

PREDICTIVE MAINTENANCE IN MANUFACTURING INDUSTRY

Deepanshu

30/08/2024

Problem Statement

Predictive maintenance in manufacturing. This project is to find the early fault detection and diagnosis in the manufacturing industries and industry Manufacturers increasingly collect data from machines in their factories and products and with that observed data we can detect warning signs of expensive failures before they occur. Generally this technology is used in large scale companies but it is possible to use in medium and small scale companies too. The problem statement attempted in this study may be divided into the following specific objectives.

- Assess the health state of the equipment (State of Health – SoH): Prediction of whether equipment is in its' last 'n' periods of life.
- Predicting the remaining useful life (RUL) of equipment. Prediction of number of cycles remaining before the equipment completely breaks down.
- Outlier Analysis: Finding the anomalies in the data.
- Pareto Alerts: analysis identifies the “vital few” features that contribute the most to plant maintenance and distinguishes them from the “trivial many”.

Business Need Assessment

Predictive maintenance is a transformative application of the AI, Machine learning, IIoT with tremendous advantages. Below, we explore five benefits that can serve as differentiators for your organization:

Decreased downtime

Predictive maintenance enables technicians to detect issues in advance and resolve problems before equipment failure can occur, so you can:

- Cut unplanned downtime by as much as up to 30%Schedule multiple service procedures at one time
- Avoid the risk of reputation-damaging outages
- Reduce costly truck rolls required by unexpected downtime

Greater worker productivity

There is no need to disrupt worker productivity for an unexpected malfunction or breakdown. Predictive maintenance plans around workers' schedules, and:

- Enables up to 83% faster service time-to-resolution
- Maximizes uptime and prevents productivity lags
- Increases asset utilization

Reduced field services cost

By anticipating machine maintenance, service departments can generate major cost-savings and increased ROI through:

- Reduction of costly service truck rolls
- Increased first-time fix rates
- Streamlined maintenance costs through reduced labor, equipment, and inventory costs

Improved Product design

Harnessing the power of IIoT data collected through your machine's sensors, product designers can use this vital information to:

- Extend asset lifespans
- Improve equipment durability and reliability
- Build more efficient machines in the future

Improved worker safety

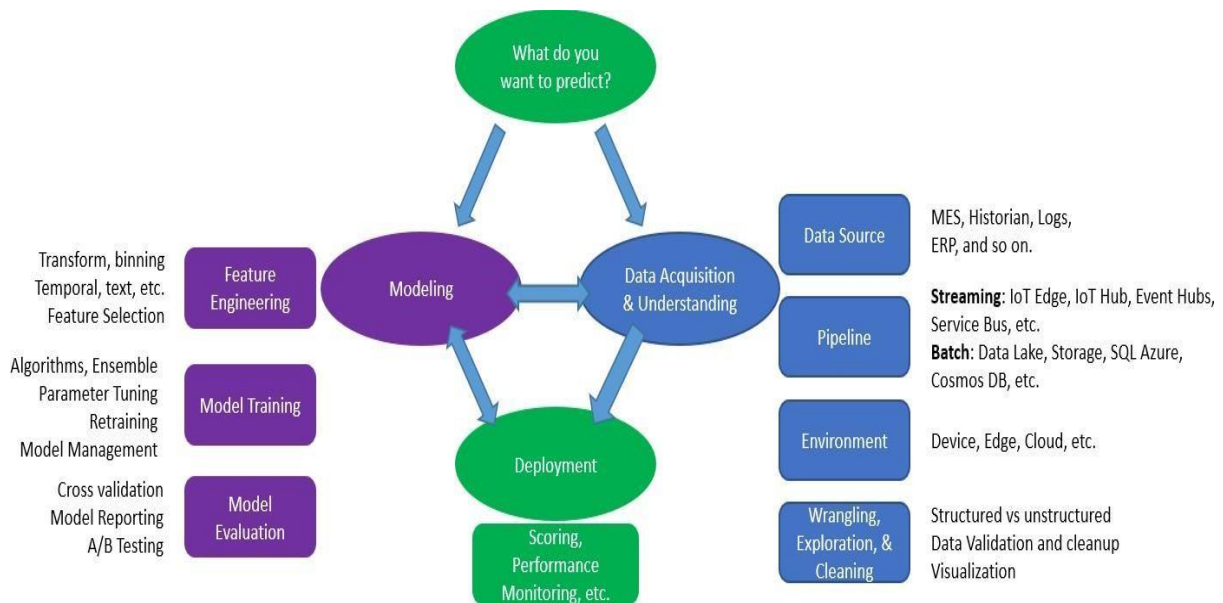
An unexpected breakdown or malfunction can lead to hazardous working conditions for your employees. By predicting when a malfunction may occur, you can ensure:

- Employees are nowhere near a machine when it breaks down
- Technicians can carry out service before a machine becomes dangerous

Target specification and characteristics

To build a PdM solution, we start with data. Ideally the data shows normal operation and the state of the equipment before, during, and after failures. The data comes from sensors, notes maintained by equipment operators, run information, environmental data, machine specifications, and so on. Systems of record can include historians, manufacturing execution systems, enterprise resource planning (ERP), and so on. The data is made available for analytics

in a variety of ways. The following diagram illustrates The Team Data Science Process (TDSP). The process is customized for manufacturing and does an excellent job of explaining the various concerns that one has when building and executing machine learning models.



Your first task is to identify the types of failures you want to predict. With that in mind, you then identify the data sources that have relevant data about that failure type. The pipeline gets the data into the system from your environment. The data scientists use their favourite machine learning tools to prepare the data. At this point, they're ready to create and train models that can identify diverse types of issues. The models answer questions like:

- For the asset, what's the probability that a failure occurs within the next X hours?
- What's the remaining useful life of the asset?
- Is this asset behaving in an unusual way?
- Which asset requires servicing most urgently?

Benchmarking Alternate Product

Predictive maintenance is primarily used to detect upcoming system failures and prevent them using appropriate corrective measures. Using machine learning with predictive maintenance, we can analyse a massive volume of data and detect all possible failures that may lead to various financial and business losses. There are several predictive maintenance applications with machine learning, including manufacturing plants, power plants, railways, aviation, oil & gas industries, logistics & transportation, etc.

- **Manufacturing and IoT:** Predictive maintenance is widely used in manufacturing industries to supervise the production procedure through timely detection of faults and eliminate them before they malfunction using IoT. Hence, it increases the overall efficiency of the manufacturing process.

- **Automotive and Vehicles:** Various technologies connect vehicles to the sensor already enabled in the vehicle by manufacturers or dealers. These sensors collect all information and produce a massive amount of data directly retrieved by manufacturers or dealers, who warn us about any possible failure and corrective measures to prevent them before any malfunction.
- **Utility Suppliers:** Predictive maintenance techniques help utility suppliers to perform better internal work, such as predicting early traits of supply, demand issues, outage issues, etc.
- **Insurance:** Various banking and financial institutions use predictive maintenance techniques to predict accurate analytics on disastrous weather conditions.

Applicable patents

S. Matzka, "Explainable Artificial Intelligence for Predictive Maintenance Applications," 2020 Third International Conference on AI for Industries (AI4I), 2020, pp. 69-74, doi: 10.1109/AI4I49448.2020.00023.

Since real predictive maintenance datasets are generally difficult to obtain and in particular difficult to publish, we present and provide a synthetic dataset that reflects real predictive maintenance data encountered in industry to the best of our knowledge and experience.

Business Model

Predictive maintenance strategies can help determine the condition of equipment in order to predict when maintenance should be performed. This approach to maintenance can ultimately lead to cost savings over routine preventive maintenance, because maintenance is only performed when warranted.

For those in manufacturing, you know that when machinery breaks, downtime and costly repairs ensue. By implementing predictive maintenance strategies, you can maintain equipment before it breaks down saving on downtime and maintenance costs. This enables optimal asset maintenance and further improves a planet's safety.

Smart predictive maintenance is a modern maintenance technique that leverages multiple technologies and maintenance approaches, including predictive maintenance. Smart predictive maintenance goes beyond traditional preventive and predictive maintenance in three ways:

- It monitors a network of connected assets
- It automates some maintenance tasks
- It integrates with other maintenance management systems (CMMS, ERP, MES)

steps to reach smart predictive maintenance:

Step 1: **Start small with a pilot**

A pilot should generally take about three to four weeks on one or two critical assets. This initial effort will include sensor implementation and data streaming connections, as well as initial performance visualization dashboards.

Step 2: **Asset health monitoring**

It takes time to collect performance data, so patience is key. You'll want this information, as well as any asset failure data in order to generate better predictions.

Step 3: **Optimize failure threshold**

Once data can be reliably connected remotely and an asset has provided enough failure data, the failure thresholds can be optimized.

Step 4: **Leverage Data Science**

Then, a data scientist can create predictive models, along with machine learning technology to update algorithms- increasing predictive capabilities with each failure until unplanned downtime can be avoided.

Step 5: **Reaching Smart Predictive Maintenance**

For those with long-term vision, achieving steps 1-4 can lead you to smart predictive maintenance, which can help your business maintain a competitive edge

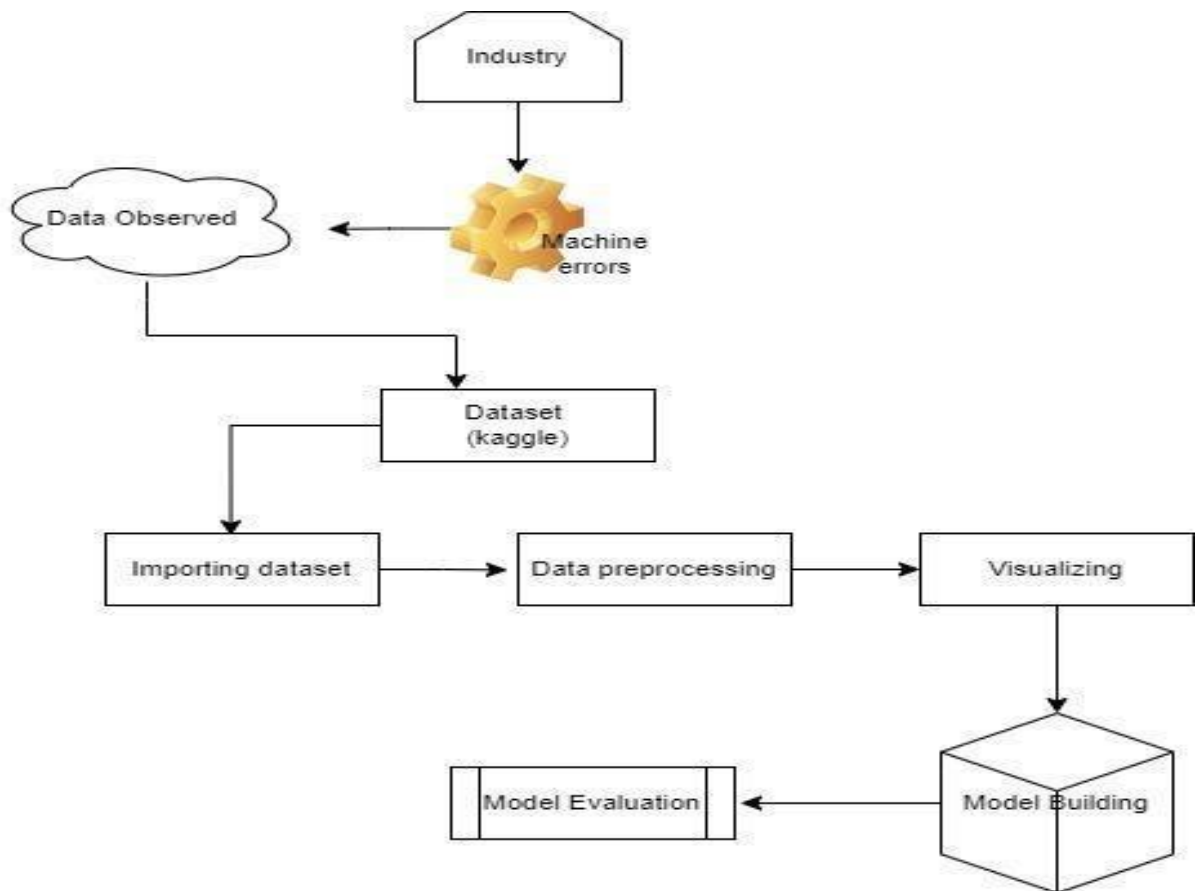
<https://www.ge.com/digital/blog/5-steps-reaching-smart-predictive-maintenance>

Predictive Maintenance for Manufacturing Industry Market : Segment Analysis

Predictive Maintenance For Manufacturing Industry Market is Segmented on the basis of Deployment, Verticals, and Geography.



Final Product (abstract) with Schematic Diagram

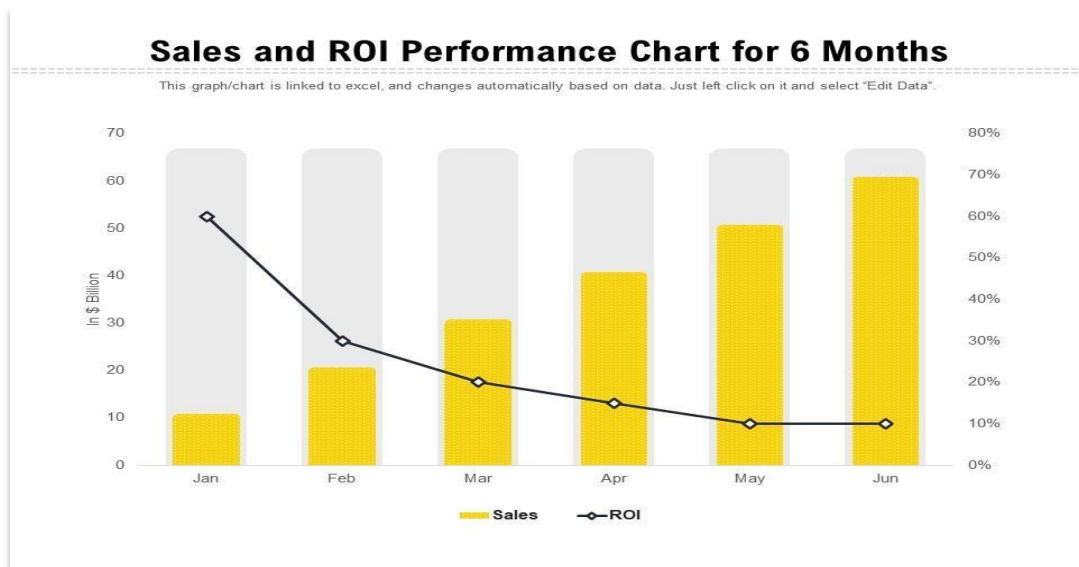


Financial Model

The Return on Investment (ROI) model is a prominent financial model for predictive maintenance in manufacturing. The financial benefits of implementing predictive maintenance are calculated by comparing the cost of implementation to the financial benefits of averting costly failures.

The ROI model is defined as follows:

$$(\text{Financial Benefits} - \text{Implementation Costs}) / \text{Implementation Costs} = \text{ROI}$$



ROI = (Net Profit / Cost of Investment) x 100

Where:

- Net Profit = Total Revenue - Total Expenses
- Cost of Investment = Total Cost of the Project

Code Implementation/Validation

IMPORTING LIBRARIES & READING DATASET

```
[1]: import pandas as pd
data = pd.read_csv('predictive_maintenance (1).csv')
data
```

	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
...
9995	9996	M24855	M	298.8	308.4	1604	29.5	14	0	No Failure
9996	9997	H39410	H	298.9	308.4	1632	31.8	17	0	No Failure
9997	9998	M24857	M	299.0	308.6	1645	33.4	22	0	No Failure
9998	9999	H39412	H	299.0	308.7	1408	48.5	25	0	No Failure
9999	10000	M24859	M	299.0	308.7	1500	40.2	30	0	No Failure

10000 rows x 10 columns

PERFORMING DATA PREPROCESSING

```
[2]: data.columns

[2]: Index(['UDI', 'Product ID', 'Type', 'Air temperature [K]',
        'Process temperature [K]', 'Rotational speed [rpm]', 'Torque [Nm]',
        'Tool wear [min]', 'Target', 'Failure Type'],
        dtype='object')

[3]: data.isnull().sum()

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

[4]: data.info()

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

```

9 Failure Type      10000 non-null object
dtypes: float64(3), int64(4), object(3)
memory usage: 781.4+ KB

```

```
[5]: data.describe().T
```

```
[5]:
```

	count	mean	std	min	25%	50%	75%	max
UDI	10000.0	5000.50000	2886.895680	1.0	2500.75	5000.5	7500.25	10000.0
Air temperature [K]	10000.0	300.00493	2.000259	295.3	298.30	300.1	301.50	304.5
Process temperature [K]	10000.0	310.00556	1.483734	305.7	308.80	310.1	311.10	313.8
Rotational speed [rpm]	10000.0	1538.77610	179.284096	1168.0	1423.00	1503.0	1612.00	2886.0
Torque [Nm]	10000.0	39.98691	9.968934	3.8	33.20	40.1	46.80	76.6
Tool wear [min]	10000.0	107.95100	63.654147	0.0	53.00	108.0	162.00	253.0
Target	10000.0	0.03390	0.180981	0.0	0.00	0.0	0.00	1.0

```
[6]: df = data.groupby(['Failure Type', 'Type'])
df.first()
```

```
[6]:
```

	UDI	Product ID	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	
Failure Type	Type								
Heat Dissipation Failure	H	3830	H33243	302.3	310.9	1366	48.4	130	1
	L	3761	L50940	302.3	310.9	1377	46.8	166	1
	M	3237	M18096	300.8	309.4	1342	62.4	113	1
No Failure	H	11	H29424	298.4	308.9	1782	23.9	24	0
	L	2	L47181	298.2	308.7	1408	46.3	3	0
	M	1	M14860	298.1	308.6	1551	42.8	0	0
Overstrain Failure	H	5400	H34813	302.8	312.4	1411	53.8	246	1
	L	161	L47340	298.4	308.2	1282	60.7	216	1
	M	3936	M18795	302.6	311.6	1227	68.2	187	1
Power Failure	H	1124	H30537	296.6	307.7	1386	62.3	100	1
	L	51	L47230	298.9	309.1	2861	4.6	143	1
	M	195	M15054	298.2	308.5	2678	10.7	86	1
Random Failures	H	1749	H31162	298.4	307.7	1626	31.1	166	0
	L	1303	L48482	298.6	309.8	1505	45.7	144	0
	M	1222	M16081	297.0	308.3	1399	46.4	132	0
Tool Wear Failure	H	1088	H30501	296.9	307.8	1549	35.8	206	1
	L	78	L47257	298.8	308.9	1455	41.3	208	1
	M	1997	M16856	298.4	308.0	1416	38.2	198	1

```
[7]: print(data['Type'].unique())
print(data['Failure Type'].unique())
```

```

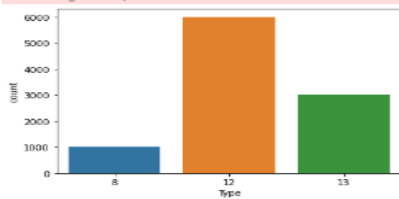
['M' 'L' 'H']
['No Failure' 'Power Failure' 'Tool Wear Failure' 'Overstrain Failure'
 'Random Failures' 'Heat Dissipation Failure']

```

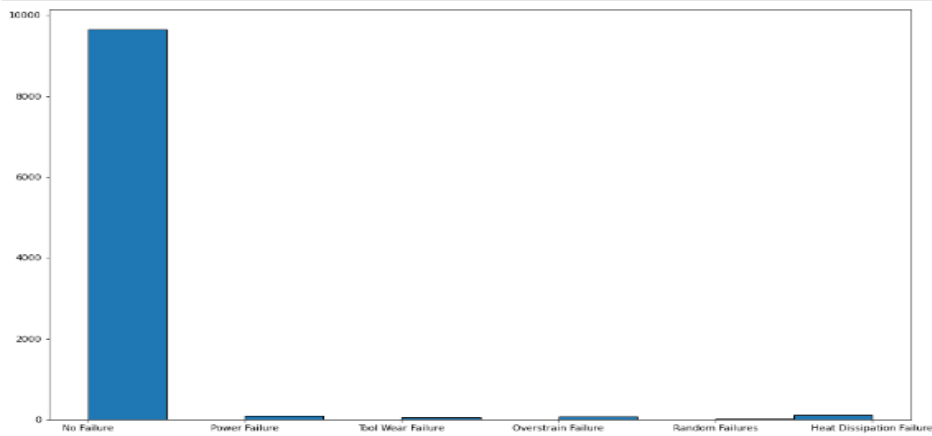

VISUALIZING THE DATASET

```
[39]: sns.countplot(data['Type'])
plt.show()
```

C:\Users\MY-PC\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be 'data', and passing other arguments without an explicit keyword will result in an error or misinterpretation.



```
[8]: import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (15,10))
plt.hist(x = data['Failure Type'], ec = 'black')
plt.show()
```

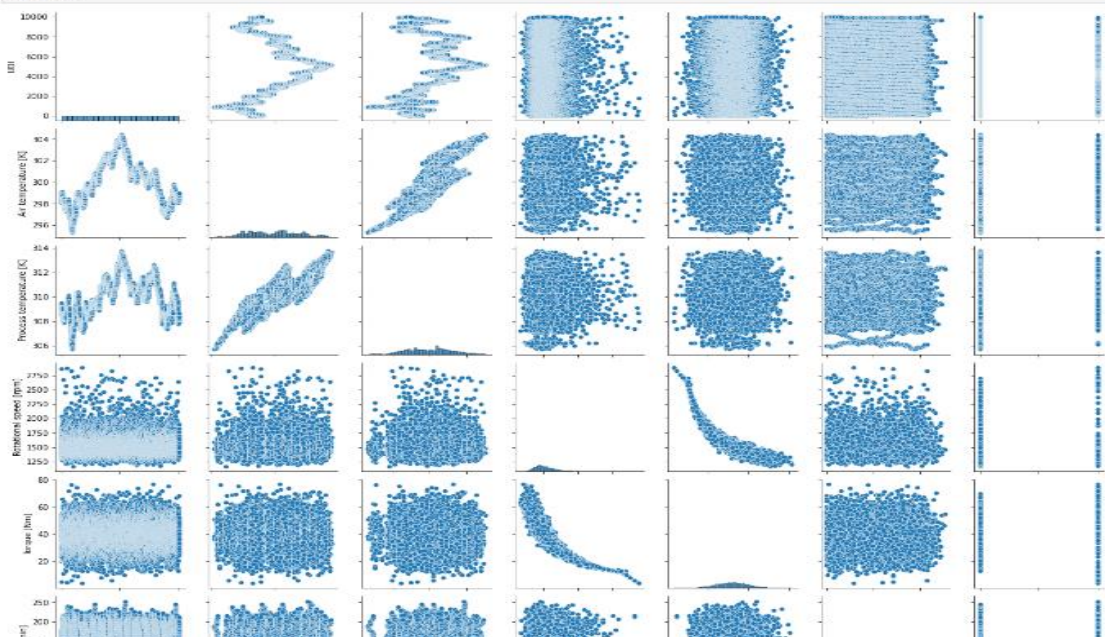


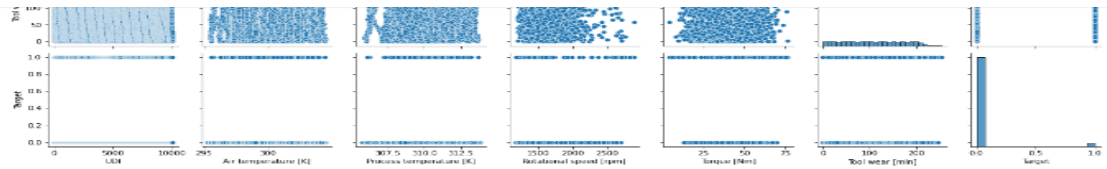
```
[9]: sns.countplot(data['Target'])
plt.show()
```

C:\Users\MY-PC\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be 'data', and passing other arguments without an explicit keyword will result in an error or misinterpretation.



```
[10]: sns.pairplot(data)
plt.show()
```





```
[11]: data.corr()
```

```
[11]:
```

	UDI	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
UDI	1.000000	0.117428	0.324428	-0.006615	0.003207	-0.010702	-0.022892
Air temperature [K]	0.117428	1.000000	0.876107	0.022670	-0.013778	0.013853	0.082556
Process temperature [K]	0.324428	0.876107	1.000000	0.019277	-0.014061	0.013488	0.035946
Rotational speed [rpm]	-0.006615	0.022670	0.019277	1.000000	-0.875027	0.000223	-0.044188
Torque [Nm]	0.003207	-0.013778	-0.014061	-0.875027	1.000000	-0.003093	0.191321
Tool wear [min]	-0.010702	0.013853	0.013488	0.000223	-0.003093	1.000000	0.105448
Target	-0.022892	0.082556	0.035946	-0.044188	0.191321	0.105448	1.000000

```
[12]: data.min()
```

```
[12]:
```

UDI	1
Product ID	H29424
Type	H
Air temperature [K]	295.8
Process temperature [K]	305.7
Rotational speed [rpm]	1168
Torque [Nm]	3.8
Tool wear [min]	0
Target	0
Failure Type	Heat Dissipation Failure
dtype: object	

```
[13]: data.max()
```

```
[13]:
```

UDI	18889
Product ID	M24859
Type	H
Air temperature [K]	304.5
Process temperature [K]	311.8
Rotational speed [rpm]	2886
Torque [Nm]	76.6
Tool wear [min]	253
Target	1
Failure Type	Tool Wear Failure
dtype: object	

```
[14]: data.median()
```

```
C:\Users\WY-PC\AppData\Local\Temp\ipykernel_7884\4184645713.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.
```

```
[14]:
```

UDI	5080.5
Air temperature [K]	300.1
Process temperature [K]	310.1
Rotational speed [rpm]	1503.0
Torque [Nm]	40.1
Tool wear [min]	188.0
Target	0.0
dtype: float64	

```
[15]: print('Target vs Air temperature [K] : ', data['Target'].corr(data['Air temperature [K]'))),
print('Target vs Process temperature [K] : ', data['Target'].corr(data['Process temperature [K]'))),
print('Target vs Rotational speed [rpm] : ', data['Target'].corr(data['Rotational speed [rpm]'))),
print('Target vs Torque [Nm] : ', data['Target'].corr(data['Torque [Nm]'))))

Target vs Air temperature [K] : 0.08255568978323986
Target vs Process temperature [K] : 0.03594597332977692
Target vs Rotational speed [rpm] : -0.0441875597343754
Target vs Torque [Nm] : 0.19132077505949352
```

```
[43]: data['Failure Type'].replace(['No Failure', 'Power Failure', 'Tool Wear Failure', 'Overstrain Failure', 'Random Failures', 'Heat Dissipation Failure'], [1, 2, 3, 4, 5, 6])
```

```
[44]: print('Target vs Air temperature [K] : ', data['Failure Type'].corr(data['Air temperature [K]'))),
print('Target vs Process temperature [K] : ', data['Failure Type'].corr(data['Process temperature [K]'))),
print('Target vs Rotational speed [rpm] : ', data['Failure Type'].corr(data['Rotational speed [rpm]'))),
print('Target vs Torque [Nm] : ', data['Failure Type'].corr(data['Torque [Nm]'))))

Target vs Air temperature [K] : 0.11848590323491176
Target vs Process temperature [K] : 0.05557937441938225
Target vs Rotational speed [rpm] : -0.11967960602784054
Target vs Torque [Nm] : 0.1903461103885093
```

```
[16]: plt.figure(figsize = (12, 6))
plt.subplot(1,2,1)
plt.scatter(data['Target'], data['Air temperature [K]'], color = 'navy', alpha = 0.5)

plt.subplot(1,2,2)
plt.scatter(data['Target'], data['Process temperature [K]'], color = 'navy', alpha = 0.5)

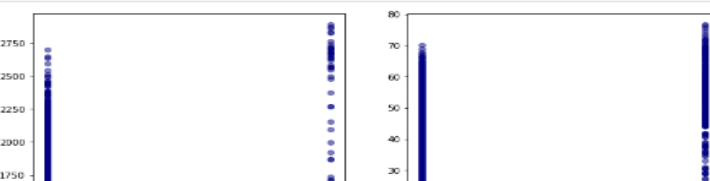
plt.show()
```



```
[17]: plt.figure(figsize = (12, 6))
plt.subplot(1,2,1)
plt.scatter(data['Target'], data['Rotational speed [rpm]'], color = 'navy', alpha = 0.5)

plt.subplot(1,2,2)
plt.scatter(data['Target'], data['Torque [Nm]'], color = 'navy', alpha = 0.5)

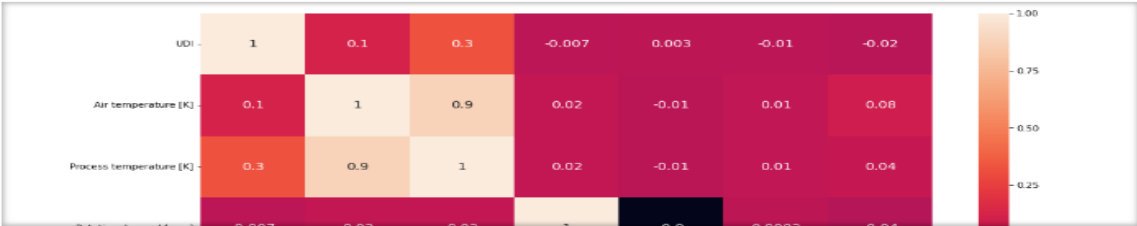
plt.show()
```



```

[18]: plt.figure(figsize = (16,11))
sns.heatmap(data,corr(), annot = True, annot_kws={'size': 14}, fmt = '.1g', mask=None)
plt.show()

```



```

[19]: !pip install pandas_profiling
from pandas_profiling import ProfileReport

Requirement already satisfied: pandas_profiling in c:\users\my-pc\anaconda3\lib\site-packages (3.6.6)
Requirement already satisfied: ydata-profiling in c:\users\my-pc\anaconda3\lib\site-packages (from pandas_profiling) (4.0.0)
Requirement already satisfied: numpy<1.24,>=1.16.0 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (1.20.3)
Requirement already satisfied: htmlmin<=0.1.12 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (0.1.12)
Requirement already satisfied: scipy<1.10,>=1.4.1 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (1.7.1)
Requirement already satisfied: requests<2.29,>=2.24.0 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (2.26.0)
Requirement already satisfied: statsmodels<0.14,>=0.13.2 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (0.13.5)
Requirement already satisfied: Jinja2<3.2,>=2.11.1 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (2.11.3)
Requirement already satisfied: visions[type_image_path]>=0.7.5 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (0.7.5)
Requirement already satisfied: pydantic<1.11,>=1.8.1 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (1.10.2)
Requirement already satisfied: tqdm<4.65,>=4.48.2 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (4.62.3)
Requirement already satisfied: multiset<1.10,>=1.4 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (1.9.1)
Requirement already satisfied: pandas<1.4.0,<1.6,>=1.1 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (1.3.4)
Requirement already satisfied: matplotlib<3.7,>=3.2 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (3.4.3)
Requirement already satisfied: typeguard<2.14,>=2.13.2 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (2.13.3)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (0.11.2)
Requirement already satisfied: phik<0.13,>=0.11.1 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (0.12.3)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in c:\users\my-pc\anaconda3\lib\site-packages (from ydata-profiling->pandas_profiling) (6.0)
Requirement already satisfied: tangled-up-in-unicode>=0.0.4 in c:\users\my-pc\anaconda3\lib\site-packages (from visions[type_image_path]>=0.7.5->ydata-profiling->pandas_profiling) (0.2.0)
Requirement already satisfied: networkx>=2.4 in c:\users\my-pc\anaconda3\lib\site-packages (from visions[type_image_path]>=0.7.5->ydata-profiling->pandas_profiling) (2.6.3)
Requirement already satisfied: attrs>=19.3.0 in c:\users\my-pc\anaconda3\lib\site-packages (from visions[type_image_path]>=0.7.5->ydata-profiling->pandas_profiling) (21.2.0)
Requirement already satisfied: imagehash in c:\users\my-pc\anaconda3\lib\site-packages (from visions[type_image_path]>=0.7.5->ydata-profiling->pandas_profiling) (4.3.1)
Requirement already satisfied: Pillow in c:\users\my-pc\anaconda3\lib\site-packages (from visions[type_image_path]>=0.7.5->ydata-profiling->pandas_profiling) (8.4.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\my-pc\anaconda3\lib\site-packages (from Jinja2<3.2,>=2.11.1->ydata-profiling->pandas_profiling) (1.1.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\my-pc\anaconda3\lib\site-packages (from matplotlib<3.7,>=3.2->ydata-profiling->pandas_profiling) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\my-pc\anaconda3\lib\site-packages (from matplotlib<3.7,>=3.2->ydata-profiling->pandas_profiling) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\my-pc\anaconda3\lib\site-packages (from matplotlib<3.7,>=3.2->ydata-profiling->pandas_profiling) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\my-pc\anaconda3\lib\site-packages (from matplotlib<3.7,>=3.2->ydata-profiling->pandas_profiling) (1.3.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\my-pc\anaconda3\lib\site-packages (from pandas<1.4.0,<1.6,>=1.1->ydata-profiling->pandas_profiling) (2021.3)
Requirement already satisfied: patsy>=0.5.2 in c:\users\my-pc\anaconda3\lib\site-packages (from statsmodels<0.14,>=0.13.2->ydata-profiling->pandas_profiling) (0.5.2)
Requirement already satisfied: colorama in c:\users\my-pc\anaconda3\lib\site-packages (from tqdm<4.65,>=4.48.2->ydata-profiling->pandas_profiling) (0.4.4)
Requirement already satisfied: six in c:\users\my-pc\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib<3.7,>=3.2->ydata-profiling->pandas_profiling) (1.16.0)
Requirement already satisfied: PyWavelets in c:\users\my-pc\anaconda3\lib\site-packages (from imagehash->visions[type_image_path]>=0.7.5->ydata-profiling->pandas_profiling) (1.1.1)

[notice] A new release of pip available: 22.3.1 -> 23.0
[notice] To update, run: python.exe -m pip install --upgrade pip
C:\Users\WV\AppData\Local\Temp\ipykernel_7004\4228493054.py:2: DeprecationWarning: 'import pandas_profiling' is going to be deprecated by April 15
c. Please use 'import ydata_profiling' instead.
from pandas_profiling import ProfileReport

[20]: sktime
profile = ProfileReport(data,
                        title="Predictive Maintenance",
                        dataset={"description": "This profiling report was generated for Akshay Kumar",
                                "copyright_holder": "Akshay Kumar",
                                "copyright_year": "2023",
                                },
                        explorative=True,
                        )
profile

Wall time: 48.7 ms
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]

[20]:
[21]: data['Type'].replace(['M', 'L', 'H'], [13, 12, 8], inplace=True)

```

BUILDING THE MODEL

```

[22]: target = data['Failure Type']
target

[22]: 0      No Failure
1      No Failure
2      No Failure
3      No Failure
4      No Failure
...
9995    No Failure
9996    No Failure
9997    No Failure
9998    No Failure
9999    No Failure
Name: Failure Type, Length: 10000, dtype: object

[23]: features = data.drop(['Failure Type', 'Product ID'], axis = 1)
features

```

	UDI	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
0	1	13	298.1	308.6	1551	42.8	0	0
1	2	12	298.2	308.7	1408	46.3	3	0
2	3	12	298.1	308.5	1498	49.4	5	0
3	4	12	298.2	308.6	1433	39.5	7	0
4	5	12	298.2	308.7	1408	40.0	9	0
...
9995	9996	13	298.8	308.4	1604	29.5	14	0
9996	9997	8	298.9	308.4	1632	31.8	17	0

```

9997 9998 13 299.0 308.6 1645 33.4 22 0
[24]: from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=10)

[25]: from sklearn import svm
model = svm.LinearSVC()
model.fit(x_train, y_train)
C:\Users\MY-PC\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
warnings.warn("Liblinear failed to converge, increase "
linearSVC()

[26]: model.score(x_train, y_train)

[26]: 0.966375

[27]: model.score(x_test, y_test)

[27]: 0.971

```

EVALUATING THE MODEL

```

[28]: prediction = model.predict(x_test)
prediction

[28]: array(['No Failure', 'No Failure', 'No Failure', ..., 'No Failure',
       'No Failure', 'No Failure'], dtype=object)

[29]: score = model.score(x_train, y_train)
score

[29]: 0.966375

[30]: score_val = model.score(x_test, y_test)
score_val

[30]: 0.971

[31]: from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(prediction, y_test)
conf_mat

[31]: array([[ 0,  0,  0,  0,  0,  0],
       [15, 1935, 14, 12,  5,  8],
       [ 1,  0,  0,  1,  0,  0],
       [ 0,  2,  0,  7,  0,  0],
       [ 0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0]], dtype=int64)

[32]: conf_mat.shape

[32]: (6, 6)

[33]: nr_rows = conf_mat.shape[0]
nr_cols = conf_mat.shape[1]

[34]: import itertools
plt.figure(figsize=(6,6), dpi=95)

plt.imshow(conf_mat, cmap = plt.cm.Greens)

plt.title('Confusion Matrix', fontsize=16)
plt.ylabel('Actual Labels', fontsize=12)
plt.xlabel('Predicted Labels', fontsize=12)

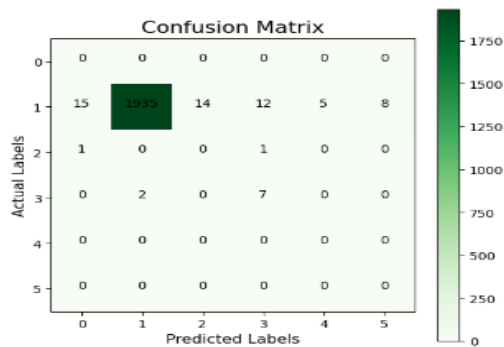
for i, j in itertools.product(range(nr_rows), range(nr_cols)):
    plt.xlabel('Predicted Labels', fontsize=12)

for i, j in itertools.product(range(nr_rows), range(nr_cols)):
    plt.text(j,i,conf_mat[i,j],horizontalalignment = 'center')

plt.colorbar()

plt.show()

```



```

[35]: # True Positives
import numpy as np
np.diag(conf_mat)

[35]: array([ 0, 1935,  0,  7,  0,  0], dtype=int64)

[36]: recall = np.diag(conf_mat) / np.sum(conf_mat, axis=1)
recall
C:\Users\MY-PC\AppData\Local\Temp\ipykernel_7004\2268408136.py:1: RuntimeWarning: invalid value encountered in true_divide
recall = np.diag(conf_mat) / np.sum(conf_mat, axis=1)

[36]: array([ nan, 0.97285068,  0.          , 0.77777778,  nan,
       nan])

[37]: precision = np.diag(conf_mat) / np.sum(conf_mat, axis=0)
precision

[37]: array([0.          , 0.99896748,  0.          , 0.35          , 0.          ,
       0.          ])

[38]: avg_recall = np.mean(recall)
print(avg_recall)
avg_precision = np.mean(precision)
print(avg_precision)

nan
0.22482791257959045

```

Conclusion

Predictive maintenance presents you with the best time to work on an asset so that maintenance frequency is minimal and reliability is as high as possible while eliminating unnecessary costs. However, there are few disadvantages to predictive maintenance like high start-up costs and the need for specialized personnel.

Clearly, predictive maintenance is not apt for every company, especially those that have not yet implemented planned maintenance activities. However, larger organizations that have outgrown conventional maintenance practices and have additional budgets should leverage predictive maintenance. Predictive maintenance has been shown to result in a tenfold increase in ROI, 25%-30% reduction in maintenance costs, a 70%-75% decrease in breakdowns, and a 35%-45% reduction in downtime. These statistics are evidence of why predictive maintenance is gaining prominence quickly.