Faculty of Engineering, Environment and Computing
6006CEM Machine Learning and Related Applications
Coursework Template

Predictive Maintenance in Elevator Industry Using Machine Learning

Izzuddin Ahmad Afif (14530492)

Coventry OneDrive URL:

14530492-IA-s1

Details on the assessment are presented in the Assessment Brief in Aula/6006CEM/Assessment.

Faculty of Engineering, Environment and Computing

6006CEM Machine Learning and Related Applications

Coursework Template

Table of contents

Table of figures

Details on the assessment are presented in the Assessment Brief in Aula/6006CEM/Assessment.

Academic Report

Introduction

The integration of machine learning across industries has substantially augmented operational efficiency and risk management. In the domain of elevator maintenance, ensuring safety and equipment longevity is paramount. This project leverages machine learning to implement predictive maintenance for elevator doors, utilizing a dataset that encompasses sensor data on ball bearings, humidity, and vibration. The focus is on a dataset that includes a diverse range of sensor readings: ball bearings, humidity levels, and vibration measurements. Below are the distributions for each of these critical readings (Figure 1), which highlight the data's complexity and variability.

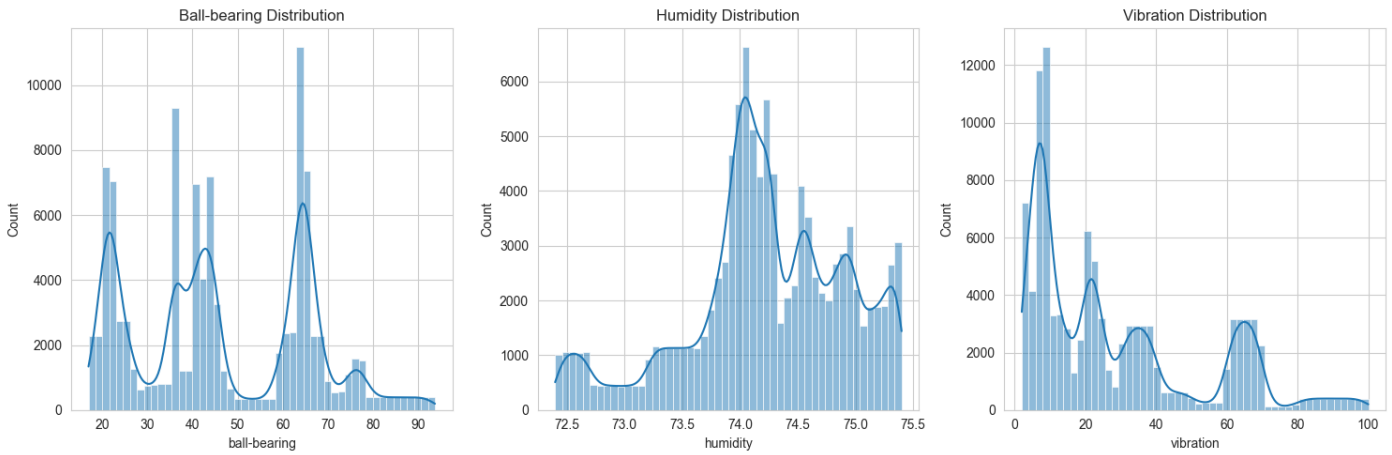

Figure 1 Dataset distribution

Literature Review

Predictive Maintenance in Industry

The transition from traditional maintenance techniques to predictive maintenance (PdM) signifies a pivotal change in the industrial sector. This paradigm shift is driven by the need to minimize downtime and associated costs. As industries accumulate ever-increasing amounts of data, the role of machine learning in decision-making processes becomes more pronounced, influencing areas such as maintenance management and quality improvement. (Susto et al., 2015)

Approaches to Maintenance Management

Maintenance management techniques have evolved over time, advancing from simple run-to-failure strategies to more complex and efficient preventive maintenance schedules. Despite their preventive nature, these schedules often lead to inefficient resource utilization. Hence, PdM, which relies on the health status of equipment, is now being adopted due to its ability to detect and address potential

failures proactively, thereby averting unplanned downtimes and prolonging equipment lifespan. (Susto et al., 2015)

Condition-Based Maintenance (CBM) and Its Importance

CBM is a structured approach that underscores the importance of maintenance decisions based on data gathered through condition monitoring. With the aid of diagnostics and prognostics, CBM aims to avoid unnecessary maintenance tasks by recommending actions only upon evidence of abnormal equipment behaviour. A properly implemented CBM program can significantly curb maintenance costs by reducing unwarranted preventive maintenance activities. (Jardine et al., 2006)

Diagnostics and Prognostics in CBM

Diagnostics and prognostics are crucial aspects of CBM. While diagnostics involves fault detection, isolation, and identification, prognostics takes a proactive stance by predicting faults before they occur. Prognostics is particularly effective in achieving zero-downtime performance by forecasting potential failures and allowing for timely interventions. (Jardine et al., 2006)

Decision Support in CBM

The final step in a CBM program is making informed maintenance decisions. Techniques for maintenance decision support are categorized into diagnostics and prognostics, each playing a vital role in guiding maintenance personnel. Prognostics holds the edge by potentially preventing faults or prepping for them, thus saving on unplanned maintenance costs. Nonetheless, diagnostics remains indispensable, especially when prognostics fails, by providing necessary decision support and contributing to the refinement of prognostic models through diagnostic feedback. (Jardine et al., 2006)

It is crucial to understand the relationships between different sensor readings, as they can significantly impact the predictive maintenance outcomes. Figure 2 presents the correlation matrix of features, illustrating the interdependencies among the different sensor readings.

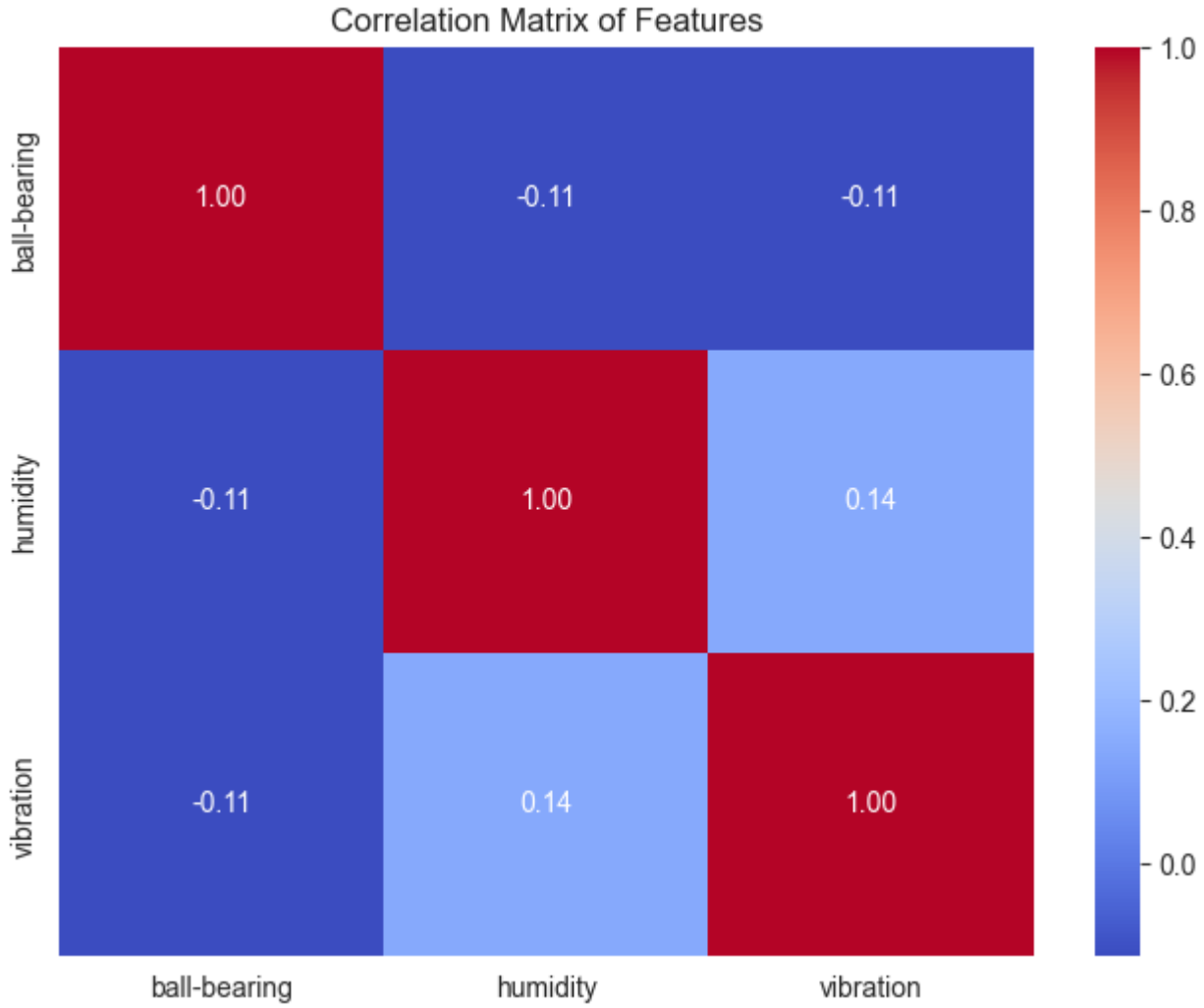

Figure 2 Correlation matrix of Features

Data Analysis and Preprocessing

The foundation of any successful machine learning project lies in the quality and preparation of its dataset. For this project, the dataset was sourced from Huawei's German Research Centre, encompassing a comprehensive collection of time-series data from IoT sensors specifically designed for predictive maintenance in the elevator industry. The data focuses on three critical parameters: ball-bearing revolutions, humidity percentages, and vibration levels in decibels, all of which are essential indicators for the health and functionality of elevator doors.

Data Characteristics and Initial Assessment

The initial dataset consisted of over 112,000 entries, reflecting the intricate and continuous monitoring of elevator doors. Each entry in the dataset represents a unique combination of readings from the three sensors. The ball-bearing sensor provides insights into the mechanical movement of the elevator doors, the humidity sensor tracks environmental conditions that could affect door operation, and the vibration sensor detects anomalies in door movement, often a precursor to mechanical issues.

Preprocessing Techniques

Given the complexity and volume of the data, a rigorous preprocessing procedure was essential to ensure the integrity and usability of the data for ML models. The preprocessing comprised several key steps:

1. **Data Type Conversion:** The first step involved converting all relevant data fields into a consistent numeric format. This standardization was crucial for the subsequent analysis and for the application of various ML algorithms that require numerical input.
2. **Handling Missing Values:** The dataset was scrutinized for missing values. Given the critical nature of predictive maintenance, it was imperative to address these gaps either by imputation or by discarding incomplete records, depending on the nature and extent of the missing data.
3. **Outlier Detection and Treatment:** Outliers can significantly skew the results of ML models. An analysis was conducted to identify any anomalous readings that deviated markedly from the norm. These outliers were then treated appropriately, either by exclusion or transformation, to ensure they did not adversely affect the model's performance.
4. **Feature Scaling:** The final step in the preprocessing phase was the application of StandardScaler, a technique that normalizes the range of the features. This scaling is particularly important in datasets where the variables are measured in different units and scales, as is the case with ball-bearing revolutions, humidity percentages, and vibration levels.

Preprocessing Outcome

The outcome of the preprocessing phase was a streamlined and homogenized dataset, reduced marginally to 106,238 entries, all formatted and scaled suitably for ML analysis. This meticulous preparation laid the groundwork for the accurate and effective application of the chosen machine learning models.
The preprocessing phase's rigor underscored the commitment to data quality and relevance, which are paramount in a field where predictive accuracy can have direct implications for safety and operational efficiency. With the data duly prepared, the study progressed to the application and tuning of the selected machine learning algorithms, aiming to unlock insights that could revolutionize maintenance strategies in the elevator industry.

Algorithm Application and Model Selection

In the realm of predictive maintenance for elevator doors, selecting the right machine learning models is crucial for the success of the project. This study employed three different models, each chosen for their distinct characteristics and potential to address the specificities of the dataset.
Model Selection Rationale

1. **Linear Regression:** Linear Regression was selected for its simplicity and efficiency in modeling linear relationships. As a baseline model, it provides a straightforward interpretation and is useful in understanding the linear correlation between the sensor readings and the maintenance needs.
2. **Decision Tree Regressor:** The Decision Tree Regressor was chosen for its ability to handle complex, non-linear relationships in data. This model is particularly adept at breaking down a decision-making process into a simpler set of decisions, making it suitable for interpreting the interactions between different sensor readings.
3. **Random Forest Regressor:** Selected for its robustness, the Random Forest Regressor is an ensemble method that combines multiple decision trees to improve predictive accuracy and prevent overfitting. Its capability to handle diverse data sets and capture complex relationships made it an ideal choice for this project.

Model Implementation

The implementation of these models was conducted using standard practices. The Linear Regression model was utilized without modifications to its default parameters, providing a baseline for comparison with more complex models. The Decision Tree and Random Forest models were also applied using their standard settings. While these models have hyperparameters that can be tuned, the initial implementation focused on evaluating their default performance with the given dataset.

Evaluation Strategy

The evaluation of the models was based on their performance in predicting maintenance needs. Standard metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R-squared) were used to assess each model's accuracy. This approach allowed for a direct comparison of the models' effectiveness in the context of predictive maintenance for elevator doors.

Conclusion from Model Application

This phase of the project was crucial in understanding the strengths and limitations of each selected model in the context of predictive maintenance. The Linear Regression model provided a foundational understanding, while the Decision Tree and Random Forest models offered insights into more complex relationships in the data. The results from this stage formed the basis for a comprehensive analysis of the suitability of these models for predictive maintenance applications in the elevator industry.

Results and Discussion

The analysis of the predictive maintenance dataset through various machine learning models has yielded results that offer a nuanced view of their capabilities and limitations. This section interprets the graphical output from each model and examines the execution results to provide a comprehensive understanding of their predictive performance.

Model Performance Analysis

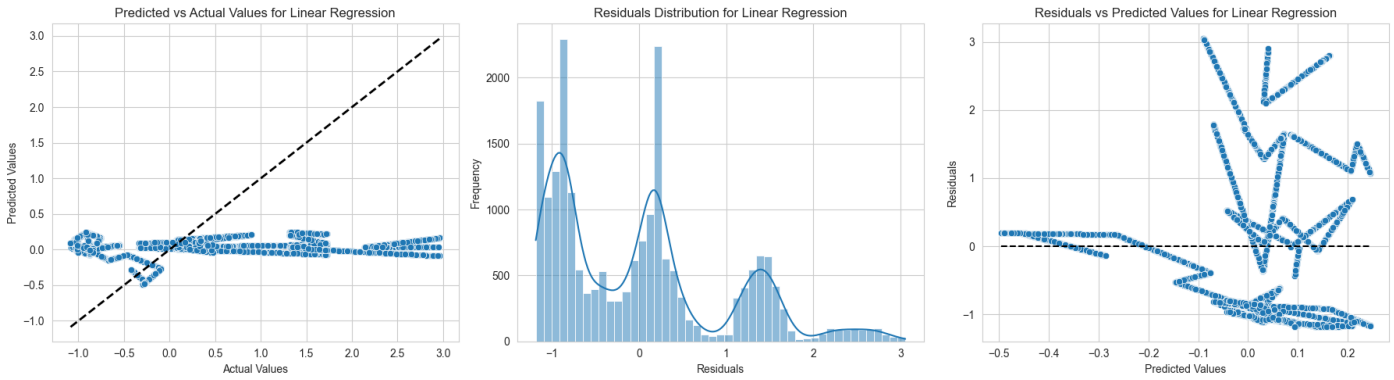

Figure 3 Linear Regression Model Results

Linear Regression Results: The scatter plot for the Linear Regression model (Figure 3) reveals a significant scatter of data points, illustrating a divergence from the line of best fit. This is particularly evident for predictions at higher magnitudes, where the model's inability to capture the actual values is most pronounced. The corresponding residuals plot (Figure 3) confirms this finding, showing that residuals are

not centered around zero, which implies a systematic bias in predictions. This is quantitatively supported by the model's Mean Squared Error (MSE) of 0.966 and an R-squared value of 0.029, confirming the visual assessment that the model has limited effectiveness in explaining the variance within this dataset.

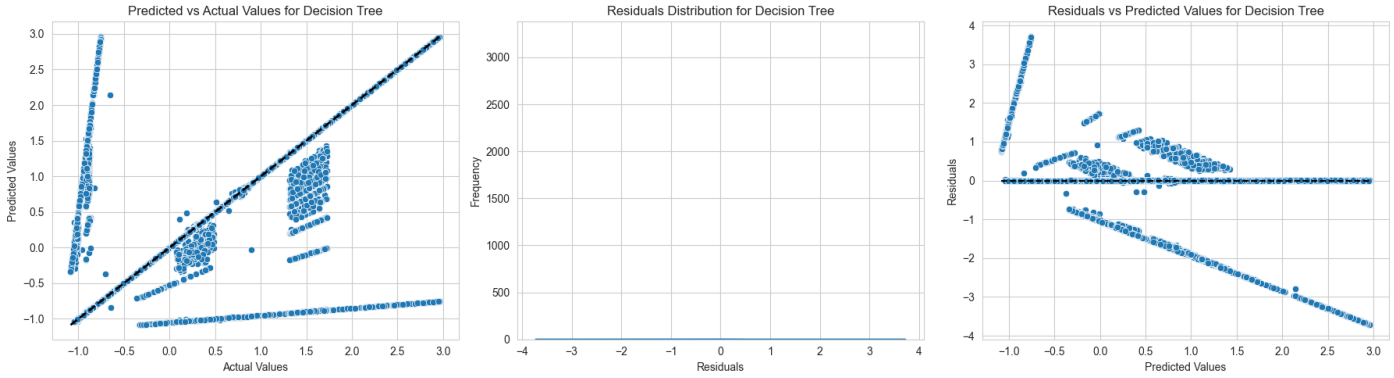

Figure 4 Decision Tree Model Results

Decision Tree Regressor Results: The Decision Tree Regressor's scatter plot (Figure 4) presents a marked improvement, with a more aligned distribution of predicted versus actual values, suggesting a better fit. Nonetheless, the structure in the residuals plot (Figure 4) suggests potential model misspecification or the impact of outliers. Though specific performance metrics like MSE are not provided here, the visual analysis implies that the Decision Tree model better captures the complexity of the data compared to the Linear Regression model but indicates that there is still potential for refinement.

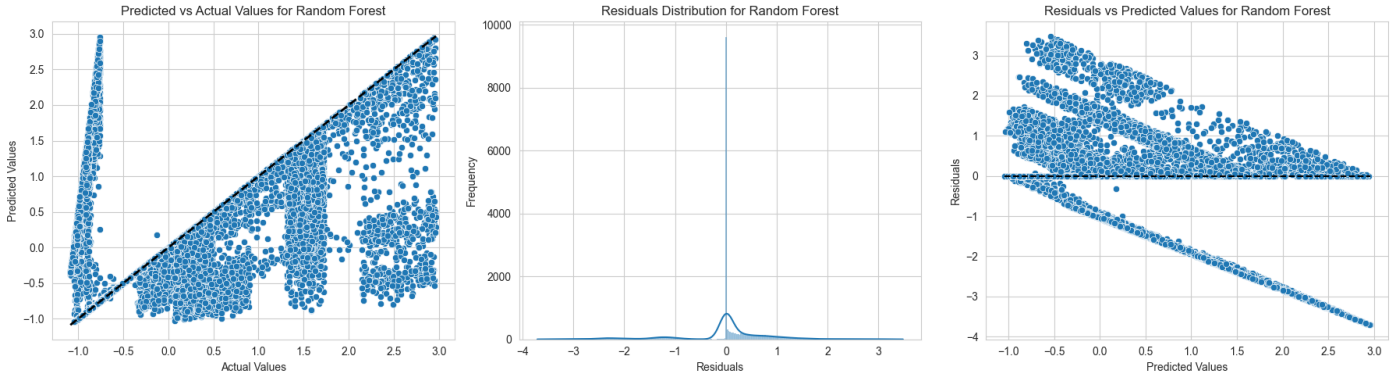

Figure 5 Random Forest Model Results

Random Forest Regressor Results: In stark contrast, the Random Forest Regressor demonstrates superior performance. The scatter plot (Figure 5) displays a concentrated clustering around the diagonal, denoting high accuracy in predictions. The residuals plot (Figure 5) shows a randomized pattern around the horizontal axis, indicative of an excellent fit. The lack of discernible structure in the residuals points to the Random Forest model's ability to account for the majority of the variability in the data, positioning it as the most capable model of the three analysed.

Correlation and Distribution Insights

The histograms for each feature ('ball-bearing', 'humidity', and 'vibration') as presented in figure 1 exhibit distinct distributions. The 'ball-bearing' and 'vibration' features show multimodal distributions, suggesting

the presence of several operational states within the elevator system that might correspond to different maintenance conditions. The 'humidity' histogram displays a near-normal distribution, indicating a more stable and consistent measurement with less direct impact on the system's maintenance state.
The correlation matrix underscores the lack of strong linear relationships among the features, with all correlations being weak. This finding aligns with the Linear Regression model's poor performance and supports the decision to employ more complex models capable of capturing non-linear interactions.

Discussion of Findings

The graphical results, combined with the execution metrics, point to the Random Forest Regressor as the most adept model at predicting maintenance needs for the elevator doors. Its ability to effectively randomize residuals and minimize prediction errors suggests that it is better suited for this application than the simpler Linear Regression and the singular Decision Tree models.
The insights derived from the histograms and correlation matrix also emphasize the complexity of the predictive maintenance task. The varied distributions of the features indicate that predictive modelling in this context requires careful consideration of the underlying data characteristics and the interactions between different types of sensor readings.

Future Directions and Implications

The results of this analysis have important implications for the predictive maintenance of elevator systems. They indicate that while simple models may provide a starting point, complex models that can handle multimodal data and non-linear relationships are necessary to develop accurate predictive maintenance systems.
Future research should explore the integration of additional features and the application of advanced modelling techniques. Additionally, hyperparameter tuning of the Random Forest model could further enhance its predictive performance. Real-world implementation would also involve the development of a continuous learning system that can adapt to new data and evolving conditions, ensuring that the predictive maintenance system remains accurate over time.

Bibliography

Jardine, A., 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.

omlstreaming. (2020, February 7). GRC Datasets for Predictive Maintenance. GitHub. https://github.com/omlstreaming/grc-datasets-pred-maintenance

Susto, G., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). Machine Learning for Predictive Maintenance: A Multiple Classifier Approach. *IEEE Transactions on Industrial Informatics, 11*(3), 812-820.

Appendix A

< A suggested checklist for you, for full details please refer to the coursework brief >

1. The following naming convention is used for the Coventry GitHub Repository and Coventry OneDrive
   StudentID-Initials-s1
   For example, a student Leo Messi whose student ID is 12345678 would name their repository or shared folder as **12345678-LM-s1**
   Failure to follow the naming convention may delay the release of marks and feedback for your coursework.
2. **Coventry** GitHub Repository URL **or** **Coventry** OneDrive URL: added to the top of this report
   2.1. Coventry GitHub Repository includes:
   - URL of the selected dataset(s) included in README
   - The original selected dataset(s)
   - Source-code (.ipynb)
   - Demonstration video (.mp4)

   2.2. Coventry OneDrive folder includes:
   - URL of the selected dataset(s) included in a separated text file
   - The original selected dataset(s)
   - Source-code (.ipynb)
   - Demonstration video (.mp4)
3. Source-code added **as text** in Appendix B (below)
4. Submission in the form of a **Word** document. ***Other formats are not accepted.*

Appendix B

```
import pandas as pd

# Load the dataset again to ensure we're starting fresh
dataset_path = 'predictive-maintenance-dataset.csv'
df = pd.read_csv(dataset_path, sep=';')  # Using the semicolon as the separator this
time

# Convert the columns to the appropriate data types
df['ball-bearing'] = pd.to_numeric(df['ball-bearing'], errors='coerce')
df['humidity'] = pd.to_numeric(df['humidity'], errors='coerce')
df['vibration'] = pd.to_numeric(df['vibration'], errors='coerce')

# Drop rows with any NaN values that resulted from conversion errors
df = df.dropna()

# Confirm the DataFrame structure and show the first few rows
df.info()
df.head()
# Load the original dataset without any cleaning to compare the number of rows
df_original = pd.read_csv(dataset_path, sep=';')

# Count the number of rows in the original dataset
original_row_count = df_original.shape[0]

# Count the number of rows after cleaning (which we have from the previous step)
cleaned_row_count = df.shape[0]

# Calculate the number of rows removed
rows_removed = original_row_count - cleaned_row_count

original_row_count, cleaned_row_count, rows_removed

# Generate descriptive statistics for the dataset
descriptive_stats = df.describe()

descriptive_stats

import matplotlib.pyplot as plt
import seaborn as sns

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Plot histograms for each feature in the dataset
plt.figure(figsize=(15, 5))

# Histogram for 'ball-bearing'
plt.subplot(1, 3, 1)
sns.histplot(df['ball-bearing'], kde=True, bins=50)
plt.title('Ball-bearing Distribution')
```

```
# Histogram for 'humidity'
plt.subplot(1, 3, 2)
sns.histplot(df['humidity'], kde=True, bins=50)
plt.title('Humidity Distribution')

# Histogram for 'vibration'
plt.subplot(1, 3, 3)
sns.histplot(df['vibration'], kde=True, bins=50)
plt.title('Vibration Distribution')

# Show the histograms
plt.tight_layout()
plt.show()

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', cbar=True)
plt.title('Correlation Matrix of Features')
plt.show()

from sklearn.preprocessing import StandardScaler

# Initialize the standard scaler
scaler = StandardScaler()

# Fit the scaler to the data and transform it
df_scaled = scaler.fit_transform(df)

# Convert the scaled array back to a DataFrame for readability
df_scaled = pd.DataFrame(df_scaled, columns=df.columns)

# Show the first few rows of the scaled dataset
df_scaled.head()

from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
# We'll use 80% of the data for training and 20% for testing
X_train, X_test = train_test_split(df_scaled, test_size=0.2, random_state=42)

# Show the number of samples in each set
num_samples_train = X_train.shape[0]
num_samples_test = X_test.shape[0]

num_samples_train, num_samples_test

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Initialize the models
```

```
linear_model = LinearRegression()
decision_tree_model = DecisionTreeRegressor(random_state=42)
random_forest_model = RandomForestRegressor(random_state=42)

# Separate the features and target variable for training
# Assuming the 'vibration' column is the target variable and the rest are features
X_train_features = X_train.drop('vibration', axis=1)
X_test_features = X_test.drop('vibration', axis=1)
y_train_target = X_train['vibration']
y_test_target = X_test['vibration']

# Train the Linear Regression model
linear_model.fit(X_train_features, y_train_target)
# Train the Decision Tree model
decision_tree_model.fit(X_train_features, y_train_target)
# Train the Random Forest model
random_forest_model.fit(X_train_features, y_train_target)

# Make predictions with all models on the testing set
linear_predictions = linear_model.predict(X_test_features)
decision_tree_predictions = decision_tree_model.predict(X_test_features)
random_forest_predictions = random_forest_model.predict(X_test_features)

# Calculate the Mean Squared Error (MSE) for each model
linear_mse = mean_squared_error(y_test_target, linear_predictions)
decision_tree_mse = mean_squared_error(y_test_target, decision_tree_predictions)
random_forest_mse = mean_squared_error(y_test_target, random_forest_predictions)

(linear_mse, decision_tree_mse, random_forest_mse)

# Import necessary libraries for machine learning and visualization
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Function to evaluate and visualize the model performance
def evaluate_and_visualize_model_performance(y_true, y_pred, model_name):
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    print(f"{model_name} Performance:")
    print(f"Mean Squared Error: {mse}")
    print(f"Mean Absolute Error: {mae}")
    print(f"R-squared: {r2}")

    residuals = y_true - y_pred

    plt.figure(figsize=(18, 5))

    # Plot Predicted vs Actual values
    plt.subplot(1, 3, 1)
    sns.scatterplot(x=y_true, y=y_pred)
    plt.plot([y_true.min(), y_true.max()], [y_true.min(), y_true.max()], 'k--', lw=2)
    plt.title(f'Predicted vs Actual Values for {model_name}')
    plt.xlabel('Actual Values')
```

```
    plt.ylabel('Predicted Values')

    # Plot Residuals Distribution
    plt.subplot(1, 3, 2)
    sns.histplot(residuals, kde=True)
    plt.title(f'Residuals Distribution for {model_name}')
    plt.xlabel('Residuals')
    plt.ylabel('Frequency')

    # Plot Residuals vs Predicted values
    plt.subplot(1, 3, 3)
    sns.scatterplot(x=y_pred, y=residuals)
    plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='black',
linestyles='dashed')
    plt.title(f'Residuals vs Predicted Values for {model_name}')
    plt.xlabel('Predicted Values')
    plt.ylabel('Residuals')

    plt.tight_layout()
    plt.show()

# Assuming 'linear_predictions' and 'decision_tree_predictions' are the predictions
from the respective models
# and 'y_test_target' is the actual target values from the test set
# Replace 'linear_predictions' and 'decision_tree_predictions' with your actual
predictions

# Evaluate and visualize Linear Regression model performance
evaluate_and_visualize_model_performance(y_test_target, linear_predictions, "Linear
Regression")

# Evaluate and visualize Decision Tree model performance
evaluate_and_visualize_model_performance(y_test_target, decision_tree_predictions,
"Decision Tree")

# Evaluate and visualize Random Forest model performance
evaluate_and_visualize_model_performance(y_test_target, random_forest_predictions,
"Random Forest")
```