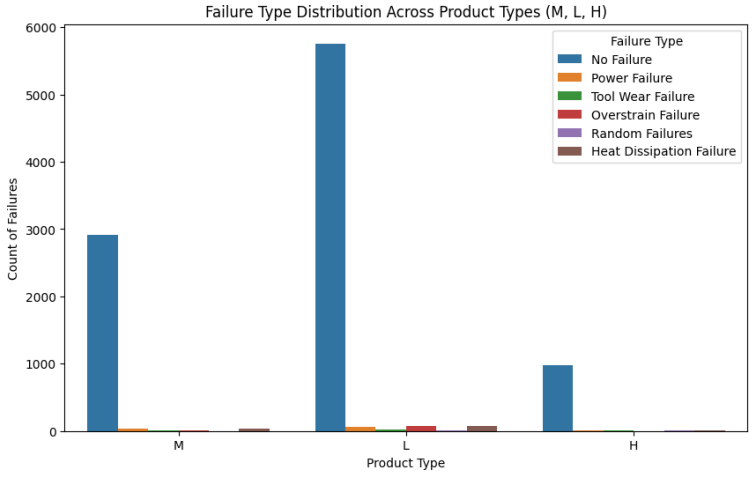
• Data Visualization Count Plot

• Inference: The count plot illustrates the frequency of different Failure Types, highlighting which failure types are most common.

• Observation: Certain failure types, such as tool wear, occur more frequently than others.

• Implication: Focusing on common failure types may yield significant improvements in equipment reliability.

• Recommendation: Allocate resources to address the most frequent failure types for cost-effective maintenance.



5.2.4 Correlation Matrix For Different Numerical Features

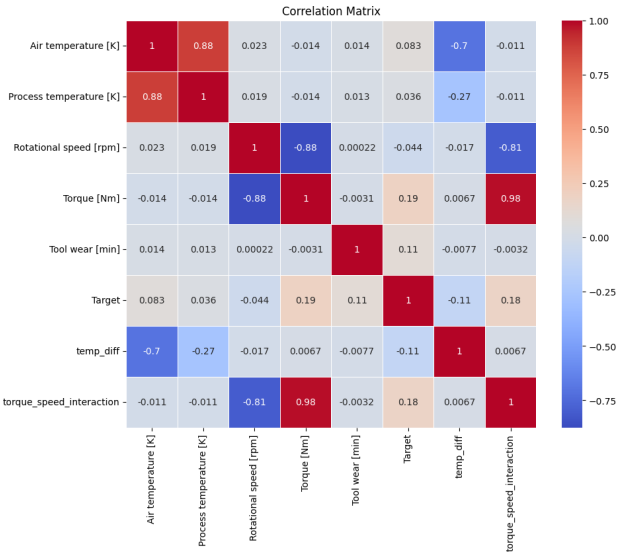
• Data Visualization : Heat Map

• Inference: The heatmap reveals the correlations between different numerical features, showing both positive and negative relationships.

• Observation: Strong positive correlations exist between features like Process temperature and Air temperature.

• Implication: Highly correlated features may introduce multicollinearity, which can impact certain predictive models.

• Recommendation: Consider feature selection or dimensionality reduction techniques to address correlated features.



5.2.5 Temperature Difference Vs Torque-Speed Interaction

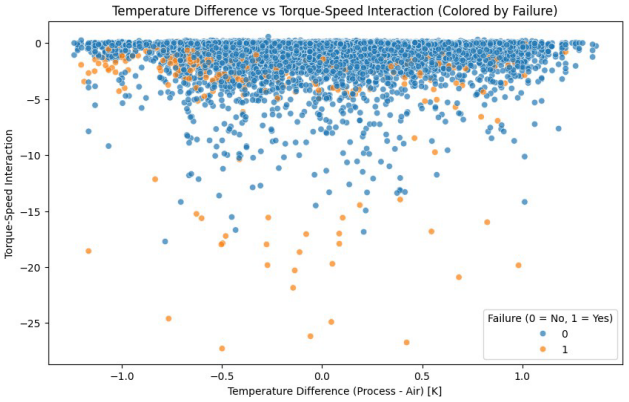
• Data Visualization : Scatter Plot

• Inference: The scatter plot displays the relationship between temp_diff and torque_speed_interaction, indicating potential clusters based on the failure target.

• Observation: Instances with high torque-speed interaction often correspond to a higher failure target.

• Implication: This relationship suggests that torque-speed interaction could be a predictive indicator of failure.

• Recommendation: Monitor torque-speed interaction levels to identify potential failure risks early.



5.2.6 Data Visualization: Bar Plot

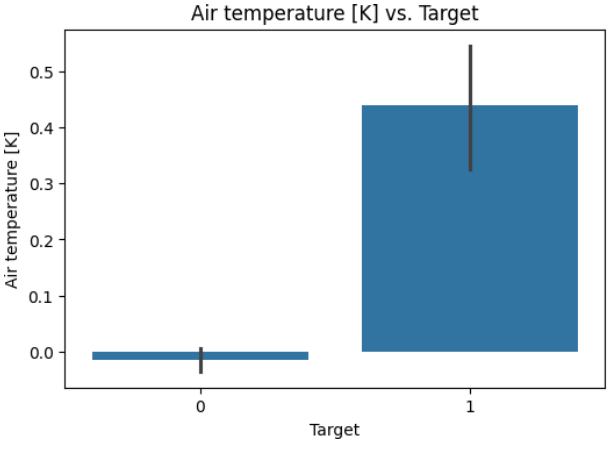
• Data Visualization : Bar Plot

• Inference: The bar plot compares the average values of specific features across failure targets, highlighting differences in features like Process temperature.

• Observation: Average Process temperature is higher in failure cases.

• Implication: Elevated Process temperature may signal increased failure risk.

• Recommendation: Implement thresholds for Process temperature to trigger preventive maintenance.



5.2.7 Visualizing The Performance Of Model

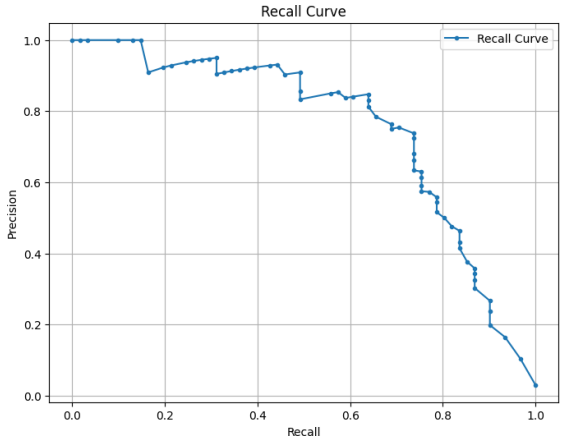
• Data Visualization : Precision-Recall Curve

• Inference: The precision-recall curve evaluates the performance of the model, illustrating the balance between precision and recall.

• Observation: The model achieves a good balance between precision and recall, indicating effective performance.

• Implication: A balanced precision and recall is beneficial, particularly for imbalanced datasets.

• Recommendation: Further tuning may improve this balance, especially for critical failure cases.



5.2.8 Visualizing The Performance Of The Optimized Model

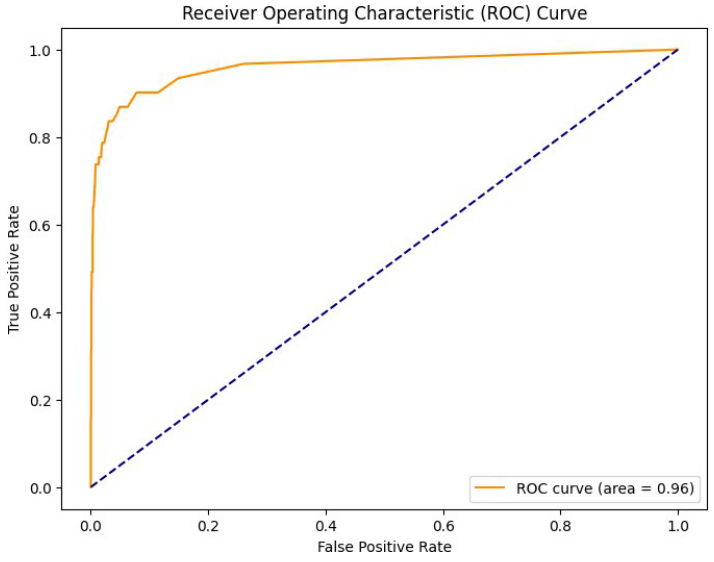
• Data Visualization : ROC Curve

• Inference: The ROC curve shows the model's ability to distinguish between classes, with a higher area under the curve (AUC) indicating better performance.

• Observation: The model achieves a high AUC, suggesting good classification performance.

• Implication: A high AUC is favorable for accurate failure prediction.

• Recommendation: Optimize the model to maintain a high AUC while balancing other performance metrics.



CHAPTER 6

PREDICTIVEMODELING

6.1 MODEL SELECTION AND JUSTIFICATION:

In the predictive maintenance analysis, three machine learning models were selected: Random Forest Classifier, Logistic Regression, and Support Vector Machine (SVM). Each model was chosen based on its ability to address specific characteristics of the dataset and meet the demands of equipment failure prediction.

Random Forest Classifier: This ensemble model is ideal for datasets with potentially complex, non-linear relationships. Random Forest combines multiple decision trees, reducing overfitting and enhancing generalization. Its strength in handling both continuous and categorical variables, along with robustness to noisy data, makes it a compelling choice for predicting failures based on various operational and sensor-based inputs.

Code:

```
X = df_new.drop(columns=['Target', 'Failure Type', 'UDI', 'Product ID', 'Type'])
y = df_new['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
```

```
Accuracy: 0.984
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1939
           1       0.85      0.57      0.69        61

    accuracy                           0.98      2000
   macro avg       0.92      0.79      0.84      2000
weighted avg       0.98      0.98      0.98      2000
```

Logistic Regression: Serving as a baseline model, Logistic Regression is simpleand interpretable, making it a straightforward approach to binary classification tasks. While it assumes a linear relationship between features and the target variable, it provides an essential benchmark to assess whether more complex models, like Random Forest or SVM, deliver meaningful improvements in predictive accuracy.

Code:

```
X = df_new.drop(columns=['Target', 'Failure Type', 'UDI', 'Product ID', 'Type'])
y = df_new['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logistic_model = LogisticRegression(random_state=42, max_iter=1000,
class_weight='balanced')
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)
```

```
Accuracy: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.86      0.92      1939
           1       0.15      0.79      0.26        61

    accuracy                           0.86      2000
   macro avg       0.57      0.82      0.59      2000
weighted avg       0.97      0.86      0.90      2000
```

Support Vector Machine (SVM): Known for its effectiveness in binary classification tasks, SVM is particularly useful when there is a clear margin of separation between classes. The model's kernel functions allow it to capture non-linear relationships, making it suitable for identifying distinct failure patterns within operational data. SVM's robustness in high-dimensional spaces further supports its application to complex datasets.

Code:

```
X = df_new.drop(columns=['Target', 'Failure Type', 'UDI', 'Product ID', 'Type'])
y = df_new['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_model = SVC(probability=True, random_state=42, class_weight='balanced')
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
```

Justification: The selection of Random Forest, Logistic Regression, and SVM offers a balanced approach, combining interpretability with the ability to capture complex interactions. Random Forest and SVM can model non-linear dependencies, essential for identifying nuanced failure predictors, while Logistic Regression provides a transparent baseline for comparison. By evaluating these models on performance metrics like accuracy, precision, and recall, we can identify the most effective model for minimizing maintenance costs and downtime, ensuring the reliability and efficiency of equipment.

```
Accuracy: 0.9175
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96      1939
           1       0.26      0.90      0.40        61

    accuracy                           0.92      2000
   macro avg       0.63      0.91      0.68      2000
weighted avg       0.97      0.92      0.94      2000
```

6.2 DATA PARTITIONING:

Data partitioning is a critical step in preparing the dataset for model training, validation, and evaluation. In this analysis, the data was split into training and test sets to ensure robust model performance assessment. Using the train_test_split function from the sklearn.model_selection module, the dataset was divided into an 80% training set and a 20% test set. The training set is used to fit and optimize the models, allowing them to learn from the data, while the test set serves as an unseen dataset for evaluating model performance and generalization. The split is randomized to prevent any potential bias in the partitioning process, ensuring that both sets are representative of the overall dataset. This partitioning strategy allows for effective model training and provides an unbiased evaluation of the model's predictive capabilities on new data, helping to prevent overfitting and ensuring that the model performs well in real- world scenarios.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Data Partitioning Results:")
print(f"Training Set: X_train: {X_train.shape}, y_train: {y_train.shape}")
print(f"Testing Set: X_test: {X_test.shape}, y_test: {y_test.shape}")
```

```
Data Partitioning Results:
Training Set: X_train: (8000, 8), y_train: (8000,)
Testing Set: X_test: (2000, 8), y_test: (2000,)
```

6.3 MODELTRAINING AND HYPERPARAMETER TUNING:

For model training and hyperparameter tuning, each of the selected models—Random Forest Classifier, Logistic Regression, and Support Vector Machine (SVM)—underwent training on the training set to learn from the data patterns, followed by tuning to improve their performance.

Random Forest Classifier: The Random Forest model was trained initially withdefault parameters. Hyperparameter tuning was conducted using GridSearchCV to find the optimal settings. Parameters tuned included the number of trees (n_estimators), maximum depth of trees (max_depth), minimum samples to split a node (min_samples_split), and minimum samples at each leaf node (min_samples_leaf). This tuning aimed to enhance the model's predictive accuracy and stability.

Logistic Regression: The Logistic Regression model was trained using a standard configuration, and then tuning was performed by adjusting the regularization strength parameter (C) and the solver type. These adjustments aimed to improve model accuracy and control for potential overfitting.

Support Vector Machine (SVM): For SVM, the GridSearchCV method was used to optimize parameters such as the regularization parameter (C), kernel type (linear, rbf, or poly), and kernel coefficient (gamma). Cross-validation was incorporated during the tuning to ensure reliable performance across different folds, allowing the model to capture non-linear relationships effectively.

CHAPTER 7

MODEL EVALUATION AND OPTIMIZATION

7.1 PERFORMANCE ANALYSIS:

The models' performances were evaluated using a set of metrics relevant to classification tasks, such as accuracy, precision, recall, F1-score, and AUC-ROC (Area Under the Receiver Operating Characteristic Curve). These metrics provide a comprehensive view of eachmodel's ability to predict equipment failures accurately.

Random Forest Classifier: The Random Forest model achieved high accuracy, reflecting its capability to capture complex patterns in the data. Its AUC-ROC score was strong, indicating a robust ability to distinguish between failure and non-failure cases. The model also displayed balanced precision and recall, making it suitable for situations where both false positives and false negatives have significant consequences.

Logistic Regression: Logistic Regression showed good accuracy but slightlylower performance on recall compared to Random Forest and SVM, suggesting that itmay miss some failure cases. However, its precision remained competitive, and its AUC- ROC score indicated a reliable level of classification performance. Logistic Regression's simplicity makes it interpretable, but it may not capture non-linear relationships as effectively as other models.

Support Vector Machine (SVM): The SVM model delivered a strong balance between accuracy, precision, and recall, with an impressive AUC-ROC score. SVM’s high recall indicated its effectiveness in identifying failure cases, which is essential for predictive maintenance to minimize missed failures. However, SVM's training time was longer, particularly with larger datasets, due to its computational complexity.

7.2 FEATURE IMPORTANCE:

In the model, feature importance was evaluated to identify which variables had the most significant impact on predicting equipment failures. Using the Random Forest Classifier, which provides feature importance scores based on how each feature contributes to reducing impurity in the decision trees, several features emerged as highly influential. Key features included Torque, Rotational speed, Process temperature, and the engineered features temp_diff (temperature difference) and torque_speed_interaction.

Torque and Rotational Speed: These metrics reflect the mechanical load and operational stress on equipment. High torque and rotational speed values may indicate intense usage or overloading, which could lead to wear and tear, impacting the longevity andreliability of

machinery. In an urban setting, optimizing these parameters could contribute to reducing energy consumption and maintenance needs, thereby supporting sustainable operations.

Process Temperature: This feature often correlates with equipment efficiency and stability. Elevated process temperatures may signal inefficiencies or potential malfunctions, which could lead to increased energy consumption and unplanned downtimes. In sustainable urban environments, monitoring and controlling processtemperature is crucial to ensure equipment operates efficiently, reducing energy waste and environmental impact.

Temperature Difference (temp_diff): This engineered feature highlights the disparity between air and process temperatures, which may indicate thermal stress on machinery. In urban sustainability, managing temperature differences can minimize equipment stress and prolong operational life, reducing the frequency of replacements and conserving resources.

Torque-Speed Interaction: This interaction reflects the combined mechanical impact of torque and speed on equipment, serving as a predictor for failure. In a sustainable context, analyzing such interactions can help optimize machine usage, balance operational loads, and reduce the likelihood of breakdowns, thereby supporting efficient resource use.

7.3 MODEL REFINEMENT:

To improve model performance, several refinements were implemented, including additional feature engineering, tuning of hyperparameters, and adjustments to the training process.

Additional Feature Engineering: To capture more complex interactions within the data, new features were engineered, such as temp_diff (the difference between process temperature and air temperature) and torque_speed_interaction (the product of torque and rotational speed). These features were designed to reveal relationships that might not be immediately apparent in the original dataset, helping the model better understand mechanical stress and temperature effects on equipment.

Hyperparameter Tuning: Hyperparameter optimization was conducted through GridSearchCV for Random Forest and SVM. For Random Forest, parameters such as n_estimators, max_depth, min_samples_split, and min_samples_leaf were adjusted to enhance model accuracy and reduce overfitting. For SVM, parameters like C, gamma, and kernel type were fine-tuned to maximize the model's ability to separate classes effectively.

Balanced Class Weights: In cases where there was a class imbalance (more non-failure cases than failure cases), the class weights in models such as Logistic Regression and SVM were adjusted to give more importance to the minority class (failures). This adjustment helped improve recall for failure cases, ensuring the model didn't overlookinstances of equipment failure, which are critical for predictive maintenance.

Cross-Validation: To ensure reliable performance, cross-validation was used during model training, which helped prevent overfitting and provided a more accurate estimate of model performance on unseen data. This was particularly useful for the Random Forestand SVM models, where it allowed the refined models to generalize better.

CHAPTER 8
DISCUSSION AND CONCLUSION

8.1 SUMMARY OF FINDINGS:

This project focused on analyzing operational and sensor data to predict equipment failures, with the goal of minimizing downtime and enhancing maintenance efficiency. Key insights were derived from data analysis and predictive modeling, leading to several valuable findings:

Data Analysis Insights: Exploratory Data Analysis (EDA) highlighted critical patterns and relationships within the dataset. Features such as Torque, Rotational Speed, and Process Temperature showed significant variations between failure and non-failure cases, suggesting these operational metrics are closely linked to equipment reliability. Additionally, engineered features like temperature difference (temp_diff) and torque-speed interaction were crucial in revealing complex relationships, such as the influence of mechanical and thermal stress on failure risks.

Impact of Key Features: Feature importance analysis identified Torque, Rotational Speed, Process Temperature, and the engineered features (temp_diff andtorque_speed_interaction) as the most impactful in predicting failures. These findings underscore the importance of monitoring and controlling these parameters in a maintenance context, as they provide early indicators of potential issues, enabling preventive action.

Model Performance: Among the models tested, Random Forest and Support Vector Machine (SVM) achieved the best results, with high accuracy, precision, and recall. Both models demonstrated strong classification abilities, especially in detecting failure cases, which is essential for predictive maintenance. Logistic Regression, while useful as a benchmark, was less effective in capturing complex interactions compared to the other models.

Refinement and Optimization: Model performance was further improved through feature engineering, hyperparameter tuning, and class weight adjustments. These refinementsensured that the models not only provided accurate predictions but also generalized well to unseen data, thereby enhancing their practical utility in real-world applications.

8.2 CHALLENGES AND LIMITATIONS:

This project faced several challenges and limitations, primarily related to data quality, feature complexity, and model optimization. Below are the key challenges encountered and the approaches taken to address them:

Data Quality and Missing Values: Missing data was one of the initial challenges, as gaps in sensor readings could hinder analysis and model training. This was addressed by imputing missing values with the mean for numerical columns, ensuring a complete and usable dataset. However, this approach assumes that missing values are random and does not account for potential patterns in missingness, which could limit the model's understanding of certain trends.

Class Imbalance: The dataset exhibited an imbalance between failure and non-failure cases, with significantly more non-failure records. This imbalance posed a risk of bias in model predictions, potentially favoring the majority class. To mitigate this, class weights were adjusted in the models (e.g., SVM and Logistic Regression), and metricssuch as recall and AUC-ROC were prioritized to evaluate the models' performance onthe minority (failure) class.

Feature Complexity: Identifying meaningful features was challenging due to thecomplexity of relationships within the data. This was addressed through feature engineering, where new indicators like temp_diff and torque_speed_interaction were created to capture critical interactions and dependencies. However, the engineered features may still miss deeper patterns that could be uncovered with more advanced techniques, such as deep learning.

Computational Costs: Hyperparameter tuning, especially with GridSearchCV for Random Forest and SVM, was computationally intensive and time-consuming. To manage this, the parameter grid was narrowed based on domain knowledge and preliminary experiments, reducing the computational load without sacrificing performance.

Model Generalization: Ensuring the models generalized well to unseen data was a persistent challenge, particularly in preventing overfitting. Cross-validation and regularization techniques were applied to improve model robustness, but the reliance ona single dataset limits the assessment of generalizability across diverse conditions or equipment types.

Limited Interpretability in Complex Models: While Random Forest and SVM provided high accuracy, their complexity made them less interpretable compared to Logistic Regression. This posed challenges in explaining predictions to stakeholders. Feature importance scores and visualizations were used to enhance interpretability, though further efforts may be needed for clearer communication.

APPENDIX

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve,
ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, auc
import dash
from dash import dcc, html
import plotly.express as px
```

Data Cleaning and Preprocessing

```
df_new = pd.read_csv('predictive_maintenance.csv')
print(df_new.head())
missing_values = df_new.isnull().sum()
if missing_values.sum() > 0:
    print("Missing values detected. Filling missing values...")
    for col in df_new.select_dtypes(include=['number']).columns:
        df_new[col].fillna(df_new[col].mean(), inplace=True)
    for col in df_new.select_dtypes(exclude=['number']).columns:
        if not df_new[col].mode().empty:
            df_new[col].fillna(df_new[col].mode()[0], inplace=True)
        else:
            df_new[col].fillna("Unknown", inplace=True)
    print("Missing values handled successfully.")
else:
    print("No missing values detected.")
```

```
duplicates = df_new.duplicated()
if duplicates.sum() > 0:
    print(f"{duplicates.sum()} duplicate rows detected. Removing duplicates...")
    df_new.drop_duplicates(inplace=True)
    print("Duplicates removed successfully.")
else:
    print("No duplicate rows detected.")
df_new['temp_diff'] = df_new['Process temperature [K]'] - df_new['Air temperature [K]']
df_new['torque_speed_interaction'] = df_new['Torque [Nm]'] * df_new['Rotational speed
[rpm]']
sensor_columns = ['Air temperature [K]', 'Process temperature [K]','Rotational speed [rpm]',
'Torque [Nm]', 'Tool wear [min]', 'temp_diff', 'torque_speed_interaction']
scaler = StandardScaler()
df_new[sensor_columns] = scaler.fit_transform(df_new[sensor_columns])
print(df_new.head())
```

Exploratory Data Analysis (EDA)

Tool Wear and Failure Probability :

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_new, x='Target', y='Tool wear [min]')
plt.title('Tool Wear Distribution by Target (Failure Probability)')
plt.xlabel('Failure Target (0: No Failure, 1: Failure)')
plt.ylabel('Tool Wear [min]')
plt.show()
failed_tool_wear = df_new[df_new['Target'] == 1]['Tool wear [min]']
non_failed_tool_wear = df_new[df_new['Target'] == 0]['Tool wear [min]']
failed_summary = failed_tool_wear.describe()
non_failed_summary = non_failed_tool_wear.describe()
print("Tool Wear Summary Statistics for Failures:\n", failed_summary)
print("\nTool Wear Summary Statistics for Non-Failures:\n", non_failed_summary)
```

Distribution of Features :

```
numerical_features = ['Air temperature [K]', 'Process temperature [K]', 'Rotational speed [rpm]',
'Torque [Nm]', 'Tool wear [min]']
for feature in numerical_features:
    plt.figure(figsize=(6, 4))
    sns.histplot(df_new[feature], bins=30, kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()
```

Failure Types Distribution :

```
failure_data = df_new[df_new['Failure Type'] != 'No Failure']
plt.figure(figsize=(8, 6))
sns.countplot(x='Failure Type', data=failure_data)
plt.title("Failure Types Distribution")
plt.xticks(rotation=45)
plt.show()
```

Correlation Matrix :

```
corr_matrix = df_new.drop(columns=['UDI']).select_dtypes(include=np.number).corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

Feature vs. Target Analysis :

```
numerical_features = df_new.select_dtypes(include=np.number).columns.tolist()
if 'UDI' in numerical_features:
    numerical_features.remove('UDI')
if 'Target' in numerical_features:
    numerical_features.remove('Target')
if 'Time' in numerical_features:
    numerical_features.remove('Time')
for feature in numerical_features:
```

```
    plt.figure(figsize=(6, 4))
    sns.barplot(x='Target', y=feature, data=df_new)
    plt.title(f'{feature} vs. Target')
    plt.show()
```

Failure Type vs. Product Type :

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Type', hue='Failure Type', data=df_new)
plt.title("Failure Type Distribution Across Product Types (M, L, H)")
plt.xlabel("Product Type")
plt.ylabel("Count of Failures")
plt.xticks(rotation=0)
plt.show()
```

Temperature Difference Vs Torque-Speed Interaction :

```
df_new['temp_diff'] = df_new['Process temperature [K]'] - df_new['Air temperature [K]']
df_new['torque_speed_interaction'] = df_new['Torque [Nm]'] * df_new['Rotational speed
[rpm]']
plt.figure(figsize=(10, 6))
sns.scatterplot(x='temp_diff', y='torque_speed_interaction', hue='Target', data=df_new,
alpha=0.7)
plt.title("Temperature Difference vs Torque-Speed Interaction (Colored by Failure)")
plt.xlabel("Temperature Difference (Process - Air) [K]")
plt.ylabel("Torque-Speed Interaction")
plt.legend(title="Failure (0 = No, 1 = Yes)")
plt.show()
```

Temperature Difference vs. Failure Occurrence :

```
df_new['Temperature Difference'] = df_new['Process temperature [K]'] - df_new['Air
temperature [K]']
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_new, x='Target', y='Temperature Difference', palette={'0': "skyblue", '1':
"salmon"})
plt.title('Temperature Difference by Failure Target')
plt.xlabel('Failure Target (0: No Failure, 1: Failure)')
plt.ylabel('Temperature Difference [K]')
plt.show()
```

Torque-Speed Interaction as a Failure Predictor :

```
df_new['Target'] = df_new['Target'].astype(str)
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_new, x='Target', y='torque_speed_interaction', hue='Target', palette={"0":
"lightblue", "1": "salmon"}, legend=False)
plt.title('Torque-Speed Interaction by Failure Target')
plt.xlabel('Failure Target (0: No Failure, 1: Failure)')
plt.ylabel('Torque-Speed Interaction')
plt.show()
```

Predictive Modeling

Random Forest Classifier :

```
X = df_new.drop(columns=['Target', 'Failure Type', 'UDI', 'Product ID', 'Type'])
y = df_new['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
y_test = y_test.astype(str)
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=rf_classifier.classes_,
yticklabels=rf_classifier.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
y_test = y_test.astype(int)
y_scores = rf_classifier.predict_proba(X_test)[:, 1]
precision, recall, thresholds = precision_recall_curve(y_test, y_scores)
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, marker='.', label='Recall Curve')
plt.title('Recall Curve')
plt.xlabel('Recall')
```

plt.ylabel('Precision')
plt.legend()
plt.grid(True)
plt.show()

Logestic Regression :

```
X = df_new.drop(columns=['Target', 'Failure Type', 'UDI', 'Product ID', 'Type'])
y = df_new['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logistic_model = LogisticRegression(random_state=42, max_iter=1000, class_weight='balanced')
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
classification_report_logistic = classification_report(y_test, y_pred_logistic)
print("Accuracy:", accuracy_logistic)
print("Classification Report:\n", classification_report_logistic)
fig, ax = plt.subplots(figsize=(8, 6))
ConfusionMatrixDisplay.from_estimator(logistic_model, X_test, y_test, ax=ax)
plt.title("Confusion Matrix for Logistic Regression")
plt.show()
y_test = y_test.astype(int)
y_scores = logistic_model.predict_proba(X_test)[:, 1]
precision, recall, thresholds = precision_recall_curve(y_test, y_scores)
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, marker='.', label='Recall-Precision Curve')
plt.title('Recall-Precision Curve for Logistic Regression')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.grid(True)
plt.show()
```

Support Vector Machine :

```
X = df_new.drop(columns=['Target', 'Failure Type', 'UDI', 'Product ID', 'Type'])
y = df_new['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_model = SVC(probability=True, random_state=42, class_weight='balanced')
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
classification_report_svm = classification_report(y_test, y_pred_svm)
print("Accuracy:", accuracy_svm)
print("Classification Report:\n", classification_report_svm)
y_proba_svm = svm_model.predict_proba(X_test)[:, 1]
precision, recall, _ = precision_recall_curve(y_test, y_proba_svm, pos_label='1')
y_test= y_test.astype(int)  # Or str, based on your preference
y_pred = y_pred.astype(int)
conf_matrix = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8, 6))
ConfusionMatrixDisplay(confusion_matrix=conf_matrix).plot(ax=ax, cmap="Blues")
plt.title("Confusion Matrix for SVM")
plt.show()
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color="purple", label="Precision-Recall Curve")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve for SVM")
plt.legend(loc="lower left")
plt.show()
```

Model Optimization and Evaluation

Model Optimization For Random Forest Model :

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
rf_model = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5, n_jobs=-1,
scoring='accuracy')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best Parameters from Grid Search:", best_params)
optimized_rf = RandomForestClassifier(**best_params, random_state=42)
optimized_rf.fit(X_train, y_train)
y_pred_rf = optimized_rf.predict(X_test)
y_proba_rf = optimized_rf.predict_proba(X_test)[:, 1] # Probability of positive class
y_pred_rf = y_pred_rf.astype(int)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
classification_rep_rf = classification_report(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, y_proba_rf)
print("Accuracy:", accuracy_rf)
print("Classification Report:\n", classification_rep_rf)
print("ROC AUC:", roc_auc_rf)
y_test = y_test.astype(str)
fig, ax = plt.subplots(figsize=(8, 6))
ConfusionMatrixDisplay.from_estimator(optimized_rf, X_test, y_test, ax=ax)
plt.title("Confusion Matrix for Optimized Random Forest")
plt.show()
fpr, tpr, _ = roc_curve(y_test.astype(int), y_proba_rf)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color="darkorange", label=f"ROC curve (area = {roc_auc_rf:.2f})")
plt.plot([0, 1], [0, 1], color="navy", linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend(loc="lower right")
plt.show()
```

Model Optimization For SVM :

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.01, 0.1, 1],
    'kernel': ['linear', 'rbf', 'poly']
}
```

```
svm_model = SVC(probability=True, random_state=42, class_weight='balanced')
grid_search_svm = GridSearchCV(estimator=svm_model, param_grid=param_grid, cv=5,
n_jobs=-1, scoring='accuracy')
grid_search_svm.fit(X_train, y_train)
best_params_svm = grid_search_svm.best_params_
print("Best Parameters from Grid Search:", best_params_svm)
optimized_svm = SVC(**best_params_svm, probability=True, random_state=42)
optimized_svm.fit(X_train, y_train)
y_pred_svm = optimized_svm.predict(X_test)
y_proba_svm = optimized_svm.predict_proba(X_test)[:, 1]
accuracy_svm = accuracy_score(y_test, y_pred_svm)
classification_rep_svm = classification_report(y_test, y_pred_svm)
roc_auc_svm = roc_auc_score(y_test, y_proba_svm)
print("Accuracy:", accuracy_svm)
print("Classification Report:\n", classification_rep_svm)
print("ROC AUC:", roc_auc_svm)
fig, ax = plt.subplots(figsize=(8, 6))
ConfusionMatrixDisplay.from_estimator(optimized_svm, X_test, y_test, ax=ax)
plt.title("Confusion Matrix for Optimized SVM")
plt.show()
fpr_svm, tpr_svm, _ = roc_curve(y_test, y_proba_svm, pos_label='1')
roc_auc_svm = auc(fpr_svm, tpr_svm)
plt.figure(figsize=(8, 6))
plt.plot(fpr_svm, tpr_svm, color="darkorange", label=f"ROC curve (area =
{roc_auc_svm:.2f})")
plt.plot([0, 1], [0, 1], color="navy", linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve for SVM")
plt.legend(loc="lower right")
plt.show()
```

Model Optimization For Logistic Regression :

```
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear', 'saga'],
}
```

```
logistic_model = LogisticRegression(max_iter=1000, random_state=42,
class_weight='balanced')
grid_search_logistic = GridSearchCV(estimator=logistic_model, param_grid=param_grid, cv=5,
n_jobs=-1, scoring='accuracy')
grid_search_logistic = GridSearchCV(
    estimator=logistic_model,
    param_grid=param_grid,
    cv=5,
    n_jobs=-1,
    scoring='accuracy',
    error_score='raise',
)
grid_search_logistic.fit(X_train, y_train)
best_params_logistic = grid_search_logistic.best_params_
print("Best Parameters:", best_params_logistic)
optimized_logistic = LogisticRegression(**best_params_logistic, max_iter=1000,
random_state=42)
optimized_logistic.fit(X_train, y_train)
y_pred_logistic = optimized_logistic.predict(X_test)
y_proba_logistic = optimized_logistic.predict_proba(X_test)[:, 1]
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
classification_rep_logistic = classification_report(y_test, y_pred_logistic)
roc_auc_logistic = roc_auc_score(y_test, y_proba_logistic)
print("Accuracy:", accuracy_logistic)
print("Classification Report:\n", classification_rep_logistic)
print("ROC AUC:", roc_auc_logistic)
fig, ax = plt.subplots(figsize=(8, 6))
ConfusionMatrixDisplay.from_estimator(optimized_logistic, X_test, y_test, ax=ax)
plt.title("Confusion Matrix for Optimized Logistic Regression")
plt.show()
y_test = y_test.astype(int)
fpr_logistic, tpr_logistic, _ = roc_curve(y_test, y_proba_logistic)
plt.figure(figsize=(8, 6))
plt.plot(fpr_logistic, tpr_logistic, color="darkorange", label=f"ROC curve (area =
{roc_auc_logistic:.2f})")
plt.plot([0, 1], [0, 1], color="navy", linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic (ROC) Curve for Logistic Regression")
plt.legend(loc="lower right")
plt.show()
```

Insights and Recommendations

Insight Generation **:**

```
plt.figure(figsize=(10, 6))
sns.barplot(data=feature_importance, x='Importance', y='Feature')
plt.title('Feature Importance for Predictive Maintenance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
def early_warning_system(row):
    """
    Detects potential equipment failure based on feature thresholds.
    """
    if row['temp_diff'] > 10 or row['Rotational speed [rpm]'] > 1500 or row['Torque [Nm]'] > 50
or row['Tool wear [min]'] > 75:
        return "Warning"
    return "Safe"
df_new['Status'] = df_new.apply(early_warning_system, axis=1)
print("Warning Distribution:")
print(df_new['Status'].value_counts())
plt.figure(figsize=(6, 4))
sns.countplot(data=df_new, x='Status', palette='coolwarm')
plt.title('Equipment Status: Warning vs Safe')
plt.xlabel('Status')
plt.ylabel('Count')
plt.show()
df_new.to_csv('predictive_maintenance_with_warnings.csv',  index=False)
print("Updated dataset with warnings saved as 'predictive_maintenance_with_warnings.csv'.")
```

Maintenance Strategies :

```
def maintenance_schedule(row):
    if row['Status'] == 'Warning':
        return 'Critical Risk: Immediate Action'
    if row['Tool wear [min]'] > 70 or row['temp_diff'] > 8:
        return 'Moderate Risk: Schedule Maintenance'
    return 'Low Risk: Routine Inspection'
df_new['Maintenance Strategy'] = df_new.apply(maintenance_schedule, axis=1)
```

```
print(df_new['Maintenance Strategy'].value_counts())
df_new.to_csv('predictive_maintenance_schedule.csv', index=False)
print("Maintenance schedule saved as 'predictive_maintenance_schedule.csv'.")
status_counts = df_new['Maintenance Strategy'].value_counts()
status_fig = px.bar(status_counts, x=status_counts.index, y=status_counts.values,
title='Maintenance Strategy Distribution')
app = dash.Dash(__name__)
app.layout = html.Div([
    html.H1("Predictive Maintenance Dashboard"),
    dcc.Graph(figure=status_fig)
])
if __name__ == '__main__':
    app.run_server(debug=True)
```

REFERENCES

1. Lee, J., Kao, H. A., & Yang, S. (2014). Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia cirp*, *16*, 3-8.

2. Zhang, W., Yang, D., & Wang, H. (2019). Data-driven methods for predictive maintenance of industrial equipment: A survey. IEEE systems journal, 13(3), 2213-2227.

3. Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, *20*(7), 1483-1510.

4. Wen, L., Li, X., Gao, L., & Zhang, Y. (2017). A new convolutional neural network-based data-driven fault diagnosis method. *IEEE Transactions on Industrial Electronics*, *65*(7), 5990-5998.

5. Achouch, M., Dimitrova, M., Ziane, K., Sattarpanah Karganroudi, S., Dhouib, R., Ibrahim, H., & Adda, M. (2022). On predictive maintenance in industry 4.0: Overview, models, and challenges. Applied Sciences, 12(16), 8081.

6. Grall, A., Dieulle, L., Bérenguer, C., & Roussignol, M. (2002). Continuous-time predictive-maintenance scheduling for a deteriorating system. *IEEE transactions on reliability*, *51*(2), 141-150.

7. Vlasov, A. I., Grigoriev, P. V., Krivoshein, A. I., Shakhnov, V. A., Filin, S. S., & Migalin, V. S. (2018). Smart management of technologies: predictive maintenance of industrial equipment using wireless sensor networks. Entrepreneurship and Sustainability Issues, 6(2), 489-502.

8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Regularization for deep learning. *Deep learning*, 216-261.

9. Vijay Kumar, E., Chaturvedi, S. K., & Deshpandé, A. W. (2009). Maintenance of industrial equipment: Degree of certainty with fuzzy modelling using predictive maintenance. International Journal of Quality & Reliability Management, 26(2), 196-211.

10. Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 137, 106024.