Project Report Titles

1 INTRODUCTION

1.1 Overview

A brief description about your project

1.2 Purpose

The use of this project. What can be achieved using this.

2 LITERATURE SURVEY

2.1 Existing problem

Existing approaches or method to solve this problem

2.2 Proposed solution

What is the method or solution suggested by you?

3 THEORITICAL ANALYSIS

3.1 Hardware / Software designing

Hardware and software requirements of the project

1. **EXPERIMENTAL INVESTIGATIONS**

   Analysis or the investigation made while working on the solution.

2. **FLOWCHART**

   Diagram showing the control flow of the solution

3. **RESULT**

   Final findings (Output) of the project along with screenshots.

4. **ADVANTAGES & DISADVANTAGES**

   List of advantages and disadvantages of the proposed solution

5. **APPLICATIONS**

   The areas where this solution can be applied

6. **CONCLUSION**

Conclusion summarizing the entire work and findings.

10 FUTURE SCOPE

Enhancements that can be made in the future.

Project Report Titles

11 BIBILOGRAPHY

References of previous works or websites visited/books referred for analysis about the project, solution previous findings etc.

APPENDIX

A. Source Code

Attach the code for the solution built.

Machine Learning Approach For Predictive Maintenance Aircraft Engine

1 Introduction

a. Overview

Engine failure is highly risky and needs a lot of time for repair. Unexpected failure leads to loss of money and time. Predicting the failure prior, will save time, effort, money and sometimes even lives. The failure can be detected by installing the sensors and keeping a track of the values. The failure detection and predictive maintenance can be for any device, out of which we will be dealing with the engine failure for a threshold number of

b. Purpose The use of this project.

The project aims to predict the failure of an engine by using Machine Learning to save loss of time & money thus improving productivity.

2 LITERATURE SURVEY

2.1 Existing problem

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre process/clean the data using different data pre-processing techniques.
- Apply different algorithms according to the dataset
- You will be able to know how to find the accuracy of a model.
- You will be able to Build web applications using the Flask framework.

2.2 Proposed solution

- **Data Collection.**
    - Collect the dataset

- **Data Pre processing.**
    - Import the Libraries.
    - Importing the dataset.
    - Checking for Null Values.
    - Data Visualization.
    - Taking care of Missing Data.
    - Label encoding.
    - One Hot Encoding.
    - Feature Scaling.
    - Splitting Data into Train and Test.
- **Model Building**
    - Training and testing the model
    - Evaluation of Model (Decision Tree Classification)
- **Application Building**
    - Create an HTML file
    - Build a Python Code

3 THEORITICAL ANALYSIS

3.1 Software

we will be using

- **Jupter** notebook
- **Spyder**
  because it is a free and open-source distribution of the Python for data science and machine learning related applications

- Flask (Web applications)

Hardware:

1. **Processor :** Processor Intel CORE i5 and above Internet.

2. **System architecture :** Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit aarch64 (AWS Graviton2 / arm64), 64-bit Power8/Power9, s390x (Linux on IBM Z & Linux ONE).

- **RAM**:4 GB or above.

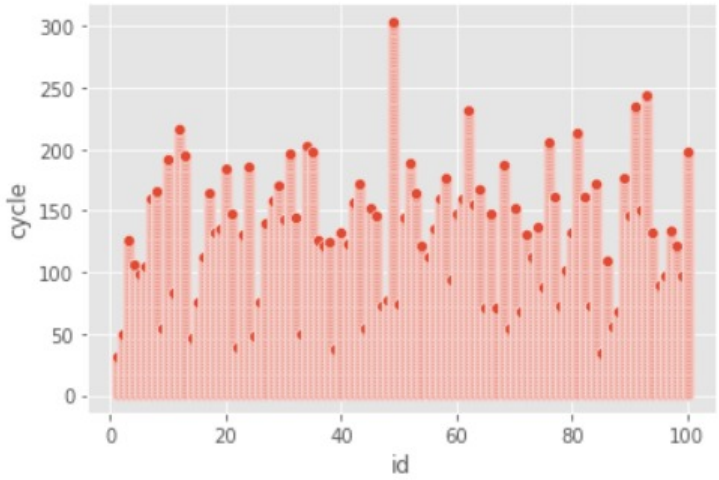
4. Analysis or the investigation made while working on the solution.

Scatterplot

A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

```
In [49]: import seaborn as sns
         sns.scatterplot(x='id',y='cycle',data=df_test)

Out[49]: <AxesSubplot:xlabel='id', ylabel='cycle'>
```



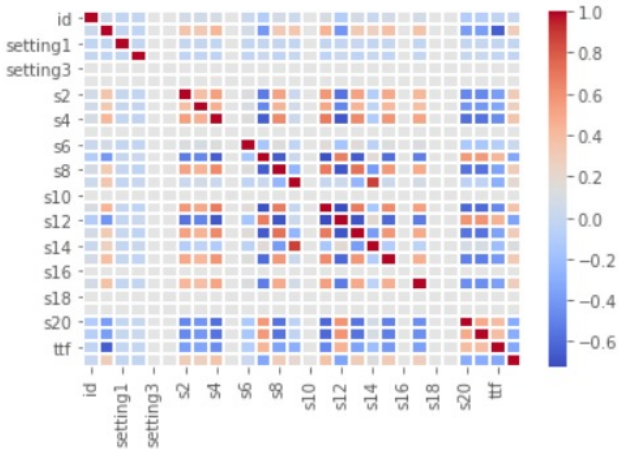
From this scatterplot, comparing the two columns we can see many flights were delayed from their arrival time.

Heatmap

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred.

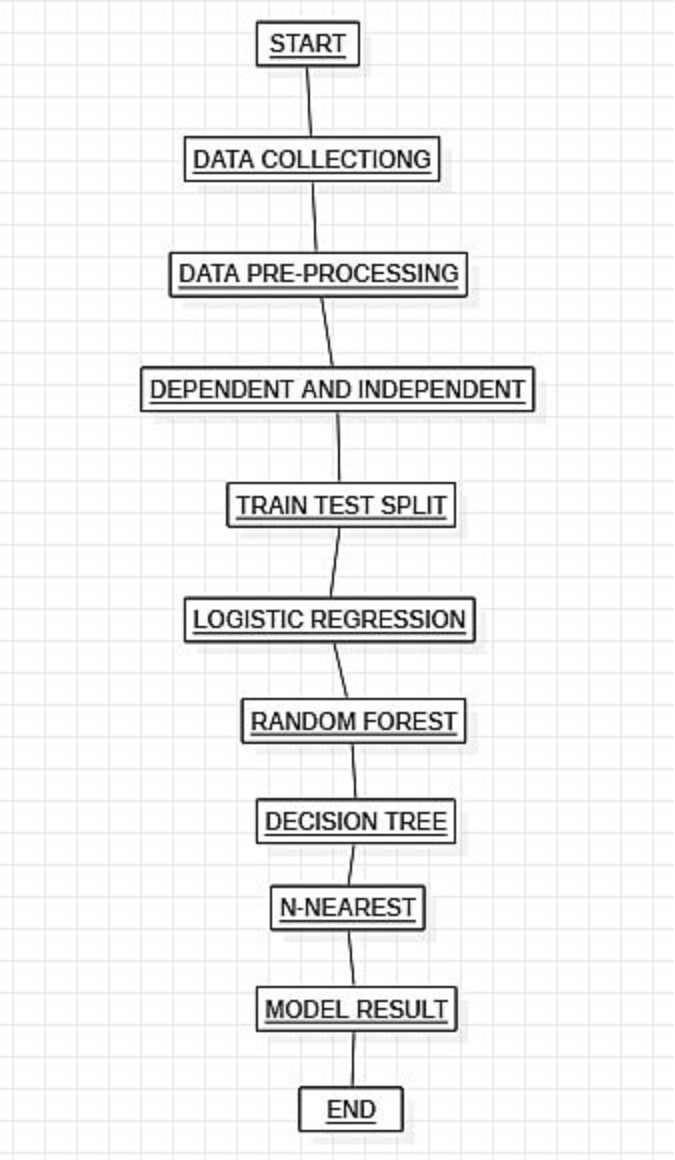
```
In [45]: sns.heatmap(df_test.corr(),cmap='coolwarm',linecolor='white',linewidths=1)

Out[45]: <AxesSubplot:>
```



If you observe the heatmap, lighter the colour the correlation between that two variables will be high. And correlation plays a very important role for extracting the correct features for building our model.

5 Flow Chart



6.Result

Result final findings (output) of the project along with screenshots.

```
y_predlog = model.predict(x_test)
```

```
y_predlog
```

```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

Decision Tree Model Accuracy

```
from sklearn.metrics import accuracy_score
accuracy_score(y_predlog,y_test)
```

```
0.9993891264508247
```

Confusion Matrix

```
In [33]: df_test['label_bc'].value_counts()
Out[33]: 0    12764
         1      332
         Name: label_bc, dtype: int64

In [35]: from sklearn.metrics import confusion_matrix
         cm1 = confusion_matrix(y_test,y_predlog)
         cm1
Out[35]: array([[12763,     1],
                [    7,   325]], dtype=int64)
```

9 CONCLUSION

1. By measuring the performance of the models using real data, we have seen interesting results on the predictability of the faliour.

2. This is the main page of Prediction of aircraft engine .where you may know about the inputs.

3. The prediction page user gives the input for predicting the output where they can give input as id,cycle,setting1,setting2,setting3,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21then click to submit the output.

4. In the prediction page user will get the output based on the inputs they given in the prediction page.

11 Biblography

References

1. Tong, M.T., “Using Machine Learning To Predict Core Sizes of High-Efficiency Turbofan Engines,” GT2019-91432, ASME Turbo-Expo 2019, June 17-21, 2019.

2. Daly, M., “Jane’s Aero-Engine,” 2017-2018.

3. Meier, N., “Civil turbojet/turbofan specifications.” http://www.jet-engine.net/civtfspec.html. Accessed August, 2018.

4. GE Aviation. https://www.geaviation.com/commercial

5. Pratt and Whitney. https://www.pw.utc.com/products-and-services/products/commercial-engines

Appendix

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score

import matplotlib.pyplot as plt
plt.style.use('ggplot')
%matplotlib inline
```

In [2]:
```
dataset_train=pd.read_csv('PM_train.txt',sep=' ',header=None).drop([26,27],axis=1)
col_names = ['id','cycle','setting1','setting2','setting3','s1','s2','s3','s4','s5','s6','s7','s8','s9','s10','s11','s12','s13',
dataset_train.columns=col_names
print('Shape of Train dataset: ',dataset_train.shape)
dataset_train.head()
```

```
Shape of Train dataset:  (20631, 26)
```

Out[2]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

5 rows × 26 columns

In [3]:
```
dataset_train['id'].value_counts()
```

Out[3]:
```
69    362
92    341
96    336
67    313
83    293
     ...
24    147
57    137
70    137
91    135
39    128
Name: id, Length: 100, dtype: int64
```

In [4]:
```
dataset_test=pd.read_csv('PM_test.txt',sep=' ',header=None).drop([26,27],axis=1)
dataset_test.columns=col_names
#dataset_test.head()
print('Shape of Test dataset: ',dataset_train.shape)
dataset_train.head()
```

```
Shape of Test dataset:  (20631, 26)
```

Out[4]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

5 rows × 26 columns

In [5]:
```
pm_truth=pd.read_csv('PM_truth.txt',sep=' ',header=None).drop([1],axis=1)
pm_truth.columns=['more']
pm_truth['id']=pm_truth.index+1
pm_truth.head()
```

Out[5]:

	more	id
0	112	1
1	98	2
2	69	3
3	82	4
4	91	5

In [6]:
```
pm_truth.shape
```

Out[6]: (100, 2)

In [7]:
```
rul = pd.DataFrame(dataset_test.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id', 'max']
rul.head()
```

Out[7]:

	id	max
0	1	31
1	2	49
2	3	126
3	4	106
4	5	98

In [8]:
```
rul.shape
```

Out[8]: (100, 2)

In [9]:
```
pm_truth['rtf']=pm_truth['more']+rul['max']
pm_truth.head()
```

Out[9]:

	more	id	rtf
0	112	1	143
1	98	2	147
2	69	3	195
3	82	4	188
4	91	5	189

In [10]:
```
pm_truth.shape
```

Out[10]: (100, 3)

In [11]:
```
pm_truth.drop('more', axis=1, inplace=True)
dataset_test=dataset_test.merge(pm_truth,on=['id'],how='left')
dataset_test['ttf']=dataset_test['rtf'] - dataset_test['cycle']
dataset_test.drop('rtf', axis=1, inplace=True)
dataset_test.head()
```

Out[11]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s13	s14	s15	s16	s17	s18	s19	s20	s21	ttf
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	...	2388.03	8125.55	8.4052	0.03	392	2388	100.0	38.86	23.3735	142
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	...	2388.06	8139.62	8.3803	0.03	393	2388	100.0	39.02	23.3916	141
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	...	2388.03	8130.10	8.4441	0.03	393	2388	100.0	39.08	23.4166	140
3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	...	2388.05	8132.90	8.3917	0.03	391	2388	100.0	39.00	23.3737	139

3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	...	2388.05	8132.90	8.3917	0.03	391	2388	100.0	39.00	23.3737	139
4	1	5	0.0014	0.0000	100.0	518.67	642.51	1587.19	1401.92	14.62	...	2388.03	8129.54	8.4031	0.03	390	2388	100.0	38.99	23.4130	138

5 rows × 27 columns

In [12]:
```
dataset_test.shape
```

Out[12]: (13096, 27)

In [13]:
```
dataset_train['ttf'] = dataset_train.groupby(['id'])['cycle'].transform(max)-dataset_train['cycle']
dataset_train.head()
```

Out[13]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s13	s14	s15	s16	s17	s18	s19	s20	s21	ttf
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190	191
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236	190
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442	189
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739	188
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	187

5 rows × 27 columns

In [14]:
```
dataset_train.shape
```

Out[14]: (20631, 27)

In [14]:
```
dataset_train.shape
```

Out[14]: (20631, 27)

In [15]:
```
dataset_train['ttf'].value_counts()
```

Out[15]:
```
0      100
87     100
118    100
14     100
31     100
      ...
361      1
346      1
347      1
348      1
351      1
Name: ttf, Length: 362, dtype: int64
```

In [16]:
```
df_train=dataset_train.copy()
df_test=dataset_test.copy()
period=30
df_train['label_bc'] = df_train['ttf'].apply(lambda x: 1 if x <= period else 0)
df_test['label_bc'] = df_test['ttf'].apply(lambda x: 1 if x <= period else 0)
df_train.head()
```

Out[16]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s14	s15	s16	s17	s18	s19	s20	s21	ttf	label_bc
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190	191	0
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236	190	0
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442	189	0
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739	188	0
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	187	0

4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	187	0

5 rows × 28 columns

In [17]:
```
df_train['label_bc'].value_counts()
```

Out[17]:
```
0    17531
1     3100
Name: label_bc, dtype: int64
```

In [18]:
```
features_col_name=['setting1', 'setting2', 'setting3', 's1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11',
                   's12', 's13', 's14', 's15', 's16', 's17', 's18', 's19', 's20', 's21']
target_col_name='label_bc'
```

In [19]:
```
sc=MinMaxScaler()
df_train[features_col_name]=sc.fit_transform(df_train[features_col_name])
df_test[features_col_name]=sc.transform(df_test[features_col_name])
```

In [20]:
```
df_train.head()
```

Out[20]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s14	s15	s16	s17	s18	s19	s20	s21	ttf	lab
0	1	1	0.459770	0.166667	0.0	0.0	0.183735	0.406802	0.309757	0.0	...	0.199608	0.363986	0.0	0.333333	0.0	0.0	0.713178	0.724662	191	
1	1	2	0.609195	0.250000	0.0	0.0	0.283133	0.453019	0.352633	0.0	...	0.162813	0.411312	0.0	0.333333	0.0	0.0	0.666667	0.731014	190	
2	1	3	0.252874	0.750000	0.0	0.0	0.343373	0.369523	0.370527	0.0	...	0.171793	0.357445	0.0	0.166667	0.0	0.0	0.627907	0.621375	189	
3	1	4	0.540230	0.500000	0.0	0.0	0.343373	0.256159	0.331195	0.0	...	0.174889	0.166603	0.0	0.333333	0.0	0.0	0.573643	0.662386	188	
4	1	5	0.390805	0.333333	0.0	0.0	0.349398	0.257467	0.404625	0.0	...	0.174734	0.402078	0.0	0.416667	0.0	0.0	0.589147	0.704502	187	

5 rows × 28 columns

In [21]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20631 entries, 0 to 20630
Data columns (total 28 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        20631 non-null  int64
 1   cycle     20631 non-null  int64
 2   setting1  20631 non-null  float64
 3   setting2  20631 non-null  float64
 4   setting3  20631 non-null  float64
 5   s1        20631 non-null  float64
 6   s2        20631 non-null  float64
 7   s3        20631 non-null  float64
 8   s4        20631 non-null  float64
 9   s5        20631 non-null  float64
 10  s6        20631 non-null  float64
 11  s7        20631 non-null  float64
 12  s8        20631 non-null  float64
 13  s9        20631 non-null  float64
 14  s10       20631 non-null  float64
 15  s11       20631 non-null  float64
 16  s12       20631 non-null  float64
 17  s13       20631 non-null  float64
 18  s14       20631 non-null  float64
 19  s15       20631 non-null  float64
 20  s16       20631 non-null  float64
 21  s17       20631 non-null  float64
 22  s18       20631 non-null  float64
 23  s19       20631 non-null  float64
 24  s20       20631 non-null  float64
 25  s21       20631 non-null  float64
 26  ttf       20631 non-null  int64
 27  label_bc  20631 non-null  int64
dtypes: float64(24), int64(4)
memory usage: 4.4 MB
```

In [22]:
```
df_train['ttf'].min()
```

Out[22]:
```
0
```

In [23]:
```
df_train['ttf'].max()
```

Out[23]:
```
361
```

In [24]:
```
df_train.iloc[0,:]
```

Out[24]:
```
id            1.000000
cycle         1.000000
setting1      0.459770
setting2      0.166667
setting3      0.000000
s1            0.000000
s2            0.183735
s3            0.406802
s4            0.309757
s5            0.000000
s6            1.000000
s7            0.726248
s8            0.242424
s9            0.109755
s10           0.000000
s11           0.369048
s12           0.633262
s13           0.205882
s14           0.199608
s15           0.363986
s16           0.000000
s17           0.333333
s18           0.000000
s19           0.000000
s20           0.713178
s21           0.724662
ttf         191.000000
label_bc      0.000000
Name: 0, dtype: float64
```

In [25]:
```
x_train = df_train.iloc[:,:-1].values
y_train = df_train.iloc[:,-1:].values
```

In [26]:
```
x_train.shape
```

Out[26]:
```
(20631, 27)
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\Praveen Raj\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\Praveen Raj\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to con
verge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[27]:
```
LogisticRegression()
```

In [28]:
```
import joblib
```

In [29]:
```
joblib.dump(model, "engine_model.sav")
```

Out[29]:
```
['engine_model.sav']
```

In [30]:
```
x_test = df_test.iloc[:,:-1].values
y_test = df_test.iloc[:,-1:].values
```

In [50]:
```
y_predlog = model.predict(x_test)
```

In [51]:
```
y_predlog
```

Out[51]:
```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

In [32]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_predlog,y_test)
```

Out[32]:
```
0.9993891264508247
```

In [33]:
```
df_test['label_bc'].value_counts()
```

Out[33]:
```
0    12764
1      332
Name: label_bc, dtype: int64
```

In [33]:
```
df_test['label_bc'].value_counts()
```

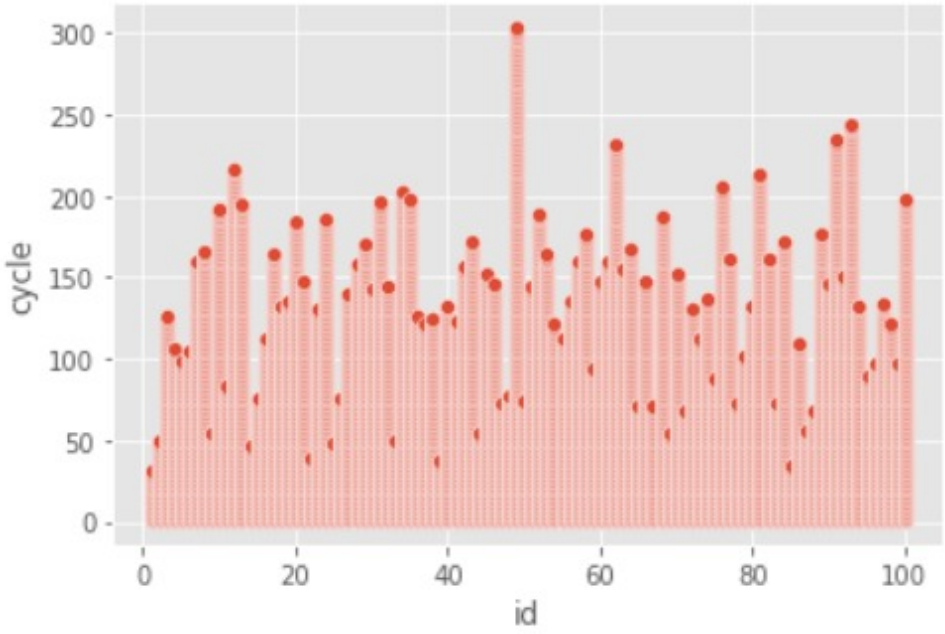
Out[33]:
```
0    12764
1      332
Name: label_bc, dtype: int64
```

In [35]:
```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test,y_predlog)
cm1
```

Out[35]:
```
array([[12763,     1],
       [    7,   325]], dtype=int64)
```

In [49]:
```
import seaborn as sns
sns.scatterplot(x='id',y='cycle',data=df_test)
```

Out[49]: <AxesSubplot:xlabel='id', ylabel='cycle'>



In [45]:
```
sns.heatmap(df_test.corr(),cmap='coolwarm',linecolor='white',linewidths=1)
```

Out[45]: <AxesSubplot:>