Machine Predication Maintenance

Hammad Anwar
Computer Science
Shaheed Zulfikar Ali Bhutto Institute of Science and Technology
Karachi, Pakistan
bscs2012119@szabist.pk

***Abstract*—The main goal of this project is to forecast machine failures. The dataset contains variables such as Product ID, Type, Air Temperature, Process Temperature, Rotational Speed, Torque, and Tool Wear Minutes. It also includes a 'Target' variable that indicates whether or not a machine failed. The basic purpose is to investigate elements such as temperature, speed, and tool wear to identify patterns that can help predict whether the machines fails or not and also future machine problems. The intention is to assist businesses in improving the way they maintain their machinery and prevent unexpected failures. We intend to provide useful insights by analyzing data using specific methods.**

I. Introduction

In the dynamic world of industrial operations, assuring the dependability and efficiency of machines is critical. This introduction provides the groundwork for exploring the novel idea of predictive maintenance, a strategy that goes beyond traditional reactive techniques. Predictive maintenance aims to identify and prevent problems before they have an impact on operations, as opposed to only addressing them when they happen. The use of advanced tools like the Random Forest model, which can be compared to an incredibly intelligent detective that evaluates vast amounts of data to identify the type of failure that a machine might be experiencing and whether it is occurring at all, is the fundamental component of this transformative approach. Finally, this could improve machine dependability and effectiveness by changing maintenance from reactive modifications to anticipating and preventing problems before they disrupt operations.

As we begin this journey, the narrative unfolds to show how predictive maintenance, powered by tools like the Random Forest model, has the ability to reshape the landscape of machinery care. It represents not simply a methodological change but also a fundamental shift in how we see industrial maintenance. We will discuss how these techniques can help ensure that machines run smoothly, prevent machine breakdowns proactively, and increase maintenance's role as a valuable resource in reaching operational excellence.

II. Literature Review

In industrial settings, predictive maintenance has developed as a disruptive paradigm with the goal of revolutionizing existing reactive procedures by proactively identifying and treating potential machinery issues. Examining important research and developments in the field, this overview of the literature focuses on predictive maintenance techniques and the use of machine learning models—most particularly the Random Forest model.

1) Predictive Maintenance Strategie: It makes more sense to anticipate when a machine might break rather than waiting for it to happen and then fixing it. According to research by Li, Zhang, and Wang et al., data can be used to anticipate problems by monitoring machine performance and making necessary adjustments on an as-needed basis rather than merely according to the schedule.

2) Machine Learning in Predictive Maintenance: With its ability to provide a data-driven method of decision-making, machine learning models have become essential to predictive maintenance. Notably, the ensemble learning algorithm Random Forest has shown effective in managing complex datasets and forecasting failures of equipment. Studies conducted by Sun et al. (2019) and Jha et al. (2020) demonstrate the effective use of Random Forest in various industrial contexts.

3) Real-World Applications: Predictive maintenance has numerous practical uses, as demonstrated by studies like Ramesh et al. (2020), which highlight how it can lower maintenance costs, increase dependability, and improve overall operational efficiency.

When these results are combined, it is clear that predictive maintenance, especially when using the Random Forest model, has enormous potential to change the way industrial machinery is managed. The knowledge gained from these studies shows how industries are dealing with machinery challenges. This will help them achieve increased efficiency, reduced downtime, and long-term operational success.

III. Problem Statement

The key problem in machine predictive maintenance is transitioning from previous reactive approaches to proactive strategies. The critical requirement is to accurately predict and recognize future failures. Ensuring model correctness, bridging the gap between predicted insights, and navigating data analysis difficulties are all necessary to meet this challenge. The ultimate goal is to increase machine dependability and operating efficiency by combining predictive maintenance approaches in a seamless manner, increasing efficiency, reducing downtime, and achieving long-term operational excellence.

IV. Methodology

The methodology includes the use of the Random Forest model for predictive maintenance. Preprocessing the data, training the model, and validation are all part of this. The approach incorporates findings from the literature review, as well as methodologies that have proven effective in real-world applications. This ensures that the predictive model is dependable and efficient in predicting machine failures by comparing its performance to historical data.

1) Data Collection: We have taken the machine predictive maintenance dataset from Kaggle.

2) Data Cleaning: By deleting null values, unneeded columns, and altering the data types of some columns, we ensured that the data we had was clean and organized.

3) Data Understanding: In order to identify patterns that can help us predict when a machine might experience problems, we thoroughly analyzed the data to determine how various things are related.

4) Data Visulaization: To have the better understanding of the data, we have used several plots and charts such as histograms, pie charts, scatter plots, heatmaps, and box plots.

5) Model Selection: Based on the features of the dataset we chose the Random Forest model for predicting the machine failure. Its capacity for group learning and track record of successfully managing complicated datasets matched with our requirements.

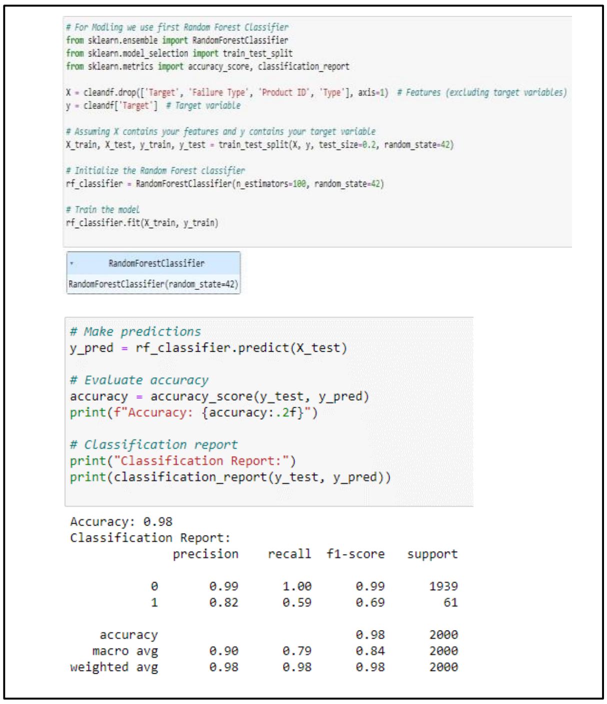
6) Model Trainig: We have used the Random Forest model to forecast machine issues during the model training phase. The dataset was divided into training and testing sets. After that, the model was trained using the training set, which helped it discover relationships and patterns among different features as well as the probability of machine failures.

7) Evalution Matrics: The evaluation phase focused on assessing the model's performance using specific metrics. Accuracy, precision, recall and F1 score, were employed followed by the importance of the features.

V. CONCLUSION

In conclusion, this predictive maintenance project offers valuable insights into enhancing industrial machinery reliability. Through meticulous data analysis, model training, and evaluation, we have developed a proactive framework capable of anticipating potential machine failures. The model, tuned to recognize intricate patterns within the dataset, demonstrates proficiency in predicting issues before they escalate. The comprehensive evaluation metrics assure the model's reliability, with considerations for accuracy, precision, recall, and F1 score providing a nuanced understanding of its strengths. The practical recommendations derived from the insights aim to streamline operational procedures and prevent costly downtimes. By fostering a proactive approach to maintenance, we pave the way for improved machinery dependability, reduced operational disruptions, and substantial cost savings.

VI. RESULTS



```
# For Modling we use first Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

X = cleandf.drop(['Target', 'Failure Type', 'Product ID', 'Type'], axis=1)  # Features (excluding target variables)
y = cleandf['Target']  # Target variable

# Assuming X contains your features and y contains your target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

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

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

```
# Make predictions
y_pred = rf_classifier.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

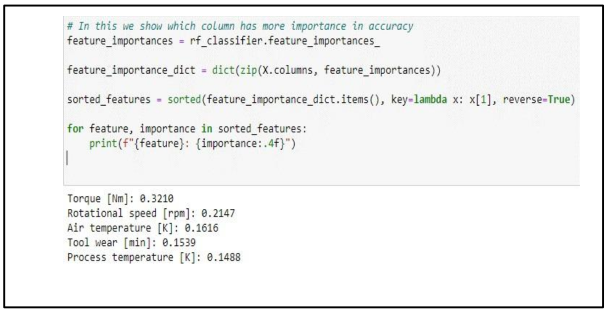
```
Accuracy: 0.98
Classification Report:
              precision    recall  f1-score   support

           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
```

Fig. 1. In this code show modling of dataset using Random Forest Classifier.

In the provided code snippet Fig 1, a Random Forest Classifier is utilized in predictive modelling with a dataset related to machine predictive maintenance. The preprocessing of the dataset involves the removal of the columns labelled "Target" and "Failure Type," as well as the categorical variables "Product ID" and "Type," from the feature set (X). The variable labelled "Target" is then assigned to the label set (y). The train_test_split function is then used to divide the dataset into training and testing sets. After being trained on the training set, the Random Forest classifier is initialized with 100 estimators. After that, predictions are made on the test set, and the accuracy score function is used to determine how accurate the model is. A comprehensive classification report is also produced, offering insights into metrics such as precision, recall, and F1 score.



```
# In this we show which column has more importance in accuracy
feature_importances = rf_classifier.feature_importances_

feature_importance_dict = dict(zip(X.columns, feature_importances))

sorted_features = sorted(feature_importance_dict.items(), key=lambda x: x[1], reverse=True)

for feature, importance in sorted_features:
    print(f"{feature}: {importance:.4f}")
```

```
Torque [Nm]: 0.3210
Rotational speed [rpm]: 0.2147
Air temperature [K]: 0.1616
Tool wear [min]: 0.1539
Process temperature [K]: 0.1488
```

Fig. 2. This code show the columns has that has more contribution in the predication.

In the provided code snippet Fig 2, focuses on estimating a Random Forest Classifier's feature importance, which aids in identifying the columns that most significantly contribute to the accuracy of the model. The feature's importance is extracted using the RandomForestClassifier instance, "rf_classifier," and they are subsequently paired with the corresponding column names. One can easily recognize which variables are more important for the model's predictive accuracy because the features are arranged in descending order of significance. The output that is produced shows each feature and its corresponding importance score, offering important information about how various input variables affect the Random Forest model's overall performance. The decision-making process of the model can be more easily understood by this information.

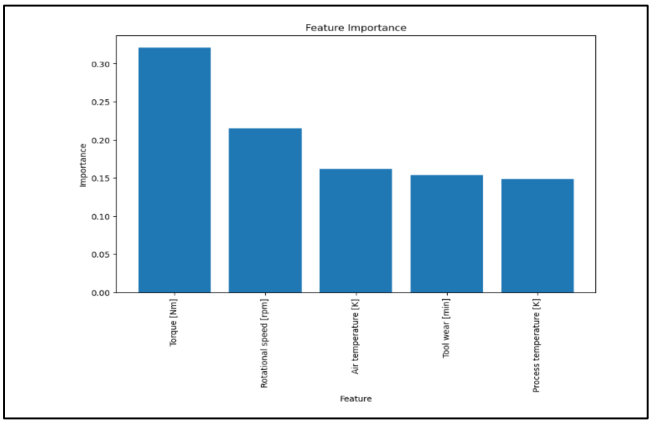


Fig. 3. Show the important variable in Histogram chart

In the Fig 3, The chart represents the feature importance's calculated by a random forest classifier for specific features. A higher value indicates a greater influence on the model's predictions. In this case, "Torque [Nm]" is the most important feature (0.3210), followed by "Rotational speed [rpm]" (0.2147), "Air temperature [K]" (0.1616), "Tool wear [min]" (0.1539), and "Process temperature [K]" (0.1488). These values help identify which features contribute more significantly to the model's decision-making.

```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20]}

# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Initialize GridSearchCV
grid_search = GridSearchCV(rf_classifier, param_grid, cv=3, scoring='accuracy')

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

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

Best Parameters: {'max_depth': None, 'n_estimators': 200}
```

Fig. 4. This code performs hyper parameters

In the Fig 4, GridSearchCV is used to tune the hyperparameters of a Random Forest Classifier. It experiments with different combinations of 'n_estimators' and'max_depth.' GridSearchCV is used to tune the hyperparameters of a Random Forest Classifier. It experiments with different combinations of 'n_estimators' and'max_depth.' GridSearchCV performs a cross-validated search using accuracy as the metric once the classifier is initialized with a random state. The optimal hyperparameter combination is printed when the grid search is fitted to the training data. This procedure improves the performance of the Random Forest model.

```
In [35]: failure_types = ['No Failure', 'Failure']

         # Input for new data point
         new_data_point = {'Air temperature [K]': float(input('Enter Air temperature [K]: ')),
                           'Process temperature [K]': float(input('Enter Process temperature [K]: ')),
                           'Rotational speed [rpm]': float(input('Enter Rotational speed [rpm]: ')),
                           'Torque [Nm]': float(input('Enter Torque [Nm]: ')),
                           'Tool wear [min]': float(input('Enter Tool wear [min]: '))}

         # Create a DataFrame from the new data point
         new_data_df = pd.DataFrame([new_data_point])

         # Use the model to make predictions
         predicted_failure = rf_classifier.predict(new_data_df)[0]

         # Use the model to predict the probabilities
         failure_probabilities = rf_classifier.predict_proba(new_data_df)[0]

         # Print the prediction and probabilities
         print(f'Predicted Failure: {failure_types[predicted_failure]}')
         print(f'Chances of Failure: {failure_probabilities[1]*100:.2f}%')

         # Print the predicted failure type and probabilities for each type
         print('Probabilities of Each Target:')
         for i, probability in enumerate(failure_probabilities):
             print(f'{failure_types[i]}: {probability * 100:.2f}%')

         Enter Air temperature [K]: 297.5
         Enter Process temperature [K]: 308.3
         Enter Rotational speed [rpm]: 2564
         Enter Torque [Nm]: 12.8
         Enter Tool wear [min]: 127
         Predicted Failure: Failure
         Chances of Failure: 92.00%
         Probabilities of Each Target:
         No Failure: 8.00%
         Failure: 92.00%
```

Fig. 5. This code show the predictive analysis of predictive maintenance.

In the Fig 5, the user is requested to provide values for essential process features (such as Air temperature, process temperature, torque, etc), which are then arranged into a DataFrame. The trained classifier is then used to forecast the failure and related probability for the given data point. The outcomes are printed, including the predicted failure, the probability of failure, and the distribution of probabilities for each failure type.

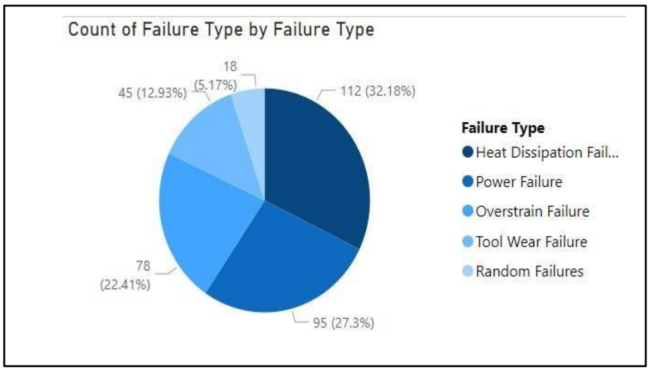


Fig. 6. Pie chart of different failure types.

In Fig 6, The chart depicts many failure types, such as Heat Dissipation Failure, Power Failure, Tool Wear Failure, Overstrain Failure, and Random Failures. Each form of failure is represented by a segment, with the size of each segment representing the proportion of the overall distribution it holds.

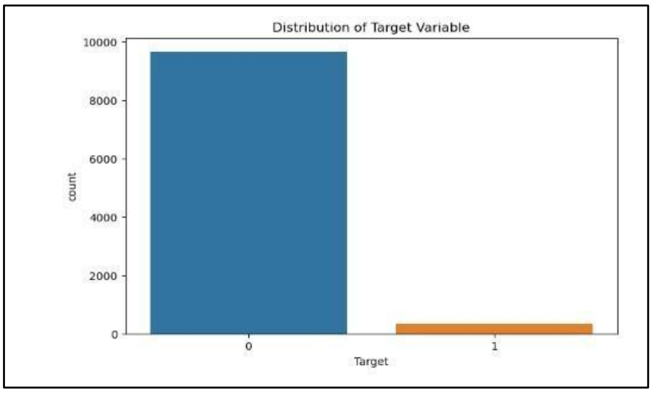


Fig. 7. Count Plot for target variable.

In Fig 7, the count plot visualizes the distribution of values in the 'Target' variable (where 1 represents no failure and 0 represents failure). Each bar represents a category, and the height of the bar corresponds to the frequency of occurrences of each category. "Target" probably indicates if a failure happened, but the count plot shows us how many examples fall into each category and gives us information about how our dataset's outcomes are distributed generally.

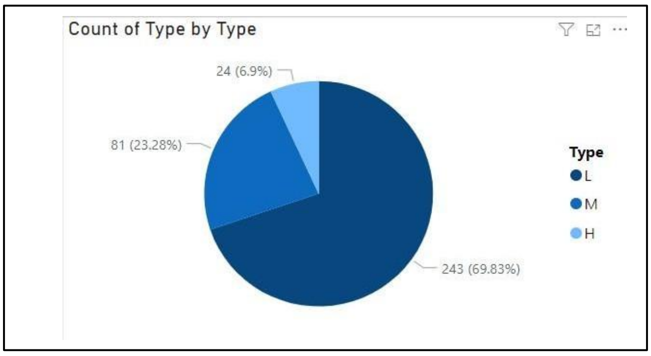


Fig. 8. Pie chart of Type variable.

In Fig 8, the pie chart depicts the distribution of distinct kinds—High (H), Medium (M), and Low (L) within the dataset. Every segment of the pie denotes a particular category, and its size indicates the proportion or percentage of occurrences of that type throughout the whole dataset. Here, Low occupies a larger portion of the pie, it signifies a higher frequency of occurrences for Low product type.

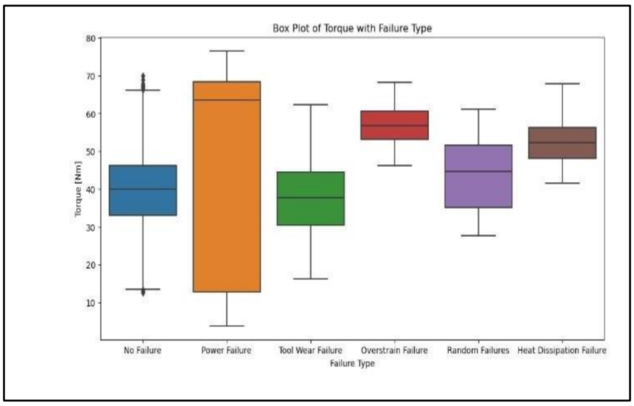


Fig. 9. Box plot of Torque with failure types.

The box plot displays the distribution of torque values across distinct failure categories in the dataset. The torque values are represented by the vertical axis, and each box in the diagram denotes a distinct type of failure. We may evaluate the torque values' central tendency, dispersion, and variability for each type of failure by looking at the box plot.

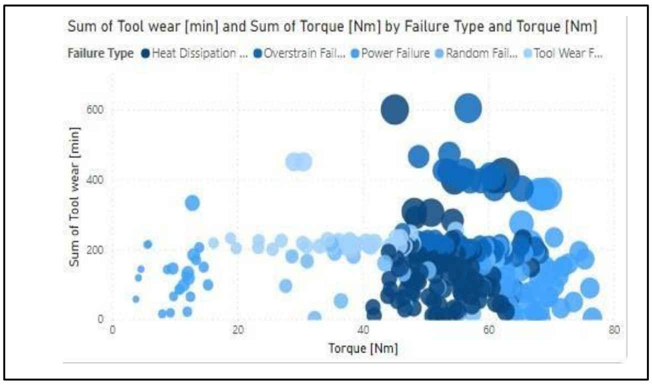


Fig. 10. Bubble chart.

The bubble chart depicts the link between the sum of tool wear and torque, as well as their correlation with various failure types. The torque and the total tool wear are shown on the chart's x- and y-axes, respectively. A distinct failure type, torque, and the associated sum of tool wear are shown by each point on the chart. The larger the bubble, the greater the sum of tool wear associated with that specific failure type and torque combination.

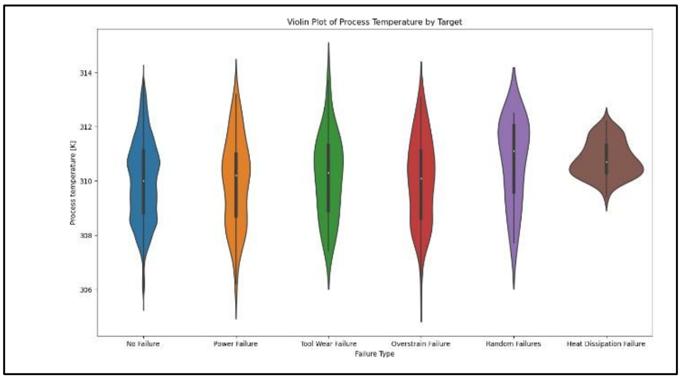


Fig. 11. Violin plot of Process Temperature with failure types.

The violin plot displays the distribution of process temperature values across distinct failure categories in the dataset. The values are represented by the vertical axis, and each box in the diagram denotes a distinct type of failure. It provides insights into the central tendency, spread, and shape of the distributions, helping to identify patterns and potential differences in process temperatures between instances of failure and non-failure.

References

[1] https://www.researchgate.net/publication/277723565_Machine_Learning_for_Predictive_Maintenance_A_Multiple_Classifier_Approach

[2] https://www.kaggle.com/datasets/shivamb/machine-predictive-maintenance-classification

[3] https://iopscience.iop.org/article/10.1088/1757-899X/954/1/012001

[4] https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/#:~:text=Correlation%20heatmaps%20are%20a%20type,the%20strength%20of%20this%20relationship