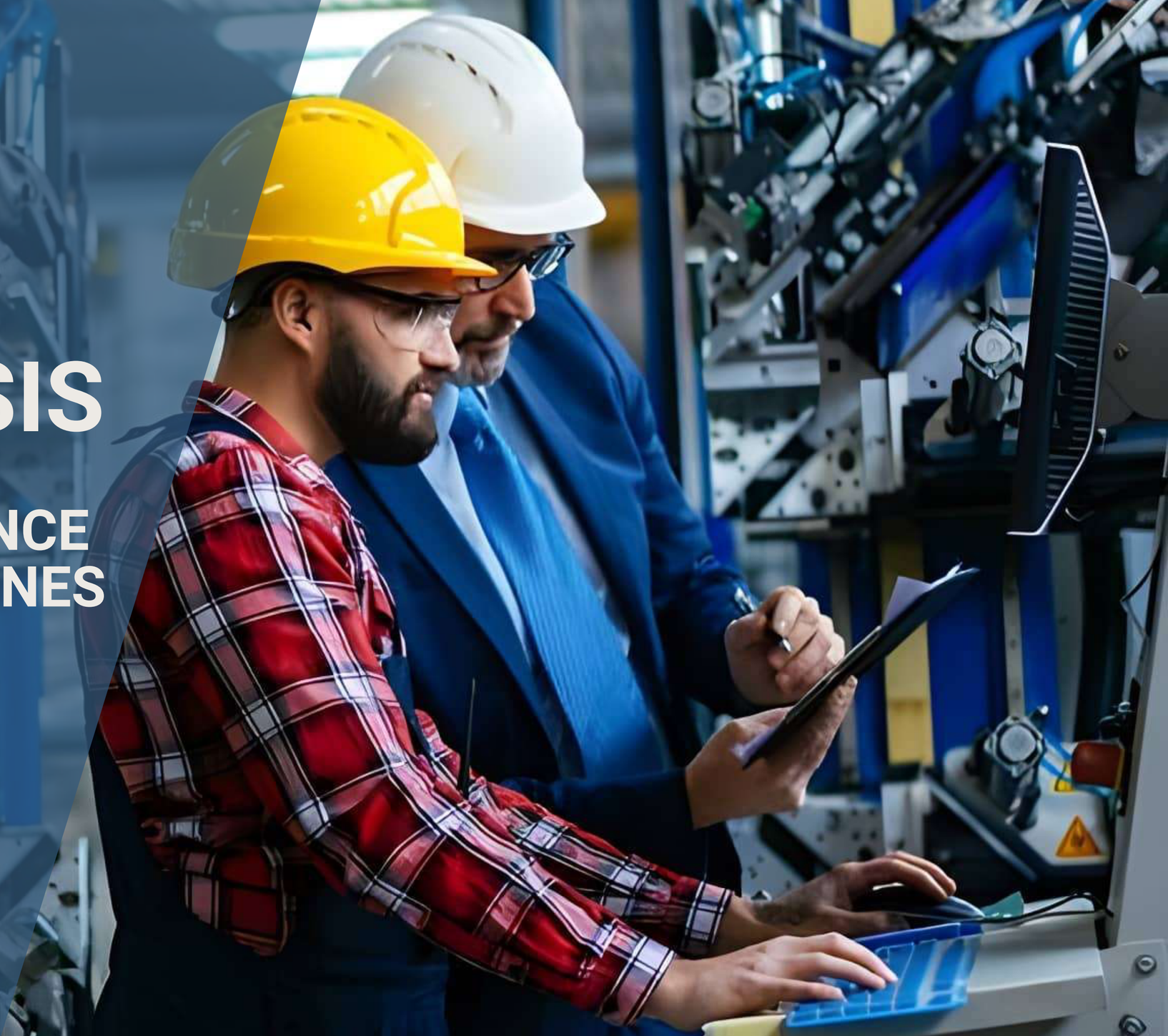
DATA ANALYSIS

PREDICTIVE MAINTENANCE FOR INDUSTRIAL MACHINES



1st STEP: Obtain the Dataset

INPUT:

```
import pandas as pd

df = pd.read_csv("predictive_maintenance.csv")
df = df.copy()

print (df.shape)

#show top 10 data
df.head(10)
```

OUTPUT:

```
(10000, 10)
```

	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
5	6	M14865	M	298.1	308.6	1425	41.9	11	0	No Failure
6	7	L47186	L	298.1	308.6	1558	42.4	14	0	No Failure
7	8	L47187	L	298.1	308.6	1527	40.2	16	0	No Failure
8	9	M14868	M	298.3	308.7	1667	28.6	18	0	No Failure
9	10	M14869	M	298.5	309.0	1741	28.0	21	0	No Failure

2nd STEP: Data Cleaning

INPUT:

```
# Check the columns data types. Are they all correct?
df.info()
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   UDI                      10000 non-null  int64  
 1   Product ID               10000 non-null  object 
 2   Type                     10000 non-null  object 
 3   Air temperature [K]      10000 non-null  float64
 4   Process temperature [K]  10000 non-null  float64
 5   Rotational speed [rpm]   10000 non-null  int64  
 6   Torque [Nm]              10000 non-null  float64
 7   Tool wear [min]          10000 non-null  int64  
 8   Target                   10000 non-null  int64  
 9   Failure Type             10000 non-null  object 
dtypes: float64(3), int64(4), object(3)
memory usage: 781.4+ KB
```

2nd STEP: Data Cleaning

INPUT:

```
# Data exploration: how many "Null" rows do we have? And Where?
df.isnull().sum()
```

OUTPUT:

```
UDI                        0
Product ID                 0
Type                       0
Air temperature [K]        0
Process temperature [K]    0
Rotational speed [rpm]     0
Torque [Nm]                0
Tool wear [min]            0
Target                     0
Failure Type               0
dtype: int64
```

3rd STEP: Data Exploration

INPUT:

```python
import pandas as pd

# Numerical columns
numerical_cols = df[["Air temperature [K]", "Rotational speed [rpm]", "Torque [Nm]"]]

# Calculate summary statistics for numerical columns
numerical_stats = numerical_cols.describe()

# Categorical columns
categorical_cols = df[["Product ID", "Type"]]

# Compute summary statistics for categorical columns
categorical_stats = categorical_cols.describe()

# Count the number of elements for each different "Type"
type_counts = df.value_counts()


# Print the summary statistics
print(f"Summary Statistics for Numerical Columns:\n{numerical_stats}\n")
print(f"Summary Statistics for Categorical Columns:\n{categorical_stats}\n")
print(f"Type counts:\n{type_counts}")
```

3rd STEP: Data Exploration

OUTPUT:

```
Summary Statistics for Numerical Columns:
       Air temperature [K]  Rotational speed [rpm]   Torque [Nm]
count         10000.000000            10000.000000  10000.000000
mean            300.004930             1538.776100     39.986910
std               2.000259              179.284096      9.968934
min             295.300000             1168.000000      3.800000
25%             298.300000             1423.000000     33.200000
50%             300.100000             1503.000000     40.100000
75%             301.500000             1612.000000     46.800000
max             304.500000             2886.000000     76.600000

Summary Statistics for Categorical Columns:
       Product ID   Type
count       10000  10000
unique      10000      3
top        M14860      L
freq            1   6000
```

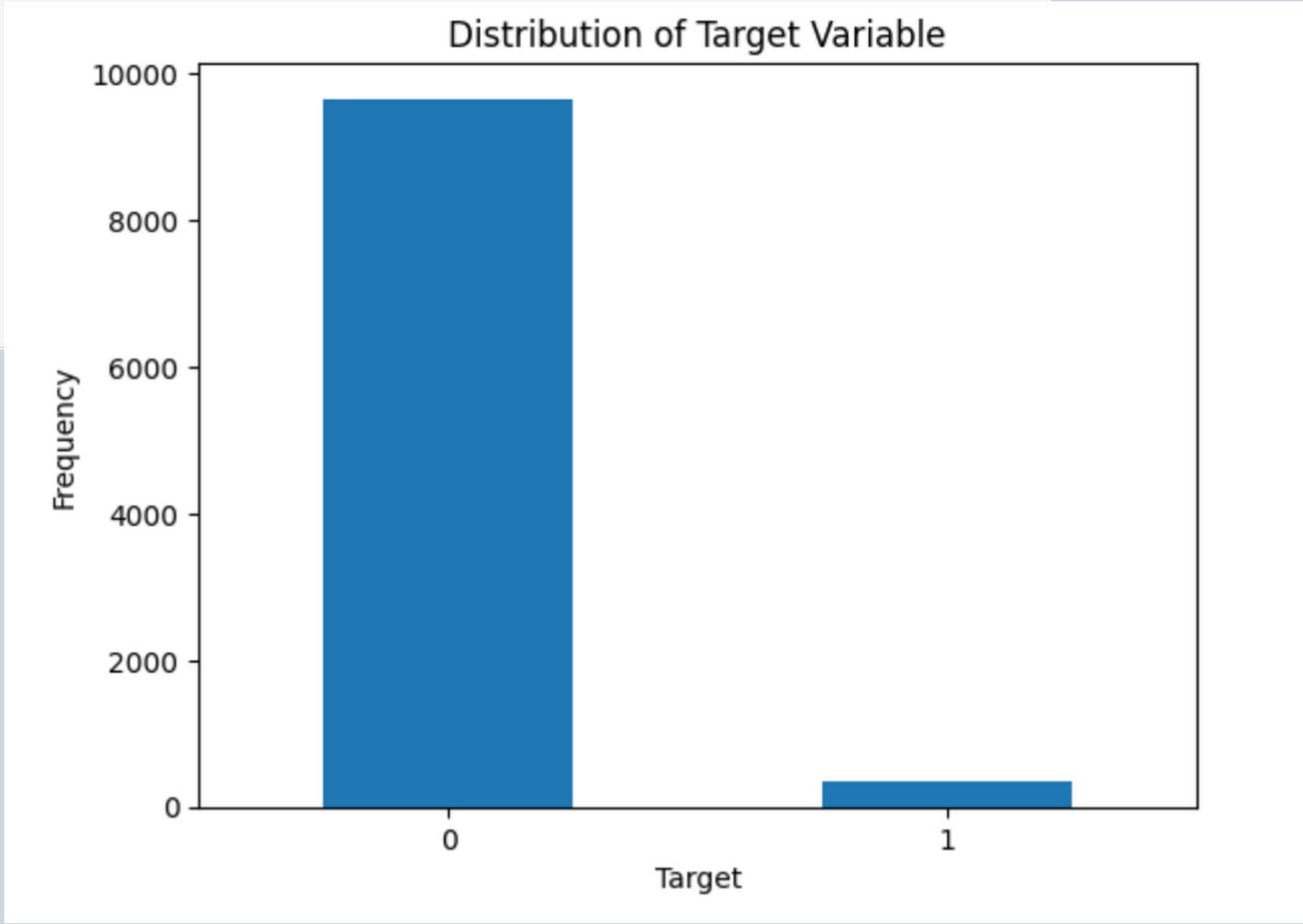
```
Type counts:
Type
L    6000
M    2997
H    1003
Name: count, dtype: int64
```

3rd STEP: Data Exploration

INPUT:

```python
# 3.2) Distribution of Target Variable: Create a histogram or bar plot
# to visualize the distribution of the target variable "Target".
# This will give you an overview of the proportion of machinery failures in the dataset.

import pandas as pd
import matplotlib.pyplot as plt

# Plot the distribution of the target variable
df['Target'].value_counts().plot.bar()
plt.title('Distribution of Target Variable')
plt.xlabel('Target')
plt.ylabel ('Frequency')
plt.xticks(rotation = 0)
plt.show()
```

OUTPUT:



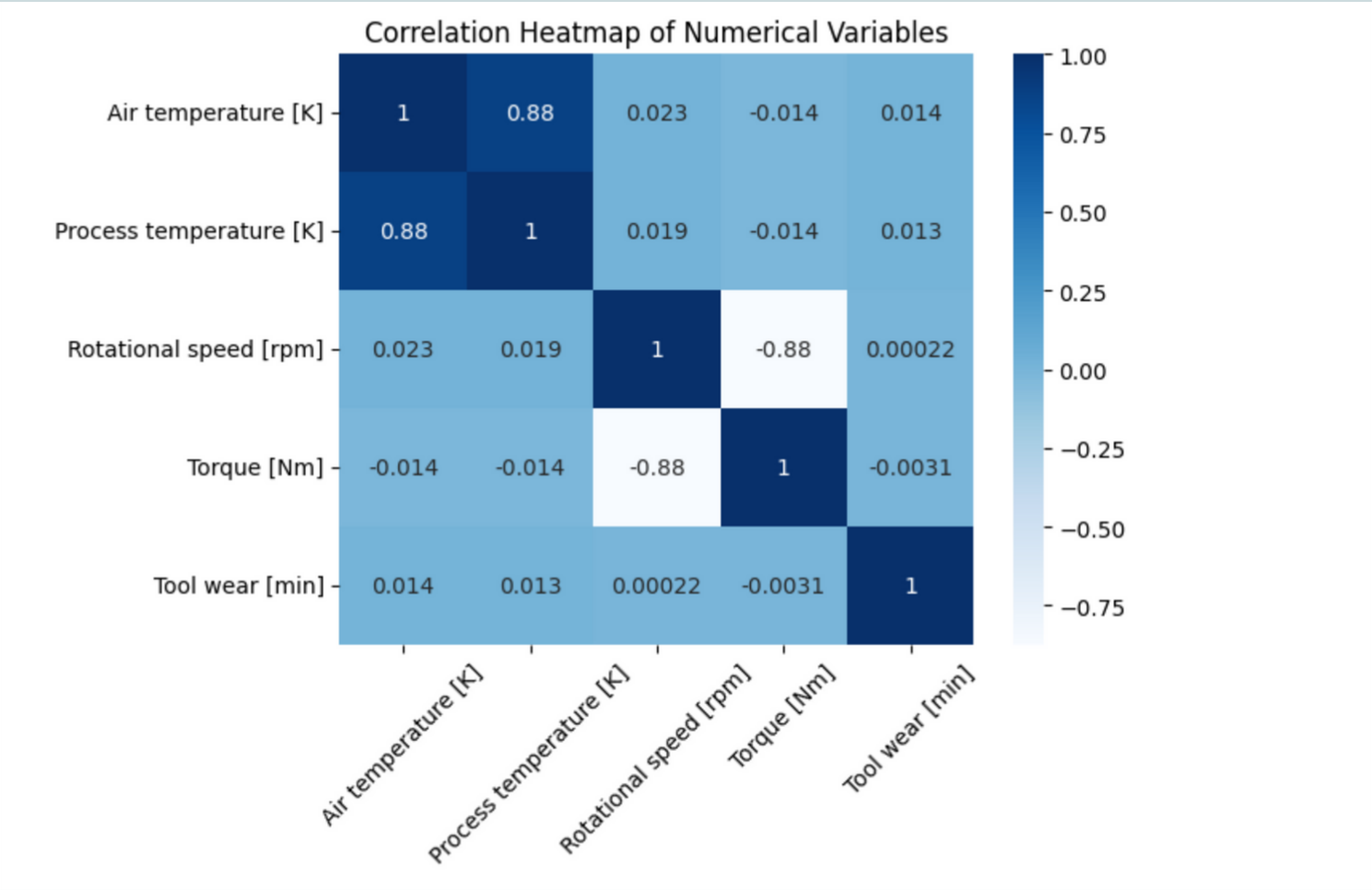
3rd STEP: Data Exploration

INPUT:

```python
# 3.3) Correlation Heatmap: Generate a correlation heatmap to visualize the relationships between different
# numerical variables (e.g., "Air temperature [K]", "Process temperature [K]", "Rotational speed [rpm]",
# "Torque [Nm]", "Tool wear [min]").
# This will help you identify any significant correlations between variables, which can be useful for feature selection.

import pandas as pd
import seaborn as sns  # requires pip install seaborn
import matplotlib.pyplot as plt

# Select the numerical variables for correlation analysis
numerical_vars = df[["Air temperature [K]", "Process temperature [K]", "Rotational speed [rpm]", "Torque [Nm]", "Tool wear [min]"]]

# Compute the correlation matrix
correlation_matrix = numerical_vars.corr()

# Generate the correlation heatmap
sns.heatmap(correlation_matrix, annot=True, cmap = 'Blues')
plt.xticks(rotation = 45)
plt.title('Correlation Heatmap of Numerical Variables')
plt.show()
```

3rd STEP: Data Exploration

OUTPUT:

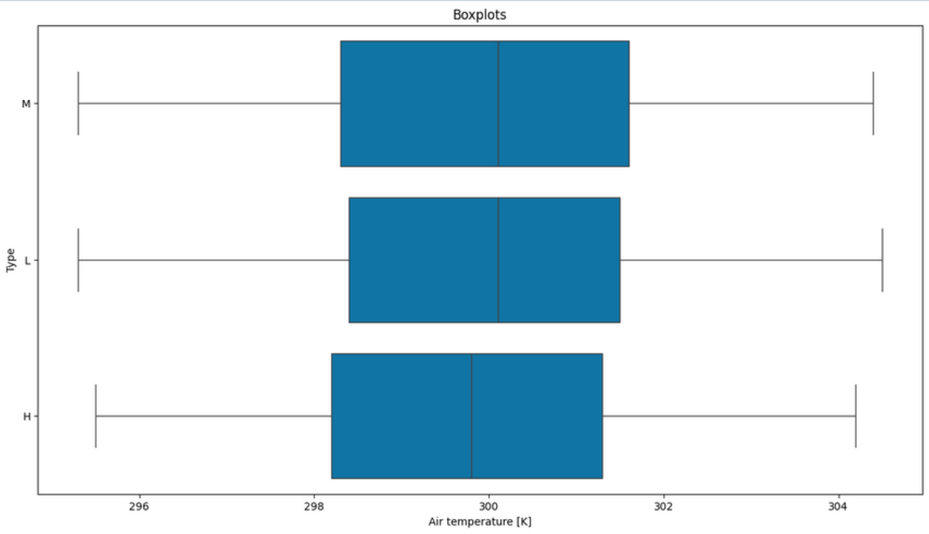


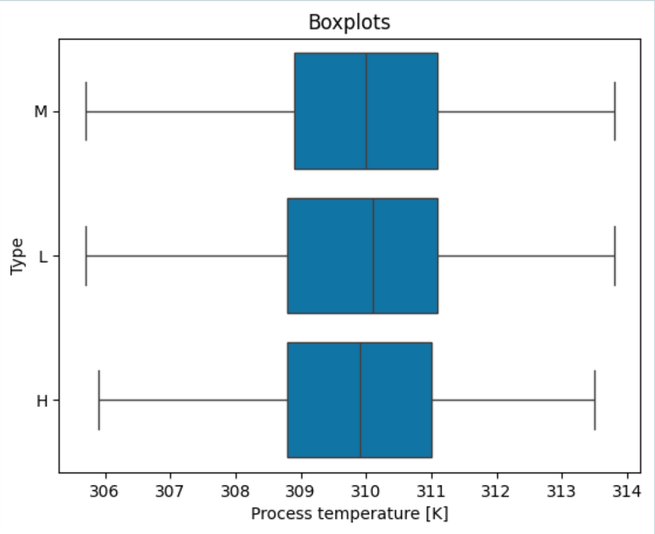
3rd STEP: Data Exploration

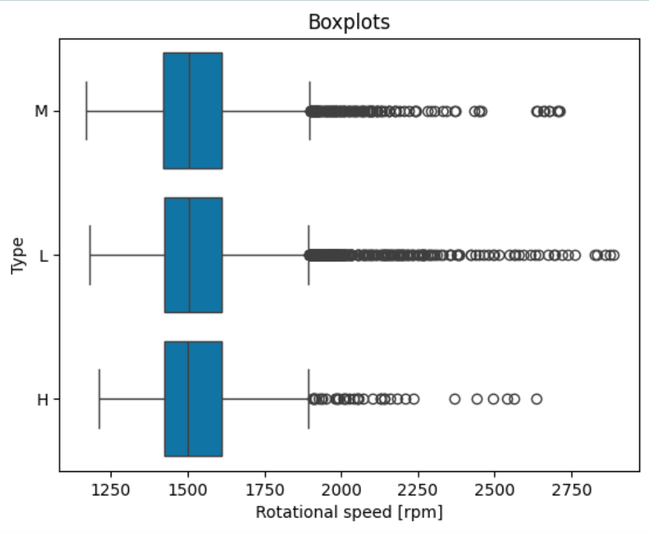
INPUT:

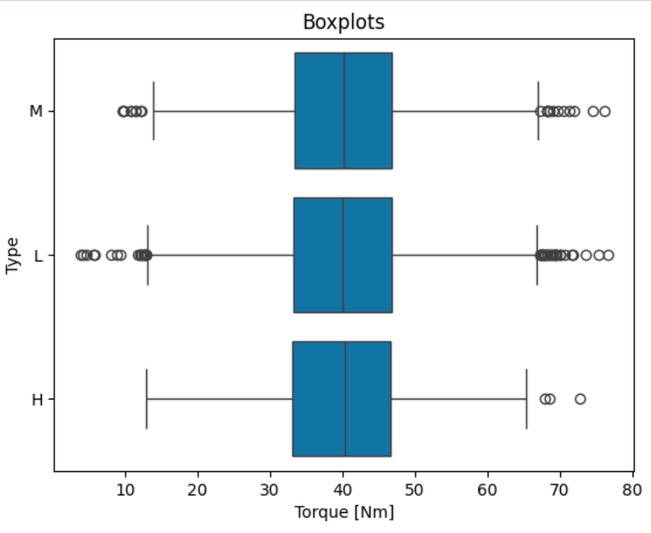
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Select the numerical and categorical variables for box plots
numerical_vars = ["Air temperature [K]", "Process temperature [K]", "Rotational speed [rpm]", "Torque [Nm]", "Tool wear [min]"]
categorical_vars = ["Type", "Failure Type"] # type and failure type

# Create box plots for numerical variables grouped by categorical variables

for cat_var in categorical_vars:

    for num_var in numerical_vars:
        plt.figure(figsize=(15, 8))
        sns.boxplot(x = num_var, y = cat_var, data = df)
        plt.title('Numerical variables grouped by categorical variables')
        plt.xlabel('Numerical var')
        plt.ylabel ('Categorical var')
        plt.show()
```

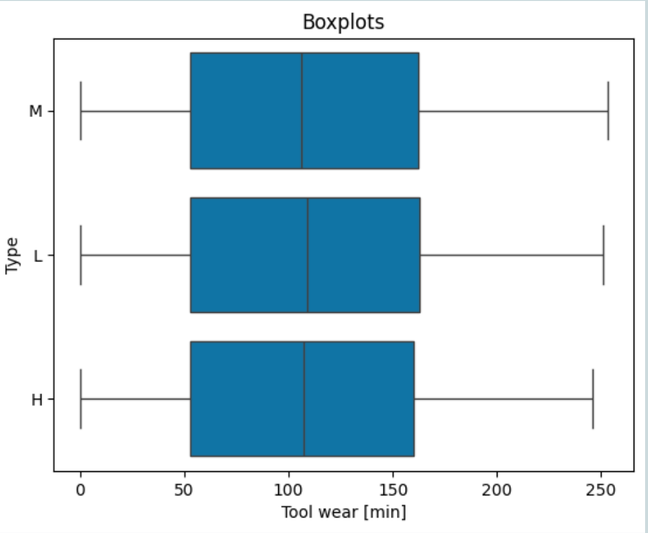
3rd STEP: Data Exploration



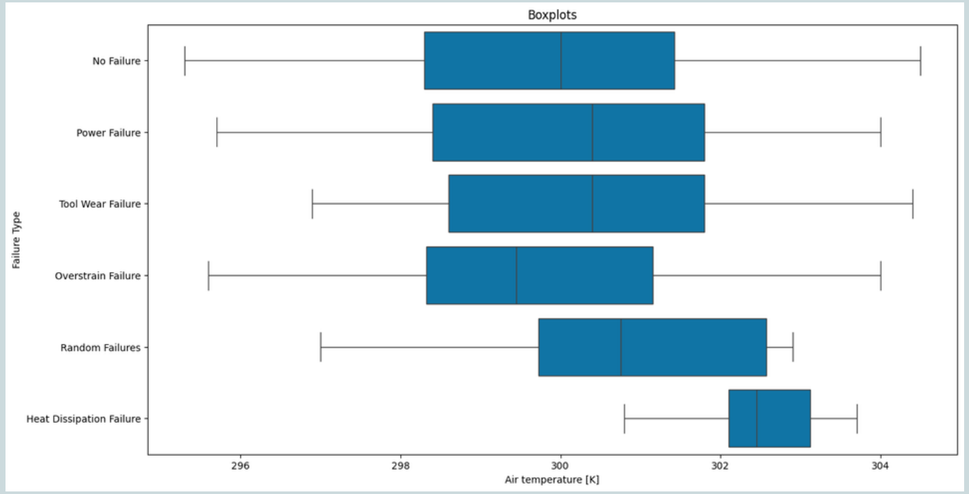


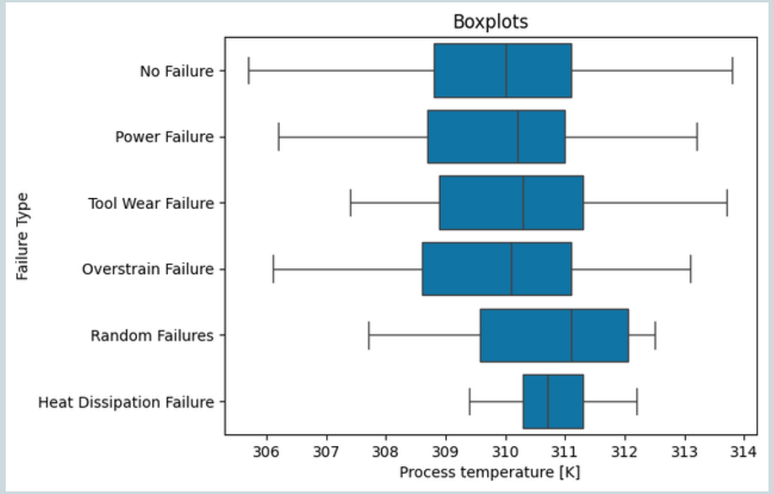


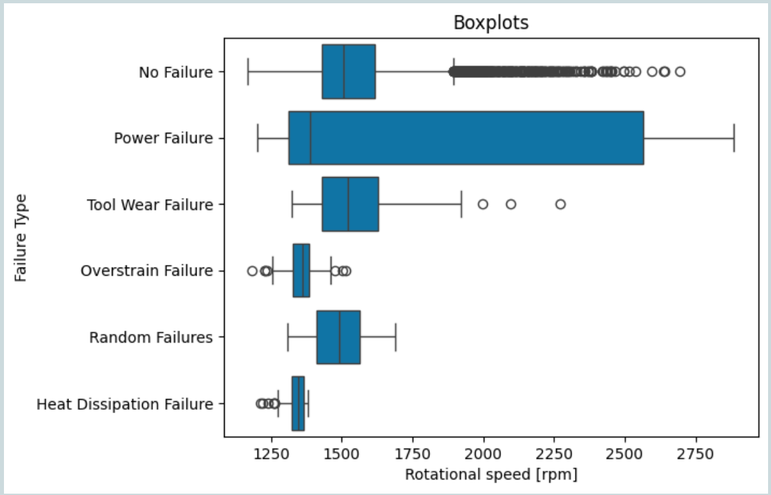


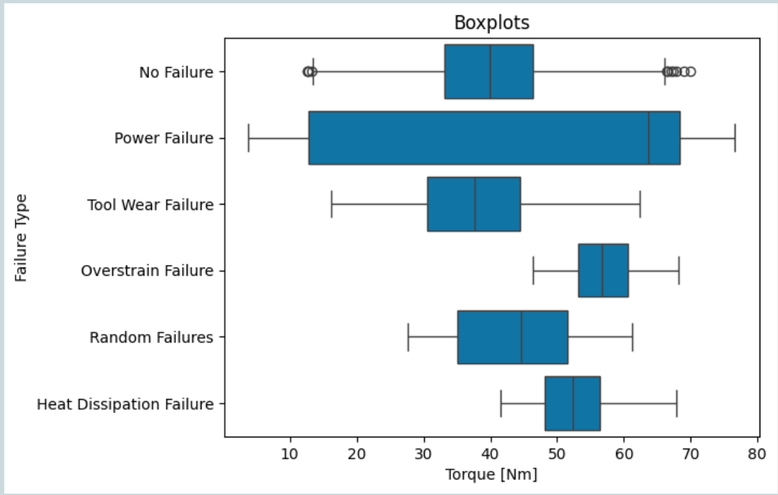


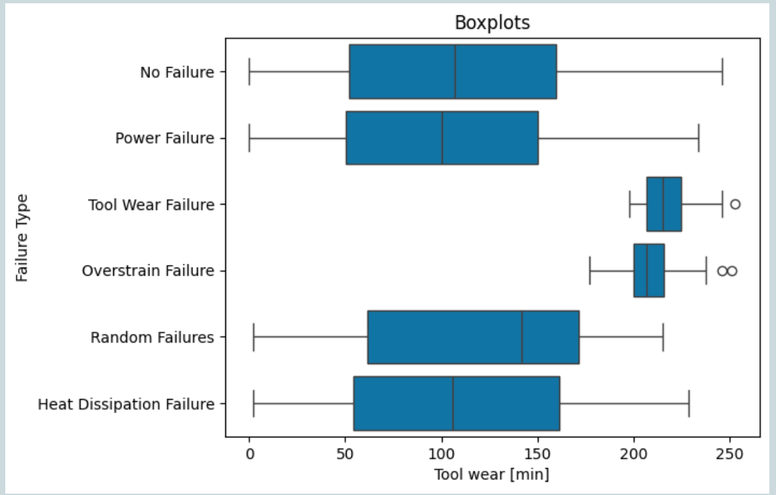
3rd STEP: Data Exploration











4th STEP: Building Classification Machine Learning Model

```
# 4.1 Create and fill two variables: features and label
# features: "Air temperature [K]", "Process temperature [K]", "Rotational speed [rpm]", "Torque [Nm]", "Tool wear [min]"
# label = "Target"  # Assuming "Target" is the binary classification target variable

features = ["Air temperature [K]", "Process temperature [K]", "Rotational speed [rpm]", "Torque [Nm]", "Tool wear [min]"]
label = ["Target"]
```

```
# 4.2 Training and Test Dataset Split
X = df[features]
y = df[label]

from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
[d.shape for d in (X_train, X_test, y_train, y_test)]
```

```
[(8000, 5), (2000, 5), (8000, 1), (2000, 1)]
```

4th STEP: Building Classification Machine Learning Model

INPUT:

```
# 4.3 Use the Random Forest classifier Algorithm to Build the Model

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state=42)

# Train the model
model.fit(X_train, y_train)
```

OUTPUT:

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Why Random Forest?

Random forest is a powerful machine-learning technique that has the potential to lead to better results than linear regression. It works by constructing numerous **decision trees**, which are much more powerful at capturing non-linear relationships between features and target variables than linear models.

5th STEP: Score the Model

INPUT:

```
# Make predictions on the test set
y_pred = model.predict(X_test)
y_pred
```

OUTPUT:

```
array([0, 0, 0, ..., 0, 1, 0])
```

6th STEP: Evaluate the Model

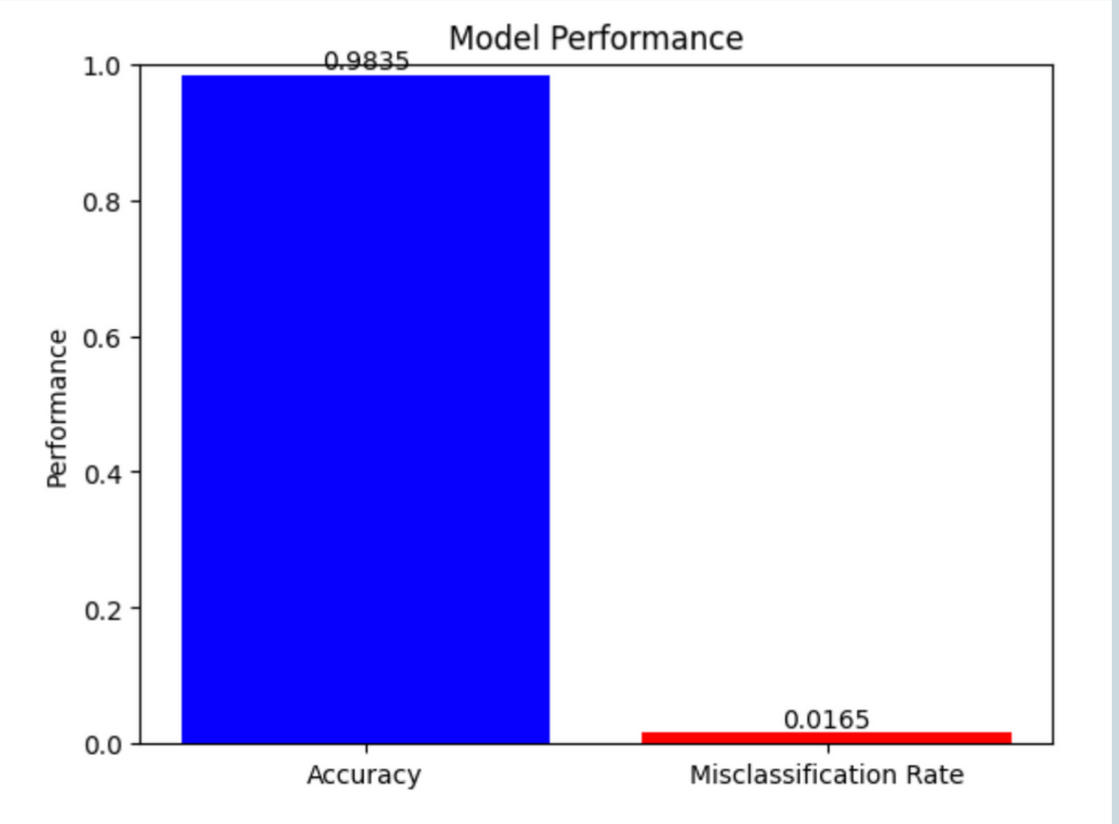
INPUT:

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

OUTPUT:

```
Accuracy: 0.9835
```

We got the **accuracy of 98%** which means that our model is pretty accurate as it predicted 98% of the values correctly.

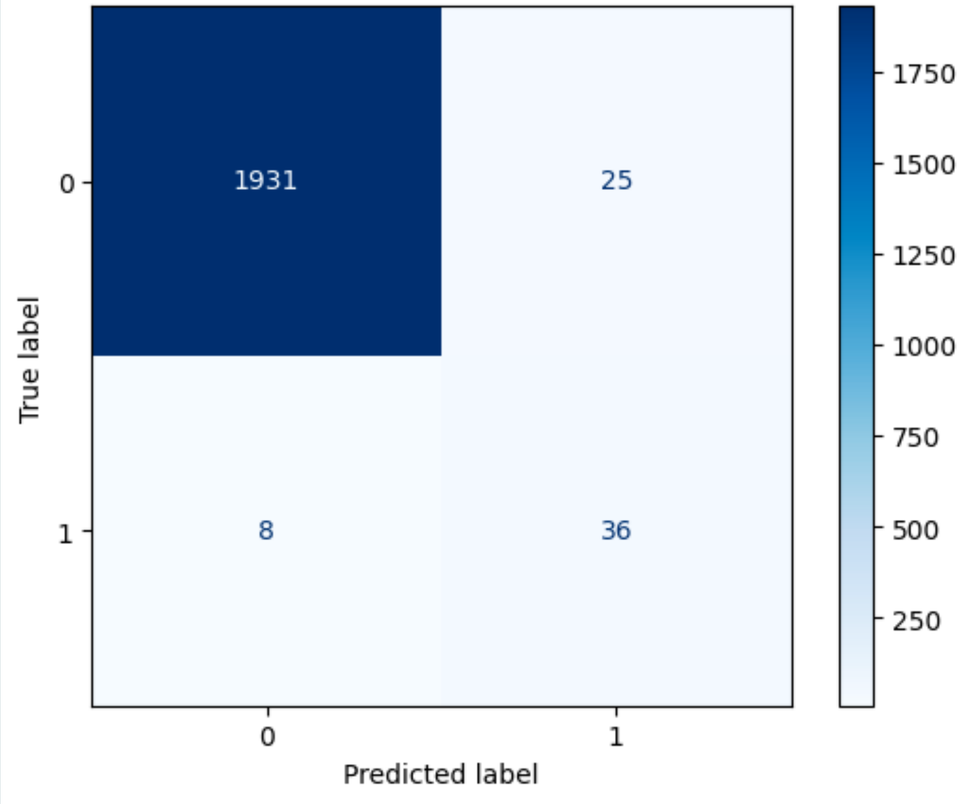


6th STEP: Evaluate the Model

INPUT:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

conf_mat = confusion_matrix(y_pred, y_test)
ConfusionMatrixDisplay(confusion_matrix=conf_mat).plot(cmap=plt.cm.Blues)
```

OUTPUT:



We have **1931 correctly predicted machinery "successes"** and **36 correctly predicted machinery failures**.

The most of the mistakes happened when classifying **25 records** as machinery failure when there were no failure .

6th STEP: Evaluate the Model

INPUT:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

OUTPUT:

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1939
           1       0.82      0.59      0.69        61

    accuracy                           0.98      2000
   macro avg       0.90      0.79      0.84      2000
weighted avg       0.98      0.98      0.98      2000
```

Precision shows that in **99%** of cases the model **correctly identified machinery 'success'** and in **82%** it **correctly identified machinery failure**.

Recall shows that **100% of machinery 'successes' were identified correctly** out of all predicted 'successes'. **59% of machinery failures were identified correctly** out of all predicted failures.

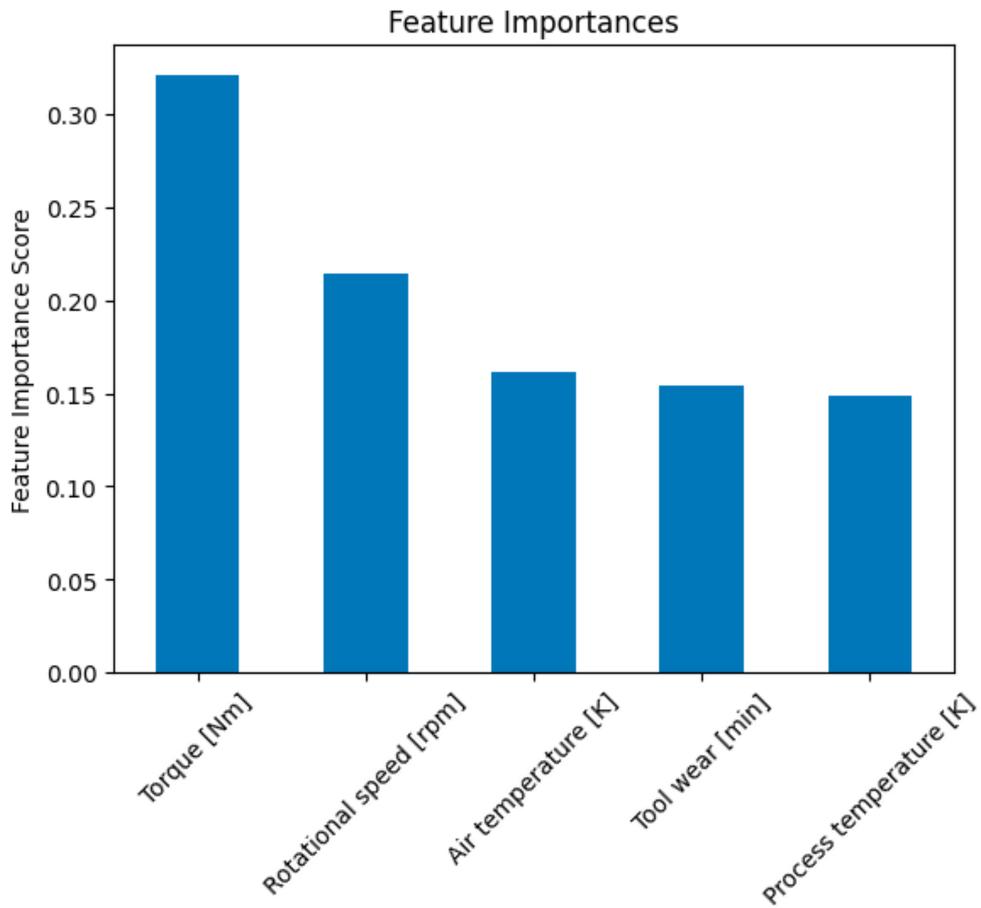
The F1 score for machinery 'successes' (99%) is higher than for failures (69%) which means that **our models predicts 'successes' better.**

6th STEP: Evaluate the Model

INPUT:

```
feature_importances = pd.Series(model.feature_importances_, index=X_train.columns).sort_values(ascending=False)

feature_importances.plot.bar()
plt.xticks(rotation=45)
```

OUTPUT:



The model is mostly considering the **Torque (33%)**, then Rotation speed (22%), Air temperature (16%), Tool wear (15%), and Process temperature (14%) when predicting whether the machinery failure will happen or not.

Recommendations for customers

1. **Air temp:** Maintain air temperature within optimal operational ranges.
2. **Process temp:** Implement tighter temperature control for machinery types with sensitivity.
3. **Rotational Speed.** Adjust operations of rotation. Investigate outliers for issues.
4. **Torque**. Standardize torque settings according to machinery type
5. **Tool Wear:** Schedule maintenance and tool replacement

Thanks for the attention