IE 7374 Statistical Learning for Engineering

Predictive Maintenance Analysis

GROUP 11

Aniket Sakharkar

Mayur Mahanta

Jwalit Shah

Dev Patel

857-230-5126 (Aniket Sakharkar)

857-437-9190 (Mayur Mahanta)

732-209-2041 (Jwalit Shah)

978-710-1327 (Dev Patel)

sakharkar.a@northeastern.edu

mahanta.m@northeastern.edu

shah.jwa@northeatern.edu

patel.dev4@northeastern.edu

Signature of Aniket: Aniket

Signature of Mayur: Mayur

Signature of Jwalit: Jwalit

Signature of Dev: Dev

Submission Date: 4th December 2022

Abstract:

Predictive Maintenance refers to the use of data driven and proactive maintenance methods that are designed to analyse the condition of equipment in the industry and help predict when maintenance is required. Predictive maintenance software uses data science and predictive analytics to estimate when a piece of equipment might fail so that corrective maintenance can be scheduled before the point of failure. The goal is to schedule maintenance at the most convenient and most cost-efficient moment, allowing the equipment's lifespan to be optimized to its fullest, but before the equipment has been compromised. Indeed, accurately modelling if a machine will break is crucial for industrial and manufacturing businesses as it can be beneficial in the following ways:

- Maintain a safe work environment by ensuring that machines are working properly.
- Increase productivity by preventing unplanned reactive maintenance and minimizing downtime.
- Optimize costs by removing the need for too many unnecessary checks or repairs of components.

Fig 1: Machine at work indicating they require maintenance over time

Introduction:

Predictive Maintenance extracts insights from data generated by shop floor equipment and then acts on these insights. It was first proposed in the early 1990s. Predictive Maintenance supplements routine preventive maintenance. Early on, it was difficult to implement Predictive Maintenance due to a lack of sensors to generate data and computational resources to gather and analyse data.

Predictive Maintenance can now enter the mainstream due to advancements on the Internet of Things (IoT), cloud computing, data analytics, and machine learning. Data from sensors that monitor the equipment, as well as other operational data, are required by Predictive Maintenance. The data is analysed and stored by the Predictive Maintenance system. Humans acts based on the analysis of these predictions.

After providing some context, we will discuss how to implement the various components of a Predictive Maintenance solution using a combination of on-premises data and machine learning models. Because Predictive Maintenance relies heavily on data to make decisions, we begin with data collection. The data must be collected and then used to evaluate what is currently happening as well as to build better predictive models in the future.

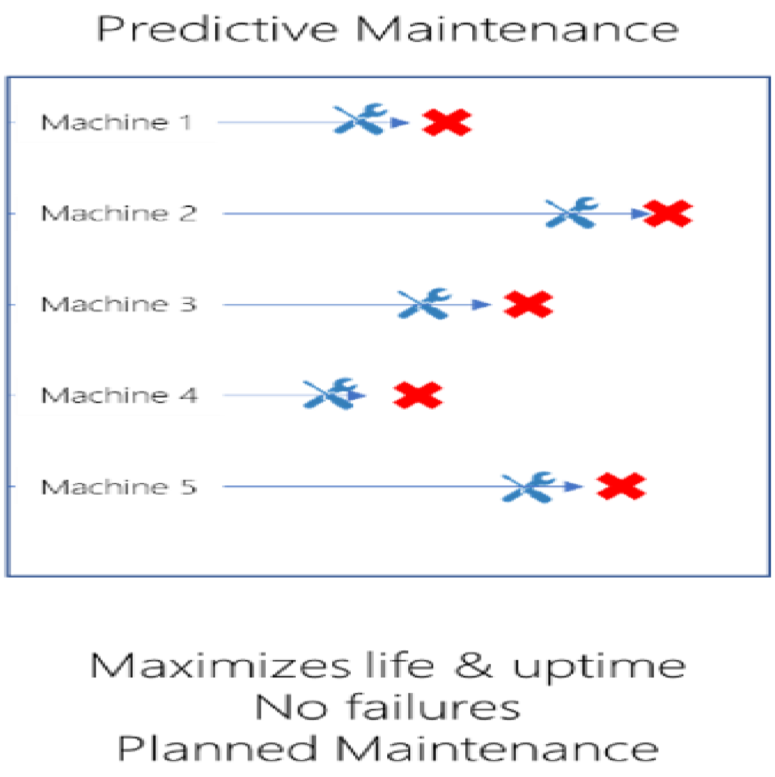


Fig 2: Demonstrating Maintenance Impact on different Machines

Predictive Maintenance employs models to predict when an asset's component is likely to fail, allowing for just-in-time maintenance to be scheduled. Predictive Maintenance outperforms previous strategies by increasing both uptime and asset life.

Because you service the equipment at times when the component's maximum lifetime is approaching, you spend less money replacing working parts.

The disadvantage is that the just-in-time nature of Predictive Maintenance makes it more difficult to implement because it necessitates a more responsive and flexible services organization.

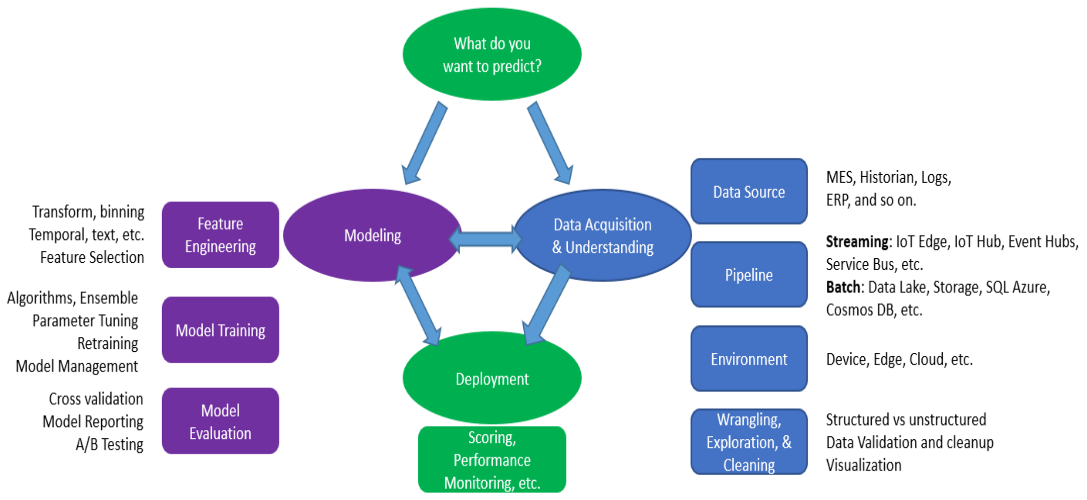

What do you want to predict?
Transform, binning
Temporal, text, etc.
Feature Selection
Feature Engineering
Modeling
Data Acquisition & Understanding
Data Source
MES, Historian, Logs, ERP, and so on.
Pipeline
Streaming: IoT Edge, IoT Hub, Event Hubs, Service Bus, etc.
Batch: Data Lake, Storage, SQL Azure, Cosmos DB, etc.
Algorithms, Ensemble
Parameter Tuning
Retraining
Model Management
Model Training
Environment
Device, Edge, Cloud, etc.
Deployment
Cross validation
Model Reporting
A/B Testing
Model Evaluation
Scoring, Performance Monitoring, etc.
Wrangling, Exploration, & Cleaning
Structured vs unstructured
Data Validation and cleanup
Visualization

Fig 3: Process Flow (Partial Overview of our Project Steps)

The following diagram illustrates the steps in a data science project model. The process is customized for manufacturing and does an excellent job of explaining the various concerns that one has when building and executing machine learning models.

Data Description:

The Predictive Maintenance Dataset from the UCI ML website. It is a synthetic dataset that reflects real predictive maintenance encountered in industry to the best of our knowledge as real predictive maintenance datasets are generally difficult to obtain and difficult to publish

Rows: - 10K Columns: - 14

Data Source:

UCI Machine Learning Repository: AI4I 2020 Predictive Maintenance Dataset Data Set

Data Features:

Features	Description
UID	unique identifier ranging from 1 to 10000
product ID	consisting of a letter L, M, or H for low (50% of all products), medium (30%) and high (20%) as product quality variants and a variant-specific serial number
air temperature [K]	generated using a random walk process later normalized to a standard deviation of 2 K around 300 K
process temperature [K]	generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K.
rotational speed [rpm]	calculated from a power of 2860 W, overlaid with a normally distributed noise
torque [Nm]	torque values are normally distributed around 40 Nm with a Ïƒ = 10 Nm and no negative values.
tool wear [min]:	The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process. and a
'machine failure'	label that indicates, whether the machine has failed in this particular datapoint for any of the following failure modes are true.
tool wear failure (TWF):	the tool will be replaced of fail at a randomly selected tool wear time between 200 â€“ 240 mins (120 times in our dataset). At this point in time, the tool is replaced 69 times, and fails 51 times (randomly assigned).
heat dissipation failure (HDF)	heat dissipation causes a process failure, if the difference between air- and process temperature is below 8.6 K and the toolâ€™s rotational speed is below 1380 rpm. This is the case for 115 data points.
power failure (PWF)	the product of torque and rotational speed (in rad/s) equals the power required for the process. If this power is below 3500 W or above 9000 W, the process fails, which is the case 95 times in our dataset.
random failures (RNF)	each process has a chance of 0,1 % to fail regardless of its process parameters. This is the case for only 5 datapoints, less than could be expected for 10,000 datapoints in our dataset
overstrain failure (OSF	if the product of tool wear and torque exceeds 11,000 minNm for the L product variant (12,000 M, 13,000 H), the process fails due to overstrain. This is true for 98 datapoints.

Method (Model) Description

The Models we will be implementing in our project are as follows:

1. **Logistic Regression:**

   This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

   Advantages: -

   - Logistic regression is easier to implement, and interpret, and very efficient to train.
   - It makes no assumptions about distributions of classes in feature space.

Disadvantages: -

- If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting
- It constructs linear boundaries.

2. **Naïve Bayes:**
   We use the Naive Bayes machine learning algorithm to solve classification problems. The Bayes Theorem underpins it. It is one of the most basic yet powerful ML algorithms in use, with applications in a wide range of industries. For example: - Assume you're working on a classification problem and have created the features and hypothesis, but your bosses want to see the model. To train the dataset, you have many data points (thousands of data points) and many variables. The Naive Bayes classifier, which is much faster than other classification algorithms, is the best solution for this situation.

   Advantages: -

   - This algorithm works quickly and can save a lot of time.
   - Naive Bayes is suitable for solving multi-class prediction problems.

Disadvantages: -

- Naive Bayes assumes that all predictors (or features) are independent, which rarely happens in real life. This limits the applicability of this algorithm in real-world use cases.
- Its estimations can be wrong in some cases, so you shouldn't take its probability outputs very seriously.

3. Decision Tree Classifier

A decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Advantages: -

- Decision trees can be used to predict both continuous and discrete values i.e. they work well in both regression and classification tasks.
- As decision trees are simple hence, they require less effort for understanding an algorithm.

Disadvantages: -

- The time complexity right for operating this operation is very huge and keeps on increasing as the number of records gets increased decision tree with numerical variables takes a lot of time for training.
- If the size of the data is too big, then one single tree may grow a lot of nodes which might result in complexity and leads to overfitting.

4. Support Vector Machine (SVM): -

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

Advantages: -

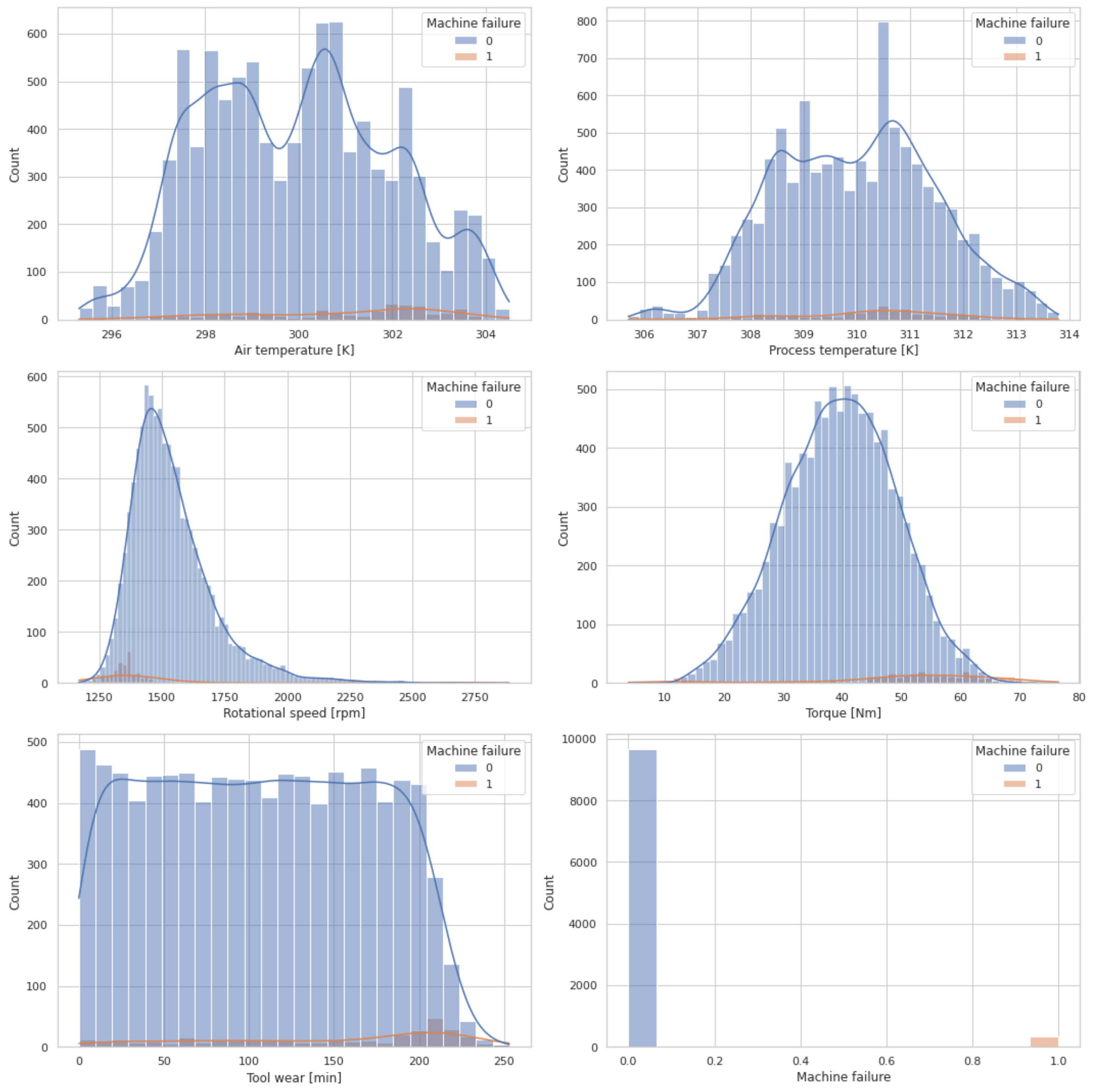
- SVM has L2 norm. So, it has good generalization capabilities which prevent it from over-fitting.
- SVM can efficiently handle non-linear data using the Kernel trick.

Disadvantages: -

- SVM takes a long training time on large datasets.
- Choosing an appropriate Kernel function (to handle the non-linear data) is not an easy task. It could be tricky and complex. In case of using a high dimension Kernel, you might generate too many support vectors which reduce the training speed drastically.

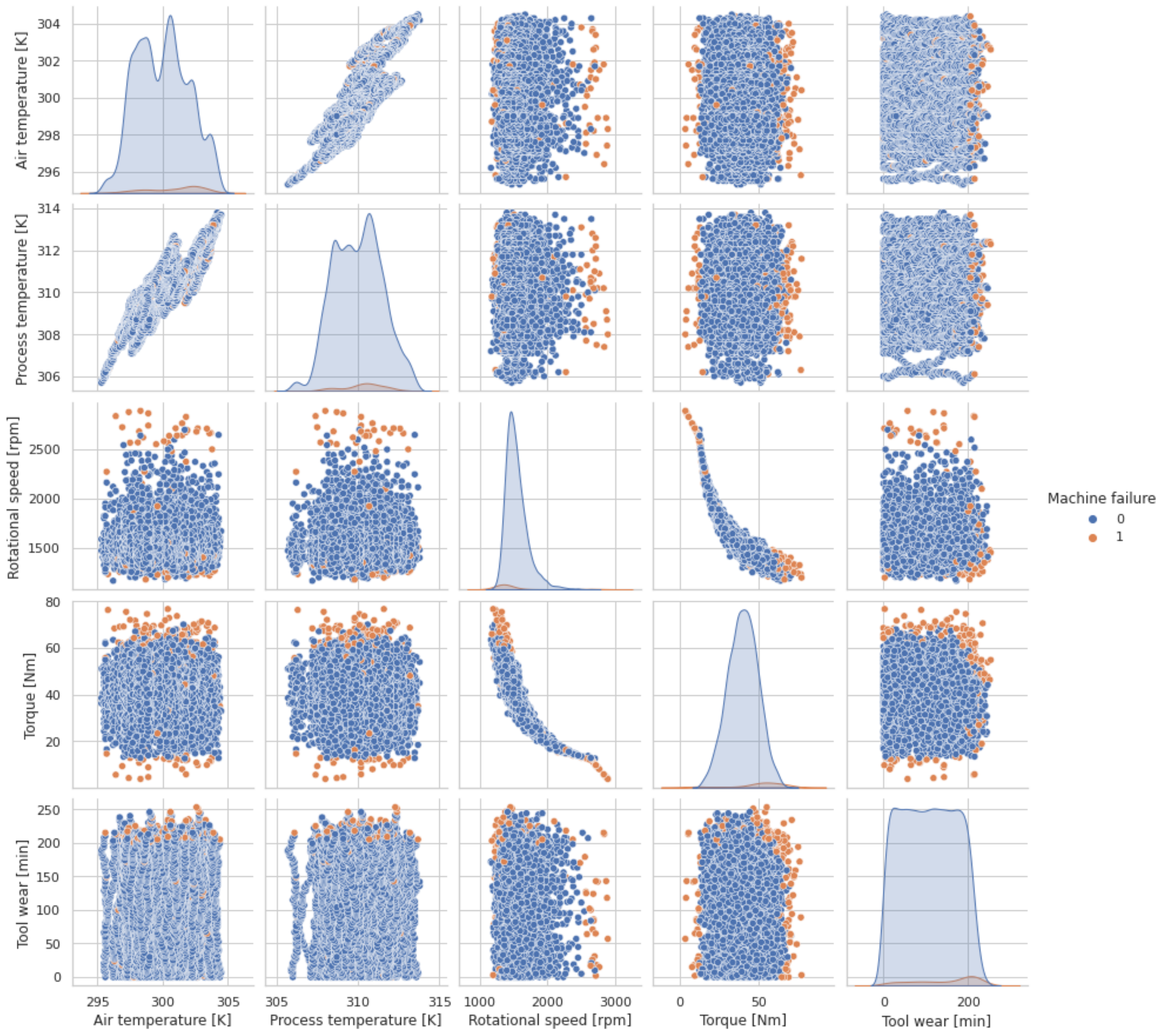
Exploratory Data Analysis (EDA) :

1) Histogram:



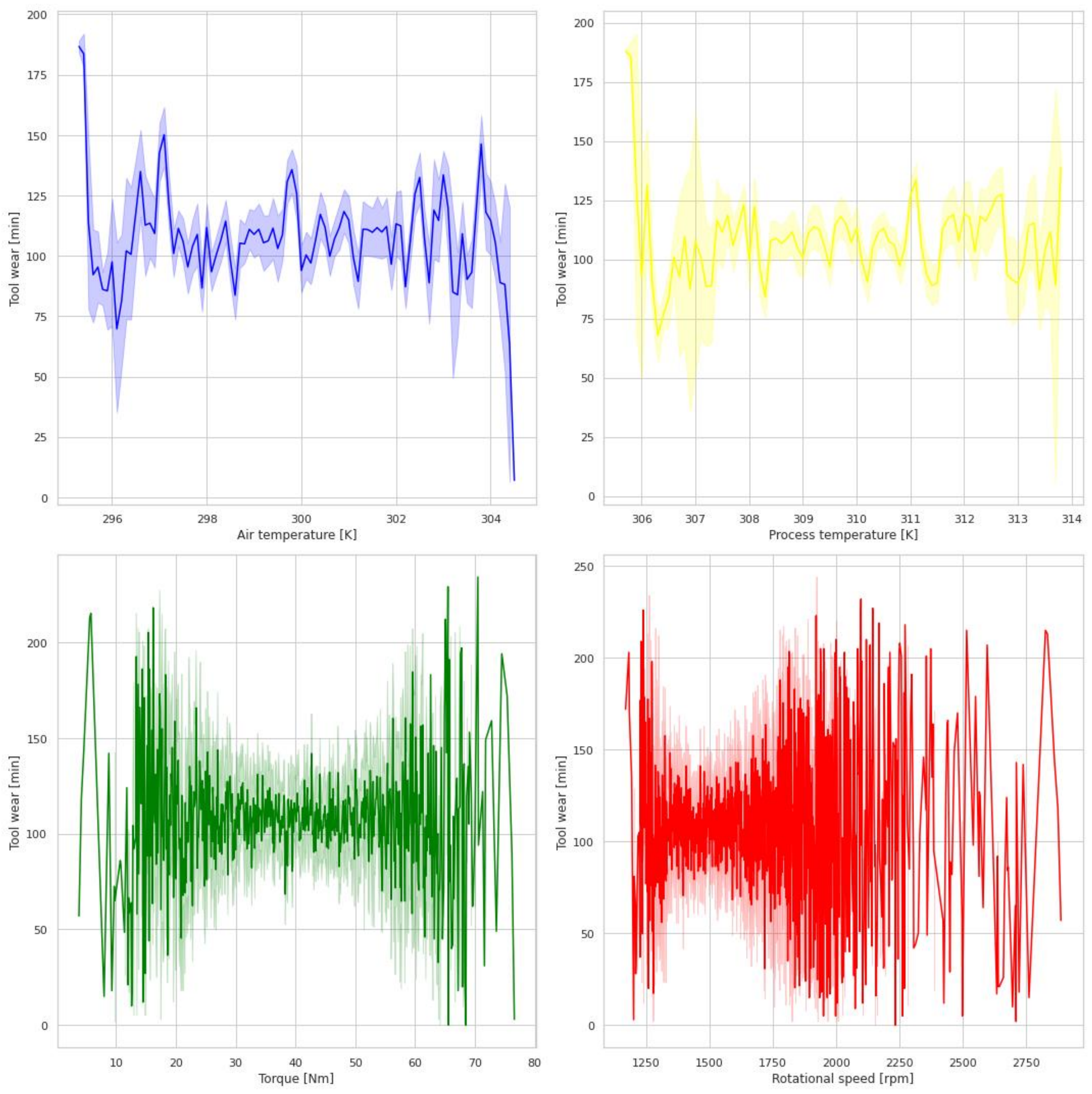
There are six different types of histograms shown above, which show how frequently data in each class appear in the dataset. There are various types of histograms visible in the figure above, such as the feature Air temperature, which we call a Bi-modal Histogram. In Histograms, Process Temperature and Torque are normally distributed. Furthermore, the skewed left histogram represents rotation speed, while the uniformly distributed histogram represents tool wear

2) .Pairplot:



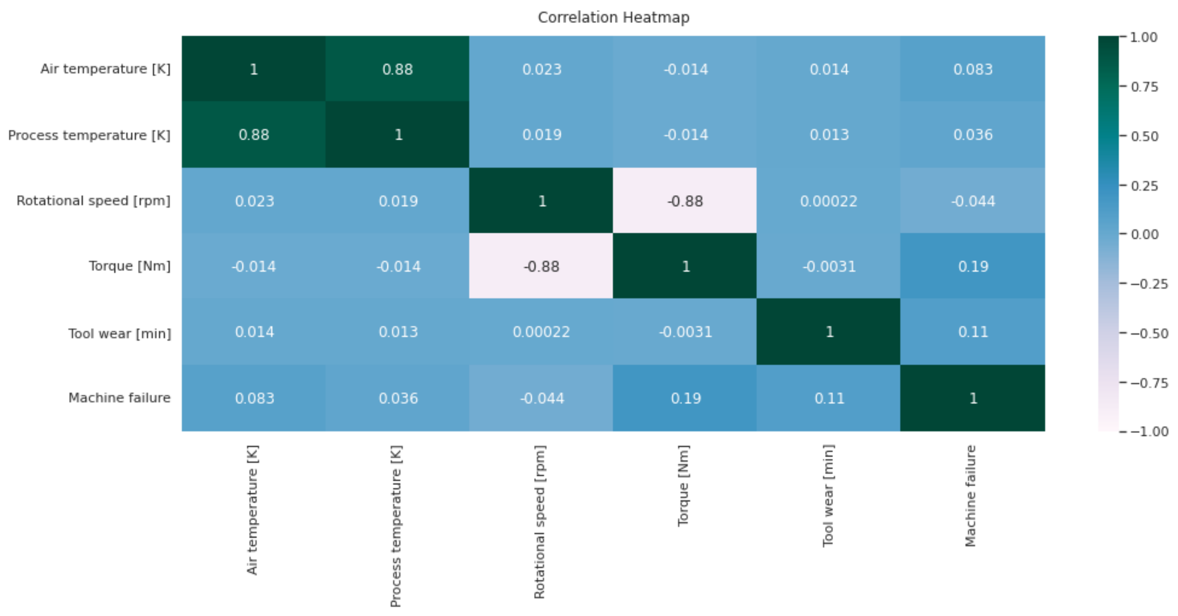
The pair plot above shows how different features are related to one another in various ways. Some features are negatively correlated, some are positively correlated, and others are randomly correlated and distributed.

3) .Line plot



The above line plot shows the relationship between tool wear and every other speed and temperature, allowing us to determine whether process temperature, air temperature, torque, or rotational speed are the causes of tool wear in the machine.

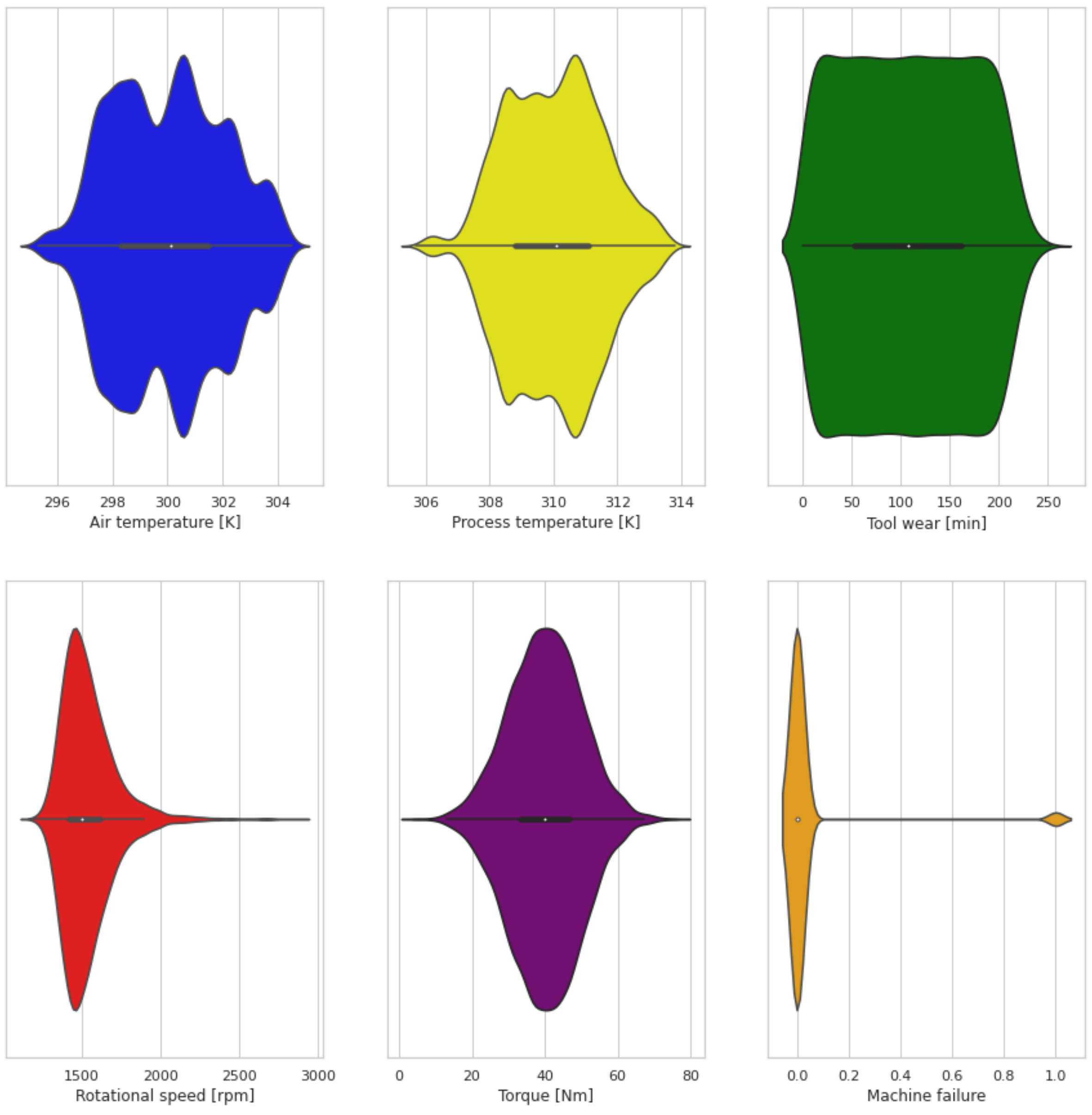
4) Heatmap



	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure
Air temperature [K]	1	0.88	0.023	-0.014	0.014	0.083
Process temperature [K]	0.88	1	0.019	-0.014	0.013	0.036
Rotational speed [rpm]	0.023	0.019	1	-0.88	0.00022	-0.044
Torque [Nm]	-0.014	-0.014	-0.88	1	-0.0031	0.19
Tool wear [min]	0.014	0.013	0.00022	-0.0031	1	0.11
Machine failure	0.083	0.036	-0.044	0.19	0.11	1

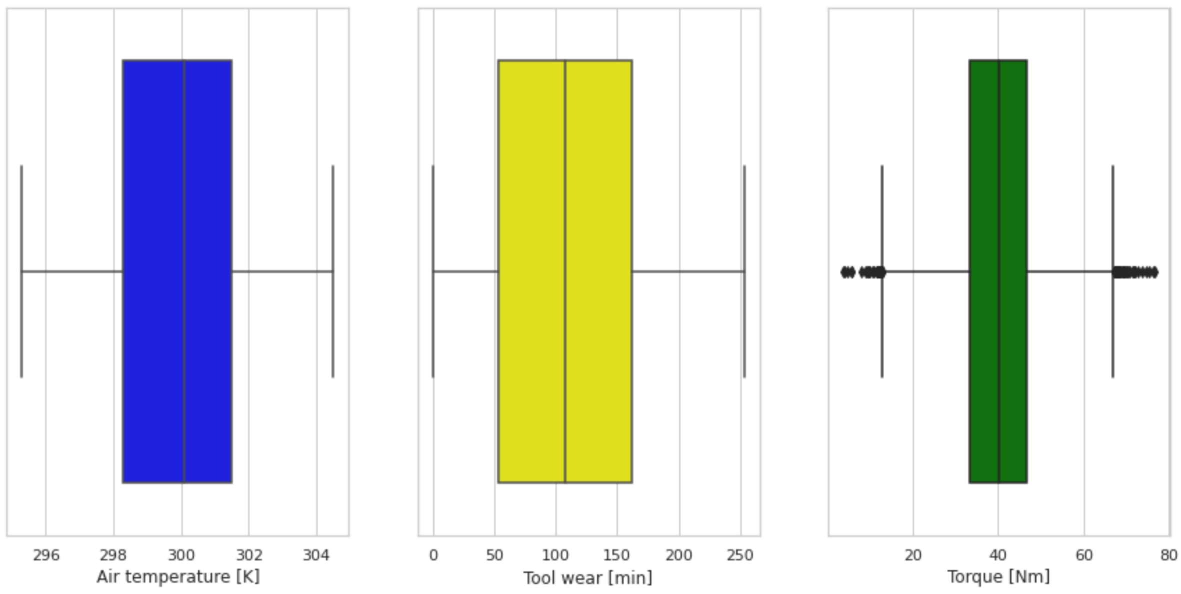
The heatmap above depicts the correlation between various features, allowing us to determine which features are highly correlated with one another. If the features are more oriented toward 1 or -1 and are positively or negatively correlated with one another, we can drop those variables, increasing the accuracy score.

5) Violin Plot



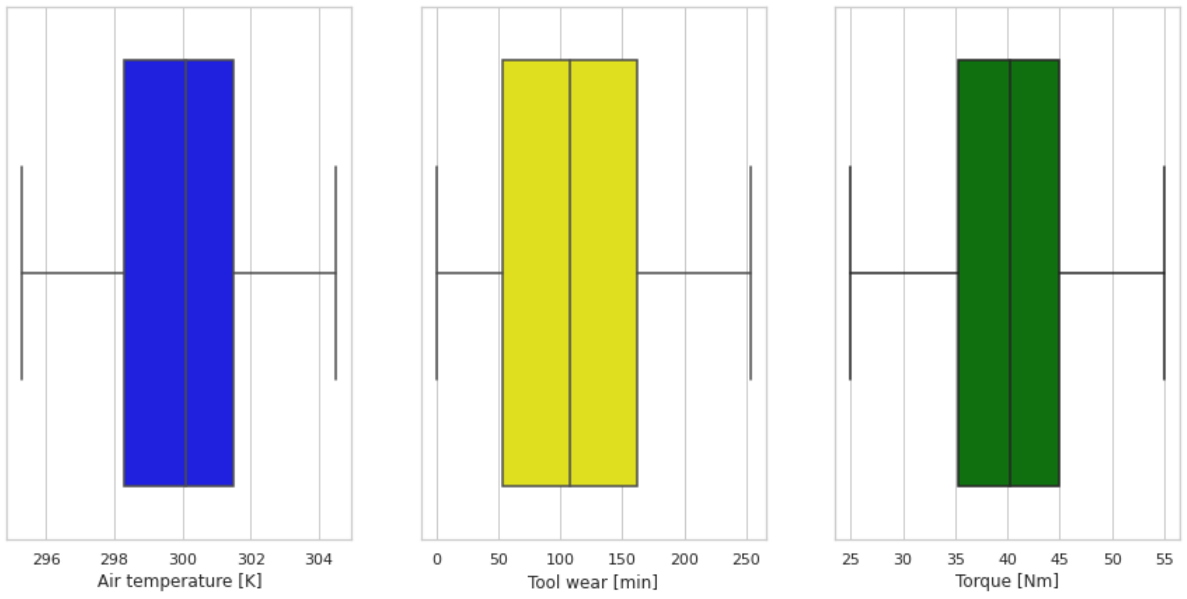
A violin plot is a hybrid of a box plot and a kernel density plot, which shows peaks in the data. It is used to visualize the distribution of numerical data. Unlike a box plot that can only show summary statistics, violin plots depict summary statistics and the density of each variable.

6) Side by Side Box Plot (With Outliers)



The above side-by-side box plot depicts numerical data distribution and skewness by displaying data quartiles and averages. The boxplot above is shown with outliers, and we can see that Torque is the only feature with outliers.

7) Side by Side Box Plot (Without Outliers)



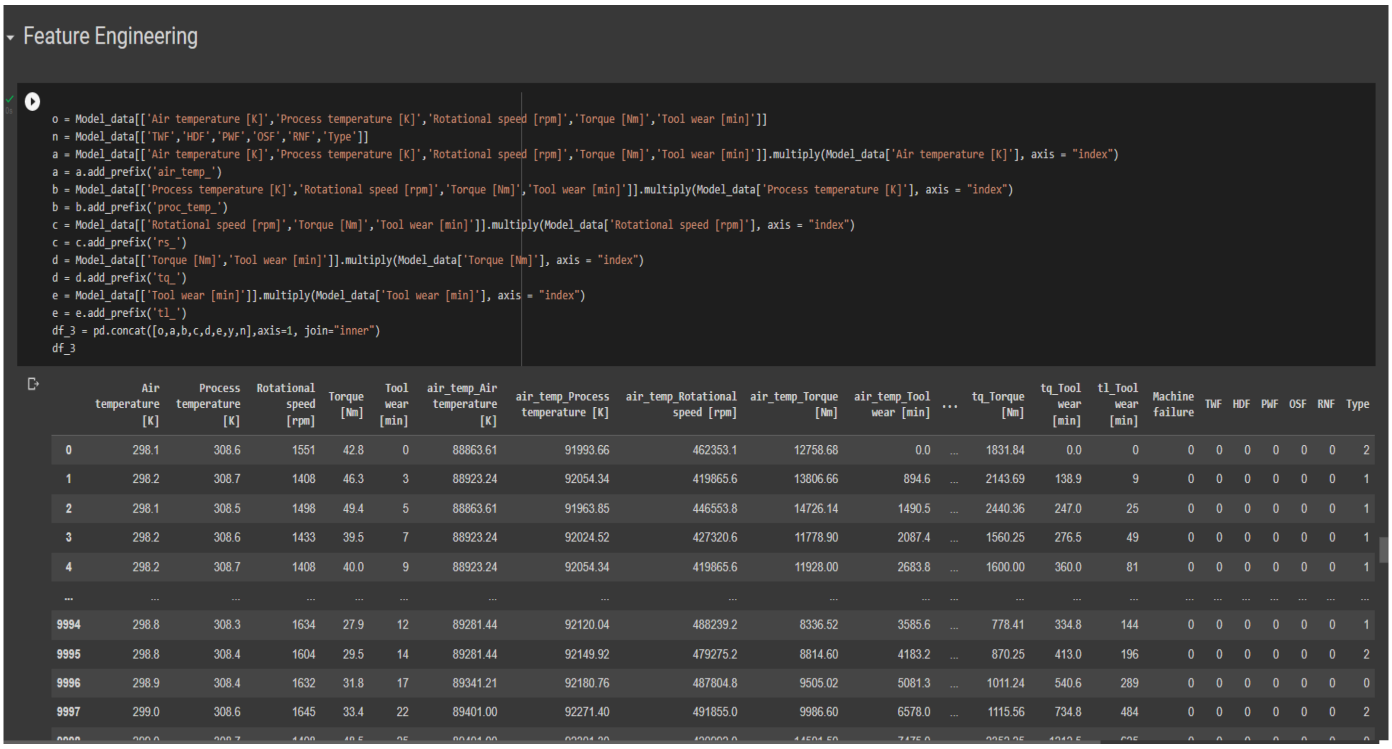
We took a few steps in this Side by Side boxplot to deal with outliers that were causing an issue earlier in the variable torque. As a result, the boxplot shown above is free of outliers.

Feature Engineering: -

Feature engineering is an important step to choose appropriate features in order to effectively train and calibrate algorithms. Feature engineering entails using data mining techniques to extract features from raw data as well as domain knowledge. Feature engineering, which is often referred to as applied machine learning, is useful for improving the performance of machine learning algorithms.

Imputation, handling outliers, log transform, feature split, and other feature engineering techniques are available. We used feature spill in our project because, as the name implies, feature spill is the process of dividing features intimately into two or more parts and performing to create new features. This technique assisted us in better understanding and delving deeper into our dataset, resulting in more meaningful patterns.

As shown in the figure below, we performed a feature engineering step and split the data accordingly.

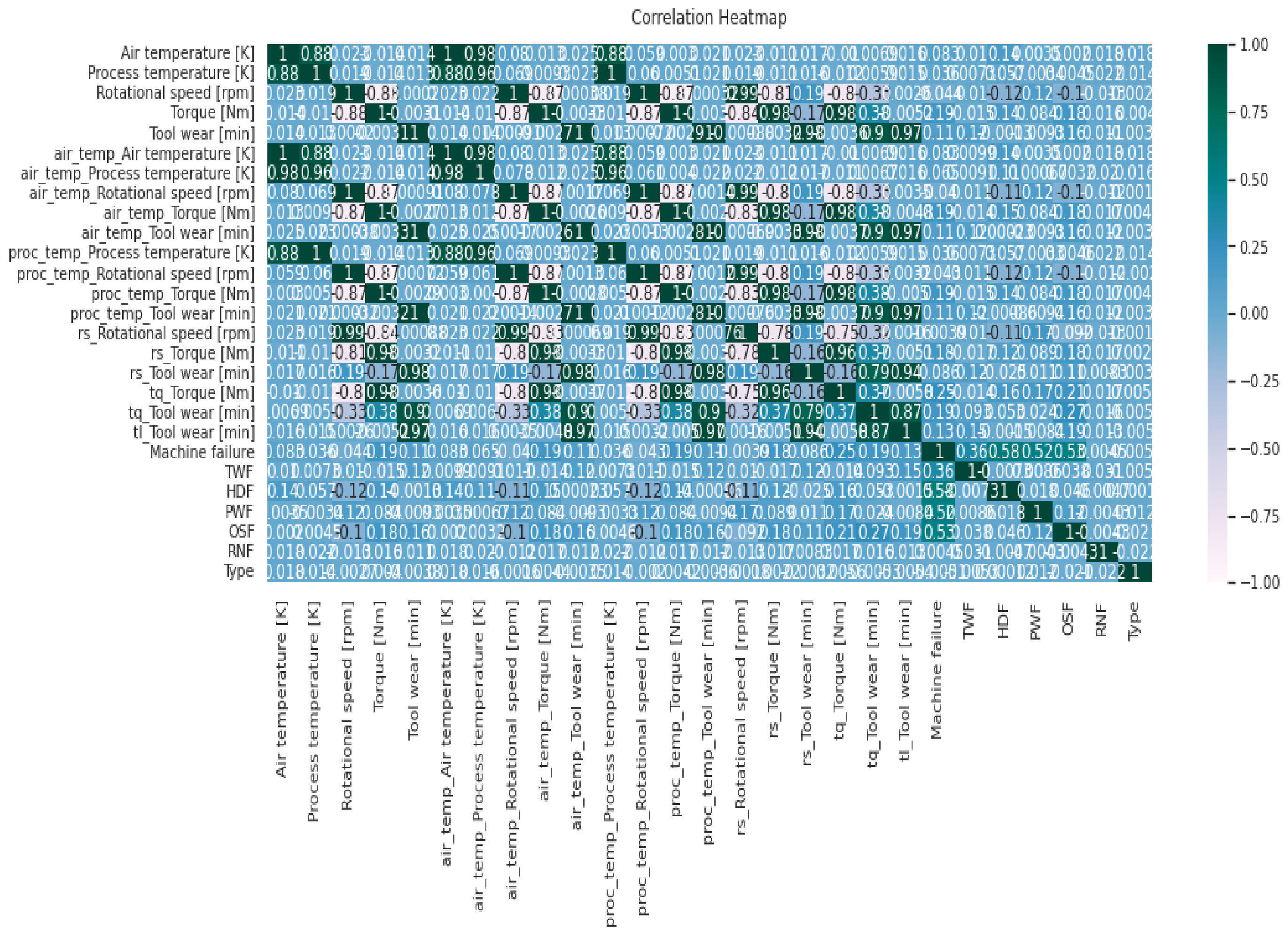


Feature Engineering

```python
o = Model_data[['Air temperature [K]','Process temperature [K]','Rotational speed [rpm]','Torque [Nm]','Tool wear [min]']]
n = Model_data[['TWF','HDF','PWF','OSF','RNF','Type']]
a = Model_data[['Air temperature [K]','Process temperature [K]','Rotational speed [rpm]','Torque [Nm]','Tool wear [min]']].multiply(Model_data['Air temperature [K]'], axis = "index")
a = a.add_prefix('air_temp_')
b = Model_data[['Process temperature [K]','Rotational speed [rpm]','Torque [Nm]','Tool wear [min]']].multiply(Model_data['Process temperature [K]'], axis = "index")
b = b.add_prefix('proc_temp_')
c = Model_data[['Rotational speed [rpm]','Torque [Nm]','Tool wear [min]']].multiply(Model_data['Rotational speed [rpm]'], axis = "index")
c = c.add_prefix('rs_')
d = Model_data[['Torque [Nm]','Tool wear [min]']].multiply(Model_data['Torque [Nm]'], axis = "index")
d = d.add_prefix('tq_')
e = Model_data[['Tool wear [min]']].multiply(Model_data['Tool wear [min]'], axis = "index")
e = e.add_prefix('tl_')
df_3 = pd.concat([o,a,b,c,d,e,y,n],axis=1, join="inner")
df_3
```

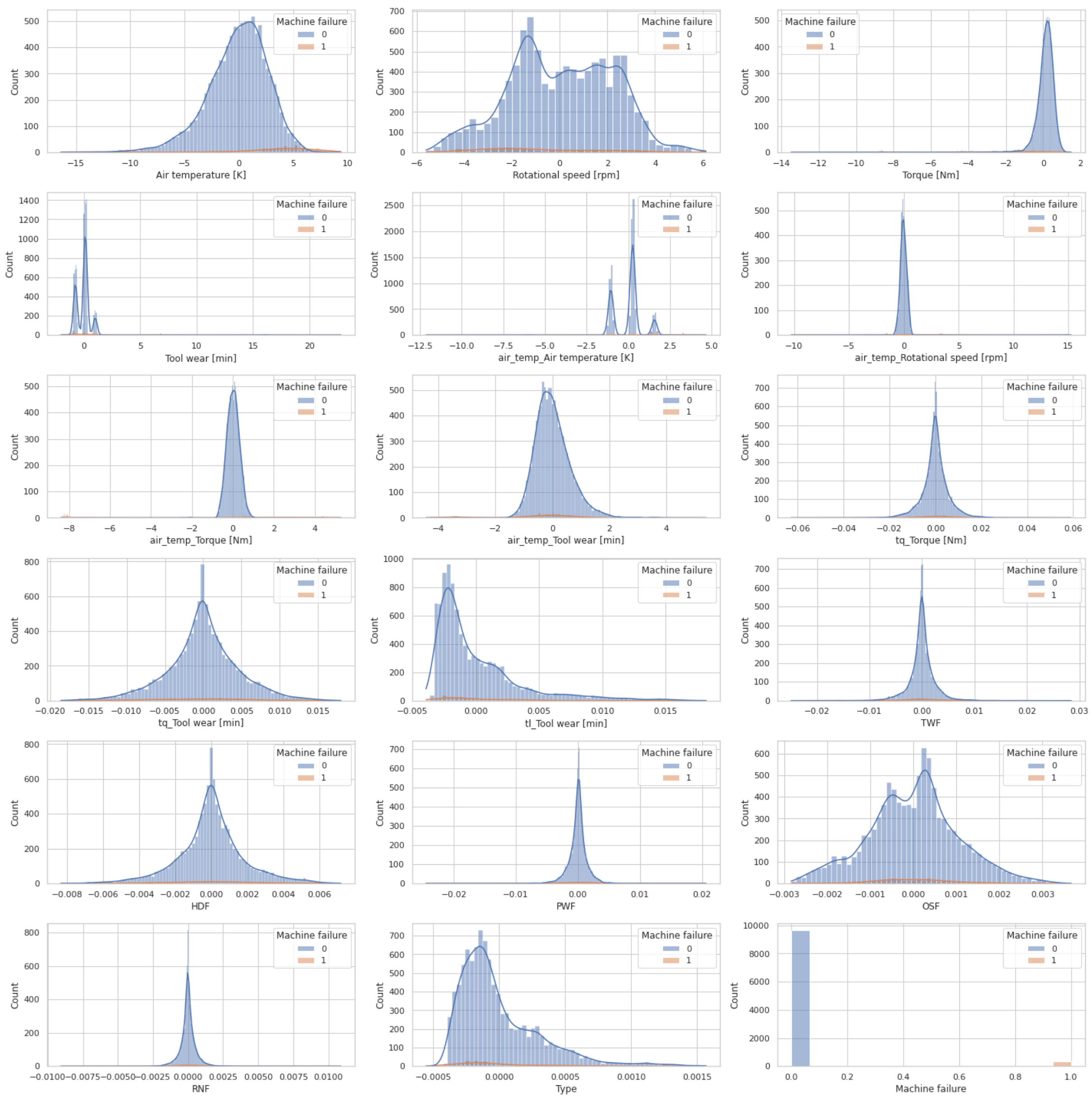
	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	air_temp_Air temperature [K]	air_temp_Process temperature [K]	air_temp_Rotational speed [rpm]	air_temp_Torque [Nm]	air_temp_Tool wear [min]	...	tq_Torque [Nm]	tq_Tool wear [min]	tl_Tool wear [min]	Machine failure	TWF	HDF	PWF	OSF	RNF	Type
0	298.1	308.6	1551	42.8	0	88863.61	91993.66	462353.1	12758.68	0.0	...	1831.84	0.0	0	0	0	0	0	0	0	2
1	298.2	308.7	1408	46.3	3	88923.24	92054.34	419865.6	13806.66	894.6	...	2143.69	138.9	9	0	0	0	0	0	0	1
2	298.1	308.5	1498	49.4	5	88863.61	91963.85	446553.8	14726.14	1490.5	...	2440.36	247.0	25	0	0	0	0	0	0	1
3	298.2	308.6	1433	39.5	7	88923.24	92024.52	427320.6	11778.90	2087.4	...	1560.25	276.5	49	0	0	0	0	0	0	1
4	298.2	308.7	1408	40.0	9	88923.24	92054.34	419865.6	11928.00	2683.8	...	1600.00	360.0	81	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9994	298.8	308.3	1634	27.9	12	89281.44	92120.04	488239.2	8336.52	3585.6	...	778.41	334.8	144	0	0	0	0	0	0	1
9995	298.8	308.4	1604	29.5	14	89281.44	92149.92	479275.2	8814.60	4183.2	...	870.25	413.0	196	0	0	0	0	0	0	2
9996	298.9	308.4	1632	31.8	17	89341.21	92180.76	487804.8	9505.02	5081.3	...	1011.24	540.6	289	0	0	0	0	0	0	0
9997	299.0	308.6	1645	33.4	22	89401.00	92271.40	491855.0	9986.60	6578.0	...	1115.56	734.8	484	0	0	0	0	0	0	2

The feature splitting process allows new features to be clustered and binned, allowing useful information to be extracted and data model performance to be improved. We were able to increase our variables from 14 to 27 after completing this step. That is, we divided the data into 13 columns.



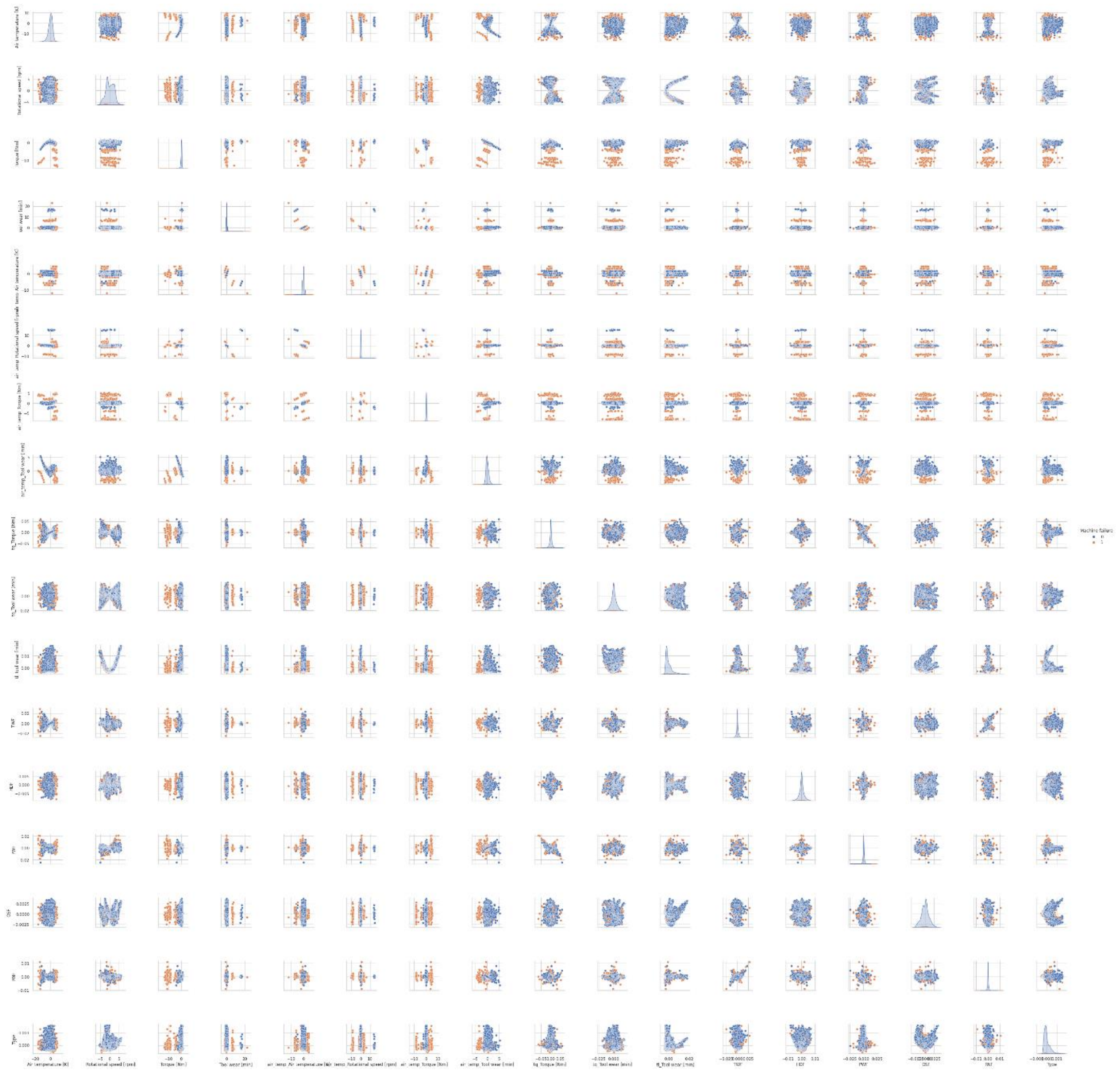
Then, using the correlation heatmap, we checked the correlation of each variable and dropped the variables that were highly correlated with each other. We made a list of the features we were going to eliminate, which were as follows: 'Process temperature [K]','air_temp_Process temperature [K]','proc_temp_Process tempera ture [K]','proc_temp_Rotational speed [rpm]','proc_temp_Torque [Nm]','proc_temp_Tool w ear [min]','rs_Rotational speed [rpm]','rs_Torque [Nm]','rs_Tool wear [min]. So, before removing these 9 variables, we ran PCA on our dataset to keep the maximum variance explained and minimize the total projection error. By doing so, we were able to reduce the number of features in a dataset while retaining as much information as possible. Now that all of these steps have been completed, we know that we will be able to display our visualizations in a more meaningful manner that conveys the best information to the users as a result of such dimensionality reduction. Some of the visualizations that we performed after splitting and applying PCA are shown below:

1) Histogram on Feature Engineered Data Set (PCA Applied)

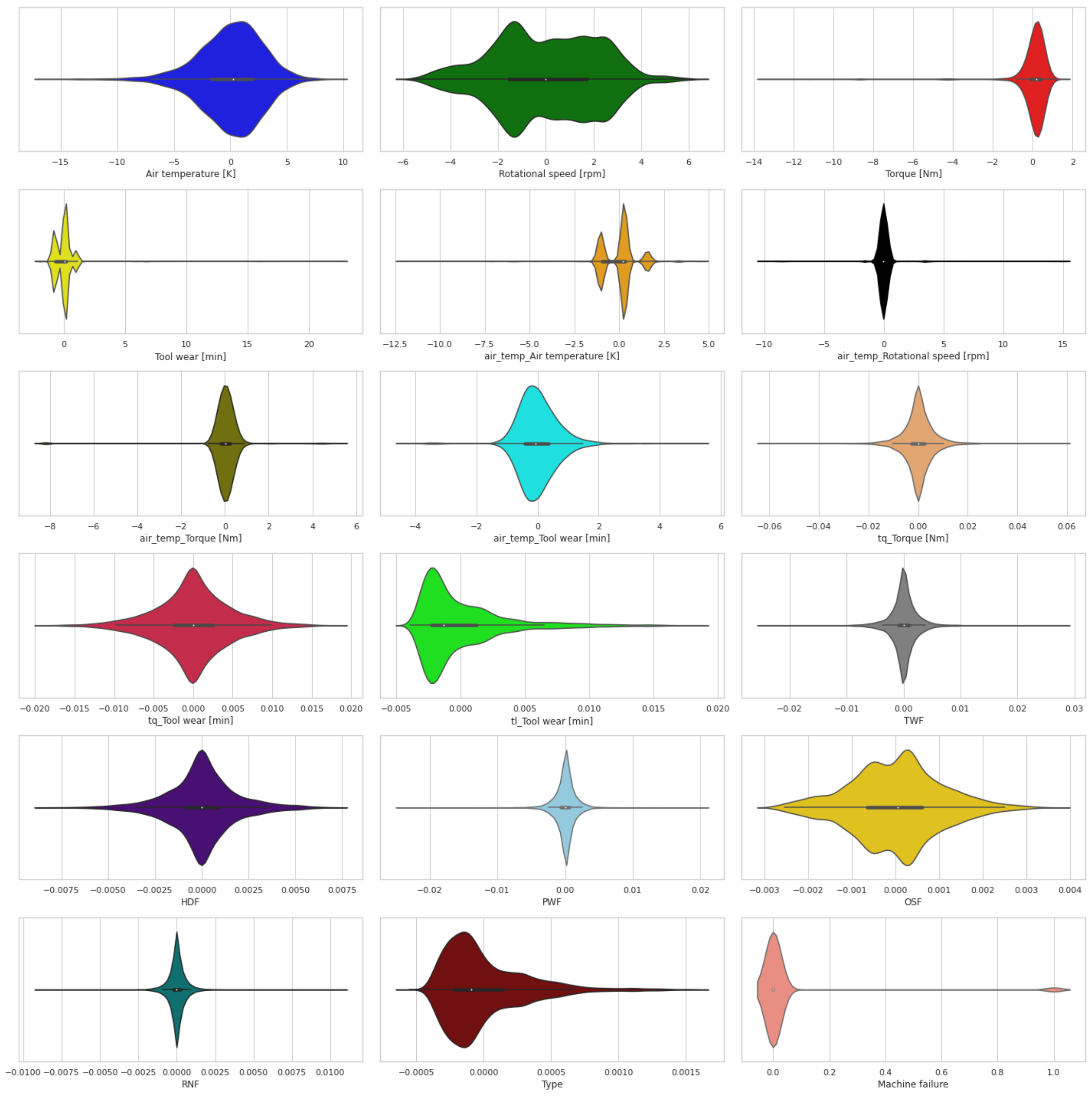


Machine failure
0
1
Count
Air temperature [K]
Rotational speed [rpm]
Torque [Nm]
Tool wear [min]
air_temp_Air temperature [K]
air_temp_Rotational speed [rpm]
air_temp_Torque [Nm]
air_temp_Tool wear [min]
tq_Torque [Nm]
tq_Tool wear [min]
tl_Tool wear [min]
TWF
HDF
PWF
OSF
RNF
Type
Machine failure

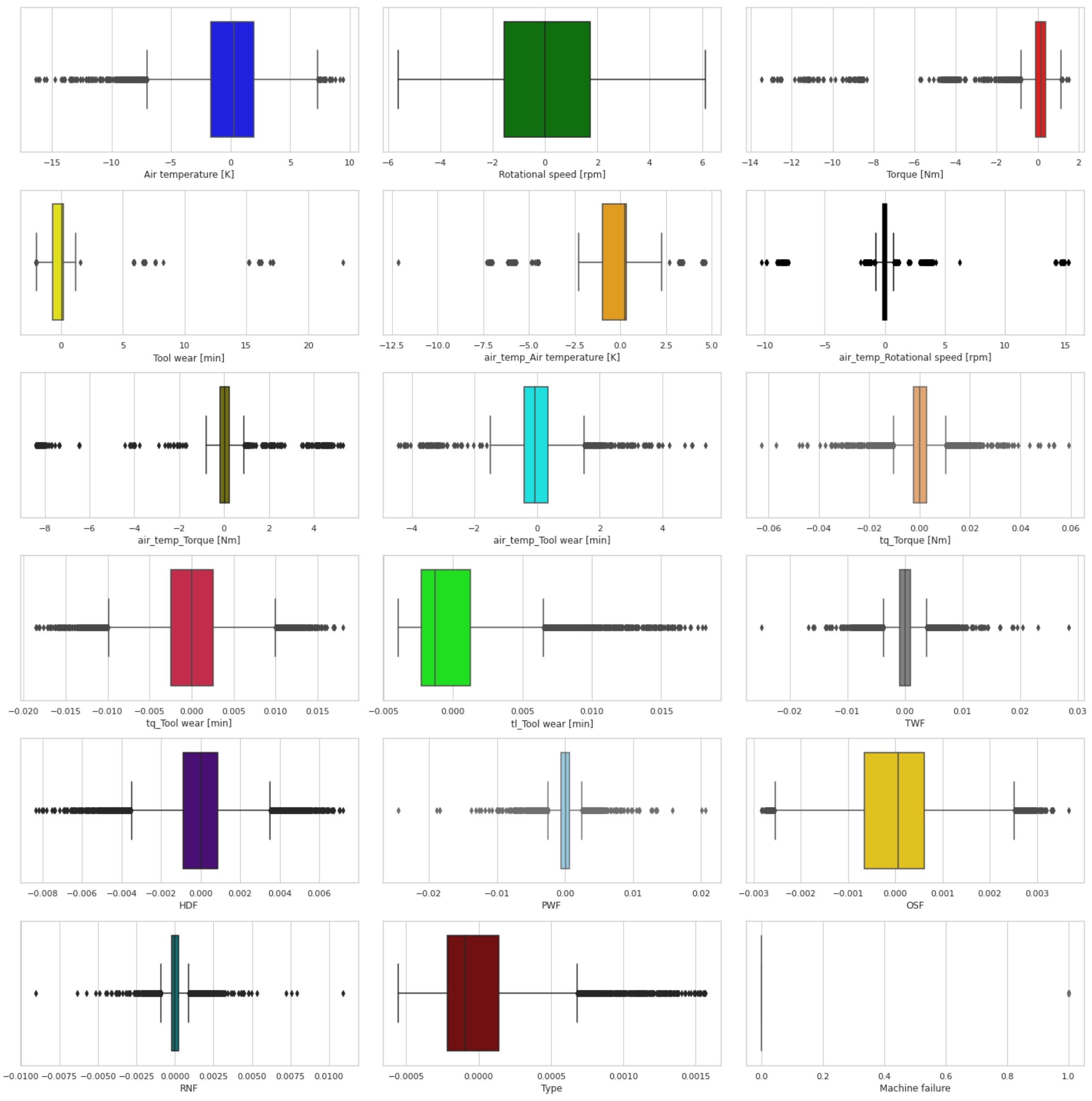
2) Pair Plot on Feature Engineered Data (PCA Applied)



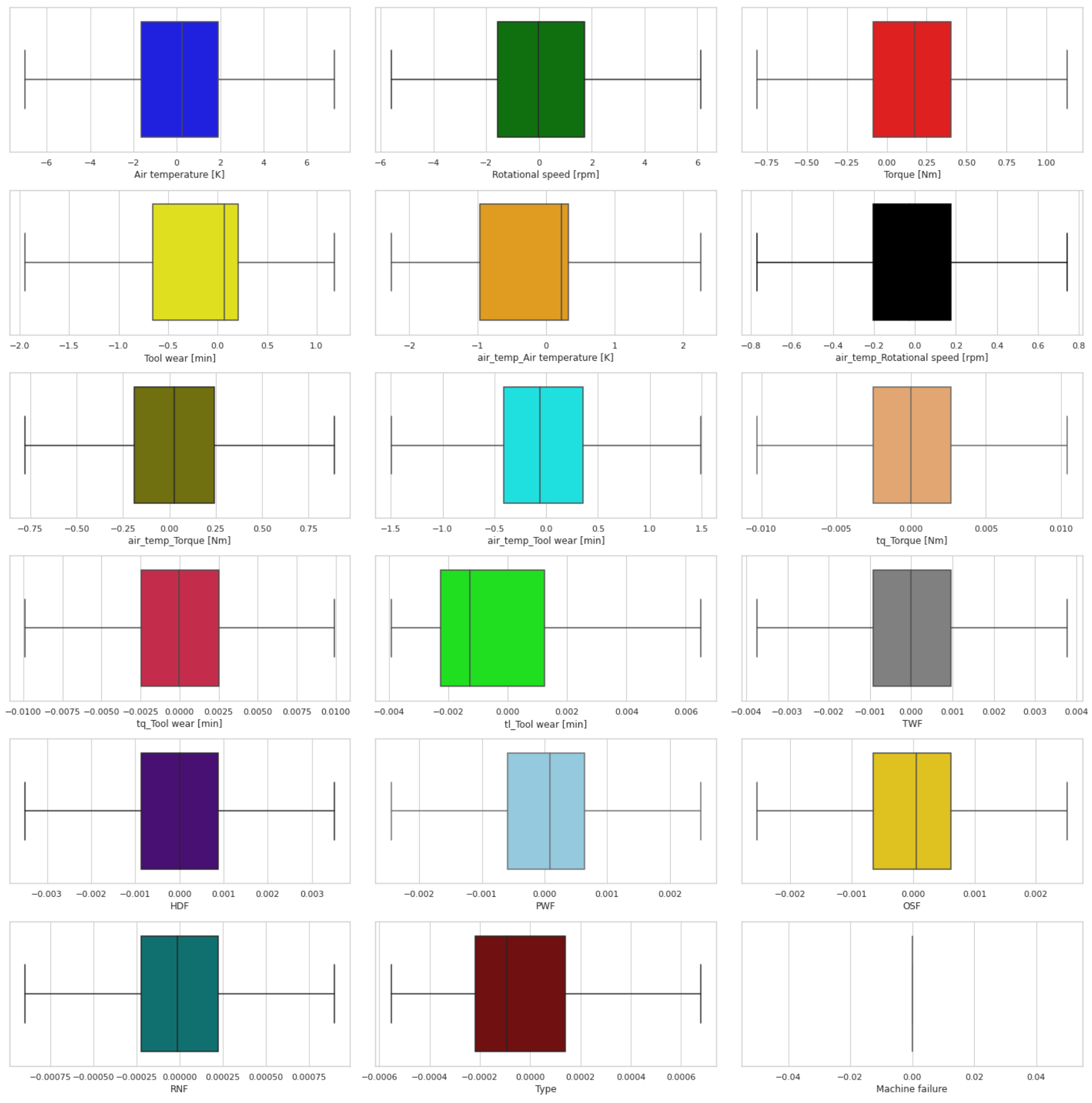
3) Violin Plot on Feature Engineered Data (PCA Applied)



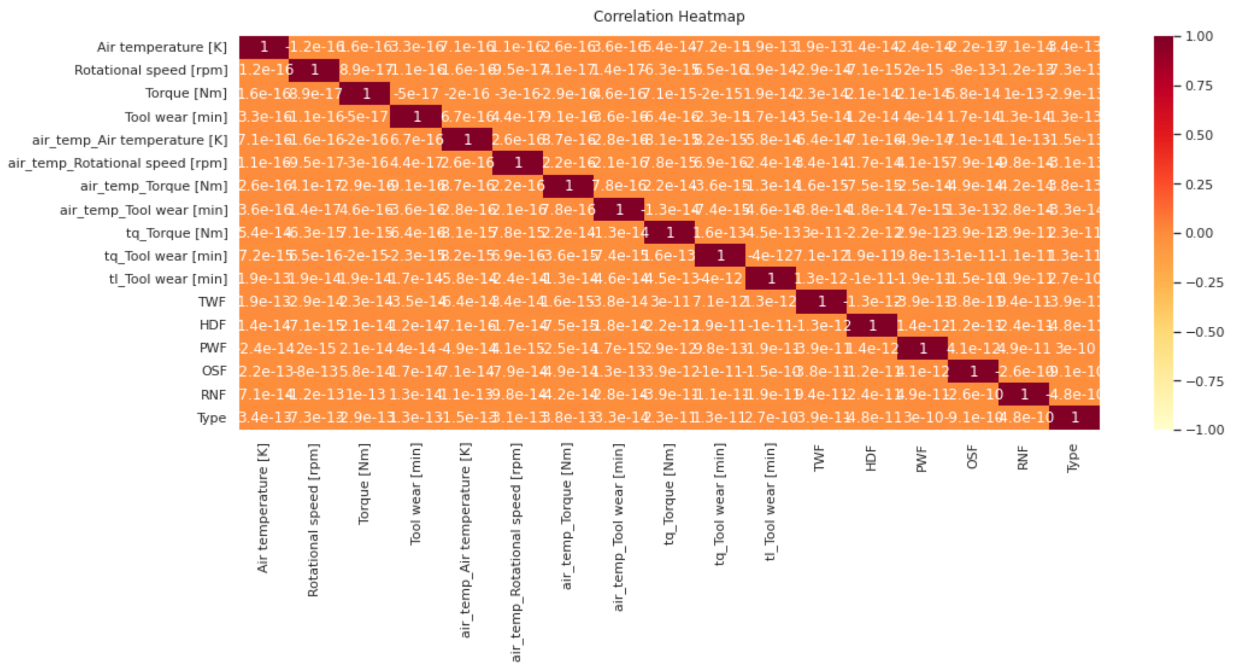
4) Side-by-Side Box Plot on Feature Engineered Data (PCA Applied) (With outliers)



5) Side-by-Side Box Plot on Feature Engineered Data (PCA Applied) (Without outliers)



6) Heatmap on Feature Engineered Data (PCA Applied)



All about Logistics Regressions

Logistic regression is a supervised machine learning algorithm that was designed to learn classification problems. When the target variable is categorical, the problem is called a classification learning problem. The goal of logistic regression is to map a function based on the characteristics of the data to the targets to predict the likelihood that a new example belongs to one of the target classes. In our project, we are developing a model that demonstrates a deep understanding of classification with logistic regression and why linear regression is unsuitable for learning categorical outputs. In our case, it was a two-class classification problem where we checked whether the machine failed or not and returned 1 if the machine failed and 0 if it did not fail.

So, to begin, we began our implementation with the logistic function, also known as the logit or sigmoid function. The Sigmoid function oversees constraining the cost function's output so that it becomes a probability output between 0 and 1.

The sigmoid function is given by:-

$$h(t) = 1 / 1 + e^{-t}$$

The next step is to write the cost function for logistic regression it is given as follows:-

Cost() h t( ), { y h = -log l ( ) ( )t if y = 1- log(1 -h (t)) if y = 0

The cost function, also known as log-loss, is configured in this manner to output the algorithm's penalty if the model predicts the incorrect class. If the algorithm predicts that the target is 1, then the cost tends toward 0. However, if the algorithm predicts incorrectly the target as 0, then the cost of the model grows exponentially large.

Next step to optimize the model that we just implemented we used the gradient descent approach which in turn reduces the cost function. Our main is to make a good predicting algorithm. So in this approach what has happened is we are finding the deepest point (global minimum) of this function. Now the deepest point is where the J() is minimum.

Now there are two approaches to find the deepest point: -

1. Derivative – to find the direction of the next step.
2. (Learning Rate) – the magnitude of the next step.

So the approach and overall idea is to first choose a random point from the function. The derivative with respect to J must then be computed (w). This will indicate the location of the local minimum. Now divide the resulting gradient by the Learning Rate. The Learning Rate has no fixed value and is determined by problems. Now, all the algorithm has to do is subtract the result from the new to get new, and the update should be done simultaneously for each theta. In our case for logistic regression, our model stopped learning after seven iterations, which means it gave us the optimal theta values on the seventh iteration.

Iterations	Learning rate	Tolerance	Accuracy
100	0.0001	0.001	0.999
1000	0.00001	0.0001	0.998
5000	0.000001	0.0001	0.997
10000	0.000001	0.0001	0.996
50000	0.0001	0.00005	0.99

	Logistic Regression
Accuracy	0.99
Precision	1
Recall	0.98
F1 Score	0.99

All about Naive Bayes Classifier

Naive Bayes Classification is a generative Classification technique, which means it has priors by default. Unlike the Discriminative Classification techniques, Generative Classification cares about the relationship between the features and classes. However, we make a strong assumption that all the features are independent of each other. The general formula used, underlined by Bayes Theorem is as follows: -

P(Class|Data)= (P(Data|Class)*P(Class)) / P(Data)

P(Class|Data)- Posterior Probability

P(Data|Class)-Likelihood

P(Class)-Prior

P(Data)-Normalizing Factor

In our case, as our data is continuous and target variable is binary categorical (0 or 1), hence we apply Gaussian Naive Bayes algorithm wherein the features are normally distributed.

$$P\ (X_k|Y=J) \sim N\ (\mu, \sigma^2)$$

$$P\ (X_k|Y=J) = [1/\ (2\pi)^{(0.5)} * \sigma^2] * e^{-[(X-\mu)\text{^}2 / (2\pi)\text{^}(0.5) * (\sigma^2)]}$$

In order to begin our implementation, we separated our training data based on two classes (0 and 1) in which the target variable was divided. This step was performed as it allows us to calculate the prior values in our problem.

P(Y=0) = (No. of data points with label ‘0’) / (Total no. of data points)

P(Y=1) = (No. of data points with label ‘1’) / (Total no. of data points)

Once the prior values are calculated, the likelihoods are calculated using the formula of Probability density function (Pdf) which follows the gaussian distribution. As features are independent, all the features have their individual likelihood value. Using these likelihood values, we calculate individual class probabilities for a given datapoint and predict the class by comparing both the class probabilities.

$$P(Y=0|X) = P(X_1|Y) * P(X_2|Y) * P(X_3|Y) * P(Y=0)$$

$$P(Y=1|X) = P(X_1|Y) * P(X_2|Y) * P(X_3|Y) * P(Y=1)$$

After calculating the class probabilities, we can evaluate the performance of model by calculating various metrics like Precision, recall and F-score for which the classification report generated is as follows: -

One of the limitations encountered in the conventional Naive Bayes Classifier is that if a given likelihood for a feature turns out to be 0, entire class probability becomes 0 as all the likelihood values are individually multiplied. As a result, for a given data point we get both class probabilities as 0, which is not desirable.

To overcome this, we implement Laplace smoothing, wherein we replace these likelihood values (that are 0) by a positive constant term ‘a’. This eventually allows to make comparison between both class probabilities and determine the class for a given data point. Mathematically, we add a constant (τ) To the numerator and (m*τ) To the denominator

	Gaussian Naïve Bayes
Accuracy	0.17
Precision	0.10
Recall	0.55
F1 Score	0.82

All about Support vector machine (SVM): -

The SVM algorithm is a supervised learning algorithm that falls under the classification technique category. It is a binary classification technique that predicts an optimal hyperplane in an n-dimensional space using the training dataset. This hyperplane is used to categorize new data sets. Because the hyperplane is a binary classifier, it divides the training data set into two classes.SVM algorithms can classify data in both a two-dimensional plane and a multidimensional hyperplane. The multidimensional hyperplane categorizes the multidimensional data using "Kernels. “It is always preferable to have the greatest possible distinction between the classified data points. This means that they should have the greatest possible distance between the data points, or that the hyperplane should have the greatest possible margin between the data points.

As we know from the model's history, it used to outperform every other model on the market 8 years ago. It is the most powerful classifier. Because our dataset contains 10,000 data points, and SVM works best with smaller datasets, we used it. As a result, we can see in the classification report below that we have excellent accuracy, precision, recall, and F1-score.

	Support Vector Machine
Accuracy	0.06
Precision	0.03
Recall	1
F1 Score	0.03

All about Decision Tree Classifier: -

Decision Tree classifiers are supervised machine learning models that use prelabelled data to make predictions. The Decision Tree follows an algorithm based on if-else-then loop. Each node of a decision tree represents a decision point that splits into two leaf nodes. Each of these nodes represents the outcome of the decision and each of the decisions can also turn into decision nodes. Eventually, the different decisions will lead to a final classification.

The split threshold and the feature on which we start the split is determined by minimizing the cost function. Decision tree is highly prone to overfitting. If a tree is too long, there is overfitting and hence addition of a penalty term is desired (regularization). There are two approaches: -

1) Set a threshold to depth

2) Look at the width (pruning the leaves)

Width is a much better representation for overfitting. Bottom-up technique and Top-down technique are the two basic methods of Prunning. The Decision Tree Classifier does not think about immediate reward. The overall goal is to get rid of branches with large error and minimize the overall error.

	Decision Tree
Accuracy	0.97
Precision	0.7
Recall	0.40
F1 Score	0.51

Final Result Table:-

	Logistic Regression	Gaussian Naïve Bayes	SVM	Decision Tree
Accuracy	0.99	0.17	0.06	0.97
Precision	1	0.10	0.03	0.7
Recall	0.98	0.55	1	0.40
F1 Score	0.99	0.82	0.03	0.51

Conclusion:-

We ran four different algorithms on our dataset and learned from their accuracy, precision, recall, and F1 score that logistic regression is the best algorithm with an F1 score of 0.99.

References: -

1. EdPrice-MSFT. "Introduction to Predictive Maintenance in Manufacturing - Azure Architecture Center." *Azure Architecture Center | Microsoft Learn*, https://learn.microsoft.com/enus/azure/architecture/industries/manufacturing/predictive-maintenance-overview.
2. Bisong, E. (2019). Logistic regression. In Building Machine Learning and Deep Learning Models on Google Cloud Platform (pp. 243-250). Apress, Berkeley, CA.