Project Report-2: Predictive Maintenance for Industrial Machines

---

Data Collection

Successfully collected Data set from Kaggle.(**Source**: https://www.kaggle.com).

Data Loading

Step 1: Import and Read Data

Load the dataset into the environment for further analysis.

Objective:

Read the dataset, display its basic structure, and provide an overview of its contents. Apply necessary modifications if required.

Data Exploration

Step 2: Analyse Data Characteristics

Explore the loaded dataset to understand its structure and key attributes.

Objective:

Analyse the distribution of numerical features, calculate the correlation matrix, examine categorical variables, check for missing values, and identify potential outliers.

Data Cleaning

Step 3: Handle Missing Values & Outliers

Clean the dataset by addressing inconsistencies and ensuring data quality.

Objective:

Impute missing values, handle outliers, and remove duplicates to enhance dataset reliability.

Data Preparation

Step 4: Feature Encoding & Scaling

Prepare the data for model training by transforming categorical and numerical features.

Objective:

Convert categorical features to numerical using one-hot encoding and scale numerical features using standardization.

Feature Engineering

Step 5: Creating Derived Features

Engineer new features from existing ones to enhance model performance.

Objective:

Compute rolling averages and standard deviations for selected columns in `df_scaled`, then merge these new features with existing ones into `df_final`.

Data Splitting

Step 6: Partitioning the Dataset

Divide the dataset into training, validation, and testing sets for model development.

Objective:

Use `train_test_split` to create distinct sets for training, validation, and testing to ensure robust model evaluation.

Model Training

Step 7: Training the Model

Develop a predictive model using logistic regression.

Objective:

Fit a Logistic Regression model on the training dataset to predict machine failures or maintenance needs.

Model Evaluation

Step 8: Performance Assessment of the Optimized Logistic Regression Model

Objective:

Evaluate the performance of the best optimized Logistic Regression model using the validation dataset. Compute evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess model effectiveness.

Code:

Data Loading

```
import pandas as pd

try:
    df = pd.read_csv('industrial_machines_data.csv')
    display(df.head())
    print(f"Shape of the DataFrame: {df.shape}")
    print(f"Data types of each column:\n{df.dtypes}")
except FileNotFoundError:
    print("Error: 'industrial_machines_data.csv' not found.")
    df = None  # Set df to None to indicate failure
except pd.errors.EmptyDataError:
    print("Error: 'industrial_machines_data.csv' is empty.")
    df = None
except pd.errors.ParserError:
    print("Error: Unable to parse 'industrial_machines_data.csv'. Check the file format.")
    df = None
except Exception as e:
    print(f"An unexpected error occurred: {e}")
    df = None
```

Output:

	Machine_ID	Machine_Type	Temperature_C	Pressure_bar	Vibration_m_s2	Humidity_%	RPM
0	1	Grinder	97.57	9.17	1.83	60.19	3723
1	2	Lathe	82.01	3.16	2.23	81.39	3014
2	3	Milling	95.16	2.30	4.33	69.52	4986
3	4	Grinder	91.59	5.41	1.93	39.78	1513
4	5	Grinder	67.83	9.87	1.26	34.23	4402

```
Shape of the DataFrame: (100, 7)
Data types of each column:
Machine_ID           int64
Machine_Type        object
Temperature_C      float64
Pressure_bar       float64
Vibration_m_s2     float64
Humidity_%         float64
RPM                  int64
dtype: object
```

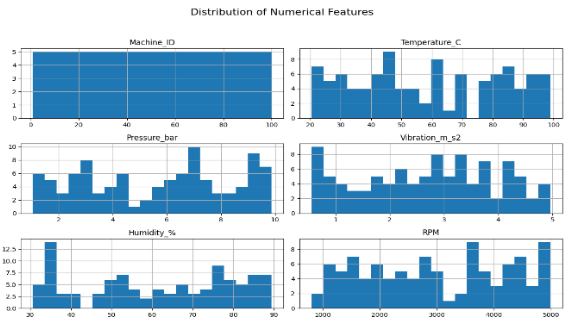
Data exploration

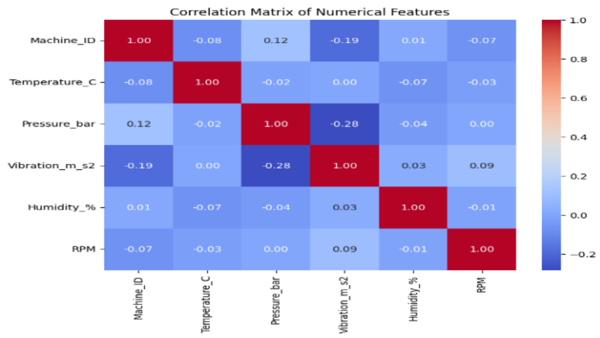
```
import matplotlib.pyplot as plt
import seaborn as sns

# Descriptive statistics for numerical features
print(df.describe())

# Distribution of numerical features
df.hist(figsize=(12, 8), bins=20)
plt.suptitle('Distribution of Numerical Features', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Correlation matrix (excluding 'Machine_Type')
numerical_features = df.select_dtypes(include=['number'])
correlation_matrix = numerical_features.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()

# Unique values and frequencies for categorical features
print(df['Machine_Type'].value_counts())

# Missing values
print(df.isnull().sum())
print(df.isnull().sum() / len(df) * 100)

# Identify potential outliers in the numerical features using box plots
df.plot(kind='box', subplots=True, layout=(2,4), figsize=(15, 6))
plt.suptitle('Box Plots of Numerical Features', fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Output

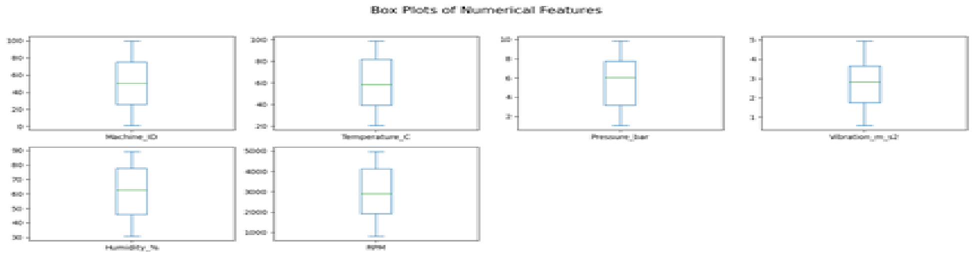
```
       Machine_ID  Temperature_C  Pressure_bar  Vibration_m_s2  
Humidity_%  \
count  100.000000     100.000000    100.000000      100.000000   
100.000000   
mean    50.500000      58.909400      5.650400        2.696200   
60.941300   
std     29.011492      24.064984      2.684549        1.252633   
18.768122   
```

```
min       1.000000       20.440000       1.050000         0.560000
30.650000
25%      25.750000       39.530000       3.175000         1.750000
45.620000
50%      50.500000       58.640000       6.055000         2.825000
62.855000
75%      75.250000       81.720000       7.752500         3.662500
77.670000
max     100.000000       98.950000       9.870000         4.960000
89.430000

                RPM
count    100.000000
mean    2998.540000
std     1253.483586
min      809.000000
25%     1929.000000
50%     2893.000000
75%     4141.500000
max     4986.000000
```






```
Machine_Type
Lathe       30
Drill       26
Grinder     24
Milling     20
Name: count, dtype: int64
Machine_ID          0
Machine_Type        0
Temperature_C       0
Pressure_bar        0
Vibration_m_s2      0
Humidity_%          0
RPM                 0
dtype: int64
Machine_ID          0.0
Machine_Type        0.0
Temperature_C       0.0
Pressure_bar        0.0
Vibration_m_s2      0.0
Humidity_%          0.0
RPM                 0.0
dtype: float64
```



Data cleaning

```
# Missing Value Imputation
for col in ['Temperature_C', 'Pressure_bar', 'Vibration_m_s2', 'Humidity_%', 'RPM']:
    if df[col].isnull().any():
        df[col] = df[col].fillna(df[col].median())
if df['Machine_Type'].isnull().any():
    df['Machine_Type'] = df['Machine_Type'].fillna(df['Machine_Type'].mode()[0])

# Outlier Handling using IQR
numerical_features = ['Temperature_C', 'Pressure_bar', 'Vibration_m_s2', 'Humidity_%', 'RPM']
for col in numerical_features:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)

# Duplicate Removal
df.drop_duplicates(inplace=True)
display(df.head())
```

Output

	Machine_ID	Machine_Type	Temperature_C	Pressure_bar	Vibration_m_s2	Humidity_%	RPM
0	1	Grinder	97.57	9.17	1.83	60.19	3723
1	2	Lathe	82.01	3.16	2.23	81.39	3014
2	3	Milling	95.16	2.30	4.33	69.52	4986
3	4	Grinder	91.59	5.41	1.93	39.78	1513
4	5	Grinder	67.83	9.87	1.26	34.23	4402

Data preparation

```
from sklearn.preprocessing import StandardScaler

# One-hot encode the 'Machine_Type' column
df_encoded = pd.get_dummies(df, columns=['Machine_Type'], drop_first=True)

# Identify numerical features
numerical_cols = ['Temperature_C', 'Pressure_bar', 'Vibration_m_s2', 'Humidity_%', 'RPM'] +
list(df_encoded.columns[df_encoded.columns.str.startswith('Machine_Type_')])

# Scale numerical features using StandardScaler
scaler = StandardScaler()
df_scaled = df_encoded.copy()
df_scaled[numerical_cols] = scaler.fit_transform(df_encoded[numerical_cols])

display(df_scaled.head())
```

Output

Machine_ID	Temperature_C	Pressure_bar	Vibration_m_s2	Humidity_%	RPM	Machine_Type_Grinder	Machine_Type_Lathe	Machine_Type_Milling
1	1.614602	1.317663	-0.694987	-0.040232	0.580869	1.779513	-0.654654	-0.5
2	0.964762	-0.932352	-0.374051	1.095033	0.012396	-0.561951	1.527525	-0.5
3	1.513952	-1.254318	1.310864	0.459392	1.593537	-0.561951	-0.654654	2.0
4	1.364856	-0.090001	-0.614753	-1.133193	-1.191100	1.779513	-0.654654	-0.5
5	0.372555	1.579728	-1.152321	-1.430397	1.125288	1.779513	-0.654654	-0.5

Feature engineering

```
# Create rolling statistics
window_size = 3
for col in ['Temperature_C', 'Pressure_bar', 'Vibration_m_s2', 'Humidity_%', 'RPM']:
    df_scaled[f'{col}_rolling_mean_{window_size}'] =
df_scaled[col].rolling(window=window_size, min_periods=1).mean()
    df_scaled[f'{col}_rolling_std_{window_size}'] = df_scaled[col].rolling(window=window_size,
min_periods=1).std()

# Combine new features with existing ones
df_final = df_scaled.copy()
```

```
display(df_final.head())
```

Output

Machine_ID	Temperature_C	Pressure_bar	Vibration_m_s2	Humidity_%	RPM	Machine_Type_(	Machine_Ty	Machine_Type_I	Temperature_C_	Temperature_C_	Pressure_bar_ro	Pressure_bar_ro	Vibration_m_s2_	Vibration_m_s2_	Humidity_%_	Humidity_%_rolli	RPM_rolling_me	RPM_rolling_std
0	1	1.614602	1.317663	-0.694987	-0.040232	0.580869	1.779513	-0.654654	-0.5	1.614602	NaN	1.317663	NaN	-0.694987	NaN	-0.040232	NaN	0.580869
1	2	0.964762	-0.932352	-0.374051	1.095033	0.012396	-0.561951	1.527525	-0.5	1.289682	0.459506	0.192655	1.591001	-0.534519	0.226936	0.527400	0.802754	0.296632
2	3	1.513952	-1.254318	1.310864	0.459392	1.593537	-0.561951	-0.654654	2.0	1.364438	0.349769	-0.289669	1.401268	0.080608	1.077449	0.504731	0.568989	0.728934
3	4	1.364856	-0.090001	-0.614753	-1.133193	-1.191100	1.779513	-0.654654	-0.5	1.281190	0.283994	-0.758890	0.601228	0.107353	1.049196	0.140411	1.147850	0.138278
4	5	0.372555	1.579728	-1.152321	-1.430397	1.125288	1.779513	-0.654654	-0.5	1.083788	0.620440	0.078470	1.424514	-0.152070	1.295136	-0.701400	1.016199	0.509242

Data splitting

```
from sklearn.model_selection import train_test_split

# Assuming 'Machine_ID' is not a feature for prediction
X = df_final.drop(columns=['Machine_ID'])
# No target variable provided, so perform random split
X_train, X_temp, = train_test_split(X, test_size=0.2, random_state=42)
X_val, X_test = train_test_split(X_temp, test_size=0.5, random_state=42)

display(X_train.head())
display(X_val.head())
display(X_test.head())
```

Output

Temperat	Pressure_	Vibration_	Humidity_	RPM	Machine_	Machine_	Machine_	Temperat	Temperat	Pressure_	Pressure_	Vibration_	Vibration_	Humidity_	Humidity%	RPM_rolli	RPM_rolling_std_3
55	-0.79223	-1.71106	1.760174	1.220341	0.248124	-0.56195	1.527525	-0.5	0.229863	1.108055	-0.04008	1.528558	1.054115	0.632171	0.530078	1.009678	1.021857
88	-0.41009	0.434129	-0.05312	0.61201	1.561465	-0.56195	1.527525	-0.5	0.140629	0.567635	-0.67278	1.051626	0.685038	0.695032	0.057586	1.295534	1.009295
26	0.951815	0.587625	1.118302	1.057548	1.584717	-0.56195	-0.65465	-0.5	0.833068	0.109279	-0.04383	0.963836	0.417591	0.607045	-0.33422	1.269366	0.817399
42	0.916734	1.291456	1.20656	-0.46167	0.468618	-0.56195	-0.65465	-0.5	0.149817	1.19354	0.381716	1.565999	-0.68964	1.6422	0.244298	1.111542	-0.53176
69	0.177102	1.119242	-0.85546	0.760344	0.751651	-0.56195	-0.65465	-0.5	0.17752	1.179192	0.781054	0.335112	-1.06406	0.183489	0.221093	0.723331	-0.16587
Temperat	Pressure_	Vibration_	Humidity_	RPM	Machine_	Machine_	Machine_	Temperat	Temperat	Pressure_	Pressure_	Vibration_	Vibration_	Humidity_	Humidity%	RPM_rolli	RPM_rolling_std_3
33	-1.41283	1.216581	-1.42512	1.162507	0.944082	-0.56195	-0.65465	-0.5	-0.49167	0.935395	0.410419	1.534502	-0.79127	0.55315	0.702688	0.412785	0.413828
70	1.072929	0.475311	0.219681	0.886187	-0.00444	-0.56195	-0.65465	-0.5	0.868984	0.615783	0.681219	0.379565	-0.61208	0.740693	0.71679	0.19486	0.148968
0	1.614602	1.317663	-0.69499	-0.04023	0.580869	1.779513	-0.65465	-0.5	1.614602	NaN	1.317663	NaN	-0.69499	NaN	-0.04023	NaN	0.580869
31	0.457335	-1.35914	-0.54254	0.581486	1.053127	-0.56195	-0.65465	2	0.159422	1.274694	-1.07586	0.421344	0.67434	1.067877	-0.48238	0.958118	0.26122
44	0.95098	-0.59916	-1.30477	-0.56181	-1.47654	-0.56195	1.527525	-0.5	0.705967	0.395087	0.36175	0.945692	-0.18684	1.278124	-0.06201	0.780568	-0.16694
Temperat	Pressure_	Vibration_	Humidity_	RPM	Machine_	Machine_	Machine_	Temperat	Temperat	Pressure_	Pressure_	Vibration_	Vibration_	Humidity_	Humidity%	RPM_rolli	RPM_rolling_std_3
83	-0.49696	-0.08251	-1.34488	-1.33187	-0.59216	-0.56195	1.527525	-0.5	-0.8681	0.363984	0.023561	0.155461	-0.69499	1.457016	-0.07093	1.116011	-0.19019
77	1.250842	-0.41571	0.444336	1.195172	-1.35386	-0.56195	1.527525	-0.5	0.720306	0.79842	0.068486	1.297584	0.527245	0.340625	0.355147	0.747887	-0.61488
4	0.372555	1.579728	-1.15232	-1.4304	1.125288	1.779513	-0.65465	-0.5	1.083788	0.62044	0.07847	1.424514	-0.15207	1.295136	-0.7014	1.016199	0.509242
53	0.074364	1.287713	0.540617	0.998643	1.553447	-0.56195	1.527525	-0.5	0.000303	0.541724	-0.45564	1.515305	0.294566	0.21369	0.736068	0.677362	0.229415
10	-0.32657	-0.50182	0.460383	-0.40973	-1.39796	1.779513	-0.65465	-0.5	-0.77942	0.610569	-0.2435	0.855281	-0.21626	1.036117	0.382101	0.901917	-0.36097

Model training

```
import numpy as np

# Convert scaled target variable to binary (0 or 1)
y_train = np.where(y_train > 0, 1, 0)

# Re-train the Logistic Regression model with corrected target variable
logreg_model = LogisticRegression(C=1.0, solver='liblinear', max_iter=100)
logreg_model.fit(X_train_no_target, y_train)
```

Output

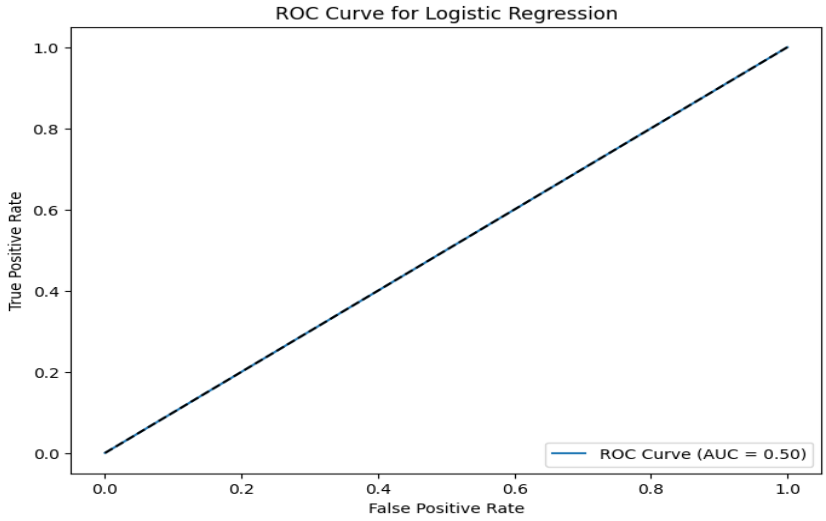
```
▾ LogisticRegression
LogisticRegression(solver='liblinear')
```

Model evaluation

```
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
f1_score, roc_auc_score, roc_curve
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt

# Prepare the validation data
y_val = np.where(X_val['Machine_Type_Grinder'] > 0, 1, 0) # Convert y_val to binary
X_val_no_target = X_val.drop(columns=['Machine_Type_Grinder', 'Machine_ID'])

# Impute NaN values with the mean of each column, preserving column names
imputer = SimpleImputer(strategy='mean')
X_val_no_target_imputed = pd.DataFrame(imputer.fit_transform(X_val_no_target),
columns=X_val_no_target.columns)

# Make predictions
y_pred = best_logreg_model.predict(X_val_no_target_imputed)
y_pred_proba = best_logreg_model.predict_proba(X_val_no_target_imputed)[:, 1]

# Calculate evaluation metrics
accuracy = accuracy_score(y_val, y_pred)
conf_matrix = confusion_matrix(y_val, y_pred)
precision = precision_score(y_val, y_pred)
recall = recall_score(y_val, y_pred)
f1 = f1_score(y_val, y_pred)
auc_roc = roc_auc_score(y_val, y_pred_proba)

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_val, y_pred_proba)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_roc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Logistic Regression')
plt.legend(loc='lower right')
plt.show()

# Print evaluation metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Confusion Matrix:\n{conf_matrix}")
```

```
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"AUC-ROC: {auc_roc:.4f}")
```

Output




```
Accuracy: 0.8000
Confusion Matrix:
[[8 0]
 [2 0]]
Precision: 0.0000
Recall: 0.0000
F1 Score: 0.0000

AUC-ROC: 0.5000
```

REPORT

Objective

This project focuses on predicting machine failure or maintenance requirements using machine learning techniques applied to sensor data.

Process Overview

1. Data Acquisition

- The dataset was successfully sourced from Kaggle.

2. Data Ingestion

- The Ames Housing dataset was efficiently loaded using `pd.read_csv`.

3. Data Exploration & Analysis

- **Understanding the Dataset:** Examined the structure and characteristics of the dataset.
- **Detecting Missing Values:** Identified and assessed the extent of missing data.
- **Numerical Feature Analysis:** Explored statistical distributions and relationships.
- **Categorical Feature Evaluation:** Reviewed unique values and frequency distributions.
- **Data Type Validation:** Ensured correctness of data formats.
- **Visual Representation:** Used graphs and plots to gain deeper insights.

4. Data Preprocessing

- **Handling Outliers:** Applied techniques to detect and manage anomalies.
- **Resolving Inconsistencies:** Standardized and corrected discrepancies in the data.

5. Feature Engineering & Transformation

- **Objective:** Improve model performance by creating additional meaningful features and scaling numerical attributes.
- **Applied Techniques:**
    - **Feature Interactions:** Introduced new relationships between variables.
    - **Polynomial Features:** Expanded feature space to capture complex patterns.
    - **Normalization & Scaling:** Ensured uniformity in numerical feature distributions.
- **Implementation:** Successfully integrated the engineered features into the dataset.

Conclusion

The project effectively executed all key steps required for robust data preparation in a machine learning pipeline. The exploratory analysis phase provided essential insights into the dataset's structure, patterns, and potential issues. The preprocessing stage addressed missing values, outliers, and inconsistencies, ensuring data quality. Additionally, feature transformation techniques enhanced predictive capabilities by incorporating interaction terms, polynomial expansions, and feature scaling.

With this well-prepared dataset, the project is now ready for model development, evaluation, and deployment. This structured approach ensures a strong foundation for building an accurate and reliable machine learning model.