Smart Industrial Predictive Solutions

Feynn Labs Internship project-3

Date: 28-sep-2023

Abstract

Smart Industrial Predictive Solutions is a forward-looking project that harnesses data science and machine learning to revolutionize industrial maintenance. With a focus on predictive maintenance, this initiative empowers industries to proactively address equipment failures and optimize maintenance schedules. Through data-driven insights and advanced analytics, it promises to enhance efficiency, reduce downtime, and transform how industries manage machinery and equipment. This abstract provides an overview of the project's objectives, methodologies, and anticipated impacts on industrial operations.

Github Link. Web-App: Link1 (render.com) and Link2 (GCP)

Contributors

- Adhiban Siddarth
- Karakavalasa venkata pranay
- Malay Vyas
- Shreyash Banduji Chacharkar
- Yash Mayur

Roles

Role	Person(s)
Business Specialists (2)	Karakavalasa venkata pranay & Shreyash Banduji Chacharkar
Data Scientists (2)	Malay Vyas & Yash Mayur
Software Developer (1)	Adhiban Siddarth

1. Problem Statement:

In various industrial sectors, machinery and equipment serve as the backbone of operations. However, these assets are susceptible to breakdowns and failures, leading to unplanned downtime, safety risks, and increased maintenance costs. Traditional maintenance practices are often reactive, causing production disruptions and financial losses. The challenge is to develop a comprehensive predictive maintenance solution, "Smart Industrial Predictive Solutions" that leverages data science and machine learning to foresee equipment failures in advance, allowing industries to proactively schedule maintenance and optimize operations. This project seeks to address the critical need for efficient, cost-effective, and data-driven maintenance strategies in industrial settings, enhancing reliability, safety, and productivity while reducing operational disruptions and financial burdens.

2. Market/Customer/Business Need Assessment:

2.1. **Market Demand and Trends:** The industrial sector is facing increasing pressure to reduce downtime, enhance operational efficiency, and minimize maintenance costs. There is a growing demand for predictive maintenance solutions that can provide actionable insights and prevent unplanned equipment failures.

2.2. **Customer Pain Points:** Industrial customers face challenges related to equipment breakdowns, production interruptions, and high maintenance expenses. They seek solutions that can proactively identify maintenance needs, improve asset reliability, and optimize resource allocation.

2.3. **Competitive Landscape:** The competitive landscape includes a mix of traditional maintenance services and emerging predictive maintenance solutions. Existing offerings often lack the sophistication and data-driven approach of "Smart Industrial Predictive Solutions."

2.4. **Regulatory Compliance:** Regulatory bodies increasingly require industries to meet specific safety and reliability standards. Predictive maintenance solutions can aid in compliance by reducing safety risks and improving equipment reliability.

2.5. **Business Opportunities:** "Smart Industrial Predictive Solutions" presents an opportunity to cater to the growing demand for predictive maintenance services. It can tap into a market where industries are actively seeking cost-effective and efficient maintenance solutions.

2.6. **Financial Impact:** For businesses, reducing downtime and maintenance costs directly impacts the bottom line. Predictive maintenance solutions promise significant cost savings and improved profitability.

2.7. **Safety and Reputation:** Predictive maintenance also contributes to improved safety records and enhanced reputation for businesses, making it an attractive proposition for industries with strict safety requirements.

The market/customer/business need assessment highlights the strong demand for predictive maintenance solutions like "Smart Industrial Predictive Solutions." Industries are actively seeking ways to reduce downtime, enhance safety, and optimize maintenance practices, presenting a compelling opportunity for this innovative project to address critical pain points and fulfill market needs.

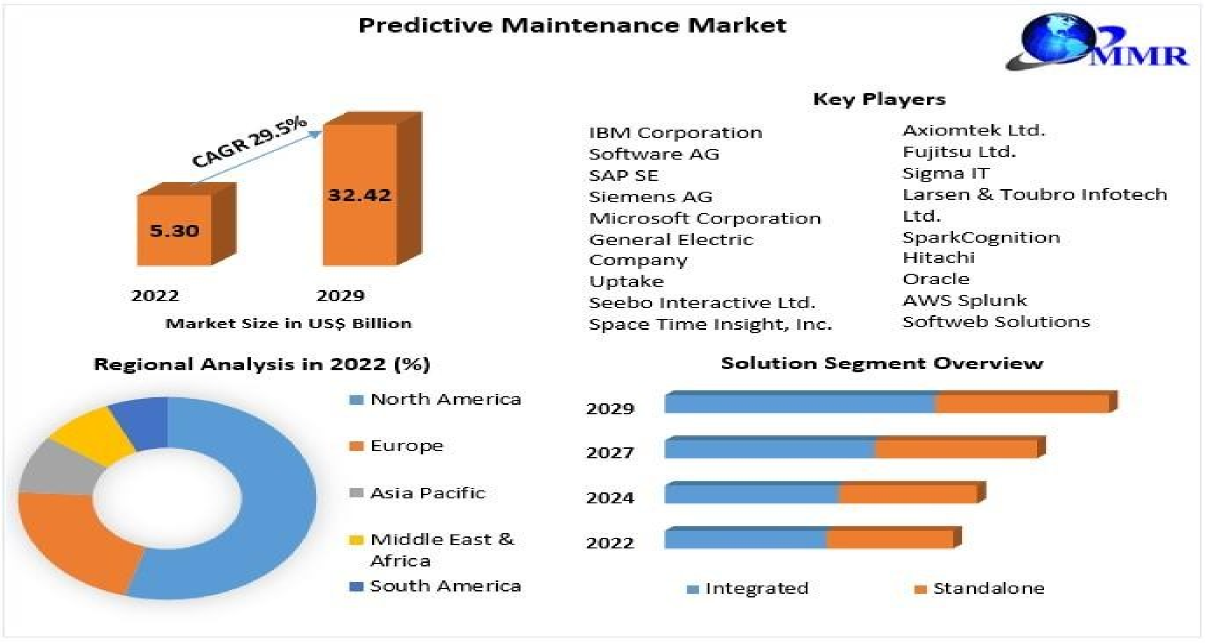
3. Market analysis:

The predictive maintenance market in India is expected to reach $4 billion by 2026, growing at a compound annual growth rate (CAGR) of 25.2%. In 2021, the market generated nearly $900 million in revenue. The growth is due to increasing demand for reducing productivity loss and maintenance costs.

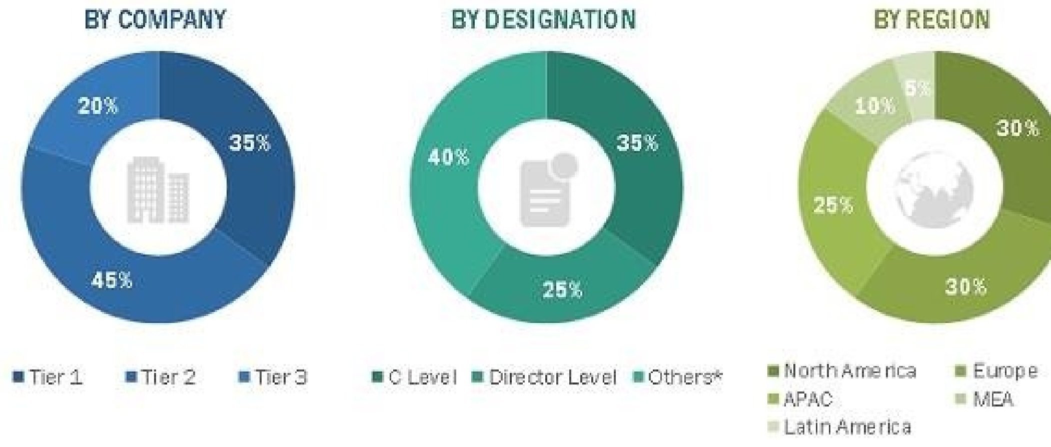
Predictive maintenance uses artificial intelligence (AI) to improve productivity and efficiency. AI can predict when a machine will break down, which allows for planning maintenance efforts where they are needed. This can reduce long-term repair costs.

The predictive maintenance market is expected to expand at a 24.2% CAGR from 2023 to 2033. The energy and utilities segment is expected to grow at a CAGR of around 32% over the forecast period. The large enterprises segment is projected to lead the market at a CAGR of 21.4% during the timeframe of 2022-2032.

The recorded data allows an engineer to estimate the eventual failure point of the observed asset, enabling it to be repaired or replaced shortly before it fails. Predictive maintenance reduces the occurrence of repair while still eliminating unexpected reactive maintenance and reducing equipment downtime and expenses associated with preventative maintenance. Predictive maintenance increases the lifespan of the equipment being observed. The report explores the Predictive Maintenance market's segments (Solution, Service, Deployment, Enterprise Size, End-Use, and Region). Data has been provided by market participants, and regions (North America, Asia Pacific, Europe, Middle East & Africa, and South America). It provides a thorough analysis of the rapid advances that are currently taking place across all industry sectors. Facts and figures, illustrations, and presentations are used to provide key data analysis for the historical period from 2018 to 2022. The report investigates the Predictive Maintenance market's drivers, limitations, prospects, and barriers. This MMR report includes investor recommendations based on a thorough examination of the Predictive Maintenance market's contemporary competitive scenario.



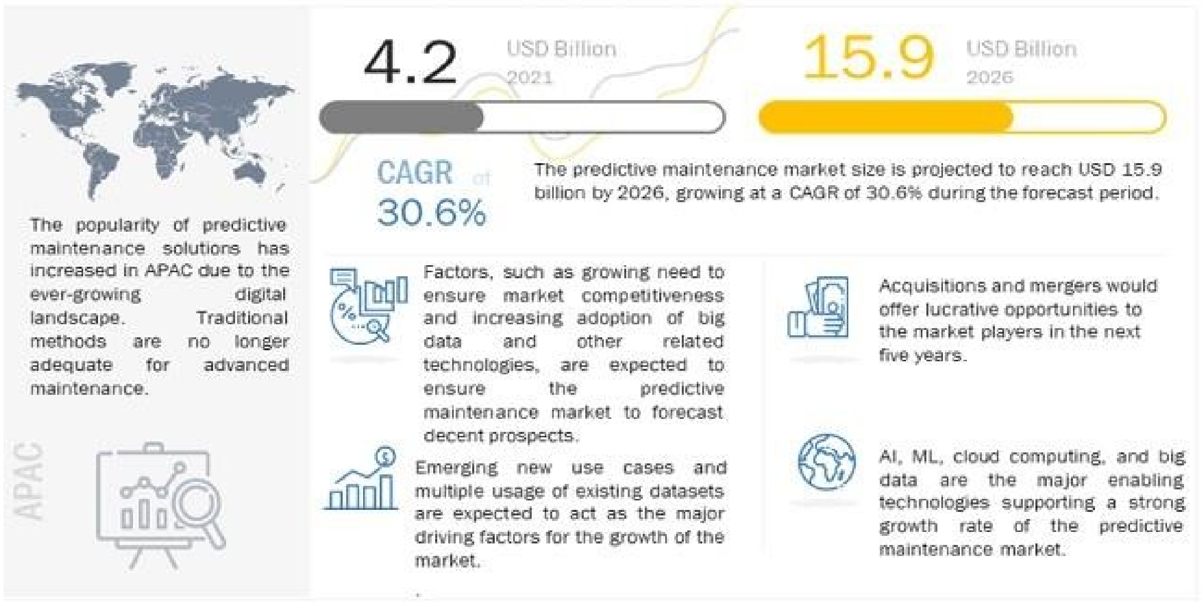
Predictive Maintenance Market size was valued at US$ 5.30 Bn.in 2022 and the total Predictive Maintenance revenue is expected to grow at 29.5% from 2023 to 2029, reaching nearly US$ 32.42 Bn.



Note: Tier 1 companies comprise the overall annual revenue of >USD 10 billion; tier 2 companies' revenue ranges in between USD 1 and 10 billion; and tier 3 companies' revenue ranges in between USD 500 million–USD 1 billion

Source: MarketsandMarkets Analysis

Attractive Opportunities in the Predictive Maintenance Market


4.2
USD Billion
2021
15.9
USD Billion
2026
CAGR of
30.6%
APAC

The predictive maintenance market size is projected to reach USD 15.9 billion by 2026, growing at a CAGR of 30.6% during the forecast period.

The popularity of predictive maintenance solutions has increased in APAC due to the ever-growing digital landscape. Traditional methods are no longer adequate for advanced maintenance.

Factors, such as growing need to ensure market competitiveness and increasing adoption of big data and other related technologies, are expected to ensure the predictive maintenance market to forecast decent prospects.

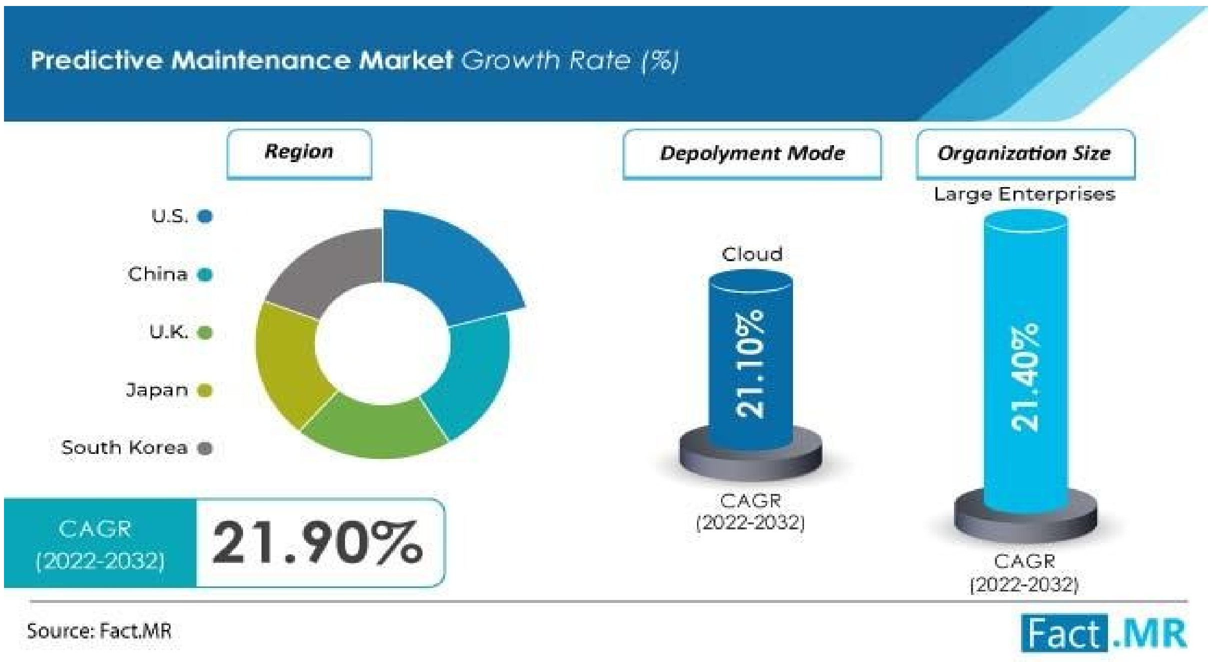
Emerging new use cases and multiple usage of existing datasets are expected to act as the major driving factors for the growth of the market.

Acquisitions and mergers would offer lucrative opportunities to the market players in the next five years.

AI, ML, cloud computing, and big data are the major enabling technologies supporting a strong growth rate of the predictive maintenance market.

e: estimated; p: projected

Source: Secondary Research, Expert Interviews, and MarketsandMarkets Analysis



By deployment, the cloud-based segment is expected to dominate the market exhibiting a CAGR of 21.1% during the forecast period. Expansion of the segment can be attributed to benefits offered such as cost-efficiency, increased asset utilization, and better safety and compliance, among others.

Based on organization size, the large enterprises segment is projected to lead the market at a CAGR of 21.4% during the timeframe of 2022-2032. Predictive maintenance allows easy access to specific details on product and application habits. It also eases expenses and offers cost-cutting solutions that inhibit the expenses.

According to the analysis, the market in the US is expected to lead the global market. The country is estimated to secure a market value worth US$ 15.8 Million by 2032. The growth of the market can be attributed to the presence of established players in the region.

Source: **https://www.marketsandmarkets.com/Market-Reports/operational-predictive-maintenance-market-8656856.html#:~:text=%5B294%20Pages%20Report%5D%20The%20Predictive,at%20a%20CAGR%20of%2030.6%25.**

4. Business Model:

4.1. Offering of Subscription Based Services:

Business model are back bones of any business model, spending time on defining a business model is good strategy rather than direct door sales. Subscription based business model can be effective in our scenario. There are many businesses use such strategy of some fixed cost and other requirement-based cost services. For our **SIPS (Smart Industrial Predictive Solutions) PDM (Predictive Maintenances)** service business

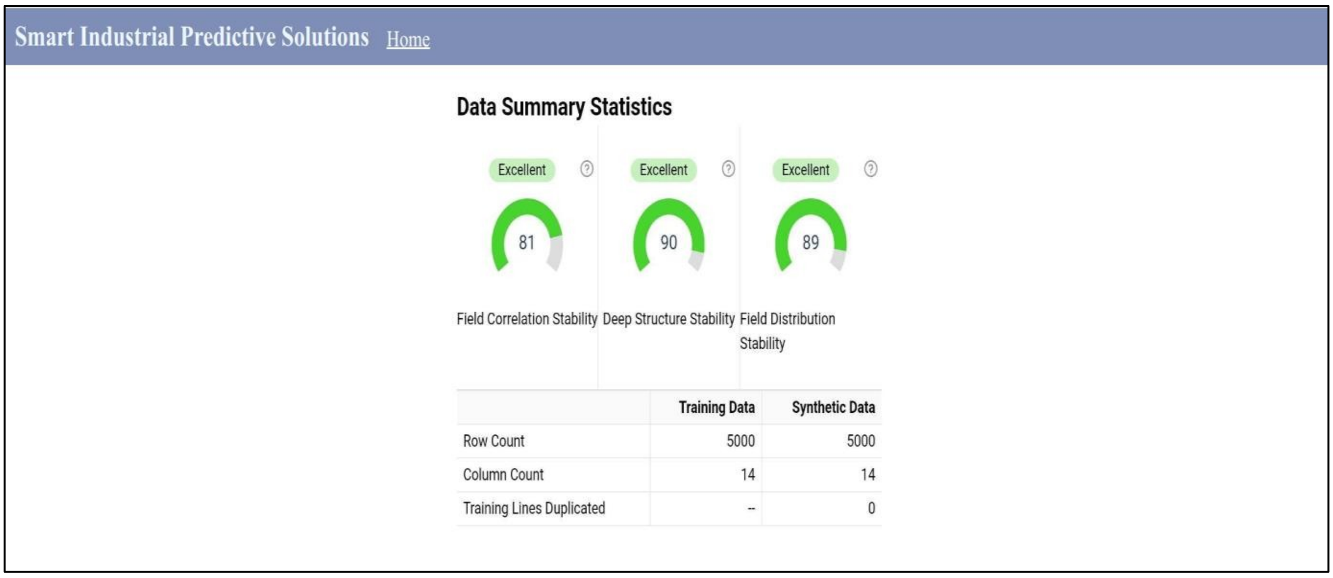
	Subscription plan 1	**Subscription plan 2**	**Subscription plan 3**
Personalised web-based monitoring system	✓	✓	✓
Personalised predictive system	✓	✓	✓
Validity	**6 months**	**1 year**	**3 years**
IOT based services*(optional)	✓	✓	✓
Services Charges**(fixed charges)	**20000**	**35000**	**80000**

***modification charges & IOT based service didn't included*

Table 2.1 : Subscription price table

4.2. Product & service description:

4.2.1. Real time Web Based monitoring and predictive maintenances app:



Data Summary Statistics

Field Correlation Stability — 81
Deep Structure Stability — 90
Field Distribution Stability — 89

	Training Data	Synthetic Data
Row Count	5000	5000
Column Count	14	14
Training Lines Duplicated	--	0

fig 2.1. prototype of real time web based predictive maintenance services

4.2.2. How it works?

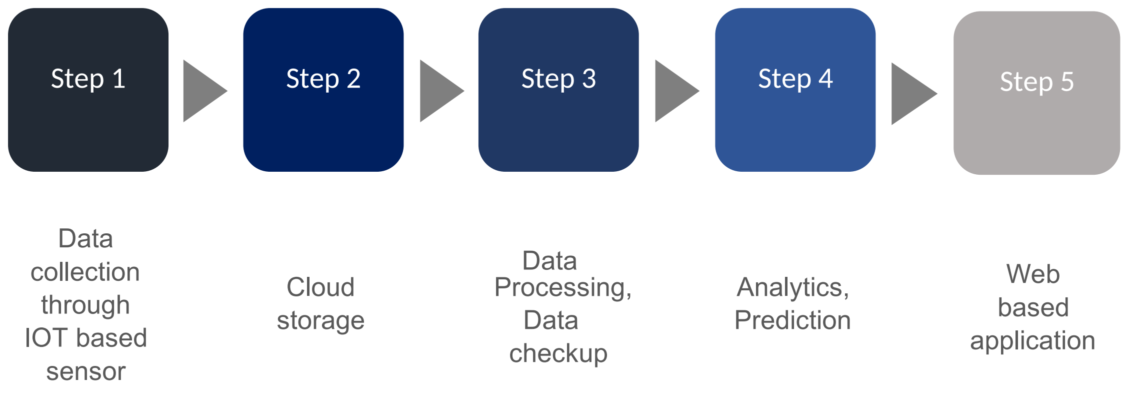
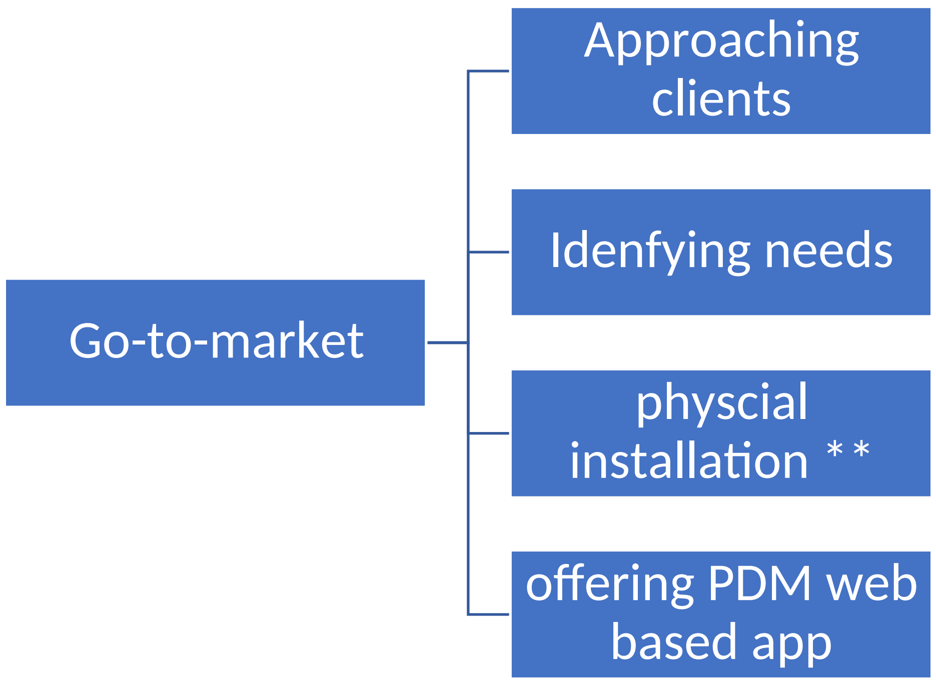


Fig 2.2. Data pipeline of web-based app

4.3 Go-to-market Strategy:



*** done by third-party*

Fig 2.3. Flow chart of Go-to-Market Strategy

4.3.1. Step 1: Approaching Clients:

Approaching potential clients as a predictive maintenance service provider can be a strategic process that involves utilizing various marketing channels and direct sales through affiliations. Here's a step-by-step approach to effectively approach clients in this industry:

4.3.1.1. Market Research and Segmentation:

Begin by conducting thorough market research to understand target audience. Identify industries and businesses that can benefit from predictive maintenance services, such as manufacturing, energy, healthcare, and transportation.

Industry	Feasibility	Viability	Monetization
Manufacturing	9	8	7
Oil and gas	8	7	6
Transportation	8	7	6
Energy	7	6	5
Healthcare	7	6	5

Table 2.2. Feasibility, viability, Monetization of top 5 PDM sales driven sector

Out of these industries Manufacturing, Industry seems to be targeting segment with good feasibility, viability, monetization

4.3.1.2. Compelling Value Proposition:

We have developed compelling value proposition of subscription service to attract customer from different segment, market size, that outlines the benefits of PDM service. Our solution can help clients reduce downtime, lower maintenance costs, and improve overall equipment efficiency**.**

4.3.1.3. Marketing:

- Developing Website predictive maintenance services, case studies, client testimonials, and industry expertise.
- Implement search engine optimization (SEO) strategies and social media marketing to ensure website ranks well on search engines for relevant keywords.
- Creating informative and educational content related to predictive maintenance, such as blog posts, whitepapers, infographics, and videos. Share industry insights, news, and success stories to engage your audience and build relationships.
- Building an email list of potential clients who have expressed interest in your services. Send personalized and informative emails that address their pain points and offer solutions.

4.3.1.4. Future development:

Showcase success stories and case studies on your website to demonstrate the real-world benefits your predictive maintenance services have delivered to clients.

Regularly assess the effectiveness of your marketing and sales strategies and make adjustments as needed. Stay up-to-date with industry trends and technologies to ensure your services remain competitive.

4.3.1 Step 2: Identifying needs:

- By active communication understands need and wants of client.
- Observe and Analyse their operations, processes, and any existing systems or equipment.
- Engage the client in collaborative problem-solving discussions and offer expertise and insights and do comprehensive documentation

4.3.2 Step 3: Physical Installation:

- Physical installation will be done by third party IOT service-based company with data security and integration

4.3.3 Step 4: Offering Solutions:

- Prioritize Needs and offer customize solutions to meet the specific needs and goals of the client.
- Offer customer support and data security

5. Financial equation:

One of best way to calculate way of profit would be by using following linear equation where each term has its sperate significance:

$$y = m * x \quad (l) + c$$

$y =$ total profit

$m =$ cost of subscription services or (expected total sales/No. of customer)

$x(l) =$ market

$c =$ constant depends on other cost including primary and secondary cost

Therefore, this formula can also be written as

$$\textit{Total profit} = \frac{\textit{Expected total sales}}{\textit{No.of customer}} * \textit{market} + \textit{constant cost}$$

5.1 Justifying Value of each variable

5.1.1 Value of C:

C is fixed cost associated with primary and supportive structure of business. It includes **supportive structure** human resource cost, technology cost, firm infrastructure, procurement and **primary structure** cost like cloud infrastructure, product development, distribution and reliability channel, marketing sales and services.

Supportive Activities	Cost	Justification
Human Resource	₹ 12,00,000.00	4 employees
Technology development	₹ 1,00,000.00	software development/ security
Firm infrastructure	₹ 3,00,000.00	office furniture, rent, electricity
Procurement	₹ 5,00,000.00	5000 per sensor
Primary Activities	**Cost**	**Justification**
Cloud Infrastructure	₹ 1,00,000.00	GCP, AWS storage bucket, compute engine
Product development	₹ 50,000.00	resources, data
Distribution and reliability	₹ 2,00,000.00	distribution channels,
Marketing and Sales	₹ 1,00,000.00	affiliate based,
Services	₹ 1,00,000.00	physical deployment of sensor, & maintenances, data quality checkup, personalised web-based app
Total Cost	₹26,50,000.00	

Table 3.1 Value chain cost analysis of SIPS

Since we are in ideation stage, let us consider cost of supportive structure would be zero, but still primary structure of business will cost around 5 to 6 lakh rupees.

Therefore, **let us consider value of C = 6 lakh.**

Value of C is doesn’t directly depend upon market.

5.1.2 Value of x(l):

This suggests that x(l) is a function that describes how market performance (x) is influenced by some variable or parameter (l).

The condition on which x(l) depends will vary depending on the factors that affect market performance in your predictive maintenance startup. These factors could include:

Customer Demand: If customer demand plays a significant role in profitability, l could represent variables related to customer behaviour, such as the number of customers, market segments, or demographics.

Industry-Specific Metrics: Industry growth and external factors, geopolitical scenario.

Since there is no direct formula for x(l), Let us consider following two scenarios for determining value of x

Current market size for predictive maintenance services in India 1.5 billion $ (by McKinsey), with currently 22% adoption rate for PDM in overall manufacturing units in India with 26.7 % CAGR growth rate.

5.1.3 Value of m:

Value of m can be determined by comprehensive market segmentation techniques on manufacturing client, by size of client company, no. of employees, no. of machines, no. of new machines, old machines, current status (closed or in operations) etc.

For time being we will take it as **35000**

Therefore equation will be,

$$y = 35000 * x \quad (l) + 600000$$

5.2 Financial equation for determine benefits of PdM:

The following financial equation can be used to estimate the benefits of predictive maintenance for the manufacturing industry:

$$\begin{aligned} &\textit{Benefits of predictive maintenance} \\ &\quad = (\textit{Cost of downtime} * \textit{Downtime reduction}) + (\textit{Cost of repairs} \\ &\quad * \textit{Repair reduction}) + (\textit{Cost of asset replacement} \\ &\quad * \textit{Asset life extension}) \end{aligned}$$

Where:

Cost of downtime: The cost of downtime is the cost of lost production, revenue, and customer satisfaction due to machine and equipment failures.

Downtime reduction: The downtime reduction is the percentage reduction in downtime achieved through predictive maintenance.

Cost of repairs: The cost of repairs is the cost of repairing and replacing failed machines and equipment.

Repair reduction: The repair reduction is the percentage reduction in repairs achieved through predictive maintenance.

Cost of asset replacement: The cost of asset replacement is the cost of replacing machines and equipment that have reached the end of their useful life.

Asset life extension: The asset life extension is the percentage increase in asset life achieved through predictive maintenance.

For example, suppose that a manufacturing company has an annual cost of downtime of $1 million. If the company implements predictive maintenance and achieves a 20% reduction in downtime, the annual savings from reduced downtime would be $200,000.

This can also be fine tune by adding

Following formula for overall maintenances

Total Maintenance Cost (TMC) equation in the form of a linear equation as follows:

TMC = PMC + CMC + PrMC

$$TMC = (m_p * x) + (m_c * x) + (m_{pr} * x) + c$$

Where:

TMC represents the Total Maintenance Cost.

PMC, CMC, and PrMC represent the Predictive Maintenance Cost, Corrective Maintenance Cost, and Preventive Maintenance Cost, respectively.

m_p, m_c, *and* m_{pr} are the coefficients representing the slopes of the respective costs in relation to some independent variable x (e.g., time or production volume).

c is the constant term, representing fixed or baseline maintenance costs.

6. Target Specifications and Characterization:

6.1. Equipment Compatibility: "Smart Industrial Predictive Solutions" should be compatible with a wide range of industrial machinery and equipment, including but not limited to manufacturing machines, motors, pumps, conveyors, and processing equipment.

6.2. Data Integration: The system must seamlessly integrate with various data sources, such as sensors, IoT devices, and historical maintenance records. It should be able to collect, process, and analyze data from different formats and protocols.

6.3. Real-time Monitoring: The solution should provide real-time monitoring capabilities to continuously assess the health and performance of equipment, enabling immediate responses to emerging issues.

6.4. Predictive Analytics: "Smart Industrial Predictive Solutions" should employ advanced predictive analytics and machine learning algorithms to forecast equipment failures with a high degree of accuracy.

6.5. User-Friendly Interface: The user interface should be intuitive and user-friendly, catering to both technical and non-technical users within industrial settings.

6.6. Customization: The system should allow for customization to adapt to specific industrial processes and equipment types, accommodating diverse user requirements.

6.7. Alerts and Notifications: It should offer automated alerts and notifications to inform maintenance teams and operators about impending issues, facilitating proactive maintenance planning.

6.8. Scalability: The solution must be scalable to accommodate the needs of small to large-scale industrial operations, ensuring flexibility for businesses of varying sizes.

6.9. Security: Robust security measures should be in place to protect sensitive industrial data and ensure compliance with industry-specific regulations.

6.10. Cost-Effectiveness: "Smart Industrial Predictive Solutions" should demonstrate a strong return on investment (ROI) by reducing maintenance costs, minimizing downtime, and optimizing resource utilization.

6.11. Reliability: The system should be highly reliable, minimizing false alarms and ensuring accurate predictions to build trust among industrial users.

6.12. Interoperability: It should support interoperability with existing industrial control systems and software, promoting seamless integration into the existing operational framework.

6.13. Maintenance Insights: In addition to predictive alerts, the solution should provide insights into the root causes of equipment issues, aiding maintenance teams in making informed decisions.

6.14. Compliance: The system should assist industries in meeting regulatory compliance requirements related to safety and equipment reliability.

6.15. Data Transparency: Transparency in data collection, processing, and analysis should be a priority, allowing users to understand and trust the system's recommendations.

These specifications and characteristics define the essential features and capabilities that "Smart Industrial Predictive Solutions" should possess to effectively address the needs and challenges of industrial customers.

Data Analysis and Model Development Report

- Yash Mayur

7. Data Collection

There were 3 types of dataset available for predictive maintenance; out of that ai4i2020 dataset was used as it has 10,000 product information. So it was good for training and testing purposes. Silicon wafer data was also collected but electronic maintenance was somehow getting diverted from actual 3 types of dataset what we already collected. Hence Silicon wafer data is kept as reference purpose.

Below is the feature description of the dataset used for analysis and model development.

a. Feature Description:

1) **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.

2) **Type**: just the product type L, M or H from column 2.

3) **Air Temperature [K]**: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K.

4) **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.

5) **Rotational Speed [rpm]:** calculated from a power of 2860 W, overlaid with a normally distributed noise.

6) **Torque [Nm]:** torque values are normally distributed around 40 Nm with a SD = 10 Nm and no negative values.

7) **Tool Wear [min]:** (breakdown and gradual failure of a cutting tool due to regular operation) The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

8) **Machine failure:** Machine Failure label that indicates, whether the machine has failed in this particular datapoint for any of the following failure modes are true. The machine failure consists of five independent failure modes as follows:

a) **Tool wear failure (TWF):** the tool will be replaced of fail at a randomly selected tool wear time between 200 - 240 mins (120 times in our dataset). At this point in time, the tool is replaced 69 times, and fails 51 times (randomly assigned).

b) **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 tools rotational speed is below 1380 rpm. This is the case for 115 data points.¶

c) **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.

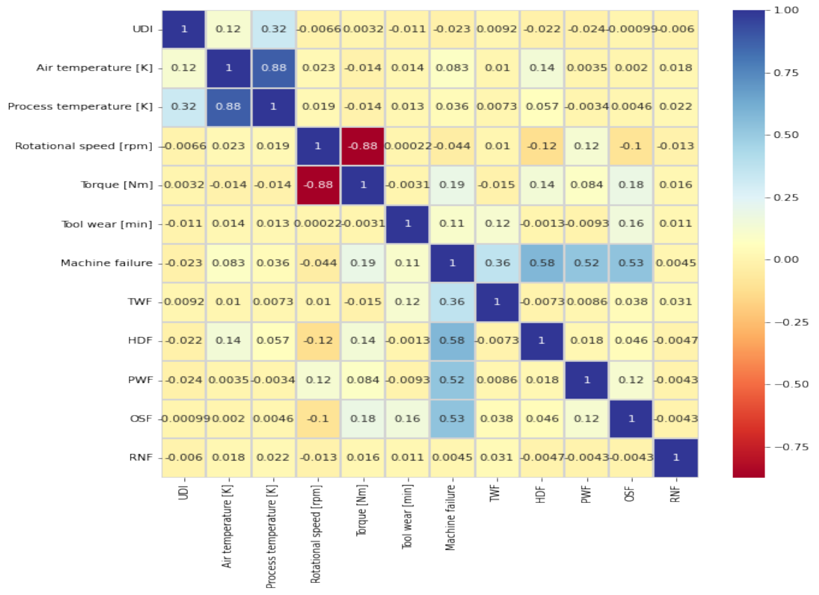
d) **Overstrain failure (OSF):** if the product of tool wear and torque exceeds 11,000 minNm for the L product variant (12,000 M, 13,000 H), the process fails due to overstrain. This is true for 98 datapoints.

e) **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.

Note that if at least one of the above failure modes is true, the process fails and the 'machine failure' label is set to 1. It is therefore not transparent to the machine learning method, which of the failure modes has caused the process to fail.

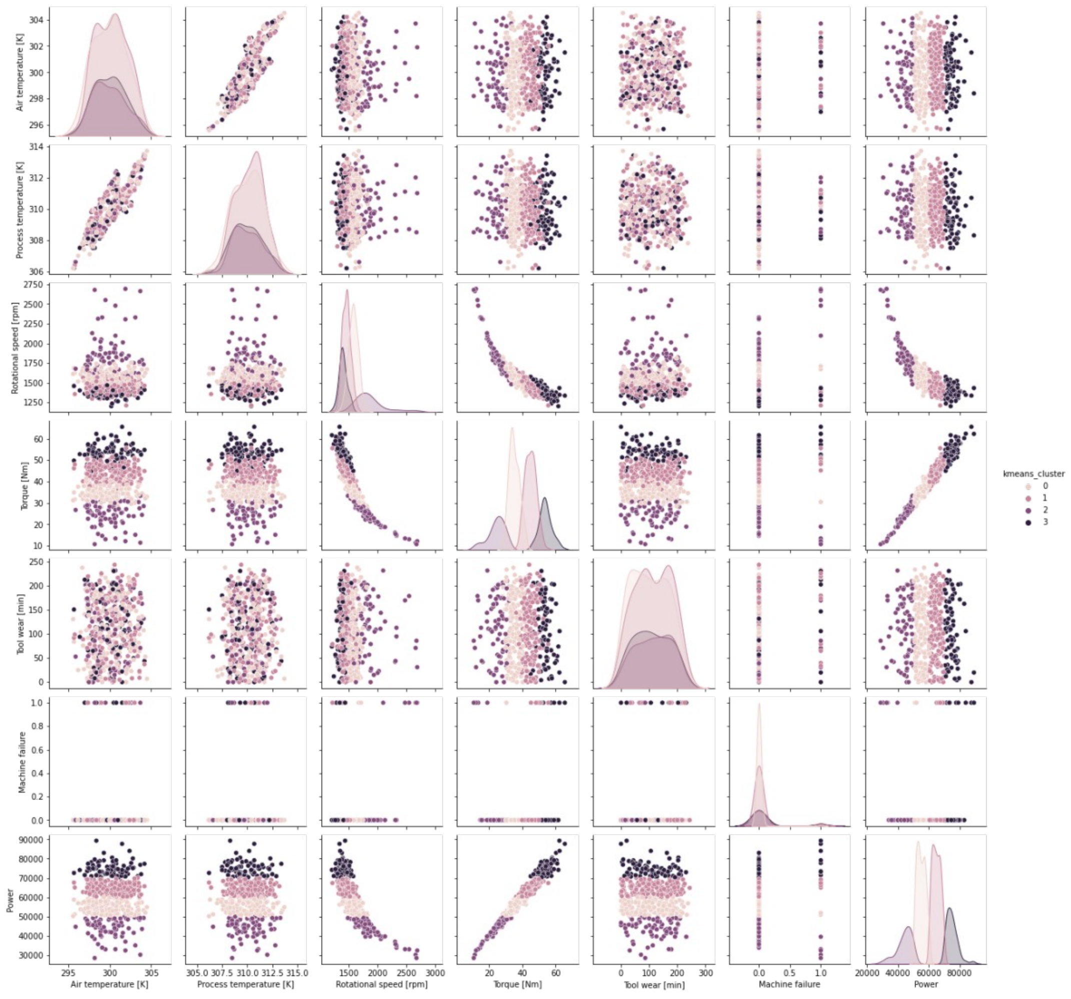
8. Exploratory Data Analysis

a. Correlation:



We can see that there are strongly correlated features namely process temperature and air temperature. Torque and rotational speed are also negatively correlated. We can drop one of the temperatures, but the torque to rotational speed difference might be a indication of a failure, so we'll keep both.

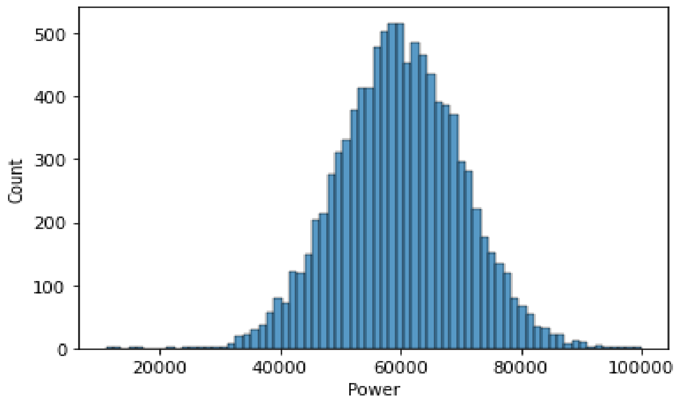
b. Linearly dependable:



Process Temperature and Air Temperature have Linear Relationship with each other and both the data has gaussian normal distribution

c. Power Attribute:

A new feature has been added to dataset



Power = Torque × Rotational speed

d. Silhouette Score:

We can say that the clusters are well apart from each other as the silhouette score is closer to 1 and greater than 0.5

9. Feature Engineering (Data Preprocessing):

a. Encoding of Categorical Feature:

Use any one from the below encoding technique

i. **Label Encoding:**

ii. **One Hot Encoding:**

b. Imbalance Dataset Handling:

As analysed and Mentioned earlier that the dataset is imbalanced so we have to use SMOTE technique on training data inorder to make balance data

c. Missing Values

There are no missing values present so the dataset is clean

10. Model Development and Evaluation:

- As this is binary classification problem, we used many types of classification algorithms like Random forest, Decision tree, KNN classifier etc (all details available in Jupiter notebook.
- Catboost classifier comes to be the best performing classifier with highest accuracy.
- Model.pkl file is created for catboost classifier for the deployment purpose.
- Classifier model is deployed on Google server (GCP) and it is attached to webapp as well.

predictive_maintainance_analysis

September 28, 2023

1 Predictive Maintenance

- By Yash Mayur

2 Stages of Data Science

1) Data Collection

2) Exploratory Data Analysis (visual and Descriptive)

3) Feature Engineering (Data Preprocessing)

4) Model Creation and Evaluation

5) Model Deployment

[1]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import warnings
from pandas_profiling import ProfileReport
from sklearn.preprocessing import (LabelEncoder,OneHotEncoder,MaxAbsScaler)

from sklearn.metrics import silhouette_score
from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SVMSMOTE
from sklearn.metrics import f1_score, precision_score, recall_score,␣
 ↪accuracy_score,roc_auc_score,confusion_matrix

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import (
    GradientBoostingClassifier,
    RandomForestClassifier,
)
```

```
from catboost import CatBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
```

```
C:\Users\Yash Mayur\AppData\Local\Temp\ipykernel_12308\1597575372.py:8:
DeprecationWarning: `import pandas_profiling` is going to be deprecated by April
1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport
```

3 1) Data Collection

[40]:
```
df = pd.read_csv('ai4i2020.csv')
df.head(10)
```

[40]:
```
   UDI Product ID Type  Air temperature [K]  Process temperature [K]  \
0    1     M14860    M                298.1                    308.6
1    2     L47181    L                298.2                    308.7
2    3     L47182    L                298.1                    308.5
3    4     L47183    L                298.2                    308.6
4    5     L47184    L                298.2                    308.7
5    6     M14865    M                298.1                    308.6
6    7     L47186    L                298.1                    308.6
7    8     L47187    L                298.1                    308.6
8    9     M14868    M                298.3                    308.7
9   10     M14869    M                298.5                    309.0

   Rotational speed [rpm]  Torque [Nm]  Tool wear [min]  Machine failure  TWF  \
0                    1551         42.8                0                0    0
1                    1408         46.3                3                0    0
2                    1498         49.4                5                0    0
3                    1433         39.5                7                0    0
4                    1408         40.0                9                0    0
5                    1425         41.9               11                0    0
6                    1558         42.4               14                0    0
7                    1527         40.2               16                0    0
8                    1667         28.6               18                0    0
9                    1741         28.0               21                0    0

   HDF  PWF  OSF  RNF
0    0    0    0    0
1    0    0    0    0
2    0    0    0    0
3    0    0    0    0
4    0    0    0    0
```

```
5   0   0   0   0
6   0   0   0   0
7   0   0   0   0
8   0   0   0   0
9   0   0   0   0
```

3.1 Feature Description

1)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.

2) Type: just the product type L, M or H from column 2.

3) Air Temperature [K]: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K.

4) 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.

5) Rotational Speed [rpm]: calculated from a power of 2860 W, overlaid with a normally distributed noise.

6) Torque [Nm]: torque values are normally distributed around 40 Nm with a SD = 10 Nm and no negative values.

7) Tool Wear [min]: (breakdown and gradual failure of a cutting tool due to regular operation) The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

8) A 'Machine failure' label that indicates, whether the machine has failed in this particular datapoint for any of the following failure modes are true. The machine failure consists of five independent failure modes as follows:

a) Tool wear failure (TWF): the tool will be replaced of fail at a randomly selected tool wear time between 200 - 240 mins (120 times in our dataset). At this point in time, the tool is replaced 69 times, and fails 51 times (randomly assigned).

b) 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 tools rotational speed is below 1380 rpm. This is the case for 115 data points.

c) 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.

d) 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.

e) 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.

3.1.1 Note: If at least one of the above failure modes is true, the process fails and the 'machine failure' label is set to 1. It is therefore not transparent to the machine learning method, which of the failure modes has caused the process to fail.

4 2) Exploratory Data Analysis

```
[41]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 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   Machine failure          10000 non-null  int64
 9   TWF                      10000 non-null  int64
 10  HDF                      10000 non-null  int64
 11  PWF                      10000 non-null  int64
 12  OSF                      10000 non-null  int64
 13  RNF                      10000 non-null  int64
dtypes: float64(3), int64(9), object(2)
memory usage: 1.1+ MB
```

4.0.1 Acronyms to be considered:

[K]: kelvin

[rpm]: revolutions per minute

[Nm]: newton-meter

[min]: minutes

```
[42]: # let's find the null values
      df.isnull().sum()
```

```
[42]: 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
      Machine failure            0
      TWF                        0
      HDF                        0
      PWF                        0
      OSF                        0
      RNF                        0
      dtype: int64
```

No null values present

```
[43]: # Let's get some description of data
      df.describe().T
```

```
[43]:                            count        mean          std     min      25%  \
      UDI                      10000.0  5000.50000  2886.895680     1.0  2500.75
      Air temperature [K]      10000.0   300.00493     2.000259   295.3   298.30
      Process temperature [K]  10000.0   310.00556     1.483734   305.7   308.80
      Rotational speed [rpm]   10000.0  1538.77610   179.284096  1168.0  1423.00
      Torque [Nm]              10000.0    39.98691     9.968934     3.8    33.20
      Tool wear [min]          10000.0   107.95100    63.654147     0.0    53.00
      Machine failure          10000.0     0.03390     0.180981     0.0     0.00
      TWF                      10000.0     0.00460     0.067671     0.0     0.00
      HDF                      10000.0     0.01150     0.106625     0.0     0.00
      PWF                      10000.0     0.00950     0.097009     0.0     0.00
      OSF                      10000.0     0.00980     0.098514     0.0     0.00
      RNF                      10000.0     0.00190     0.043550     0.0     0.00

                                  50%      75%      max
      UDI                      5000.5  7500.25  10000.0
      Air temperature [K]       300.1   301.50    304.5
      Process temperature [K]   310.1   311.10    313.8
      Rotational speed [rpm]   1503.0  1612.00   2886.0
      Torque [Nm]                40.1    46.80     76.6
      Tool wear [min]           108.0   162.00    253.0
      Machine failure             0.0     0.00      1.0
```

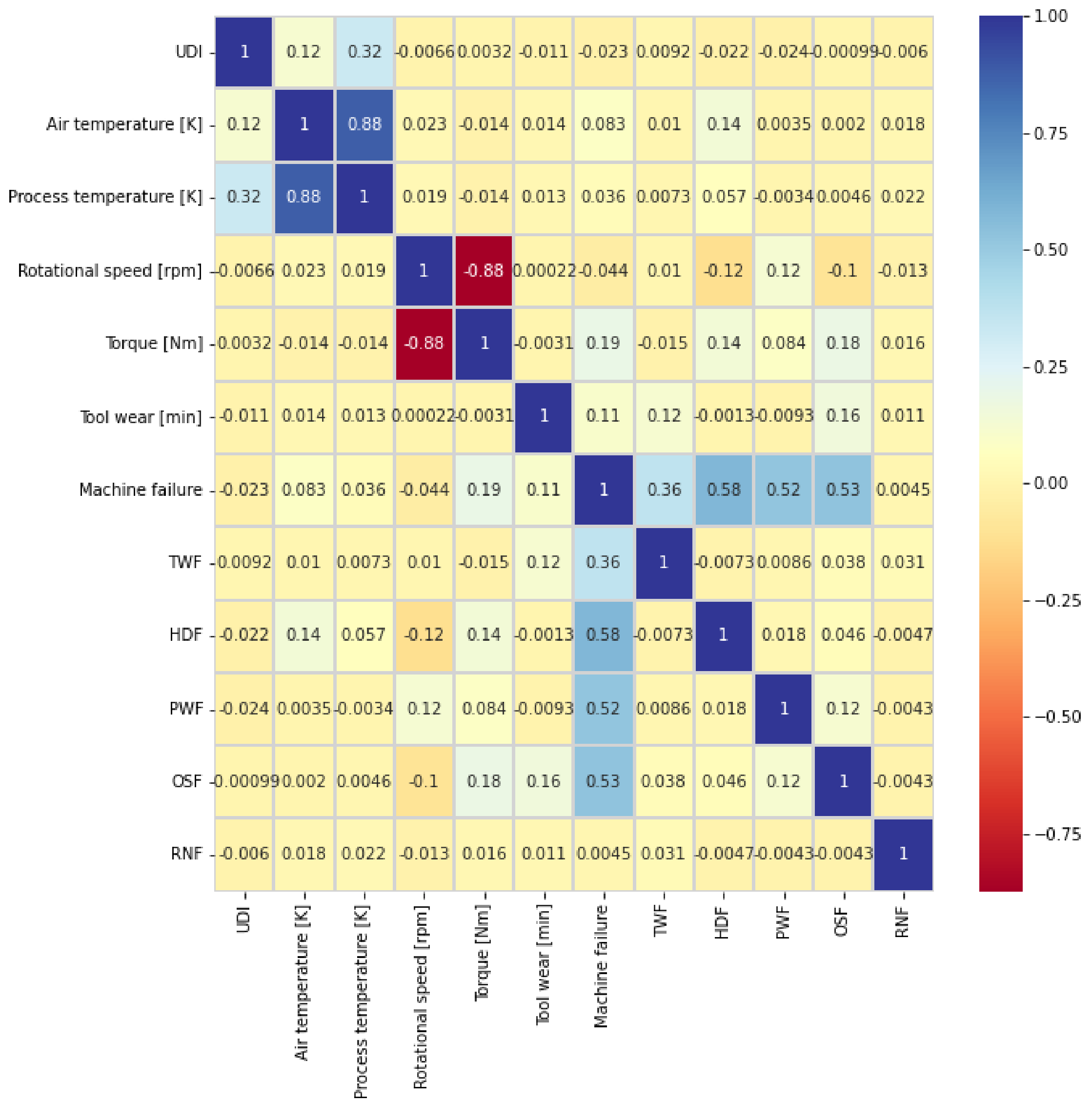
```
TWF                           0.0     0.00      1.0
HDF                           0.0     0.00      1.0
PWF                           0.0     0.00      1.0
OSF                           0.0     0.00      1.0
RNF                           0.0     0.00      1.0
```

[44]:
```
# Let's convert int datatype to float for preprocessing purposes

for feature in df.columns:
    if df[feature].dtype != 'O':
        df[feature] = df[feature].astype(float)
```

[45]:
```
# Let's find the correlation of features with each other

plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),
            cmap='RdYlBu',
            annot=True,
            linewidths=0.2,
            linecolor='lightgrey').set_facecolor('white')
```



4.0.2 We can see that there are strongly correlated features namely process temperature and air temperature. Torque and rotational speed are also negatively correlated. We can drop one of the temperatures, but the torque to rotational speed difference might be a indication of a failure, so we'll keep both.

5 So now we will drop some columns which is not usefull

dropping the indices and product id

[46]:
```
df.drop(['UDI','Product ID'],axis=1,inplace=True)
```

Drop the failure modes, as currently we are only interested whether something is failed or not.

```
[47]: df.drop(['TWF','HDF','PWF','OSF','RNF'],axis=1,inplace=True)
```

Let's check balanced relation of the independent features with thetarget feature

```
[48]: df_group = df.groupby(['Machine failure'])
      df_group.count()
```

```
[48]:                  Type  Air temperature [K]  Process temperature [K]  \
      Machine failure
      0.0              9661                 9661                     9661
      1.0               339                  339                      339

                       Rotational speed [rpm]  Torque [Nm]  Tool wear [min]
      Machine failure
      0.0                                9661         9661             9661
      1.0                                 339          339              339
```

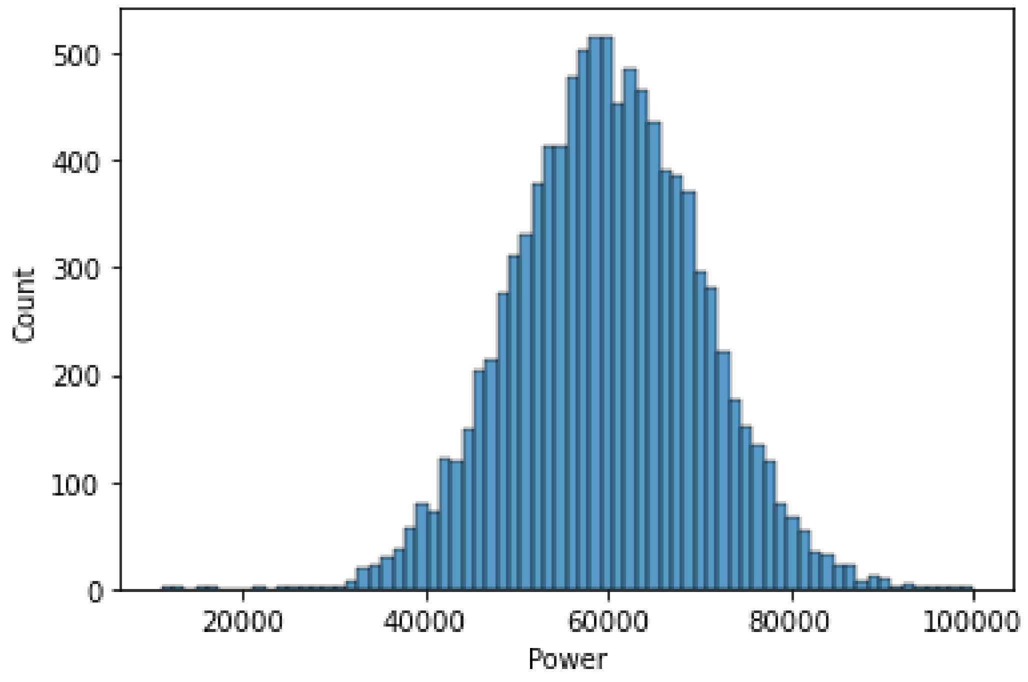
5.0.1 The above table clearly shows that the dataset is imbalanced

6 Let's derive a power attribute using this formula:

Power = Torque $\times$ Rotational speed

```
[49]: df['Power'] = df[['Rotational speed [rpm]', 'Torque [Nm]']].product(axis=1)
      sns.histplot(df['Power'])
```

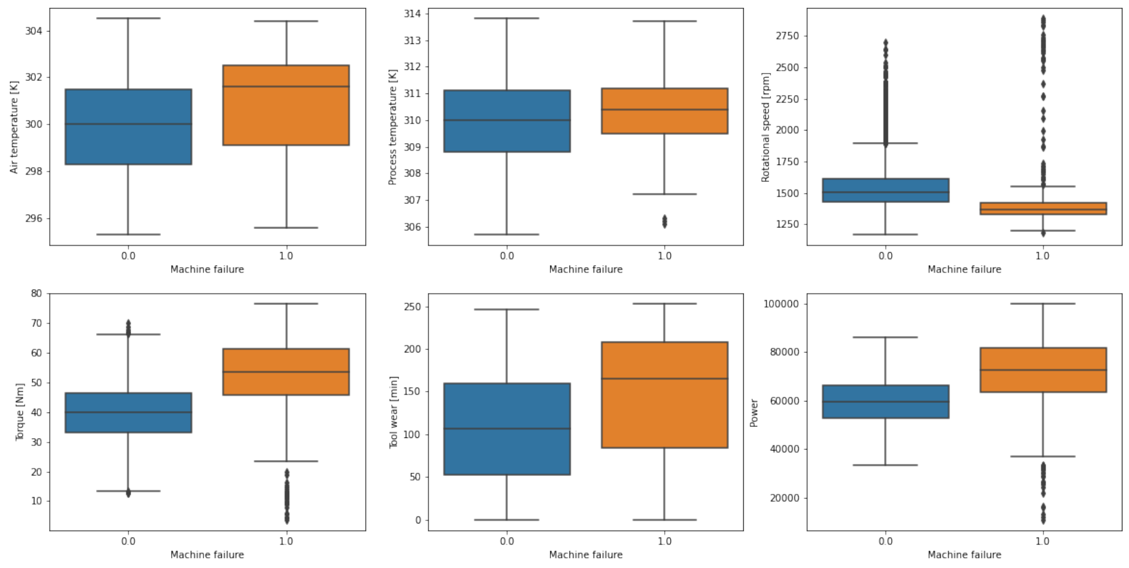
```
[49]: <AxesSubplot:xlabel='Power', ylabel='Count'>
```



We get gaussian normal distribution for the power feature

```
[50]: ## Boxplot

fig, ax = plt.subplots(2, 3, figsize=(20, 10))
i=0
for _, col in enumerate(df.columns):
    if col == "Machine failure":
        continue
    elif col == "Type":
        continue
    else:
        sns.boxplot(x="Machine failure", y=col, data=df, ax=ax[i//3][i%3])
        i += 1
```



7 3) Descriptive Analysis

7.0.1 Clustering

```
[51]: # features to use for clustering
feature_list = [feature for feature in df.columns if df[feature].dtype ==␣
  ↪'float64']
X = df[feature_list]
from sklearn.cluster import KMeans

# K-means clustering
```

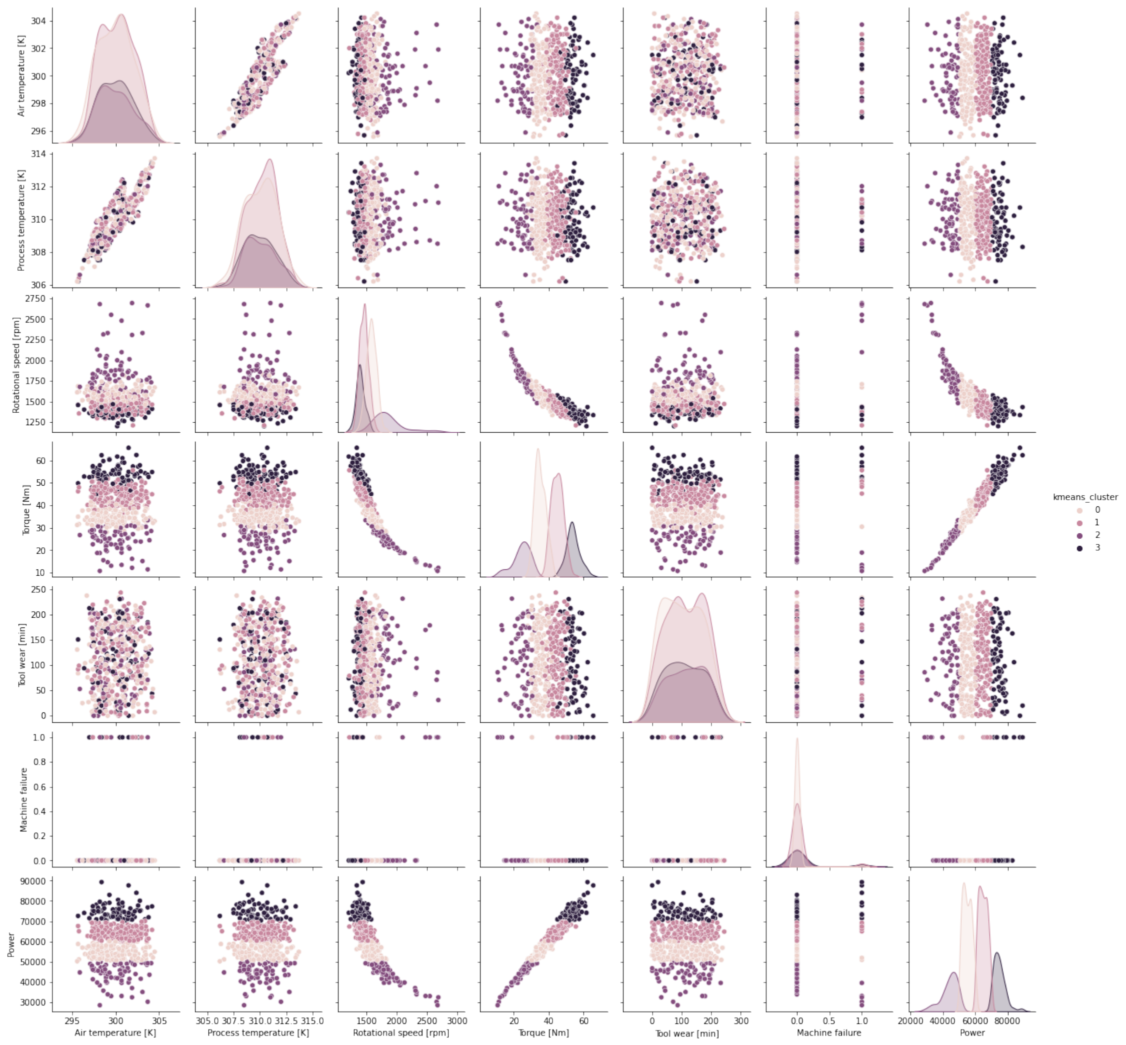
```
kmeans = KMeans(init='random',  n_clusters=4,
                n_init=10, max_iter=300, random_state=42)
kmeans.fit(X)

df["kmeans_cluster"] = kmeans.predict(X)
```

```
[52]: plt.figure(figsize=(10, 8))

      # create a pairplot of the data, colored by cluster label
      sns.pairplot(df.sample(frac=0.05), hue="kmeans_cluster", vars=feature_list)
      plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



7.1 From the above results we can clearly observe

Process Temperature and Air Temperature have Linear Relationship with each other and both the data has gaussian normal distribution

[53]:
```python
# calculate the silhouette coefficient
score = silhouette_score(X, kmeans.predict(X))

print(f"Silhouette Coefficient: {score}")
```

```
Silhouette Coefficient: 0.5191499897517547
```

7.1.1 We can say that the clusters are well apart from each other as the silhouette score is closer to 1.

[33]:
```python
## Let's convert the dataframe which can be used further in our project

os.makedirs(os.getcwd(),exist_ok=True)
df.to_csv("data.csv",header=True,index=False)
```

8 4) Feature Engineering (Data Pre-processing) and Feature Selection

8.0.1 Encoding of Categorical Feature i.e. Type

Use any one only

1) Label Encoding or 2) One Hot Encoding

[17]:
```python
## Label Encoding

le = LabelEncoder()
df['Type'] = le.fit_transform(df['Type'])
df.head()
```

[17]:
```
   Type  Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
0     2                298.1                    308.6                  1551.0
1     1                298.2                    308.7                  1408.0
2     1                298.1                    308.5                  1498.0
3     1                298.2                    308.6                  1433.0
4     1                298.2                    308.7                  1408.0

   Torque [Nm]  Tool wear [min]  Machine failure    Power  kmeans_cluster
0         42.8              0.0              0.0  66382.8               1
1         46.3              3.0              0.0  65190.4               1
2         49.4              5.0              0.0  74001.2               3
3         39.5              7.0              0.0  56603.5               0
4         40.0              9.0              0.0  56320.0               0
```

[54]:
```
# One Hot Encoding

frame = pd.get_dummies(data=df['Type'],drop_first=True)
df.drop(columns='Type',axis=1,inplace=True)
df['L'] = frame['L']
df['M'] = frame['M']
df.head()
```

[54]:
```
   Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
0                298.1                    308.6                  1551.0
1                298.2                    308.7                  1408.0
2                298.1                    308.5                  1498.0
3                298.2                    308.6                  1433.0
4                298.2                    308.7                  1408.0

   Torque [Nm]  Tool wear [min]  Machine failure    Power  kmeans_cluster  L  \
0         42.8              0.0              0.0  66382.8               1  0
1         46.3              3.0              0.0  65190.4               1  1
2         49.4              5.0              0.0  74001.2               3  1
3         39.5              7.0              0.0  56603.5               0  1
4         40.0              9.0              0.0  56320.0               0  1

   M
0  1
1  0
2  0
3  0
4  0
```

[55]:
```
X = df.drop(columns=['Machine failure','kmeans_cluster'],axis=1)
y = df['Machine failure']
X
```

[55]:
```
      Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
0                   298.1                    308.6                  1551.0
1                   298.2                    308.7                  1408.0
2                   298.1                    308.5                  1498.0
3                   298.2                    308.6                  1433.0
4                   298.2                    308.7                  1408.0
…                     …                        …                      …
9995                298.8                    308.4                  1604.0
9996                298.9                    308.4                  1632.0
9997                299.0                    308.6                  1645.0
9998                299.0                    308.7                  1408.0
9999                299.0                    308.7                  1500.0

      Torque [Nm]  Tool wear [min]    Power  L  M
```

```
0             42.8              0.0  66382.8  0  1
1             46.3              3.0  65190.4  1  0
2             49.4              5.0  74001.2  1  0
3             39.5              7.0  56603.5  1  0
4             40.0              9.0  56320.0  1  0
…            …               …       … ..  ..
9995          29.5             14.0  47318.0  0  1
9996          31.8             17.0  51897.6  0  0
9997          33.4             22.0  54943.0  0  1
9998          48.5             25.0  68288.0  0  0
9999          40.2             30.0  60300.0  0  1

[10000 rows x 8 columns]
```

[59]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
↪random_state = 42)

X_test.shape
```

[59]:
```
(2500, 8)
```

8.0.2 As analysed and Mentioned earlier that the dataset is imbalanced so we have to use SMOTE technique on training data inorder to make balance data

[60]:
```
oversample = SVMSMOTE(random_state = 42)

X_train, y_train = oversample.fit_resample(X_train, y_train)
```

[61]:
```
scale = MaxAbsScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.transform(X_test)
```

9 5) Model Creation and Evaluation

9.0.1 a) Logistic Regression

[62]:
```
# Logistic Regression

model_dict = dict()

model = LogisticRegression().fit(X_train,y_train)
y_pred = model.predict(X_test) # These are the predictions from the test data.
y_pred
```

[62]:
```
array([0., 0., 0., …, 0., 0., 0.])
```

```
[63]: accuracy = accuracy_score(y_test, y_pred)
      recall = recall_score(y_test, y_pred, average='weighted')
      precision = precision_score(y_test, y_pred, average='weighted')
      f1s = f1_score(y_test, y_pred, average='weighted')
      ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
        ↪average='weighted')
      confusion_mat = confusion_matrix(y_test,y_pred)

      model_dict.update({"Logistic Regression":accuracy})
      print("Accuracy: "+ "{:.2%}".format(accuracy))
      print("Recall: "+ "{:.2%}".format(recall))
      print("Precision: "+ "{:.2%}".format(precision))
      print("F1-Score: "+ "{:.2%}".format(f1s))
      print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
      print("Confusion Matrix: \n")
      confusion_mat
```

```
Accuracy: 88.24%
Recall: 88.24%
Precision: 96.59%
F1-Score: 91.65%
ROC AUC score: 86.40%
Confusion Matrix:
```

```
[63]: array([[2156,  272],
             [  22,   50]], dtype=int64)
```

9.0.2 b) Decision Tree

```
[64]: # Decision Tree

      model = DecisionTreeClassifier().fit(X_train,y_train)
      y_pred = model.predict(X_test)
      y_pred
```

```
[64]: array([0., 0., 0., …, 0., 0., 0.])
```

```
[65]: accuracy = accuracy_score(y_test, y_pred)
      recall = recall_score(y_test, y_pred, average='weighted')
      precision = precision_score(y_test, y_pred, average='weighted')
      f1s = f1_score(y_test, y_pred, average='weighted')
      ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
        ↪average='weighted')
      confusion_mat = confusion_matrix(y_test,y_pred)

      model_dict.update({"Decision Tree":accuracy})
```

```
print("Accuracy: "+ "{:.2%}".format(accuracy))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
print("Confusion Matrix: \n")
confusion_mat
```

```
Accuracy: 96.20%
Recall: 96.20%
Precision: 97.35%
F1-Score: 96.66%
ROC AUC score: 82.55%
Confusion Matrix:
```

[65]:
```
array([[2356,   72],
       [  23,   49]], dtype=int64)
```

9.0.3 c) Random Forest Classifier

[66]:
```
# Random Forest Classifier

model = RandomForestClassifier(n_estimators =␣
↪100,n_jobs=-1,random_state=42,bootstrap=True,).fit(X_train,y_train)
y_pred = model.predict(X_test)
y_pred
```

[66]:
```
array([0., 0., 0., …, 0., 0., 0.])
```

[67]:
```
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted')
f1s = f1_score(y_test, y_pred, average='weighted')
ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
↪average='weighted')
confusion_mat = confusion_matrix(y_test,y_pred)

model_dict.update({"Random Forest Classifier":accuracy})
print("Accuracy: "+ "{:.2%}".format(accuracy))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
print("Confusion Matrix: \n")
confusion_mat
```

```
Accuracy: 97.64%
Recall: 97.64%
Precision: 98.07%
F1-Score: 97.81%
ROC AUC score: 96.88%
Confusion Matrix:
```

```
[67]: array([[2386,   42],
             [  17,   55]], dtype=int64)
```

9.0.4 d) Gradient Boosting classifier

```
[68]: model = GradientBoostingClassifier().fit(X_train,y_train)
      y_pred = model.predict(X_test)
      y_pred
```

```
[68]: array([0., 0., 0., …, 0., 0., 0.])
```

```
[69]: accuracy = accuracy_score(y_test, y_pred)
      recall = recall_score(y_test, y_pred, average='weighted')
      precision = precision_score(y_test, y_pred, average='weighted')
      f1s = f1_score(y_test, y_pred, average='weighted')
      ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
       ↪average='weighted')
      confusion_mat = confusion_matrix(y_test,y_pred)

      model_dict.update({"Gradient Boosting":accuracy})
      print("Accuracy: "+ "{:.2%}".format(accuracy))
      print("Recall: "+ "{:.2%}".format(recall))
      print("Precision: "+ "{:.2%}".format(precision))
      print("F1-Score: "+ "{:.2%}".format(f1s))
      print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
      print("Confusion Matrix: \n")
      confusion_mat
```

```
Accuracy: 96.80%
Recall: 96.80%
Precision: 97.98%
F1-Score: 97.23%
ROC AUC score: 97.07%
Confusion Matrix:
```

```
[69]: array([[2360,   68],
             [  12,   60]], dtype=int64)
```

9.0.5 e) K Neighbors Classifier

```
[70]: model = KNeighborsClassifier(n_neighbors=2).fit(X_train, y_train)
      y_pred = model.predict(X_test)
      y_pred
```

```
[70]: array([0., 0., 0., …, 0., 0., 0.])
```

```
[71]: accuracy = accuracy_score(y_test, y_pred)
      recall = recall_score(y_test, y_pred, average='weighted')
      precision = precision_score(y_test, y_pred, average='weighted')
      f1s = f1_score(y_test, y_pred, average='weighted')
      ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
       ↪average='weighted')
      confusion_mat = confusion_matrix(y_test,y_pred)

      model_dict.update({"KNN Classifier":accuracy})
      print("Accuracy: "+ "{:.2%}".format(accuracy))
      print("Recall: "+ "{:.2%}".format(recall))
      print("Precision: "+ "{:.2%}".format(precision))
      print("F1-Score: "+ "{:.2%}".format(f1s))
      print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
      print("Confusion Matrix: \n")
      confusion_mat
```

```
Accuracy: 94.96%
Recall: 94.96%
Precision: 97.19%
F1-Score: 95.84%
ROC AUC score: 85.81%
Confusion Matrix:
```

```
[71]: array([[2323,  105],
             [  21,   51]], dtype=int64)
```

9.0.6 f) Gaussian Naive Bayes Classifier

```
[72]: model = GaussianNB().fit(X_train, y_train)
      y_pred = model.predict(X_test)
      y_pred
```

```
[72]: array([0., 0., 0., …, 0., 0., 0.])
```

```
[73]: accuracy = accuracy_score(y_test, y_pred)
      recall = recall_score(y_test, y_pred, average='weighted')
      precision = precision_score(y_test, y_pred, average='weighted')
```

```
f1s = f1_score(y_test, y_pred, average='weighted')
ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
 ↪average='weighted')
confusion_mat = confusion_matrix(y_test,y_pred)

model_dict.update({"Gaussian Naive Bayes":accuracy})
print("Accuracy: "+ "{:.2%}".format(accuracy))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
print("Confusion Matrix: \n")
confusion_mat
```

```
Accuracy: 88.56%
Recall: 88.56%
Precision: 96.60%
F1-Score: 91.85%
ROC AUC score: 88.84%
Confusion Matrix:
```

[73]:
```
array([[2164,  264],
       [  22,   50]], dtype=int64)
```

9.0.7 g) Cat Boost Classifier

[74]:
```
model = CatBoostClassifier(verbose = False).fit(X_train,y_train)
y_pred = model.predict(X_test)
y_pred
```

[74]:
```
array([0., 0., 0., …, 0., 0., 0.])
```

[75]:
```
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted')
f1s = f1_score(y_test, y_pred, average='weighted')
ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
 ↪average='weighted')
confusion_mat = confusion_matrix(y_test,y_pred)

model_dict.update({"Catboost Classifier":accuracy})
print("Accuracy: "+ "{:.2%}".format(accuracy))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
print("Confusion Matrix: \n")
confusion_mat
```

```
Accuracy: 97.96%
Recall: 97.96%
Precision: 98.28%
F1-Score: 98.09%
ROC AUC score: 97.07%
Confusion Matrix:
```

```
[75]: array([[2392,   36],
             [  15,   57]], dtype=int64)
```

9.0.8 h) Nueral Network MLP Classifier

```
[76]: model = MLPClassifier(hidden_layer_sizes = (100,100,), activation='relu',␣
        ↪solver='adam', batch_size=2000,verbose=0).fit(X_train,y_train)
      y_pred = model.predict(X_test)
```

```
C:\Users\Yash Mayur\anaconda3\lib\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:692:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
[77]: accuracy = accuracy_score(y_test, y_pred)
      recall = recall_score(y_test, y_pred, average='weighted')
      precision = precision_score(y_test, y_pred, average='weighted')
      f1s = f1_score(y_test, y_pred, average='weighted')
      ROC_AUC = roc_auc_score(y_test, model.predict_proba(X_test)[:,1],␣
        ↪average='weighted')
      confusion_mat = confusion_matrix(y_test,y_pred)

      model_dict.update({"Nueral Network Classifier":accuracy})
      print("Accuracy: "+ "{:.2%}".format(accuracy))
      print("Recall: "+ "{:.2%}".format(recall))
      print("Precision: "+ "{:.2%}".format(precision))
      print("F1-Score: "+ "{:.2%}".format(f1s))
      print("ROC AUC score: "+ "{:.2%}".format(ROC_AUC))
      print("Confusion Matrix: \n")
      confusion_mat
```

```
Accuracy: 88.04%
Recall: 88.04%
Precision: 96.73%
F1-Score: 91.55%
```

```
ROC AUC score: 91.51%
Confusion Matrix:
```

```
[77]: array([[2148,  280],
             [  19,   53]], dtype=int64)
```

10 6) Best Model to use

```
[78]: sorted_dict = dict(sorted(model_dict.items(),key=lambda x:x[1],reverse=True))
      Model_name = list(sorted_dict.keys())[0]
      Model_accuracy = list(sorted_dict.values())[0]

      sent = "The Best Model is {0} and its accuracy is {1}".
       ↪format(Model_name,Model_accuracy*100)
      print(sent)
```

```
The Best Model is Catboost Classifier and its accuracy is 97.96000000000001
```

11 7) Quick Data Analysis Report

```
[ ]: dataframe = pd.read_csv('ai4i2020.csv')
     Report = ProfileReport(dataframe,explorative=True)
     Report
```

```
[ ]: Report.to_file("Data_Analysis_Report.pdf")
```

Data Analysis and Model Development Report

- **Malay Vyas**

1. Data Collection

There were 3 types of dataset available for predictive maintenance; out of that ai4i2020 dataset was used as it has 10,000 product information. So it was good for training and testing purposes. Silicon wafer data was also collected but electronic maintenance was somehow getting diverted from actual 3 types of dataset what we already collected. Hence Silicon wafer data is kept as reference purpose.
Below is the feature description of the dataset used for analysis and model development.

a. Feature Description:

1) **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.

2) **Type**: just the product type L, M or H from column 2.

3) **Air Temperature [K]**: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K.

4) **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.

5) **Rotational Speed [rpm]:** calculated from a power of 2860 W, overlaid with a normally distributed noise.

6) **Torque [Nm]:** torque values are normally distributed around 40 Nm with a SD = 10 Nm and no negative values.

7) **Tool Wear [min]:** (breakdown and gradual failure of a cutting tool due to regular operation) The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

8) **Machine failure:** Machine Failure label that indicates, whether the machine has failed in this particular datapoint for any of the following failure modes are true.

Generated Data Report:

Generated_Data_Report.pdf

Link: https://github.com/MalayVyas/Maintenance-Prediction/blob/main/Generated_Data_Report.pdf

Notebook File Link:

https://github.com/MalayVyas/Maintenance-Prediction/blob/main/Maintanence_Prediction%20(3).ipynb

Maintanenence_Prediction

September 28, 2023

1 Maintenance-Prediction Notebook

- By Malay Vyas

```
[96]: import pickle
      import numpy as np
      import pandas as pd
      import seaborn as sns
      from sklearn import tree
      import pandas_profiling as pp
      from tpot import TPOTClassifier
      import matplotlib.pyplot as plt
      from sklearn.preprocessing import LabelEncoder
      from sklearn.metrics import log_loss, f1_score
      from ydata_profiling import ProfileReport as pr
      from lazypredict.Supervised import LazyClassifier
      from sklearn.model_selection import train_test_split
      from sklearn.model_selection import cross_val_score
      from sklearn.ensemble import ExtraTreesClassifier
```

```
[73]: df1 = pd.read_csv('original_dataset.csv')
```

```
[74]: df2 = pd.read_csv('generated_dataset.csv')
```

```
[75]: df1 = df1.drop(['UDI', 'Product ID', 'TWF','HDF','PWF','OSF','RNF'], axis=1)
```

```
[76]: df2 = df2.drop(['UDI', 'Product ID', 'TWF','HDF','PWF','OSF','RNF'], axis=1)
```

```
[77]: df1.describe()
```

```
[77]:        Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
      count             10000.00                 10000.00                10000.00
      mean                300.00                   310.01                 1538.78
      std                   2.00                     1.48                  179.28
      min                 295.30                   305.70                 1168.00
      25%                 298.30                   308.80                 1423.00
      50%                 300.10                   310.10                 1503.00
      75%                 301.50                   311.10                 1612.00
```

```
max                   304.50                      313.80                     2886.00

       Torque [Nm]  Tool wear [min]  Machine failure
count     10000.00         10000.00         10000.00
mean         39.99           107.95             0.03
std           9.97            63.65             0.18
min           3.80             0.00             0.00
25%          33.20            53.00             0.00
50%          40.10           108.00             0.00
75%          46.80           162.00             0.00
max          76.60           253.00             1.00
```

[78]:
```
df2.describe()
```

[78]:
```
       Air temperature [K]  Process temperature [K]  Rotational speed [rpm]  \
count              5000.00                  5000.00                 5000.00
mean                300.14                   309.89                 1569.97
std                   1.97                     1.53                  229.71
min                 295.30                   305.70                 1217.00
25%                 298.50                   308.80                 1416.00
50%                 300.30                   309.80                 1505.00
75%                 301.60                   311.10                 1668.00
max                 304.50                   313.70                 2886.00

       Torque [Nm]  Tool wear [min]  Machine failure
count      5000.00          5000.00          5000.00
mean         40.51           101.90             0.02
std          12.45            59.03             0.14
min           6.40             0.00             0.00
25%          31.40            55.00             0.00
50%          41.10            85.00             0.00
75%          49.00           152.00             0.00
max          76.60           243.00             1.00
```

[56]:
```
pr(df1, title="Original Datset Report")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[56]:

[57]:
```
pr(df2, title="Original Datset Report")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
```

```
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[57]:

[79]:
```
le = LabelEncoder()
df1["Type"] = le.fit_transform(df1["Type"])
```

[80]:
```
y = df1["Machine failure"]
x = df1.drop(["Machine failure"],axis=1)
```

[81]:
```
x_train,x_test,y_train,y_test =␣
 ↪train_test_split(x,y,random_state=42,test_size=0.2)
```

[82]:
```
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(x_train, x_test, y_train, y_test)
```

```
100%|     | 29/29 [00:11<00:00,  2.50it/s]

[LightGBM] [Info] Number of positive: 278, number of negative: 7722
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000269 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 928
[LightGBM] [Info] Number of data points in the train set: 8000, number of used
features: 6
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.034750 -> initscore=-3.324208
[LightGBM] [Info] Start training from score -3.324208
```

[83]:
```
models
```

[83]:
```
                               Accuracy  Balanced Accuracy  ROC AUC  F1 Score  \
Model
DecisionTreeClassifier             0.98               0.84     0.84      0.98
XGBClassifier                      0.98               0.83     0.83      0.98
LGBMClassifier                     0.99               0.83     0.83      0.99
BaggingClassifier                  0.99               0.83     0.83      0.99
RandomForestClassifier             0.98               0.79     0.79      0.98
NearestCentroid                    0.72               0.77     0.77      0.81
AdaBoostClassifier                 0.98               0.74     0.74      0.98
LabelSpreading                     0.98               0.73     0.73      0.97
LabelPropagation                   0.97               0.73     0.73      0.97
ExtraTreeClassifier                0.97               0.73     0.73      0.97
```

```
PassiveAggressiveClassifier         0.96                 0.72     0.72      0.96
ExtraTreesClassifier                0.98                 0.71     0.71      0.98
Perceptron                          0.91                 0.70     0.70      0.93
LinearDiscriminantAnalysis          0.97                 0.69     0.69      0.97
KNeighborsClassifier                0.98                 0.69     0.69      0.98
QuadraticDiscriminantAnalysis       0.96                 0.66     0.66      0.96
SVC                                 0.98                 0.65     0.65      0.97
LogisticRegression                  0.97                 0.63     0.63      0.97
CalibratedClassifierCV              0.97                 0.62     0.62      0.97
GaussianNB                          0.96                 0.60     0.60      0.96
LinearSVC                           0.97                 0.56     0.56      0.96
SGDClassifier                       0.97                 0.52     0.52      0.96
DummyClassifier                     0.97                 0.50     0.50      0.95
RidgeClassifier                     0.97                 0.50     0.50      0.95
RidgeClassifierCV                   0.97                 0.50     0.50      0.95
BernoulliNB                         0.97                 0.50     0.50      0.95

                               Time Taken
Model
DecisionTreeClassifier               0.06
XGBClassifier                        0.52
LGBMClassifier                       0.16
BaggingClassifier                    0.22
RandomForestClassifier               0.94
NearestCentroid                      0.02
AdaBoostClassifier                   0.34
LabelSpreading                       4.85
LabelPropagation                     2.15
ExtraTreeClassifier                  0.02
PassiveAggressiveClassifier          0.03
ExtraTreesClassifier                 0.43
Perceptron                           0.03
LinearDiscriminantAnalysis           0.07
KNeighborsClassifier                 0.17
QuadraticDiscriminantAnalysis        0.03
SVC                                  0.32
LogisticRegression                   0.04
CalibratedClassifierCV               0.73
GaussianNB                           0.02
LinearSVC                            0.20
SGDClassifier                        0.05
DummyClassifier                      0.02
RidgeClassifier                      0.02
RidgeClassifierCV                    0.03
BernoulliNB                          0.02
```

[84]:
```
df_sum = models.drop(["Time Taken"], axis=1).sum(axis=1)
```

[85]:
```
df = models.iloc[:,:-1].sum(axis=1)
df_sum1 = df/4
```

[86]:
```
df_sum1.sort_values(ascending=False)
```

[86]:
```
Model
XGBClassifier                   0.91
DecisionTreeClassifier          0.91
LGBMClassifier                  0.91
BaggingClassifier               0.91
RandomForestClassifier          0.89
AdaBoostClassifier              0.86
LabelSpreading                  0.85
LabelPropagation                0.85
ExtraTreeClassifier             0.85
ExtraTreesClassifier            0.85
PassiveAggressiveClassifier     0.84
KNeighborsClassifier            0.83
LinearDiscriminantAnalysis      0.83
Perceptron                      0.81
SVC                             0.81
QuadraticDiscriminantAnalysis   0.81
LogisticRegression              0.80
CalibratedClassifierCV          0.80
GaussianNB                      0.78
NearestCentroid                 0.77
LinearSVC                       0.76
SGDClassifier                   0.74
DummyClassifier                 0.73
RidgeClassifier                 0.73
RidgeClassifierCV               0.73
BernoulliNB                     0.73
dtype: float64
```

[88]:
```
pipeline_optimizer = TPOTClassifier(generations=5, population_size=20,␣
 ↪cv=5,random_state=42, verbosity=2)
pipeline_optimizer.fit(x_train, y_train)
print(pipeline_optimizer.score(x_test, y_test))
pipeline_optimizer.export('tpot_exported_pipeline.py')
```

```
Optimization Progress:   0%|          | 0/120 [00:00<?, ?pipeline/s]


Generation 1 - Current best internal CV score: 0.9838749999999999

Generation 2 - Current best internal CV score: 0.9838749999999999

Generation 3 - Current best internal CV score: 0.983875
Generation 4 - Current best internal CV score: 0.9848750000000001

Generation 5 - Current best internal CV score: 0.9848750000000001

Best pipeline: ExtraTreesClassifier(input_matrix, bootstrap=False,
criterion=entropy, max_features=0.8500000000000001, min_samples_leaf=2,
min_samples_split=6, n_estimators=100)
0.9865
```

```
[89]: etc = ExtraTreesClassifier(bootstrap=False, criterion='entropy', max_features=0.
      ↪8500000000000001, min_samples_leaf=2, min_samples_split=6, n_estimators=100)
```

```
[90]: etc.fit(x_train,y_train)
```

```
[90]: ExtraTreesClassifier(criterion='entropy', max_features=0.8500000000000001,
                           min_samples_leaf=2, min_samples_split=6)
```

```
[91]: y_pred = etc.predict(x_test)
```

```
[92]: ll1 = cross_val_score(clf, x_test,y_test, cv=5)
      ll1
```

```
[92]: array([0.9725, 0.965 , 0.9775, 0.965 , 0.9725])
```

```
[98]: Model = 'Model.sav'
      pickle.dump(etc, open(Model, 'wb'))
```

Overview

Dataset statistics

Number of variables	7
Number of observations	10000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	547.0 KiB
Average record size in memory	56.0 B

Variable types

Categorical	2
Numeric	5

Alerts

`Air temperature [K]` is highly overall correlated with `Process temperature [K]`	High correlation
`Process temperature [K]` is highly overall correlated with `Air temperature [K]`	High correlation
`Rotational speed [rpm]` is highly overall correlated with `Torque [Nm]`	High correlation
`Torque [Nm]` is highly overall correlated with `Rotational speed [rpm]`	High correlation
`Machine failure` is highly imbalanced (78.6%)	Imbalance
`Tool wear [min]` has 120 (1.2%) zeros	Zeros

Reproduction

Analysis started	2023-09-27 09:16:11.490438
Analysis finished	2023-09-27 09:16:21.415705
Duration	9.93 seconds
Software version	pandas-profiling v3.6.6 (https://github.com/pandas-profiling/pandas-profiling)
Download configuration	config.json (data:text/plain;charset=utf-8,%7B%22title%22%3A%20%22Pandas%20Profiling%20Report%22%2C%20%22dataset%22%3A%20%7B%22description%22%3A

Variables

Select Columns

Type
Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	78.2 KiB

Length

Max length	1
Median length	1
Mean length	1

Characters and Unicode

Total characters	10000

Unique

Unique	0 ?
Unique (%)	0.0%

Sample

1st row	M
2nd row	L
3rd row	L

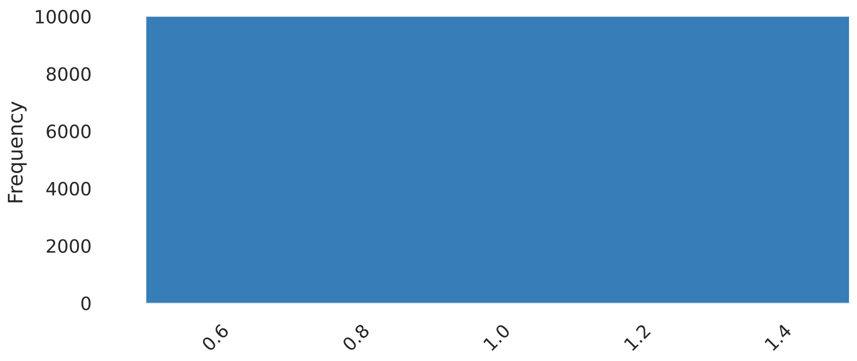
Min length	1	**Distinct characters**	3	**4th row**	L
				5th row	L
		Distinct categories	1 (https://en.wikipedia.org/wiki/Unicode_character_property#General_Category) ?		
		Distinct scripts	1 (https://en.wikipedia.org/wiki/Script_(Unicode)#List_of_scripts_in_Unicode) ?		
		Distinct blocks	1 (https://en.wikipedia.org/wiki/Unicode_block) ?		

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Common Values

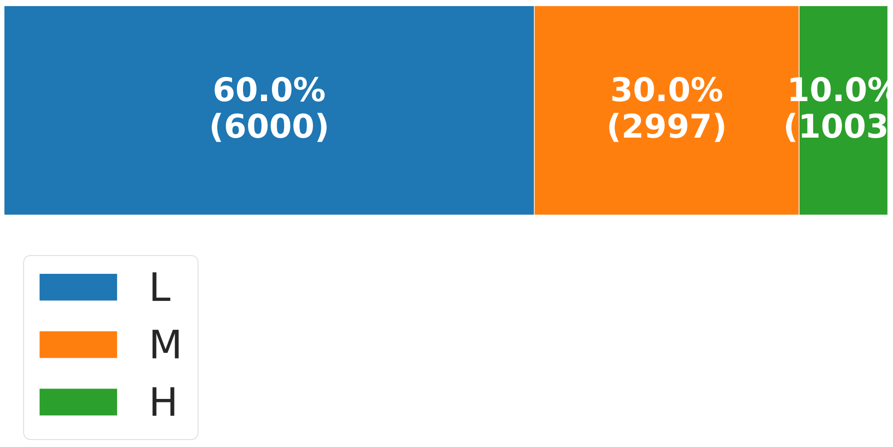
Value	Count	Frequency (%)
L	6000	60.0%
M	2997	30.0%
H	1003	10.0%

Length



Histogram of lengths of the category

Common Values (Plot)



Value	Count	Frequency (%)
l	6000	60.0%
m	2997	30.0%
h	1003	10.0%

Most occurring characters

Value	Count	Frequency (%)
L	6000	60.0%

Value	Count	Frequency (%)
M	2997	30.0%
H	1003	10.0%

Most occurring categories

Value	Count	Frequency (%)
Uppercase Letter	10000	100.0%

Most frequent character per category

Uppercase Letter

Value	Count	Frequency (%)
L	6000	60.0%
M	2997	30.0%
H	1003	10.0%

Most occurring scripts

Value	Count	Frequency (%)
Latin	10000	100.0%

Most frequent character per script

Latin

Value	Count	Frequency (%)
L	6000	60.0%
M	2997	30.0%
H	1003	10.0%

Most occurring blocks

Value	Count	Frequency (%)
ASCII	10000	100.0%

Most frequent character per block

ASCII

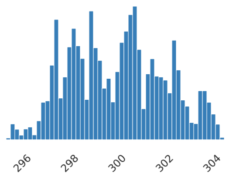
Value	Count	Frequency (%)
L	6000	60.0%
M	2997	30.0%
H	1003	10.0%

Air temperature [K]

Real number (ℝ)

Distinct	93
Distinct (%)	0.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	300.00493

Minimum	295.3
Maximum	304.5
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	78.2 KiB

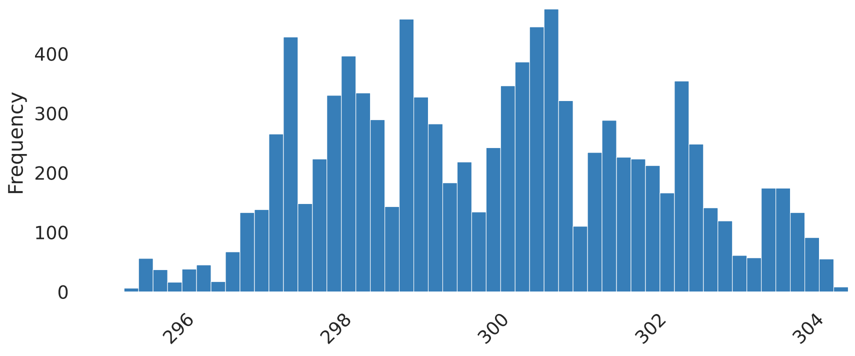


Quantile statistics

Minimum	295.3
5-th percentile	297.1
Q1	298.3
median	300.1
Q3	301.5
95-th percentile	303.5
Maximum	304.5
Range	9.2
Interquartile range (IQR)	3.2

Descriptive statistics

Standard deviation	2.0002587
Coefficient of variation (CV)	0.0066674194
Kurtosis	-0.83596167
Mean	300.00493
Median Absolute Deviation (MAD)	1.6
Skewness	0.11427392
Sum	3000049.3
Variance	4.0010348
Monotonicity	Not monotonic



Histogram with fixed size bins (bins=50)

Value	Count	Frequency (%)
300.7	279	2.8%
298.9	231	2.3%
297.4	230	2.3%
300.5	229	2.3%
298.8	227	2.3%
300.6	216	2.2%
298.2	208	2.1%
302.3	203	2.0%
297.5	198	2.0%
300.4	198	2.0%
Other values (83)	7781	77.8%

Value	Count	Frequency (%)
295.3	3	< 0.1%
295.4	3	< 0.1%

Value	Count	Frequency (%)
295.5	18	0.2%
295.6	38	0.4%
295.7	18	0.2%
295.8	19	0.2%
295.9	10	0.1%
296	6	0.1%
296.1	12	0.1%
296.2	26	0.3%

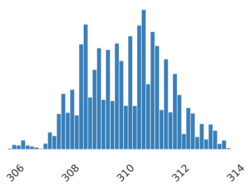
Value	Count	Frequency (%)
304.5	1	< 0.1%
304.4	7	0.1%
304.3	15	0.1%
304.2	40	0.4%
304.1	46	0.5%
304	45	0.4%
303.9	58	0.6%
303.8	75	0.8%
303.7	96	1.0%
303.6	78	0.8%

Process temperature [K]

Real number (ℝ)

Distinct	82
Distinct (%)	0.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	310.00556

Minimum	305.7
Maximum	313.8
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	78.2 KiB

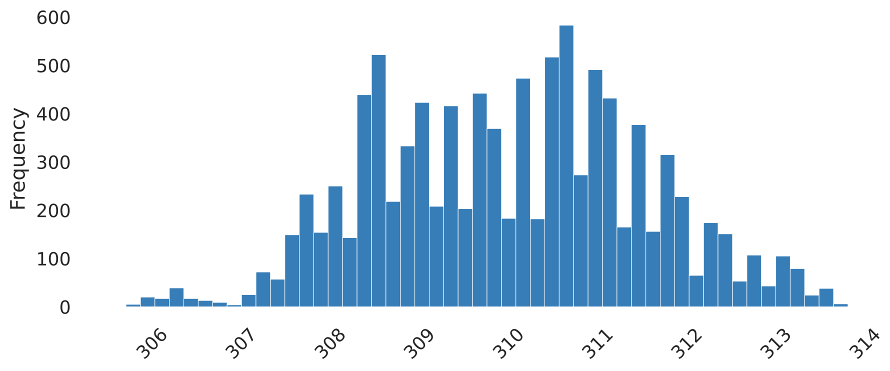


Quantile statistics

Minimum	305.7
5-th percentile	307.7
Q1	308.8
median	310.1
Q3	311.1
95-th percentile	312.5
Maximum	313.8
Range	8.1
Interquartile range (IQR)	2.3

Descriptive statistics

Standard deviation	1.4837342
Coefficient of variation (CV)	0.0047861536
Kurtosis	-0.49973437
Mean	310.00556
Median Absolute Deviation (MAD)	1.1
Skewness	0.015027268
Sum	3100055.6
Variance	2.2014672
Monotonicity	Not monotonic



Histogram with fixed size bins (bins=50)

Value	Count	Frequency (%)
310.6	317	3.2%
310.8	273	2.7%
310.7	266	2.7%
308.6	265	2.6%
310.5	263	2.6%
310.1	260	2.6%
308.5	257	2.6%
310.4	254	2.5%
311	246	2.5%
310.9	245	2.5%
Other values (72)	7354	73.5%

Value	Count	Frequency (%)
305.7	2	< 0.1%
305.8	3	< 0.1%

Value	Count	Frequency (%)
305.9	6	0.1%
306	14	0.1%
306.1	17	0.2%
306.2	26	0.3%
306.3	13	0.1%
306.4	8	0.1%
306.5	9	0.1%
306.6	13	0.1%

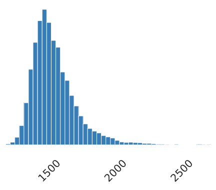
Value	Count	Frequency (%)
313.8	2	< 0.1%
313.7	4	< 0.1%
313.6	16	0.2%
313.5	22	0.2%
313.4	24	0.2%
313.3	29	0.3%
313.2	50	0.5%
313.1	50	0.5%
313	55	0.5%
312.9	43	0.4%

Rotational speed [rpm]

Real number (ℝ)

Distinct	941
Distinct (%)	9.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	1538.7761

Minimum	1168
Maximum	2886
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	78.2 KiB

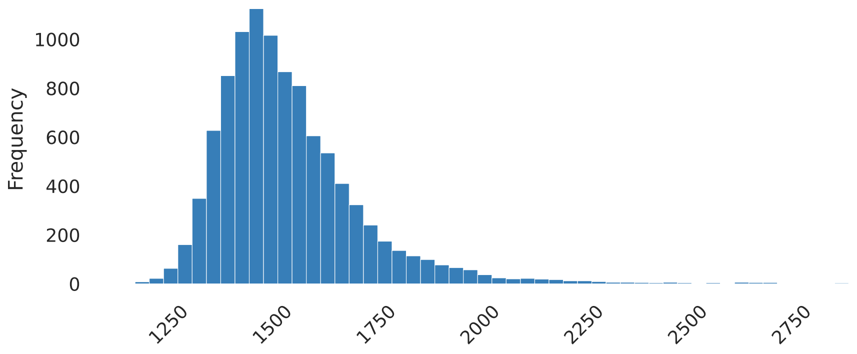


Quantile statistics

Minimum	1168
5-th percentile	1332
Q1	1423
median	1503
Q3	1612
95-th percentile	1868.05
Maximum	2886
Range	1718
Interquartile range (IQR)	189

Descriptive statistics

Standard deviation	179.2841
Coefficient of variation (CV)	0.11651084
Kurtosis	7.3929449
Mean	1538.7761
Median Absolute Deviation (MAD)	91
Skewness	1.993171
Sum	15387761
Variance	32142.787
Monotonicity	Not monotonic



Histogram with fixed size bins (bins=50)

Value	Count	Frequency (%)
1452	48	0.5%
1435	43	0.4%
1447	42	0.4%
1429	40	0.4%
1469	40	0.4%
1479	40	0.4%
1450	39	0.4%
1507	39	0.4%
1418	39	0.4%
1446	38	0.4%
Other values (931)	9592	95.9%

Value	Count	Frequency (%)
1168	1	< 0.1%
1181	1	< 0.1%

Value	Count	Frequency (%)
1183	1	< 0.1%
1192	1	< 0.1%
1200	1	< 0.1%
1202	3	< 0.1%
1207	1	< 0.1%
1208	1	< 0.1%
1212	2	< 0.1%
1217	1	< 0.1%

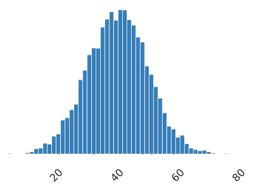
Value	Count	Frequency (%)
2886	1	< 0.1%
2874	1	< 0.1%
2861	1	< 0.1%
2833	1	< 0.1%
2825	1	< 0.1%
2760	1	< 0.1%
2737	1	< 0.1%
2721	1	< 0.1%
2710	1	< 0.1%
2709	1	< 0.1%

Torque [Nm]

Real number (ℝ)

Distinct	577
Distinct (%)	5.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	39.98691

Minimum	3.8
Maximum	76.6
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	78.2 KiB

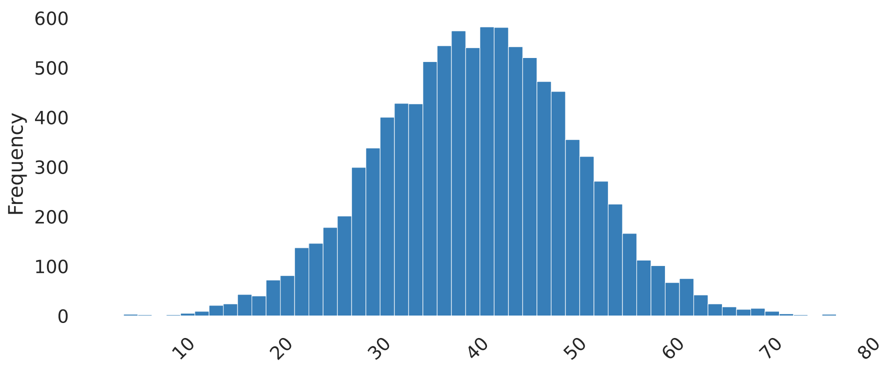


Quantile statistics

Minimum	3.8
5-th percentile	23.5
Q1	33.2
median	40.1
Q3	46.8
95-th percentile	56.1
Maximum	76.6
Range	72.8
Interquartile range (IQR)	13.6

Descriptive statistics

Standard deviation	9.9689337
Coefficient of variation (CV)	0.24930493
Kurtosis	-0.013240614
Mean	39.98691
Median Absolute Deviation (MAD)	6.8
Skewness	-0.0095165958
Sum	399869.1
Variance	99.37964
Monotonicity	Not monotonic



Histogram with fixed size bins (bins=50)

Value	Count	Frequency (%)
40.2	52	0.5%
38.5	50	0.5%
42.4	50	0.5%
35.8	50	0.5%
37.7	49	0.5%
39.9	48	0.5%
40.6	48	0.5%
38.2	48	0.5%
40	47	0.5%
36.6	47	0.5%
Other values (567)	9511	95.1%

Value	Count	Frequency (%)
3.8	1	< 0.1%

Value	Count	Frequency (%)
4.2	1	< 0.1%
4.6	1	< 0.1%
5.6	1	< 0.1%
5.8	1	< 0.1%
8	1	< 0.1%
8.8	1	< 0.1%
9.3	1	< 0.1%
9.7	2	< 0.1%
9.8	1	< 0.1%

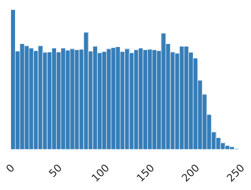
Value	Count	Frequency (%)
76.6	1	< 0.1%
76.2	1	< 0.1%
75.4	1	< 0.1%
74.5	1	< 0.1%
73.6	1	< 0.1%
72.8	1	< 0.1%
72	1	< 0.1%
71.8	1	< 0.1%
71.6	1	< 0.1%
71.3	1	< 0.1%

Tool wear [min]

Real number (ℝ)

Distinct	246
Distinct (%)	2.5%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	107.951

Minimum	0
Maximum	253
Zeros	120
Zeros (%)	1.2%
Negative	0
Negative (%)	0.0%
Memory size	78.2 KiB

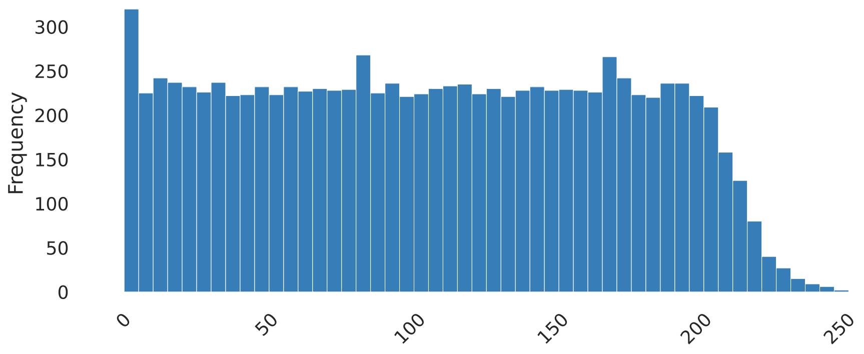


Quantile statistics

Minimum	0
5-th percentile	9.95
Q1	53
median	108
Q3	162
95-th percentile	206.05
Maximum	253
Range	253
Interquartile range (IQR)	109

Descriptive statistics

Standard deviation	63.654147
Coefficient of variation (CV)	0.58965778
Kurtosis	-1.1667371
Mean	107.951
Median Absolute Deviation (MAD)	55
Skewness	0.027292239
Sum	1079510
Variance	4051.8504
Monotonicity	Not monotonic



Histogram with fixed size bins (bins=50)

Value	Count	Frequency (%)
0	120	1.2%
2	69	0.7%
5	63	0.6%
7	58	0.6%
59	58	0.6%
166	57	0.6%
119	57	0.6%
9	55	0.5%
146	54	0.5%
96	54	0.5%
Other values (236)	9355	93.5%

Value	Count	Frequency (%)
0	120	1.2%
2	69	0.7%

Value	Count	Frequency (%)
3	34	0.3%
4	34	0.3%
5	63	0.6%
6	31	0.3%
7	58	0.6%
8	36	0.4%
9	55	0.5%
10	45	0.4%

Value	Count	Frequency (%)
253	1	< 0.1%
251	1	< 0.1%
246	3	< 0.1%
244	3	< 0.1%
242	2	< 0.1%
241	1	< 0.1%
240	3	< 0.1%
239	1	< 0.1%
238	2	< 0.1%
237	1	< 0.1%

Machine failure

Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	78.2 KiB

Length

Max length	1
Median length	1
Mean length	1
Min length	1

Characters and Unicode

Total characters	10000
Distinct characters	2
Distinct categories	1 (https://en.wikipedia.org/wiki/Unicode_character_property#General_Category) ?
Distinct scripts	1 (https://en.wikipedia.org/wiki/Script_(Unicode)#List_of_scripts_in_Unicode) ?
Distinct blocks	1 (https://en.wikipedia.org/wiki/Unicode_block) ?

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

Unique

Unique	0 ?
Unique (%)	0.0%

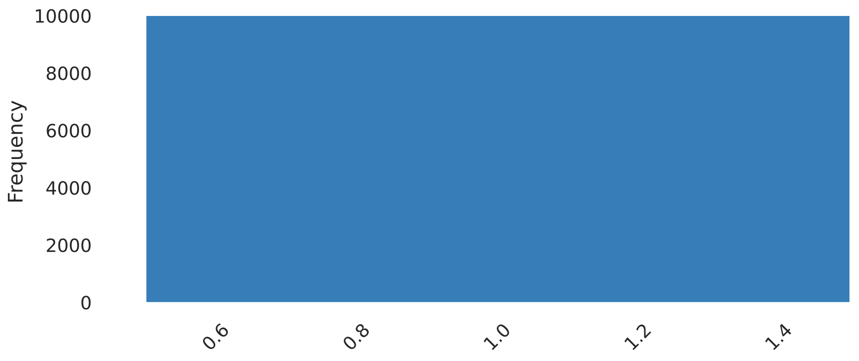
Sample

1st row	0
2nd row	0
3rd row	0
4th row	0
5th row	0

Common Values

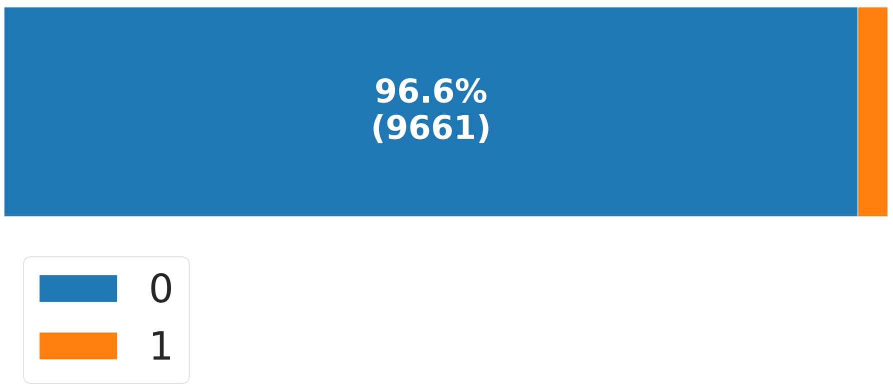
Value	Count	Frequency (%)
0	9661	96.6%
1	339	3.4%

Length



Histogram of lengths of the category

Common Values (Plot)



Value	Count	Frequency (%)
0	9661	96.6%
1	339	3.4%

Most occurring characters

Value	Count	Frequency (%)
0	9661	96.6%
1	339	3.4%

Most occurring categories

Value	Count	Frequency (%)
Decimal Number	10000	100.0%

Most frequent character per category

Decimal Number

Value	Count	Frequency (%)
0	9661	96.6%
1	339	3.4%

Most occurring scripts

Value	Count	Frequency (%)
Common	10000	100.0%

Most frequent character per script

Common

Value	Count	Frequency (%)
0	9661	96.6%
1	339	3.4%

Most occurring blocks

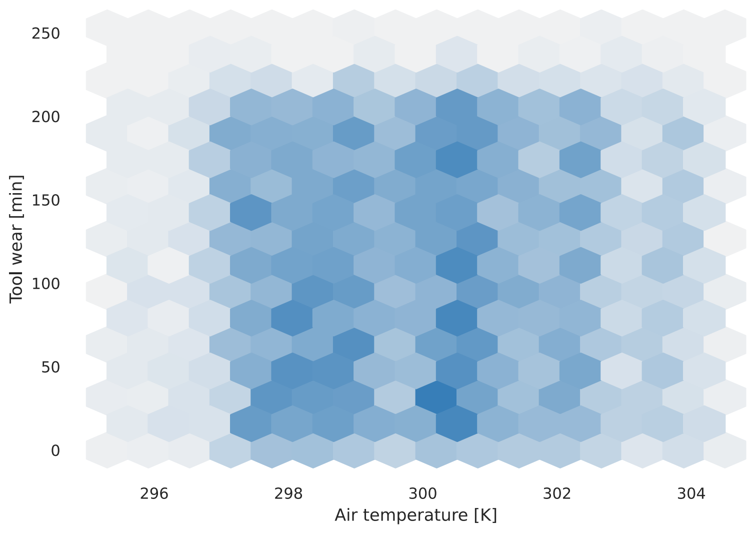
Value	Count	Frequency (%)
ASCII	10000	100.0%

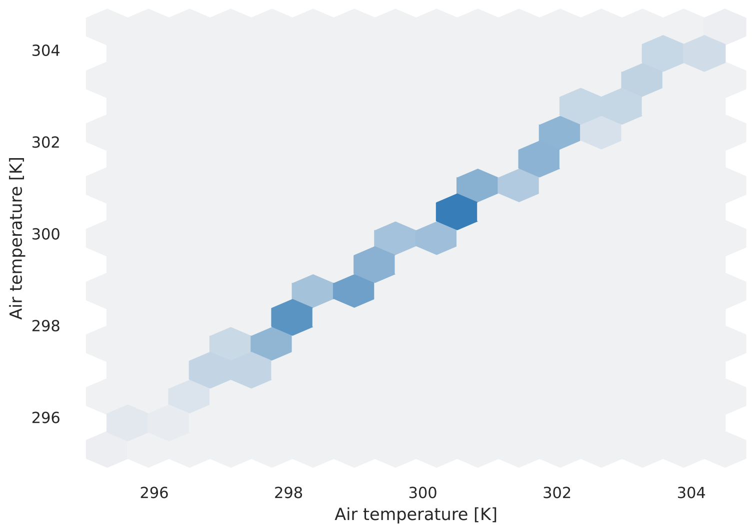
Most frequent character per block

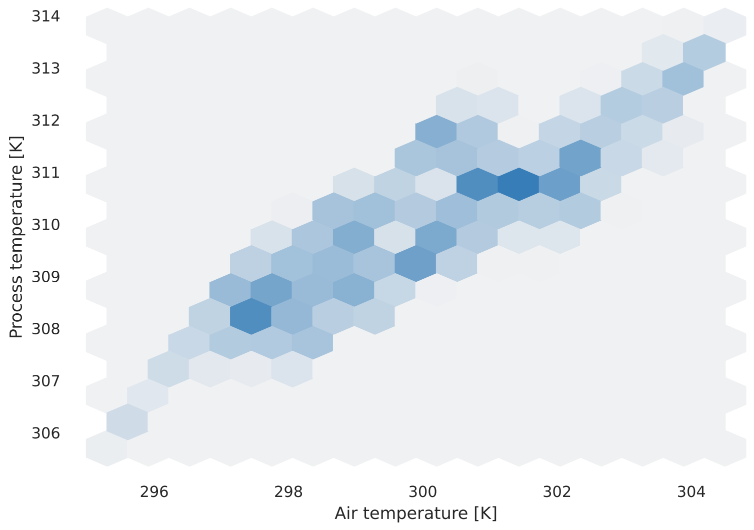
ASCII

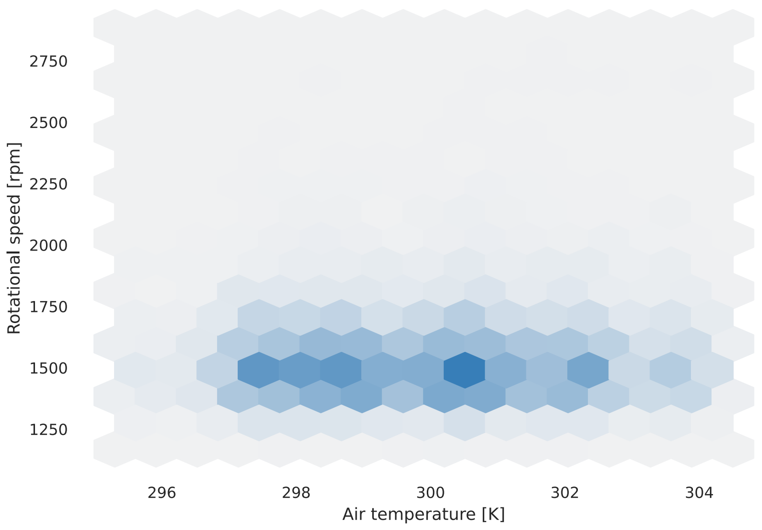
Value	Count	Frequency (%)
0	9661	96.6%
1	339	3.4%

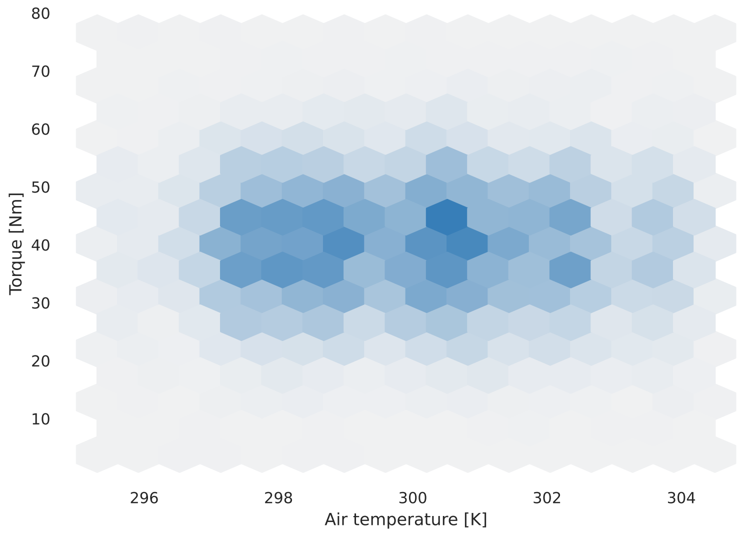
Interactions

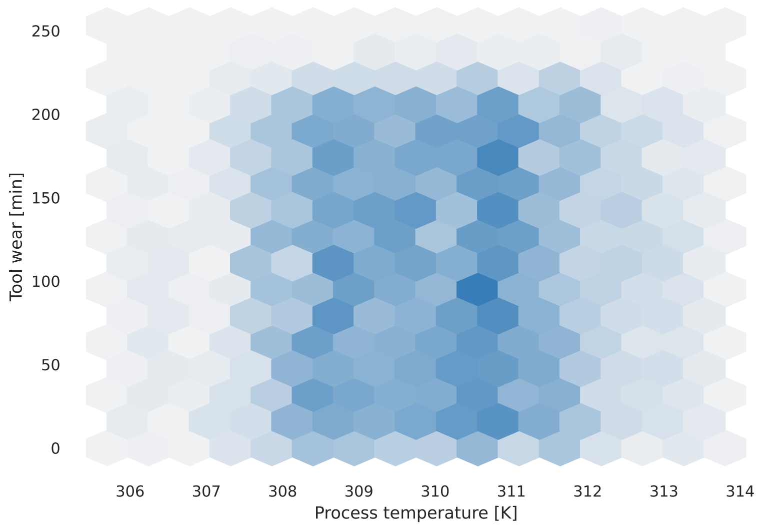


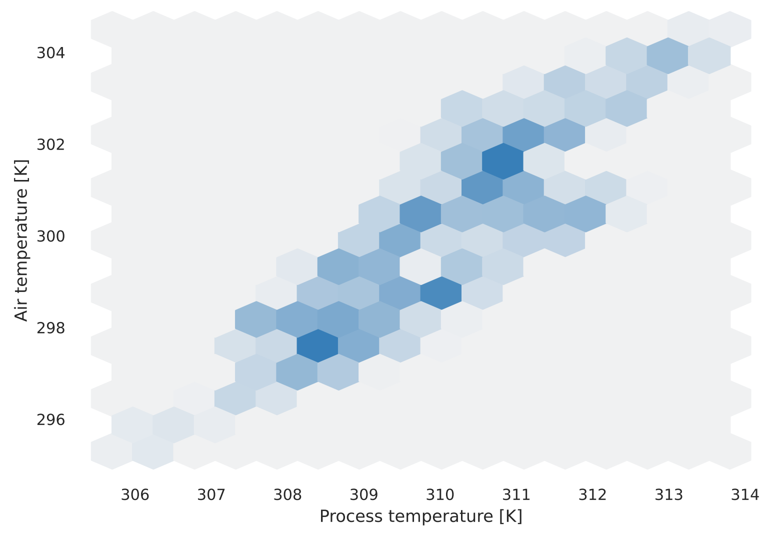




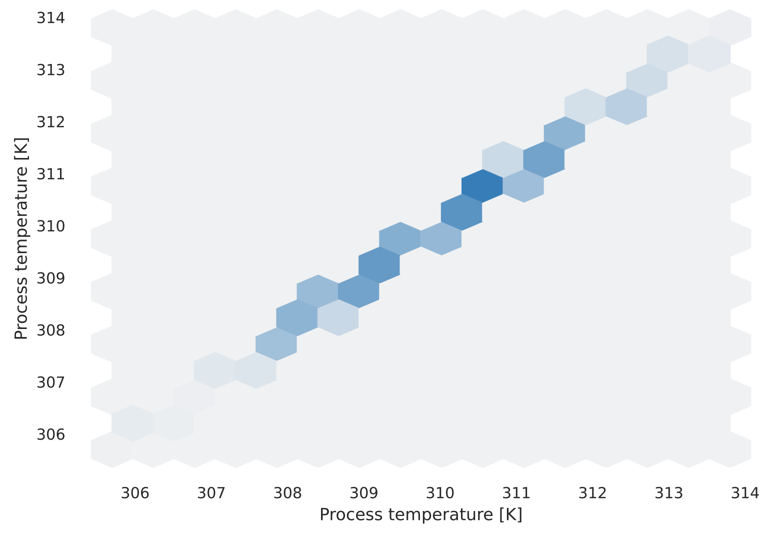

2750
2500
2250
2000
1750
1500
1250
Rotational speed [rpm]
296
298
300
302
304
Air temperature [K]


80
70
60
50
40
30
20
10
Torque [Nm]
296
298
300
302
304
Air temperature [K]

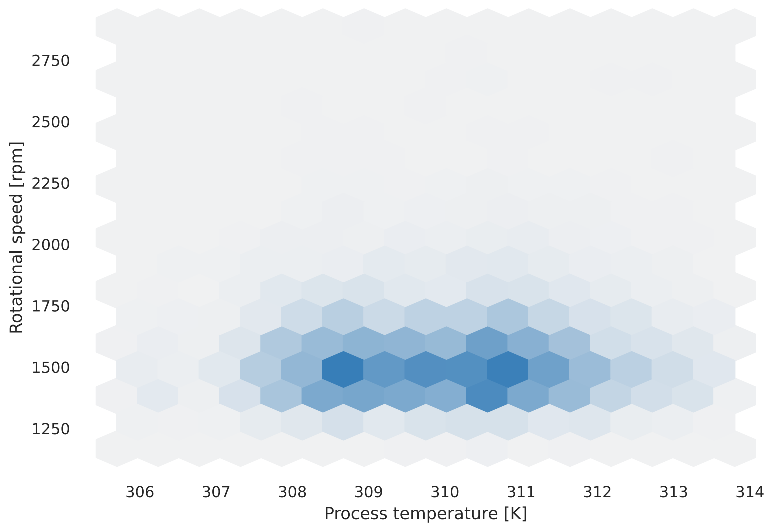

250
200
150
100
50
0
Tool wear [min]
306
307
308
309
310
311
312
313
314
Process temperature [K]



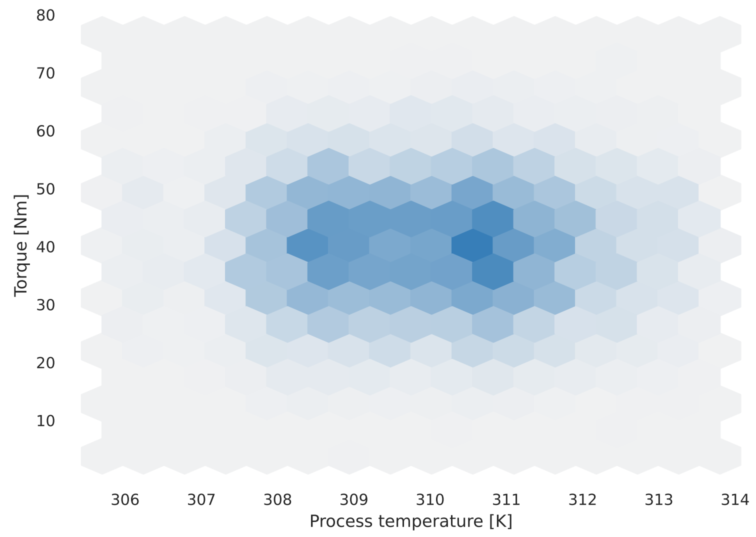
Air temperature [K]
304
302
300
298
296
306
307
308
309
310
311
312
313
314
Process temperature [K]



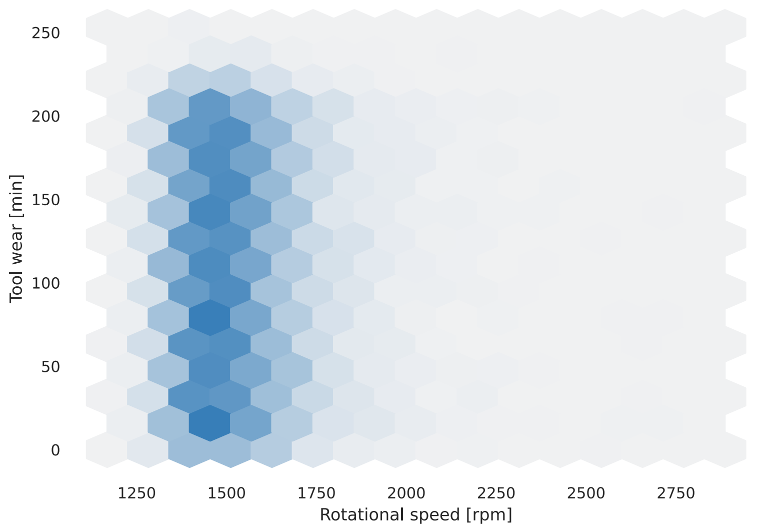
Process temperature [K]
314
313
312
311
310
309
308
307
306
306
307
308
309
310
311
312
313
314
Process temperature [K]



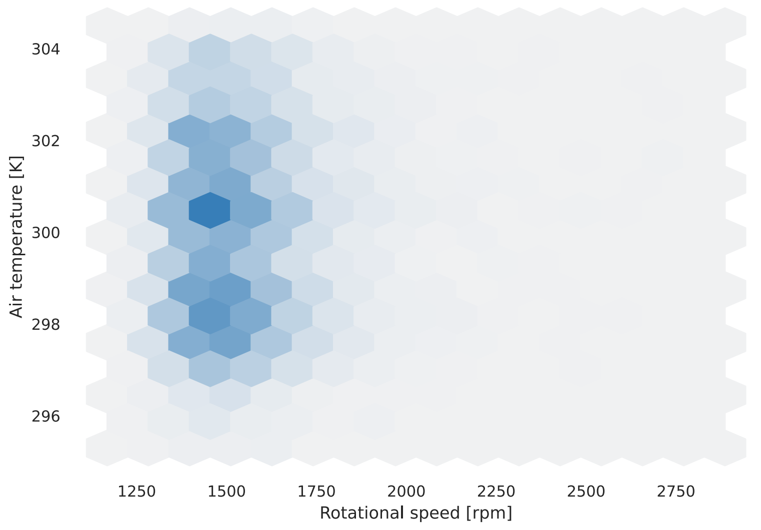
Rotational speed [rpm]
2750
2500
2250
2000
1750
1500
1250
306
307
308
309
310
311
312
313
314
Process temperature [K]



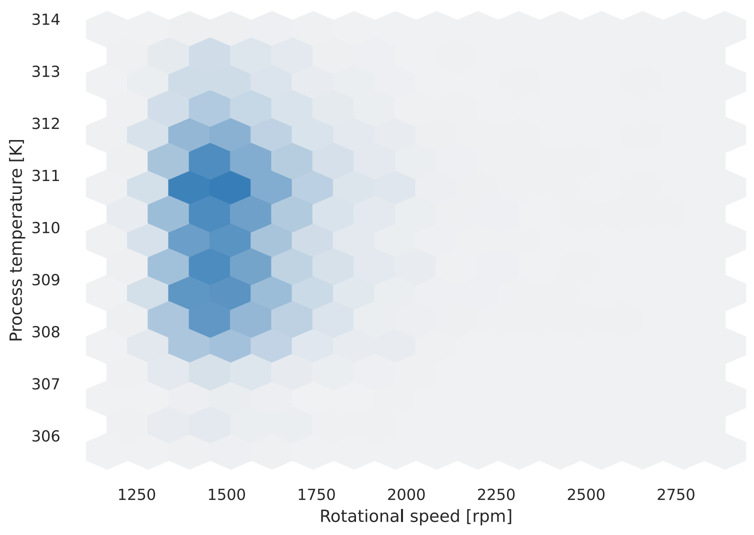
80
70
60
50
40
30
20
10
Torque [Nm]
306
307
308
309
310
311
312
313
314
Process temperature [K]



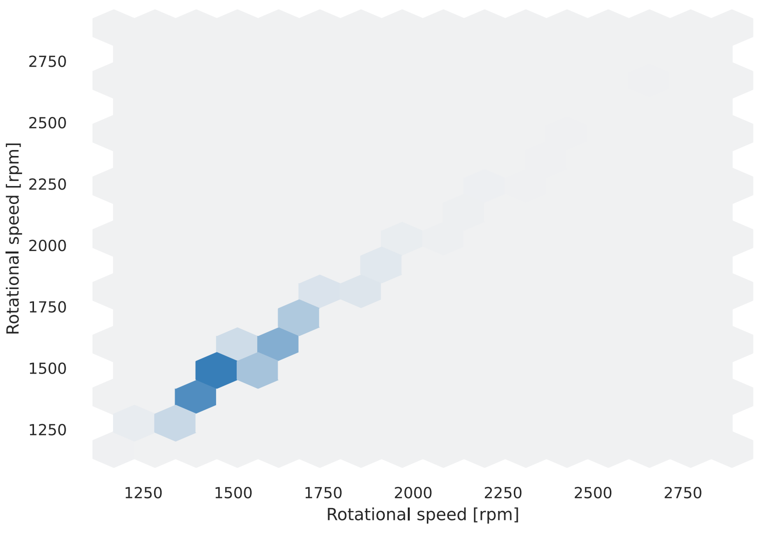
250
200
150
100
50
0
Tool wear [min]
1250
1500
1750
2000
2250
2500
2750
Rotational speed [rpm]



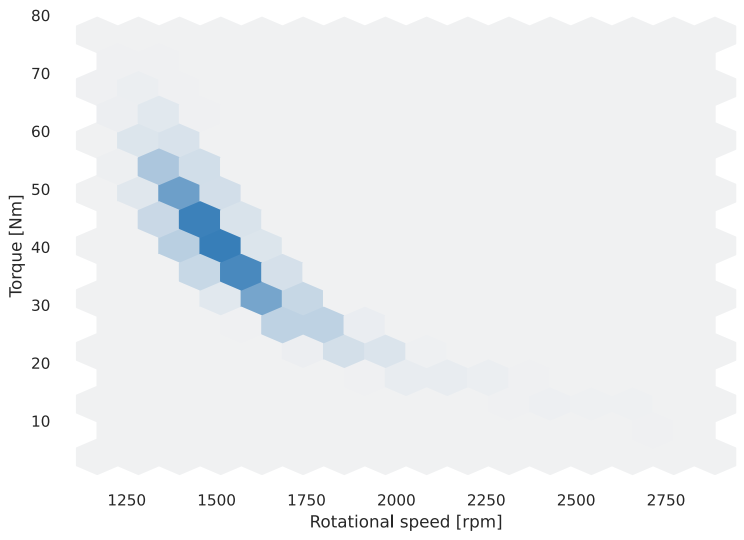
304
302
300
298
296
Air temperature [K]
1250
1500
1750
2000
2250
2500
2750
Rotational speed [rpm]



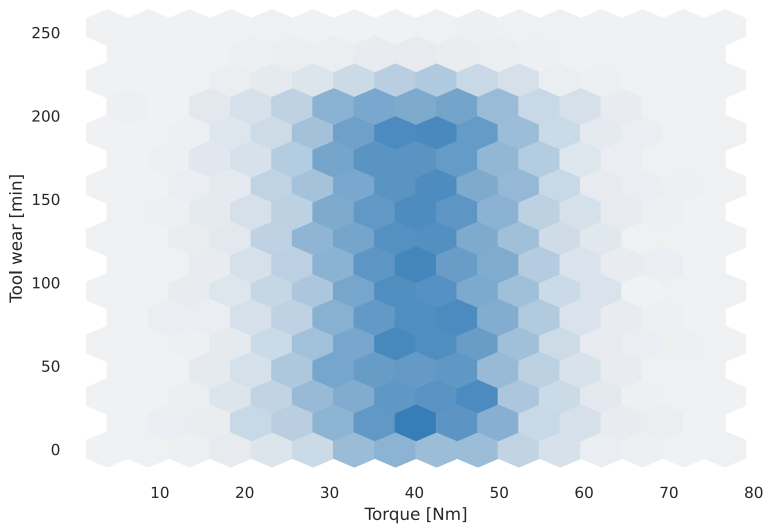
Process temperature [K]
314
313
312
311
310
309
308
307
306
1250
1500
1750
2000
2250
2500
2750
Rotational speed [rpm]



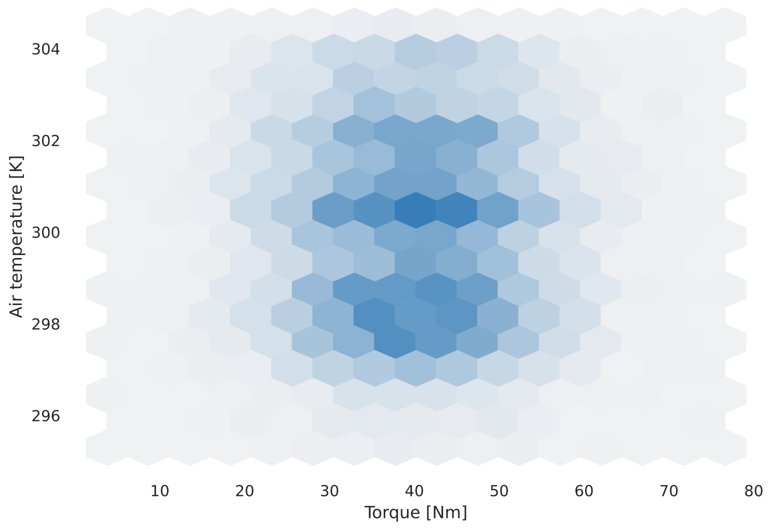
Rotational speed [rpm]
2750
2500
2250
2000
1750
1500
1250
1250
1500
1750
2000
2250
2500
2750
Rotational speed [rpm]



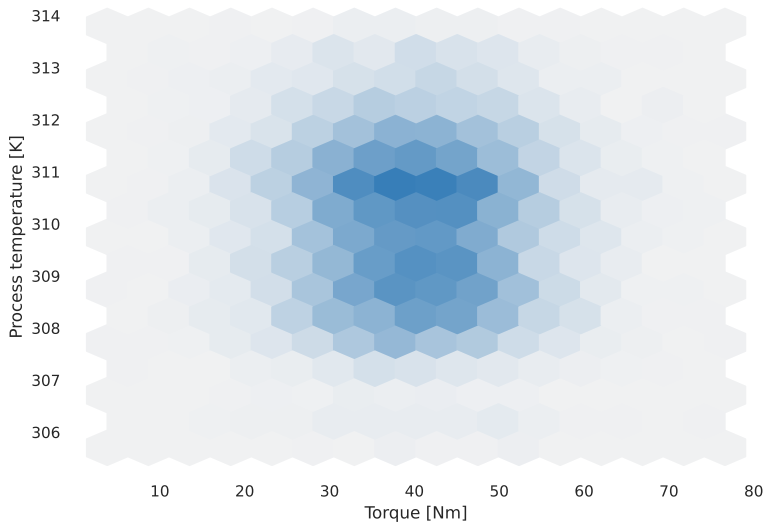
Torque [Nm]
80
70
60
50
40
30
20
10
1250
1500
1750
2000
2250
2500
2750
Rotational speed [rpm]



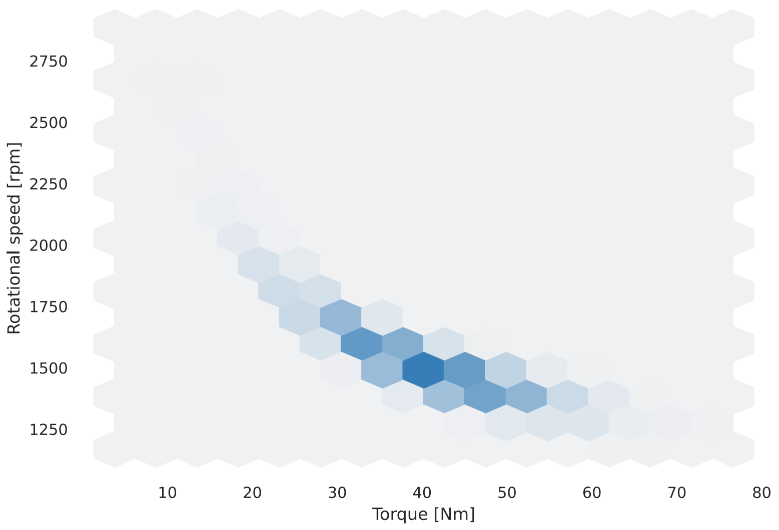
Tool wear [min]
0
50
100
150
200
250
Torque [Nm]
10
20
30
40
50
60
70
80



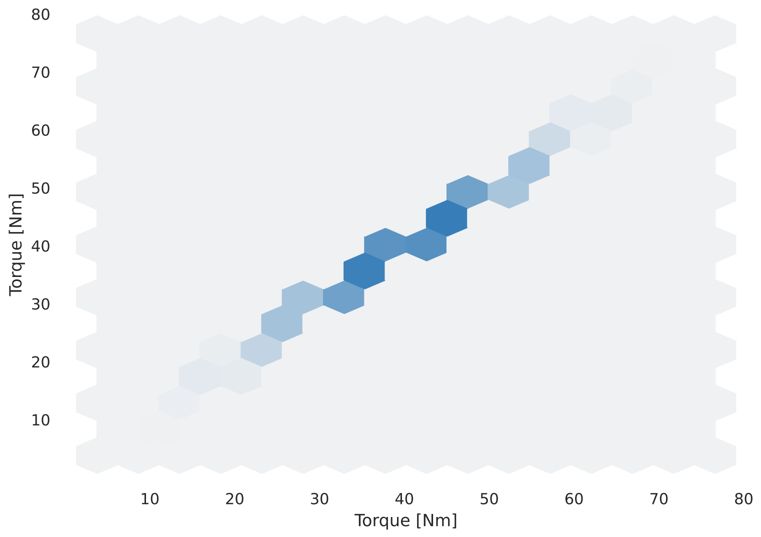
Air temperature [K]
296
298
300
302
304
Torque [Nm]
10
20
30
40
50
60
70
80



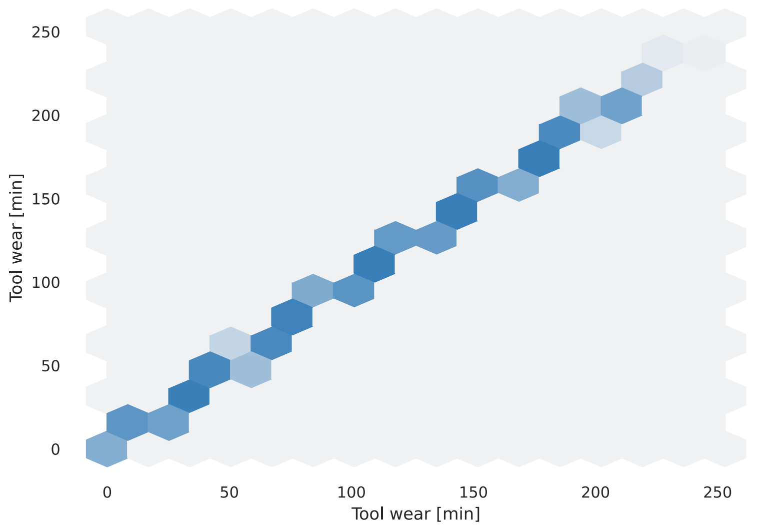
Process temperature [K]
306
307
308
309
310
311
312
313
314
Torque [Nm]
10
20
30
40
50
60
70
80



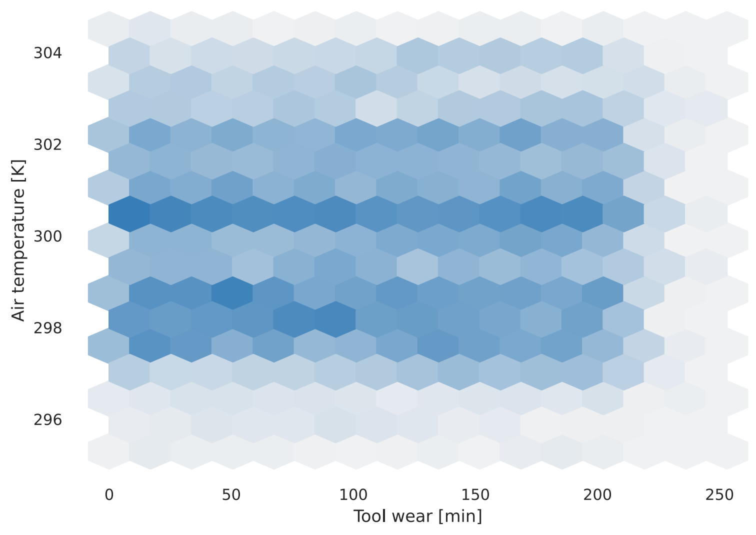
2750
2500
2250
2000
1750
1500
1250
Rotational speed [rpm]
10
20
30
40
50
60
70
80
Torque [Nm]



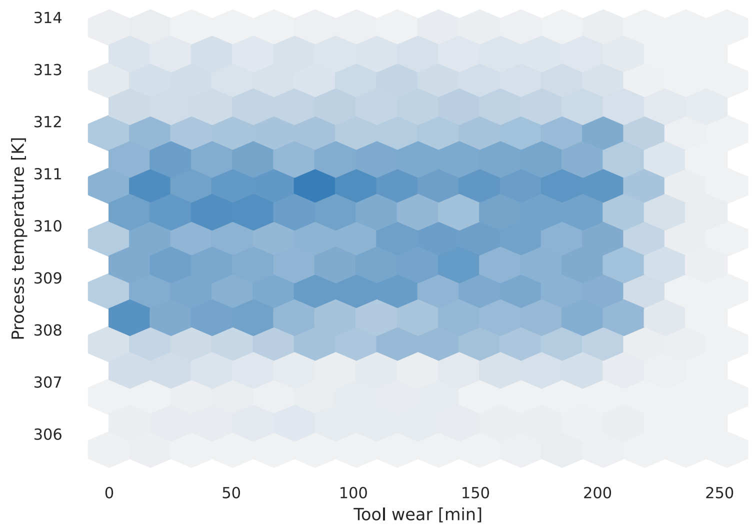
80
70
60
50
40
30
20
10
Torque [Nm]
10
20
30
40
50
60
70
80
Torque [Nm]



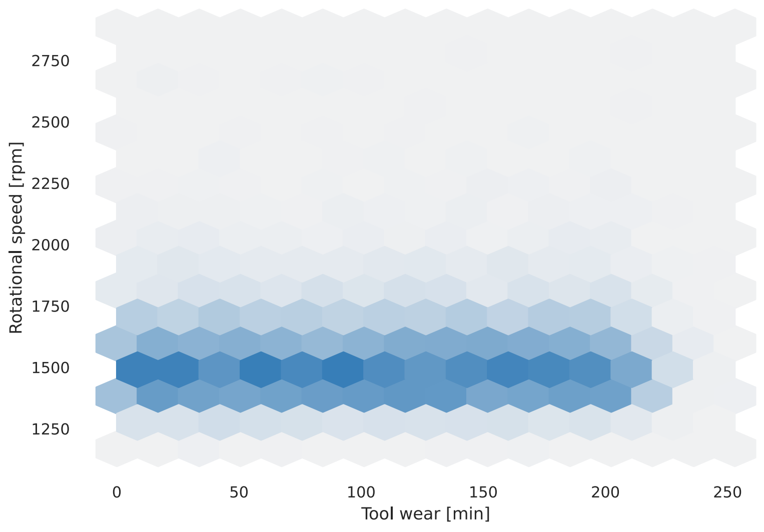
250
200
150
100
50
0
Tool wear [min]
0
50
100
150
200
250
Tool wear [min]



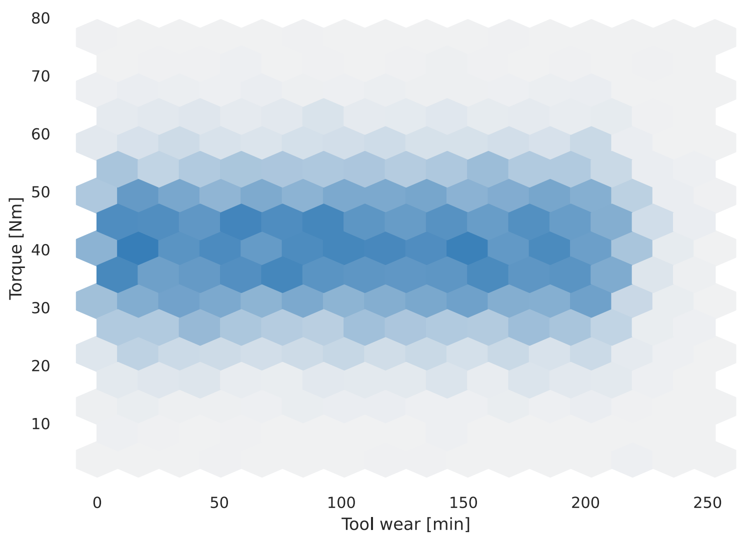
304
302
300
298
296
Air temperature [K]
0
50
100
150
200
250
Tool wear [min]



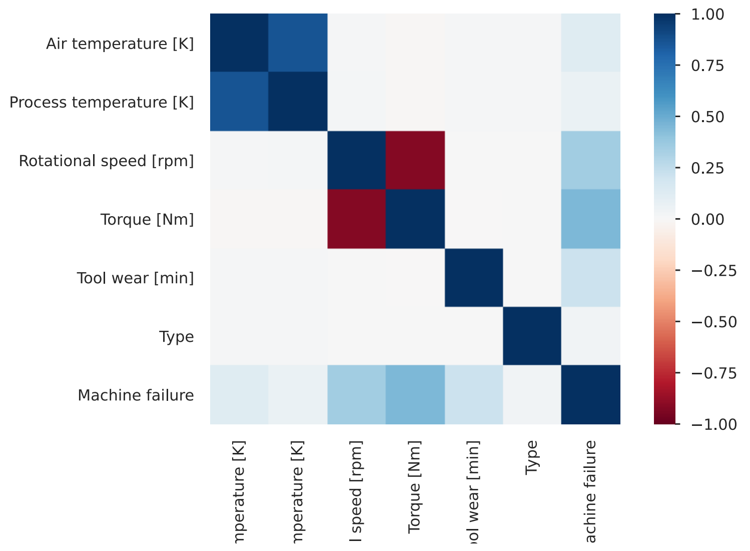
314
313
312
311
310
309
308
307
306
Process temperature [K]
0
50
100
150
200
250
Tool wear [min]



2750
2500
2250
2000
1750
1500
1250
Rotational speed [rpm]
0
50
100
150
200
250
Tool wear [min]

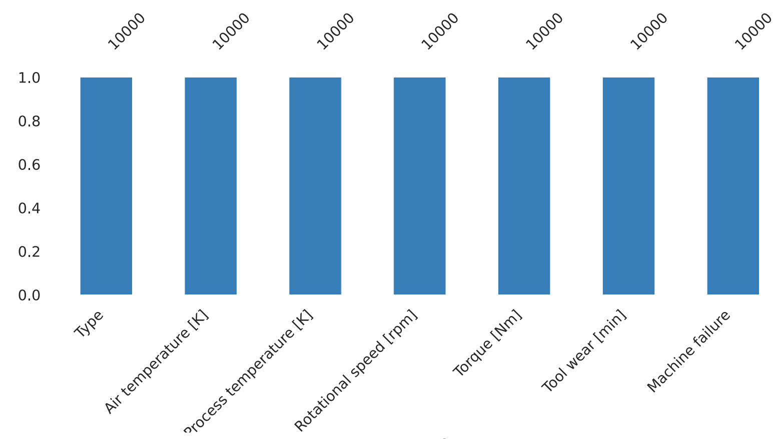


Correlations

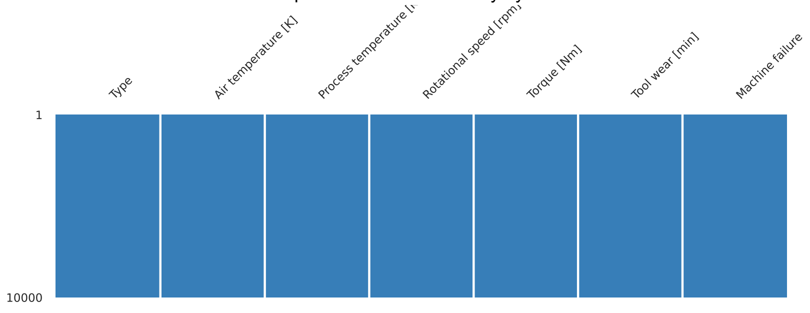


	Air temperature [K]	Process temperature [K]	Rotational speed [r
Air temperature [K]	1.000	0.864	0.014
Process temperature [K]	0.864	1.000	0.017
Rotational speed [rpm]	0.014	0.017	1.000
Torque [Nm]	-0.012	-0.014	-0.916
Tool wear [min]	0.013	0.014	0.003
Type	0.014	0.014	0.000
Machine failure	0.123	0.067	0.345

Missing values



A simple visualization of nullity by column.



Nullity matrix is a data-dense display which lets you quickly visually pick out patterns in data completion.

Sample

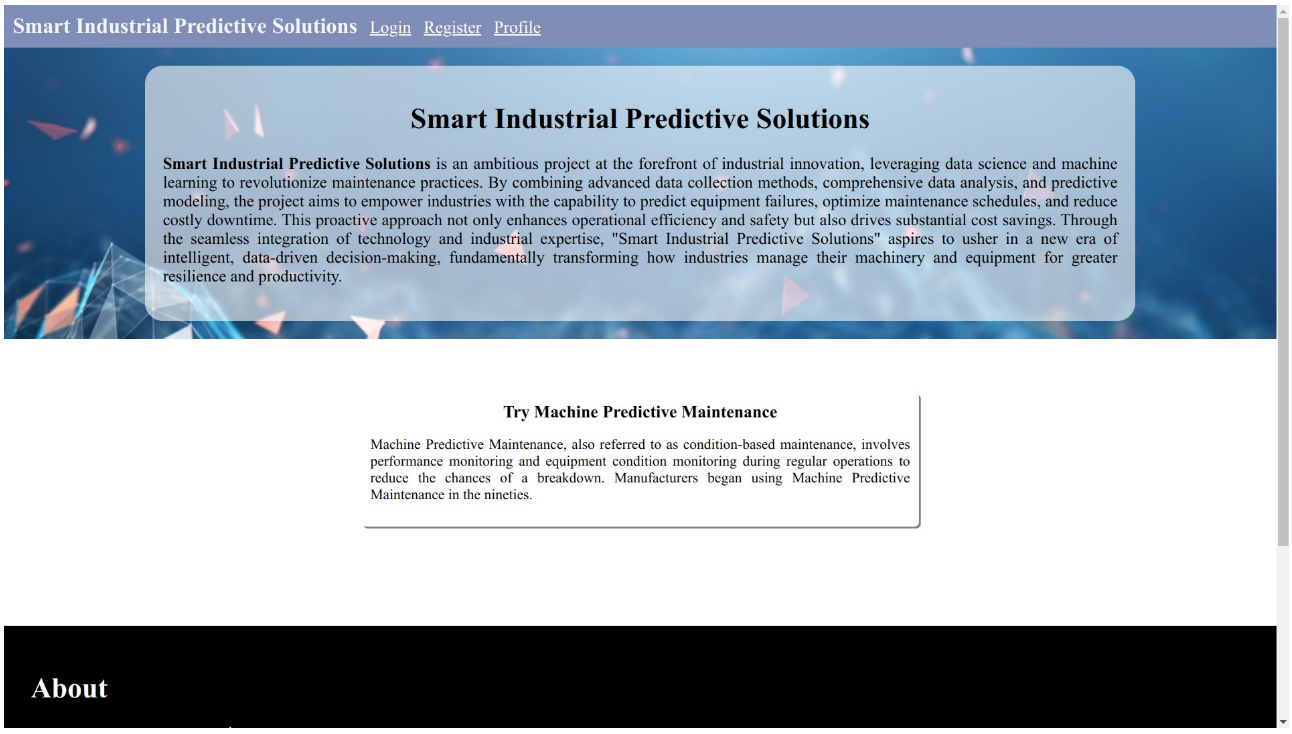
	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]
0	M	298.10	308.60	1551	42.80
1	L	298.20	308.70	1408	46.30
2	L	298.10	308.50	1498	49.40
3	L	298.20	308.60	1433	39.50
4	L	298.20	308.70	1408	40.00
5	M	298.10	308.60	1425	41.90
6	L	298.10	308.60	1558	42.40
7	L	298.10	308.60	1527	40.20
8	M	298.30	308.70	1667	28.60
9	M	298.50	309.00	1741	28.00

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nr
9990	L	298.80	308.50	1527	36.20
9991	M	298.90	308.40	1827	26.10
9992	L	298.80	308.40	1484	39.20
9993	L	298.80	308.40	1401	47.30
9994	L	298.80	308.30	1634	27.90
9995	M	298.80	308.40	1604	29.50
9996	H	298.90	308.40	1632	31.80
9997	M	299.00	308.60	1645	33.40
9998	H	299.00	308.70	1408	48.50

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nr
9999	M	299.00	308.70	1500	40.20

11. Web Application

11.1. Intro



Smart Industrial Predictive Solutions

Smart Industrial Predictive Solutions is an ambitious project at the forefront of industrial innovation, leveraging data science and machine learning to revolutionize maintenance practices. By combining advanced data collection methods, comprehensive data analysis, and predictive modeling, the project aims to empower industries with the capability to predict equipment failures, optimize maintenance schedules, and reduce costly downtime. This proactive approach not only enhances operational efficiency and safety but also drives substantial cost savings. Through the seamless integration of technology and industrial expertise, "Smart Industrial Predictive Solutions" aspires to usher in a new era of intelligent, data-driven decision-making, fundamentally transforming how industries manage their machinery and equipment for greater resilience and productivity.

Try Machine Predictive Maintenance

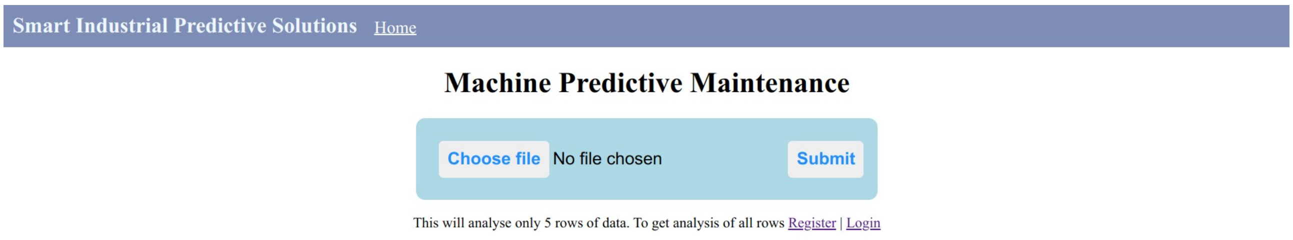
Machine Predictive Maintenance, also referred to as condition-based maintenance, involves performance monitoring and equipment condition monitoring during regular operations to reduce the chances of a breakdown. Manufacturers began using Machine Predictive Maintenance in the nineties.

About

Our web app is a subscription-based predictive maintenance application tailored for industrial settings. Leveraging cutting-edge machine learning algorithms, it offers real-time insights into machinery health, allowing proactive maintenance to prevent costly breakdowns. Key features include a user-friendly interface, limited trial version and full analysis reports. Subscribers gain access to comprehensive reports, empowering them to optimize maintenance practices, reduce downtime, and enhance operational efficiency.

The trial version offers users a valuable sneak peek into the world of predictive maintenance. It provides a limited yet insightful experience, allowing users to explore the app's capabilities and witness its potential benefits. During the trial, users can access a subset of predictive maintenance insights, enabling them to understand how the app can enhance machinery health monitoring and maintenance planning. This trial version is an excellent opportunity for users to gauge the app's value before committing to a subscription, ensuring informed decision-making and a seamless transition to the full-featured app for comprehensive predictive maintenance analysis.

11.2. Trial version

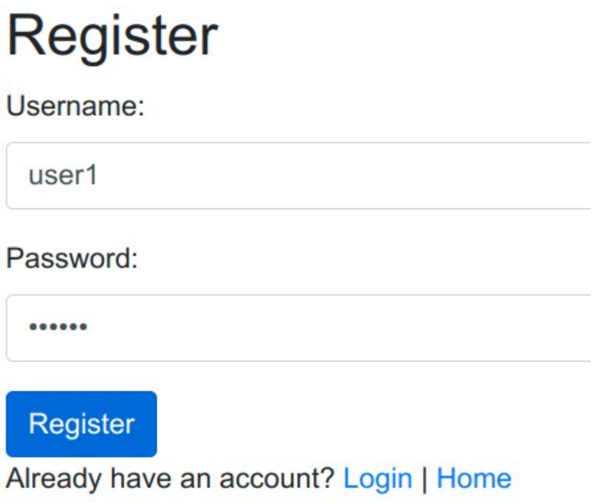

Smart Industrial Predictive Solutions Home

Machine Predictive Maintenance

This will analyse only 5 rows of data. To get analysis of all rows Register | Login

In the trial version, all you need to do is upload a CSV file with your machine's data and once you hit "submit," you'll instantly receive a failure report that provides valuable insights into your machinery's health and potential maintenance needs. It's a quick and easy way to experience how our app can help you prevent breakdowns and save on maintenance costs.

11.3. Registering

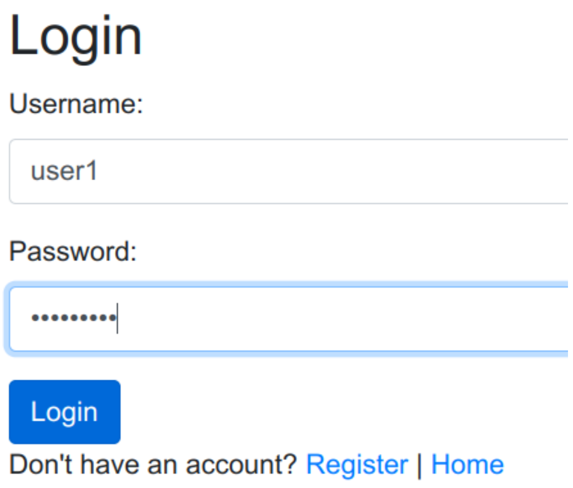

Register

Username: user1

Password: ••••••

Already have an account? Login | Home

By registering, you unlock the full potential of our app, granting you access to comprehensive analysis reports that delve deep into your machinery's health and maintenance needs.


Login

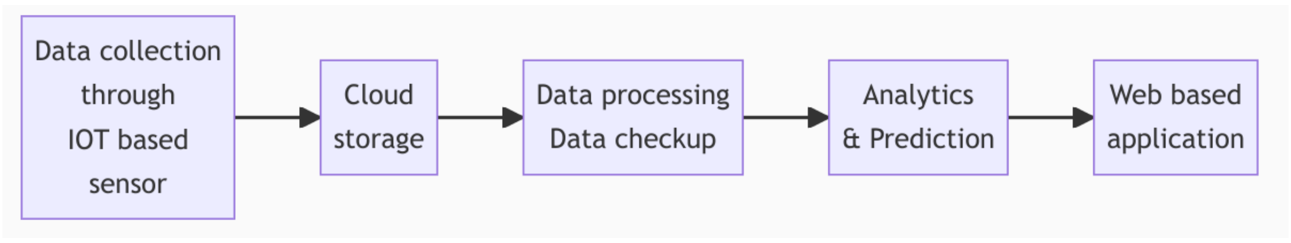
Username: user1

Password: •••••••••

Don't have an account? Register | Home

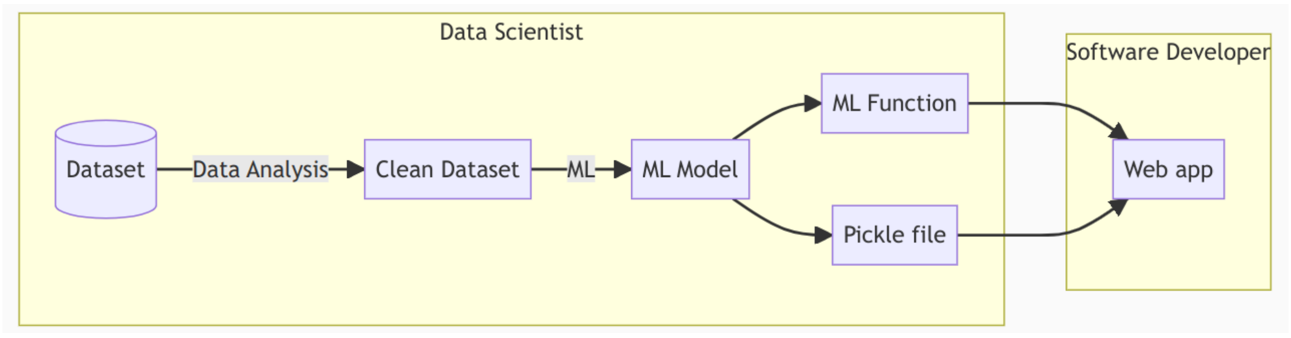
Once you log in, you'll gain immediate access to full analysis reports powered by our machine learning model. It's as straightforward as uploading a CSV file, and just like that, the detailed failure report will be available for instant download.

11.4. Business mode in web app


Data collection through IOT based sensor
Cloud storage
Data processing Data checkup
Analytics & Prediction
Web based application

Our business model simplifies predictive maintenance by following a streamlined process. It begins with IoT-based sensors that collect real-time data from your machinery, which is securely stored in the cloud. Before analysis, the data undergoes thorough processing and checks to ensure accuracy. Using advanced analytics and machine learning, we predict maintenance needs and potential issues. Finally, users can conveniently access these insights through our user-friendly web-based application. This end-to-end approach ensures efficient and effective predictive maintenance for industrial operations.

11.5. Data Analysis & Machine Learning Model in web app



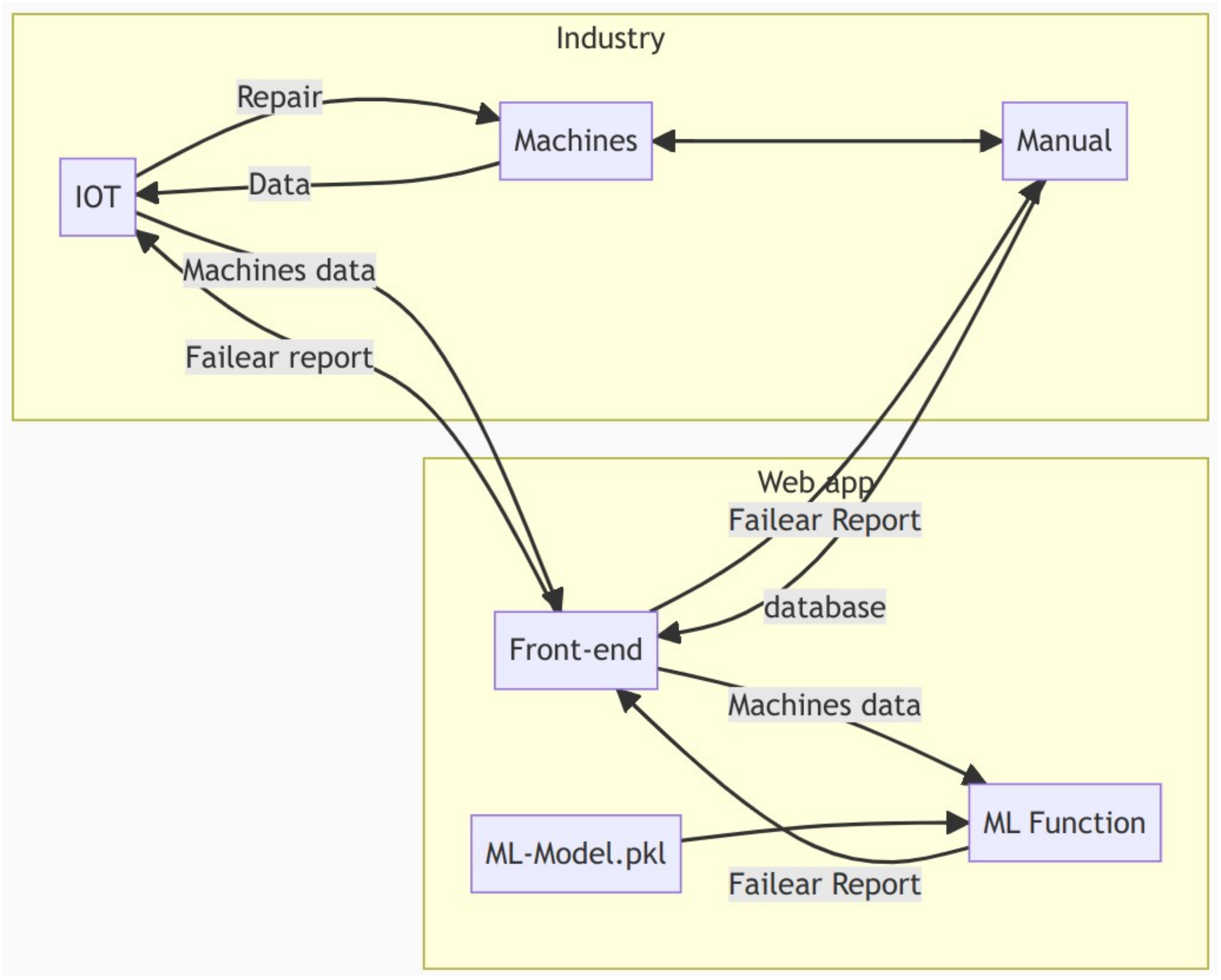
In our workflow, the machine learning (ML) model is at the core of our predictive maintenance system, serving as the brain behind the operation. Data scientists first analyze and clean the dataset to prepare it for training. They then employ the dataset to train the ML model, which includes creating a machine learning function. Once the model is trained, it's saved as a pickle file. The software developers take over from there, integrating the ML function into our web application. This collaborative effort between data scientists and software developers ensures that our web app delivers accurate and valuable predictive maintenance insights to our users.

`ML Function` gets csv file of machines' data and returns failear dataframe of all machines.

```
def ml_function(csv_path, model_path):
    # loading pre-trained model from model_path
    # pandas dataframe(df) from csv_path
    # df cleaning & preprocessing...
    ...
    ...
    # failear_df prediction from df using the ML model
    return failear_df
```

The "ML Function" is a pivotal component of our predictive maintenance system. It accepts a CSV file path containing machinery data and a pre-trained model path as inputs. Within this function, the pre-trained model is loaded, and the provided CSV data is processed and cleaned to ensure its suitability for analysis. Following data preparation, the function applies the ML model to make predictions related to machinery failure, resulting in a failure dataframe. This dataframe contains crucial insights about the potential failures of all the machines under analysis. The "ML Function" plays a critical role in our system's ability to proactively identify maintenance needs and enhance operational efficiency.

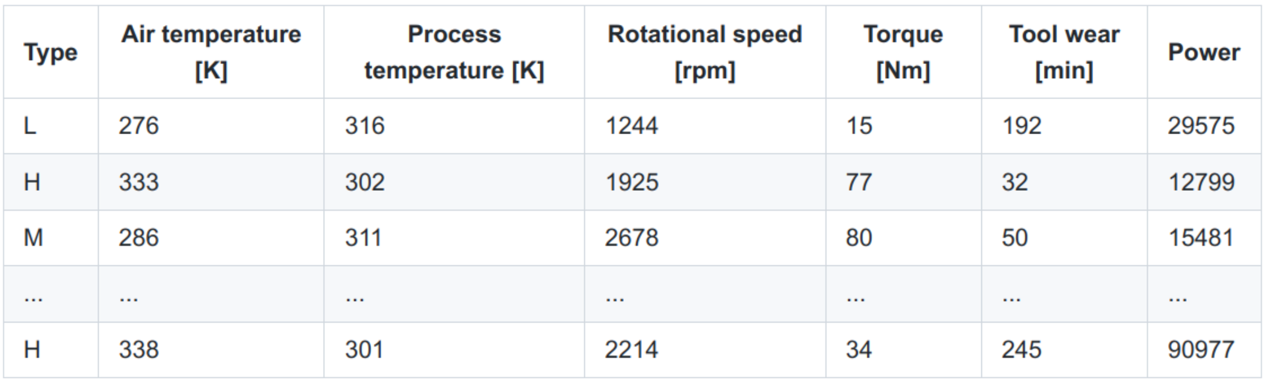
11.6. Web app architecture



In our system architecture, the web application serves as the central hub connecting various components for streamlined predictive maintenance. The journey begins with data collection, where machinery data is sourced from IoT devices (represented by "IOT") and manual input (represented by "Manual"). This data flows into the web application's front-end (represented by "Front-end"), where users can interact with the system. Within the application, the data is further processed and analyzed using the "ML Function" and a pre-trained ML model ("ML-Model.pkl"). The result of this analysis is the generation of comprehensive failure reports, which are then made available to users through the front-end. Users can access these reports, gaining insights into machinery health and maintenance needs. Additionally, the web application facilitates bidirectional communication with the industry, allowing data exchange for analysis and providing repair instructions when necessary.

The industry side of the architecture involves data provision to the IoT devices and the web application. Machinery data is sent to the web application for analysis, while repair instructions are communicated back to the industry for action. This two-way flow of data ensures a closed feedback loop, enabling proactive maintenance and optimizing industrial operations. Moreover, the web application maintains a database of machinery data, ensuring a historical record of maintenance-related information and facilitating further analysis. This comprehensive architecture streamlines predictive maintenance, minimizing downtime and enhancing machinery reliability in industrial settings.

11.7. Upload CSV format



Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Power
L	276	316	1244	15	192	29575
H	333	302	1925	77	32	12799
M	286	311	2678	80	50	15481
...	...	...	...	...	...	...
H	338	301	2214	34	245	90977

Our CSV upload format adheres to a structured table with specific columns essential for predictive maintenance analysis. These columns include "Type," "Air temperature [K]," "Process temperature [K]," "Rotational speed [rpm]," "Torque [Nm]," "Tool wear [min]," and "Power." While users have the option to include additional columns for machine names or identification numbers, the core columns mentioned above must be present in the uploaded CSV file. The resulting report CSV file exclusively contains data related to machines experiencing failures. To facilitate easy identification of the failing machines, an additional column labeled "Index" is incorporated into the report CSV, ensuring clarity and precision in pinpointing the machines in need of attention within the predictive maintenance process.

12. Future Development:

As part of our commitment to continuous improvement and staying at the forefront of technological advancements, we have identified key areas for future development that will further enhance the capabilities of "Smart Industrial Predictive Solutions." One pivotal area is the implementation of a robust CI/CD (Continuous Integration/Continuous Deployment) pipeline. This pipeline will streamline the development, testing, and deployment processes, ensuring seamless updates and enhancements to our application. By adopting best practices in CI/CD, we aim to deliver new features, improvements, and bug fixes efficiently and without disruption to our users.

This initiative aligns with industry trends and emerging technologies, such as cloud-based services like Google Cloud Platform (GCP) and Kubernetes. In the future, our CI/CD pipeline will seamlessly integrate with cloud services, facilitating automatic scaling, efficient resource management, and rapid deployment of updates. This not only enhances the reliability and performance of our application but also equips our team with valuable skills and experience in managing cloud-based solutions, including GCP's BigQuery and Kubernetes. By embracing this forward-looking approach, we ensure that "Smart Industrial Predictive Solutions" remains adaptable, agile, and ready to meet the evolving needs of our users and the industry as a whole.

13. Conclusion:

In conclusion, "Smart Industrial Predictive Solutions" represents a transformative leap forward in the realm of industrial maintenance. This innovative project combines cutting-edge technology, data-driven insights, and user-friendly accessibility to empower industries with proactive maintenance capabilities. By harnessing the power of machine learning and IoT data, this solution equips businesses with the foresight to prevent costly machinery breakdowns and optimize operational efficiency. The comprehensive web application, coupled with a seamlessly integrated data flow from sensors to analytics, offers a holistic approach to predictive maintenance. It is poised to revolutionize the way industries operate by minimizing downtime, reducing maintenance costs, and ensuring a competitive edge in an increasingly dynamic industrial landscape.

As industries continue to evolve, the need for data-driven decision-making and efficient maintenance practices becomes paramount. "Smart Industrial Predictive Solutions" rises to this challenge by providing a robust framework for informed decision-making, enhanced safety, and cost-effective maintenance. By offering trial versions, customizable options, and real-time insights, it ensures a user-centric experience that caters to businesses of all sizes. In essence, this project not only represents a technological milestone but also a strategic imperative for industries seeking to thrive in the 21st century. "Smart Industrial Predictive Solutions" is not just a tool; it's a game-changer that empowers industries to stay ahead of the curve, ensuring resilience and success in a rapidly evolving industrial landscape.