Machine Learning Approach For Predictive Maintenance Aircraft Engine Using IBM Watson Studio

1.INTRODUCTION

1.1 OVERVIEW

Aircrafts are very important part of modern age. The number of passengers traveling by airplanes has been increasing every year. So the safety of aircraft passengers' is of very much important.

It is crucial that Aircraft Engines should undergo proper maintenance.Engine failure is highly risky and needs a lot of time for repair. Doing a routine maintenance can be very expensive. Predictive maintenance is an effective alternative to it. Machine/Deep Learning are widely used for predictive maintenance. The project aims to predict the failure of an engine by using Machine Learning to save loss of time & money thus improving productivity.

1.2 PURPOSE

Predictive Maintenance techniques are used to determine the condition of an equipment to plan the maintenance/failure ahead of its time.This approach ensures cost saving.

The main purpose of predictive maintenance is to reduce the cost of time-based maintenance,and to prevent unexpected equipment failures.By knowing which equipment needs maintenance,repair work can be better planned

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

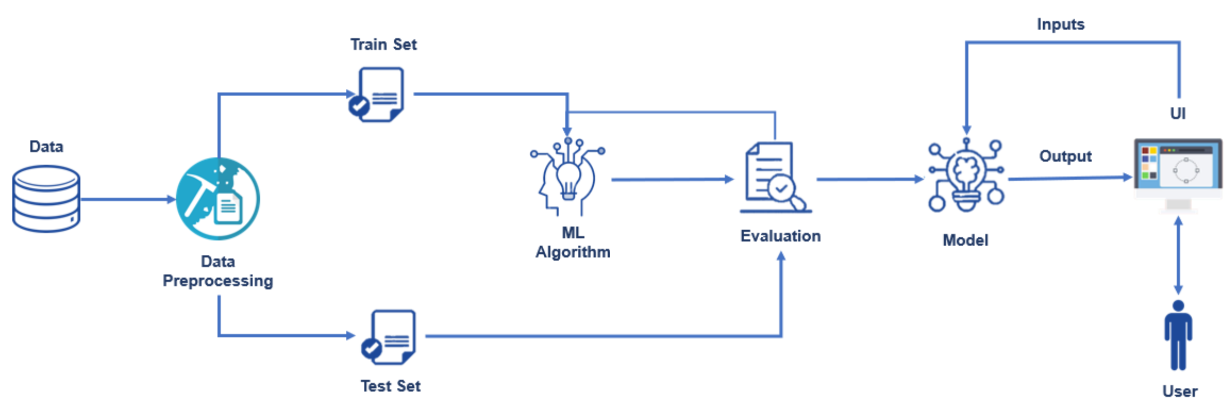
Airlines are particularly interested in predicting equipment failures in advance so that they can enhance operations and reduce flight delays.Unexpected failure leads to loss of money and time.

2.2 EXISTING SOLUTION

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 days.

3.THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE/SOFTWARE DESIGNING

Software Requirements:

- Anaconda Navigator
- Keras
- Flask

Hardware Requirements:

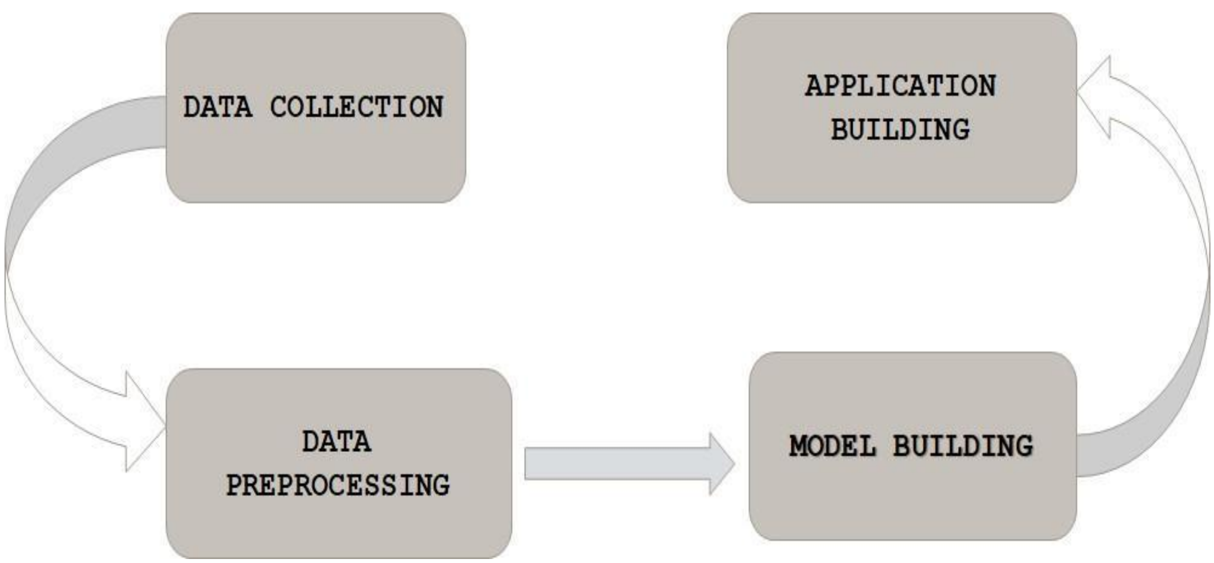
- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 8 GB
- Display : 14.1 “Color Monitor(LCD, CRT or LED)

Clock Speed : 1.67 GHz

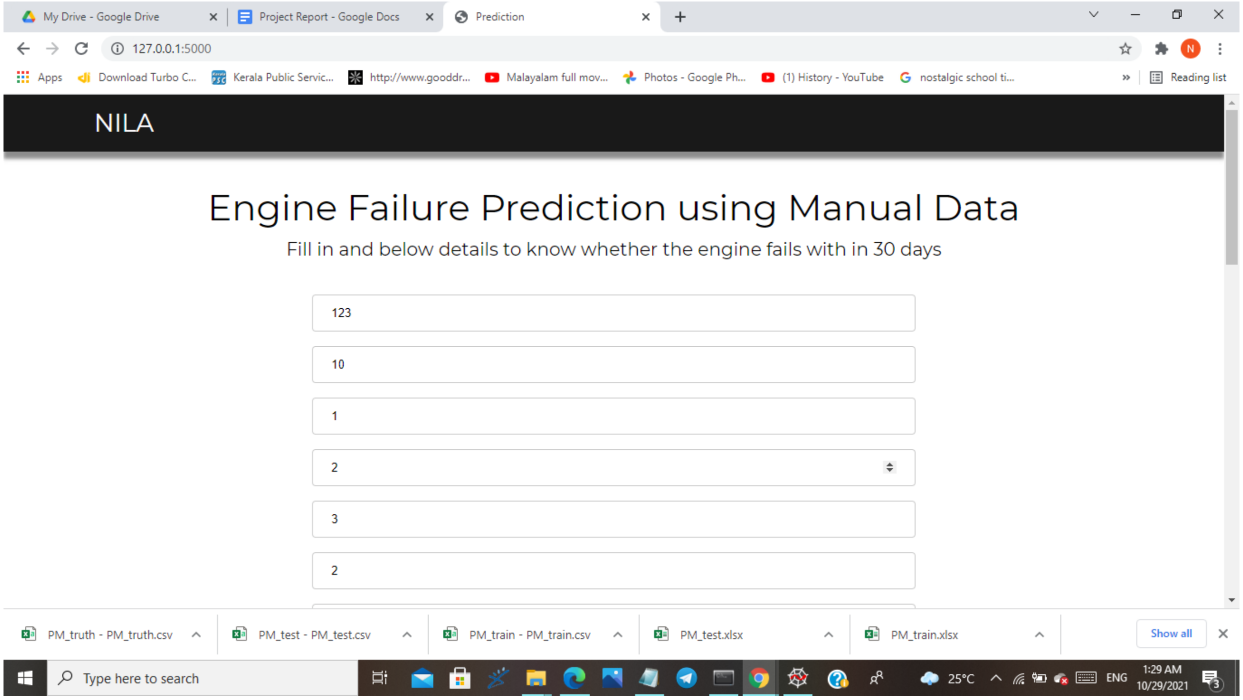
4. EXPERIMENTAL INVESTIGATIONS

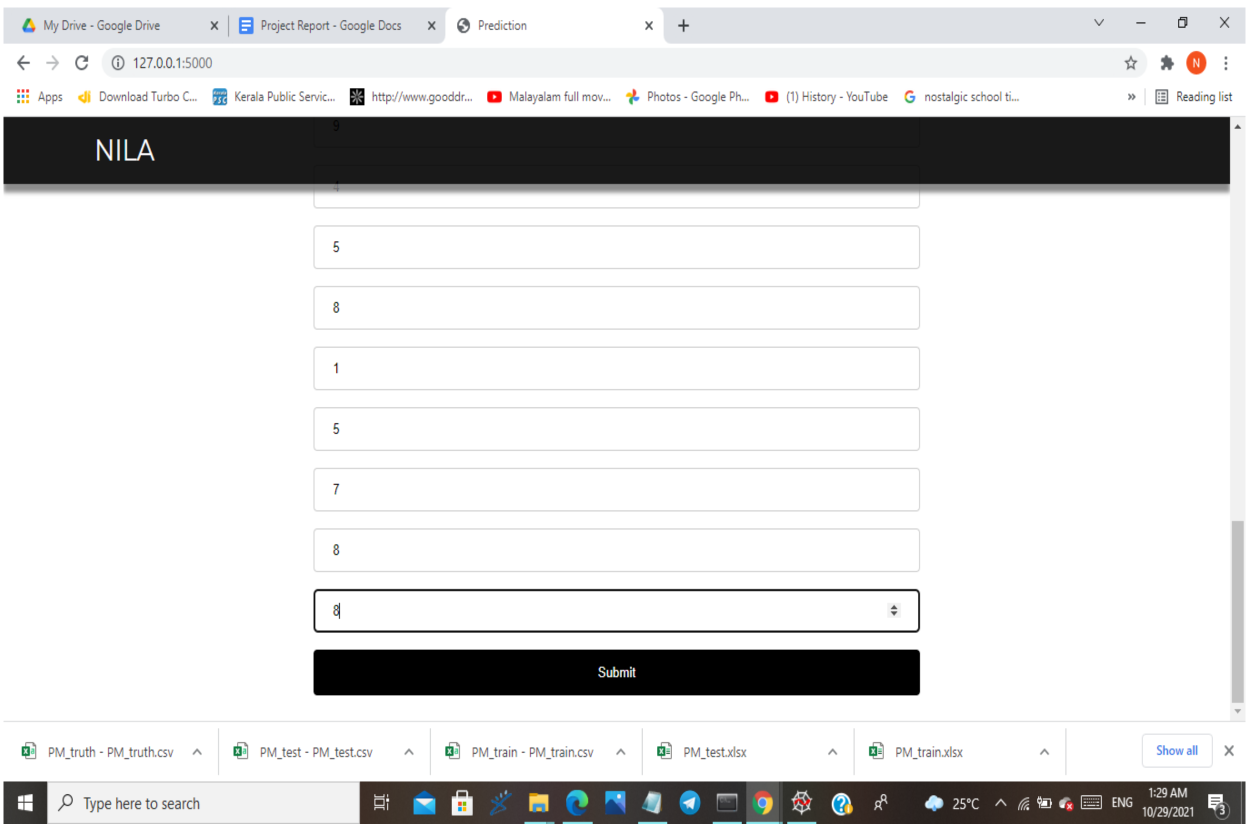
By exploring aircraft engine’s sensor values over time, machine learning algorithm can learn the relationship between sensor values and changes in sensor values to the historical failures in order to predict failures in the future.Observing engine's health and condition through sensors and telemetry data is assumed to facilitate maintenance by predicting Time-To-Failure (TTF) or Remaining Useful Life (RUL) of in-service equipment.

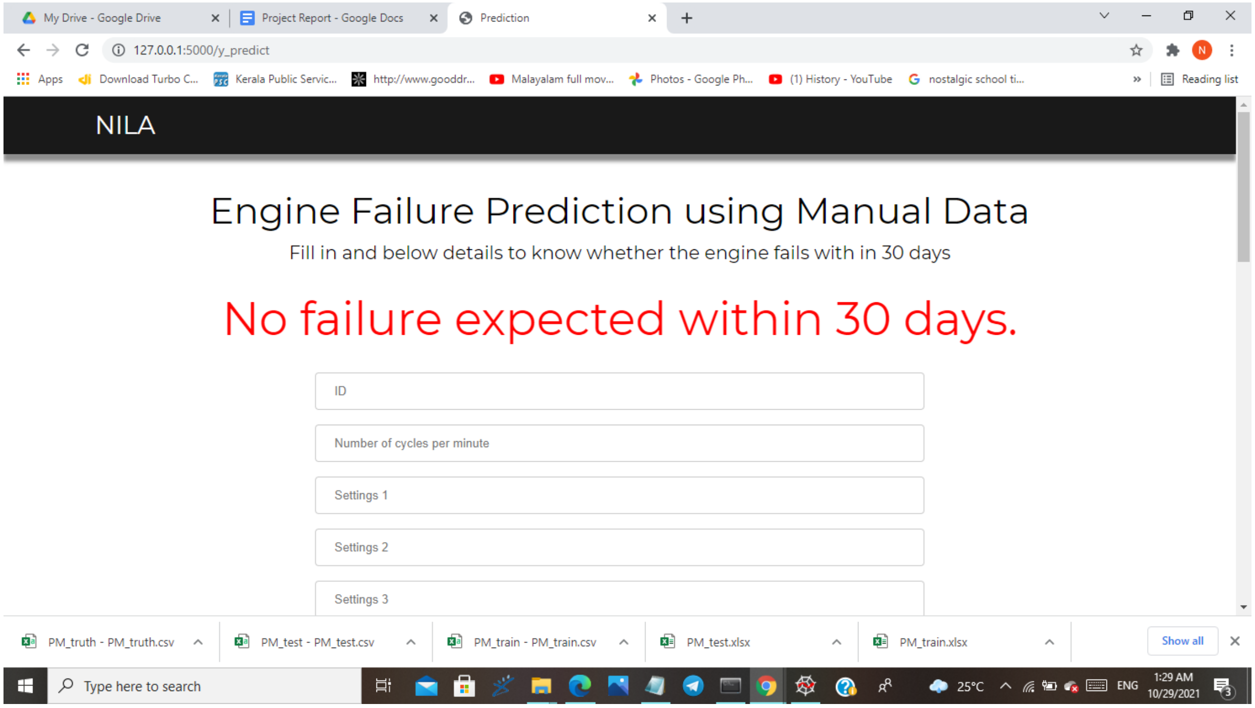
5. FLOWCHART



6.RESULT


My Drive - Google Drive
Project Report - Google Docs
Prediction
127.0.0.1:5000
NILA
Engine Failure Prediction using Manual Data
Fill in and below details to know whether the engine fails with in 30 days
123
10
1
2
3
2
PM_truth - PM_truth.csv
PM_test - PM_test.csv
PM_train - PM_train.csv
PM_test.xlsx
PM_train.xlsx
Show all
Type here to search
25°C
ENG
1:29 AM
10/29/2021


My Drive - Google Drive
Project Report - Google Docs
Prediction
127.0.0.1:5000
NILA
5
8
1
5
7
8
8
Submit
PM_truth - PM_truth.csv
PM_test - PM_test.csv
PM_train - PM_train.csv
PM_test.xlsx
PM_train.xlsx
Show all
Type here to search
25°C
ENG
1:29 AM
10/29/2021



7. ADVANTAGES & DISADVANTAGES

Advantages:

- Predicting the failure prior will save
  - Time
  - Effort
  - money
  - sometimes even lives.
- Prevents unexpected equipment failures

Disadvantages:

- Data can be misinterpreted, leading to false maintenance requests.
- Predictive analysis may not take contextual information into account, such as equipment age or weather.
- It may discourage proactive physical inspection and equipment maintenance.
- Preventative maintenance activities may be triggered by timelines rather than genuine machine condition.

8.APPLICATIONS

Predictive maintenance identifies where small issues are arising in machinery or processes. This can flag minor issues before they become serious and costly. In the past, most businesses were forced to use a ‘run to failure’ model. That is, machinery was deployed until it broke or malfunctioned. Only then could the issue be addressed. Breakdowns were unpredictable, difficult to prevent and often expensive to fix, especially if the failures caused follow-on damage to other parts, or delayed production.

9.CONCLUSION

In this project, by using machine learning algorithm and by installing the sensors and keeping a track of the values and also by exploring aircraft engine’s sensor values over time the relationship between sensor values and changes in sensor values to the historical failures are understood and the failure of an engine is predicted for a threshold number of days .

10.FUTURE SCOPE

By adopting a predictive-maintenance (PdM) strategy, you can mine your critical-asset data and identify anomalies or deviations from their standard performance. Such insights can help you discover and proactively fix issues days, weeks, or even months before they lead to failures. This can help you avoid unplanned downtime, reduce industrial maintenance overspend, and mitigate safety and environmental risks.

10.BIBLIOGRAPHY

- Predictive maintenance of turbofan engines
- A Classification case study on Aircraft Engine Failure

APPENDIX

Source Code

aircraftproject - Jupyter Notebook

localhost:8888/notebooks/aircraftproject.ipynb#

jupyter aircraftproject Last Checkpoint: Last Monday at 11:01 AM (unsaved changes)

In [101]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [102]:

```
dataset_train=pd.read_csv(r"C:\Users\91944\Desktop\Main project\Flask\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','s14','s15','s16','s17','s18','s19','s20','s21']
dataset_train.columns=col_names
print('Shape of train dataset:',dataset_train.shape)
dataset_train.head()
```

Shape of train dataset: (20631, 26)

Out[102]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20
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
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
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
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
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

5 rows × 26 columns

aircraftproject - Jupyter Notebook

localhost:8888/notebooks/aircraftproject.ipynb#

jupyter aircraftproject Last Checkpoint: Last Monday at 11:01 AM (unsaved changes)

In [106]:

```
dataset_test=pd.read_csv('PM_test.txt',sep=' ',header=None).drop([26,27],axis=1)

dataset_test.columns=col_names
print('Shape of test dataset:',dataset_test.shape)
dataset_test.head()
```

Shape of test dataset: (13096, 26)

Out[106]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	...	521.72	2388.03	8125.55	8.4052	0.03	392	2388	100.0	38.86
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	...	522.16	2388.06	8139.62	8.3803	0.03	393	2388	100.0	39.02
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	...	521.97	2388.03	8130.10	8.4441	0.03	393	2388	100.0	39.08
3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	...	521.38	2388.05	8132.90	8.3917	0.03	391	2388	100.0	39.00
4	1	5	0.0014	0.0000	100.0	518.67	642.51	1587.19	1401.92	14.62	...	522.15	2388.03	8129.54	8.4031	0.03	390	2388	100.0	38.99

5 rows × 26 columns

In [107]:

```
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[107]:

	more	id

Out[107]:

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

In [108]:
```
rul = pd.DataFrame(dataset_train.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id','max']
rul.head()
```

Out[108]:

	id	max
0	1	192
1	2	287
2	3	179
3	4	189
4	5	269

In [111]:
```
pm_truth[ 'rtf' ]=pm_truth[ 'more' ] + rul[ 'max' ]
pm_truth.head()
```

Out[111]:

	more	id	rtf
0	112	1	304
1	98	2	385
2	69	3	248
3	82	4	271
4	91	5	360

In [112]:
```
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[112]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s13	s14	s15	s16	s17	s18	s19	s20	s21
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
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
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
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
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

aircraftproject - Jupyter Notebook

localhost:8888/notebooks/aircraftproject.ipynb#

jupyter aircraftproject Last Checkpoint: Last Monday at 11:01 AM (unsaved changes)

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

5 rows × 27 columns

In [113]:
```
dataset_train['ttf'] = dataset_train.groupby(['id'])['cycle'].transform(max)-dataset_train['cycle']
dataset_train.head()
```

Out[113]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	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	...	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	...	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	...	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	...	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	...	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044

5 rows × 27 columns

In [ ]:

In [114]:
```
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)
```

1:48 AM 10/29/2021

aircraftproject - Jupyter Notebook

localhost:8888/notebooks/aircraftproject.ipynb#

jupyter aircraftproject Last Checkpoint: Last Monday at 11:01 AM (unsaved changes)

In [114]:
```
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[114]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s14	s15	s16	s17	s18	s19	s20	s21	ttf	lab
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	
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	
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	
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	
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	

5 rows × 28 columns

In [115]:
```
df_train.drop(['ttf'],axis=1,inplace=True)
df_test.drop(['ttf'],axis=1, inplace=True)
```

In [ ]:

In [ ]:

In [116]:
```
x_train = df_train.iloc[:,:-1].values
```

1:48 AM 10/29/2021

aircraftproject - Jupyter Notebook

localhost:8888/notebooks/aircraftproject.ipynb#

jupyter aircraftproject Last Checkpoint: Last Monday at 11:01 AM (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

In [116]:
```
x_train = df_train.iloc[:,:-1].values
y_train = df_train.iloc[:,-1].values
#x_test = df_train.iloc[:,:-1].values
#y_test = df_train.iloc[:,-1].values
```

In [117]:
```
#model building
```

In [118]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
C:\Users\91944\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

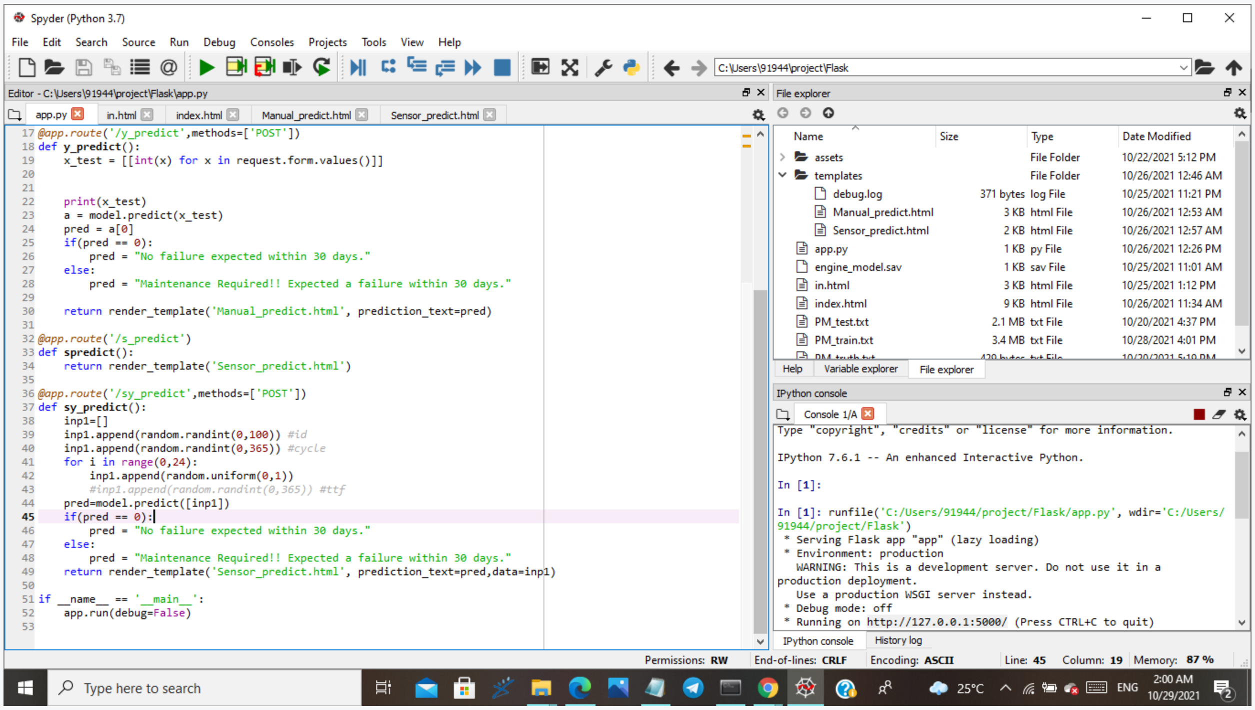
Out[118]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

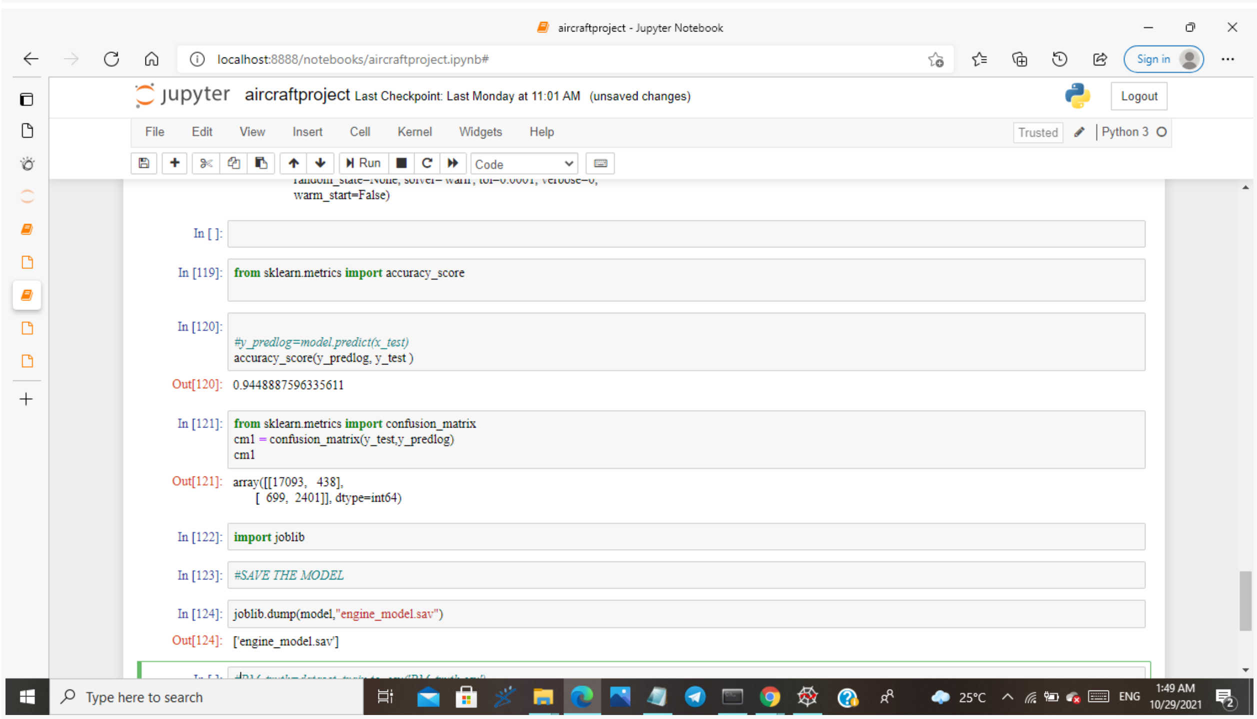
In [ ]:

In [119]:
```
from sklearn.metrics import accuracy_score
```

Type here to search

25°C ENG 1:48 AM 10/29/2021


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\91944\project\Flask
Editor - C:\Users\91944\project\Flask\app.py
app.py in.html index.html Manual_predict.html Sensor_predict.html
@app.route('/y_predict',methods=['POST'])
def y_predict():
    x_test = [[int(x) for x in request.form.values()]]
    print(x_test)
    a = model.predict(x_test)
    pred = a[0]
    if(pred == 0):
        pred = "No failure expected within 30 days."
    else:
        pred = "Maintenance Required!! Expected a failure within 30 days."
    return render_template('Manual_predict.html', prediction_text=pred)
@app.route('/s_predict')
def spredict():
    return render_template('Sensor_predict.html')
@app.route('/sy_predict',methods=['POST'])
def sy_predict():
    inp1=[]
    inp1.append(random.randint(0,100)) #id
    inp1.append(random.randint(0,365)) #cycle
    for i in range(0,24):
        inp1.append(random.uniform(0,1))
        #inp1.append(random.randint(0,365)) #ttf
    pred=model.predict([inp1])
    if(pred == 0):
        pred = "No failure expected within 30 days."
    else:
        pred = "Maintenance Required!! Expected a failure within 30 days."
    return render_template('Sensor_predict.html', prediction_text=pred,data=inp1)
if __name__ == '__main__':
    app.run(debug=False)
File explorer
Name Size Type Date Modified
assets File Folder 10/22/2021 5:12 PM
templates File Folder 10/26/2021 12:46 AM
debug.log 371 bytes log File 10/25/2021 11:21 PM
Manual_predict.html 3 KB html File 10/26/2021 12:53 AM
Sensor_predict.html 2 KB html File 10/26/2021 12:57 AM
app.py 1 KB py File 10/26/2021 12:26 PM
engine_model.sav 1 KB sav File 10/25/2021 11:01 AM
in.html 3 KB html File 10/25/2021 1:12 PM
index.html 9 KB html File 10/26/2021 11:34 AM
PM_test.txt 2.1 MB txt File 10/20/2021 4:37 PM
PM_train.txt 3.4 MB txt File 10/28/2021 4:01 PM
Help Variable explorer File explorer
IPython console
Console 1/A
Type "copyright", "credits" or "license" for more information.
IPython 7.6.1 -- An enhanced Interactive Python.
In [1]:
In [1]: runfile('C:/Users/91944/project/Flask/app.py', wdir='C:/Users/91944/project/Flask')
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 45 Column: 19 Memory: 87 %
Type here to search
25°C ENG 2:00 AM 10/29/2021


aircraftproject - Jupyter Notebook
localhost:8888/notebooks/aircraftproject.ipynb#
jupyter aircraftproject Last Checkpoint: Last Monday at 11:01 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
warm_start=False)
In [ ]:
In [119]: from sklearn.metrics import accuracy_score
In [120]: #y_predlog=model.predict(x_test)
accuracy_score(y_predlog, y_test)
Out[120]: 0.9448887596335611
In [121]: from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test,y_predlog)
cm1
Out[121]: array([[17093, 438],
[699, 2401]], dtype=int64)
In [122]: import joblib
In [123]: #SAVE THE MODEL
In [124]: joblib.dump(model,"engine_model.sav")
Out[124]: ['engine_model.sav']
Type here to search
25°C ENG 1:49 AM 10/29/2021