Machine Learning approach for Predictive Maintenance Aircraft Engine using IBM Watson Studio

1. INTRODUCTION

Overview

Aircraft engines are the most expensive parts of an aircraft, and it is in airlines' interest to keep their power plants in tip-top condition. Apart from the fact that they are rather crucial to actual flight and safety, unscheduled service interruptions due to engine problems can quickly become costly affairs.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 manual sensor data 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.

Purpose

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

2. LITERATURESURVEY

Existing problem

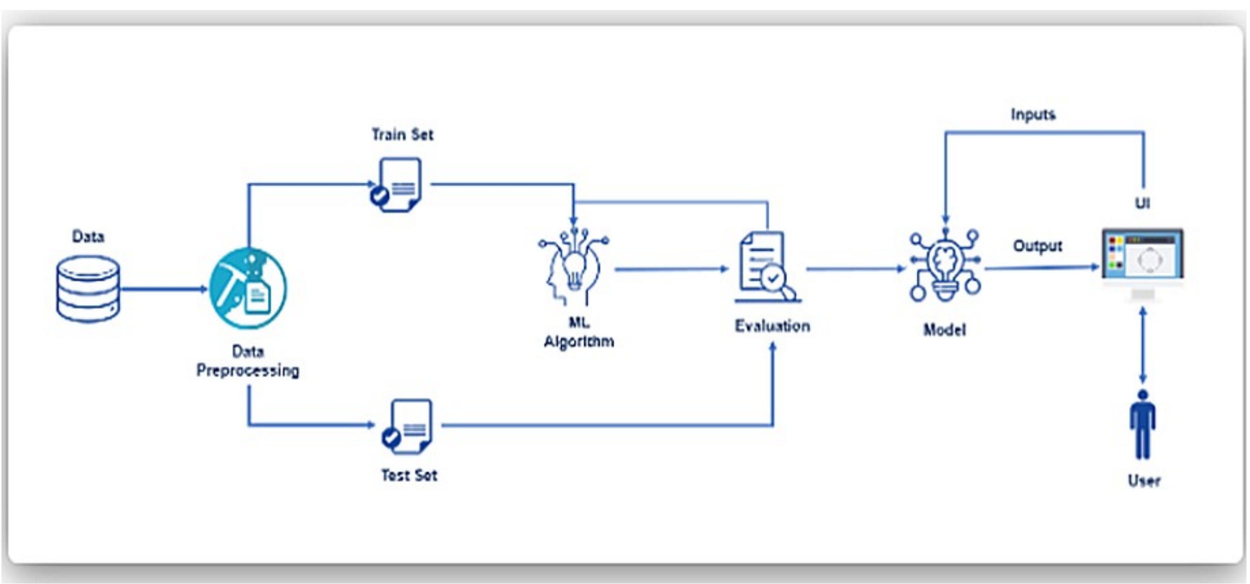
Ideally, pilots and mechanics should work together to make sure the aircraft is operated and maintained properly. As a pilot, you are encouraged to take an active role in maintenance by reviewing inspection results and discussing Airworthiness Directives and Service Bulletins with your mechanic. The existing system is highly risky and needs a lot of time for repair and heavy risk for human lives. we cannot ensure the result from existing system. Maintenance delays are major inconvenience for passengers, and they are serious issue for airlines as well.

Proposed solution

The project aims to predict the failure of an engine by using Machine Learning to save loss of time & money thus improving productivity .today's engines have hundreds of sensors and signals that transmit gigabytes of data for each flight. Planes generate a lot of data that can be used to make such predictions we have access to data like this, we can generate predictions using those data. This is a perfect opportunity to use predictive analytics: modern aircraft generate a wealth of data -- a 787 generates as much as a terabyte of sensor data per flight -- and airlines have extensive records of flight delays and their causes. So it makes sense to look the sensor data from flights that had unexpected maintenance issues, and see if you can find patterns in the data that indicate a likelihood of a maintenance problem, so you can fix any such issues before they become a delay

3. THEORITICALANALYSIS

Block diagram



Hardware / Software designing

Hardware Requirements:
Processor : Intel Core I3
RAM : 4.00 GB
OS : Windows/Linux/MAC

Software Requirements:

Anaconda
Jupyter Notebook
Spyder

4. EXPERIMENTALINVESTIGATIONS

- Download the dataset.

  Import required libraries such as pandas, numpy, sklearn and Datasets are collected.

- Preprocess or clean the data.

  After collecting the datasets we preprocess the data, preprocess involves treating the dataset , preprocessing the dataset , splitting the dataset. we split the data for training , testing purposes and analyze the pre-processed data.

  Handling the null values

  Handling the categorical values if any

  Normalize the data if required.

  Identify the dependent and independent variables.

  Split the dataset into train and test sets.

- Train the machine with pre-processed data using an appropriate machinelearning algorithm.

In this project ,We will be initially considering the Logistic Regression model and fit the data.Here we will be evaluating the model built. We will be using the test set for evaluation. The test set is given to the model for prediction and prediction values are stored in another variable called y_predlog. The actual and predicted values are compared to know the accuracy of the model using the accuracy_score function from sklearn.metrics package. Follow the below steps to find the accuracy of the model. The accuracy for logistic regression in this is .998 . Confusion matrix can be generated to know the true positives.

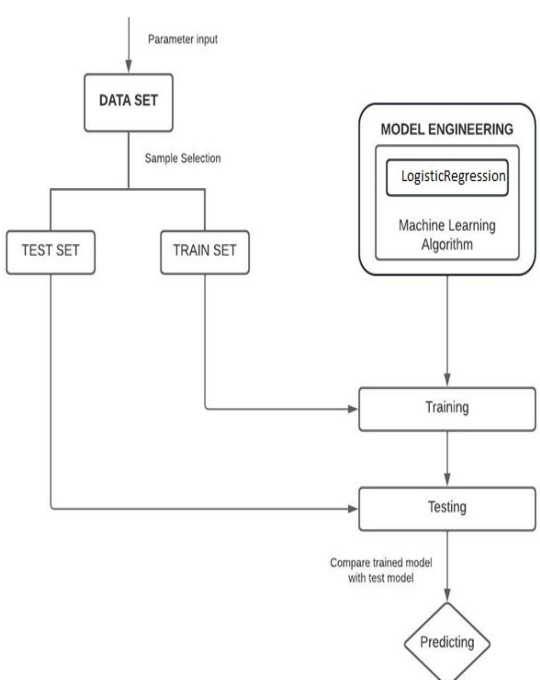
- Save the model and its dependencies.

The finalised model is now to be saved. We will be saving the model as a sav file.

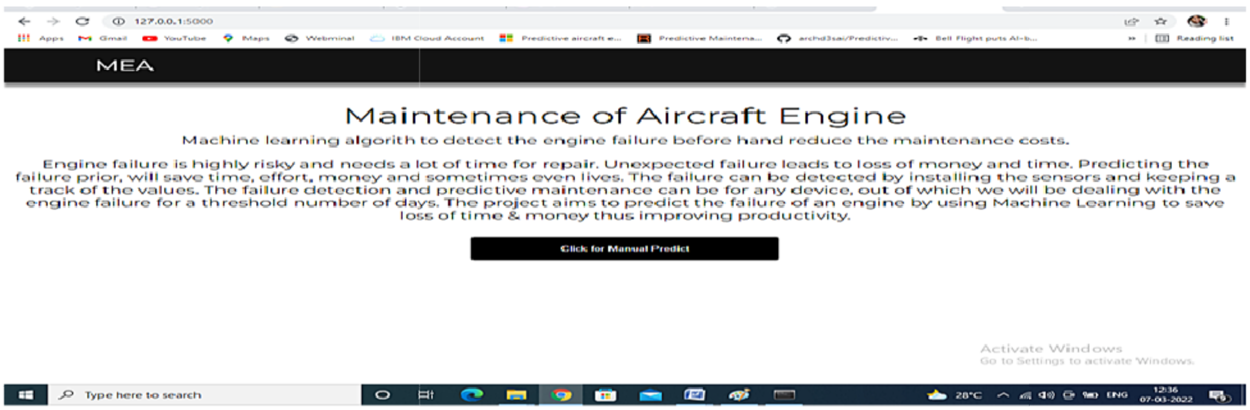
- Build a Web application using flask that integrates with the model built.

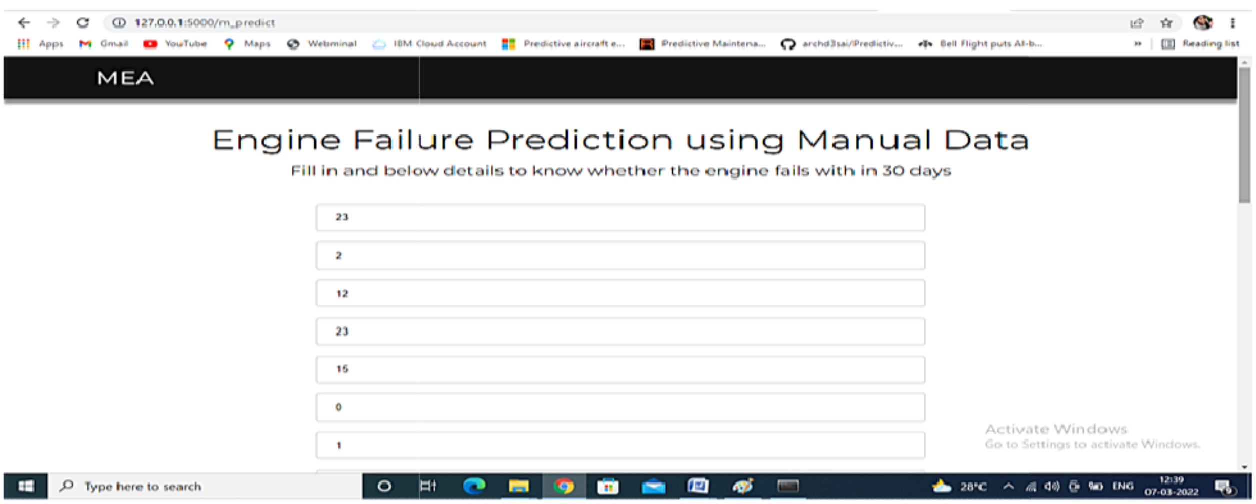
After the model is built, we will be integrating it to a web application where a user can give the sensor data and get to know if the engine fails in the next 30 days or we can even generate the random values and know the prediction for those random data.

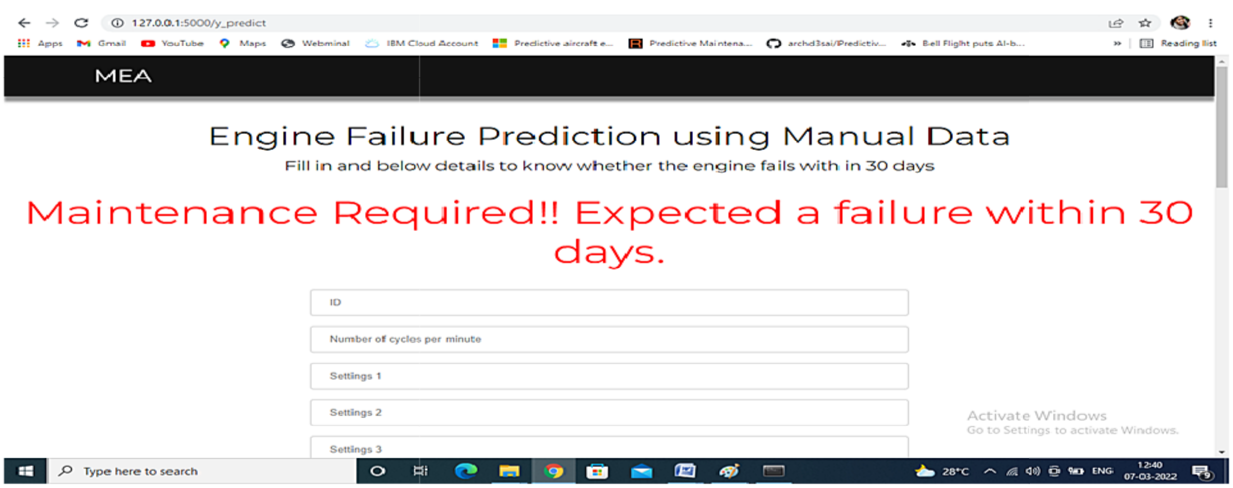
5. FLOWCHART



RESULT


127.0.0.1:5000
MEA
Maintenance of Aircraft Engine
Machine learning algorith to detect the engine failure before hand reduce the maintenance costs.
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 days. The project aims to predict the failure of an engine by using Machine Learning to save loss of time & money thus improving productivity.
Click for Manual Predict
Activate Windows
Go to Settings to activate Windows.
Type here to search


127.0.0.1:5000/m_predict
MEA
Engine Failure Prediction using Manual Data
Fill in and below details to know whether the engine fails with in 30 days
23
2
12
23
15
0
1
Activate Windows
Go to Settings to activate Windows.
Type here to search


127.0.0.1:5000/y_predict
MEA
Engine Failure Prediction using Manual Data
Fill in and below details to know whether the engine fails with in 30 days
Maintenance Required!! Expected a failure within 30 days.
ID
Number of cycles per minute
Settings 1
Settings 2
Settings 3
Activate Windows
Go to Settings to activate Windows.
Type here to search

7. ADVANTAGES &DISADVANTAGES

ADVANTAGES

Save time ,effort and money-unexpected failure leads to loss of money and time. Predict the failure of an engine by using machine learning to save loss of time and money thus improving productivity

3-5% increased machine useful life: Since predictive maintenance reduces machine breakdowns and ensures operation in optimum settings

Reduced environmental impact: As machines remain useful for longer periods and as their efficiency increase with advanced analytics, companies will waste less natural resources. Predictive maintenance is one of the few initiatives that both help companies bottom line and their corporate social responsibility goals.

Advanced analytics: Setting up predictive maintenance involves collecting sensor data from diverse machinery. Once that data starts to be automatically collected, analysts have a trove of information ready for analysis. This data can be used to identify parameter and process optimization opportunities.

DISADVANTAGES

Data security: In terms of predictive maintenance, it is critical to guarantee that equipment performance data is not subject to access by outside parties, and that outside parties are not able to control predictive maintenance systems. At a more baseline level, it also remains important to protect information such as customer data.

Additional costs

Given the complex nature of predictive maintenance, plant personnel needs to be trained on using the equipment and interpreting the analytics. It also involves investment in maintenance tools and systems. Tersely, condition monitoring has a high upfront cost.

8. APPLICATIONS

Aircraft engine maintenance is a step-by-step process similar to a person's health check-up. It consists of washing and drying jet engine parts, exterior and interior visual inspections, a dismantling of the engine, the repair and replacement of any parts, and then the re-assembling and testing of the engine.

Machine Learning play a key part in security as they typically use predictive analysis to improve services and performance, but also to detect anomalies, fraud, understand consumer behavior and enhance data security. 4. ML Algorithms are also known for it's recommendation algorithms like in Facebook or YouTube where similar content will be suggested to engage the user

9. CONCLUSION

In this work, Predictive Maintenance aircraft engine using machine learning, Logistic Regression are considered for determining the status of maintenance of aircraft engine applications. In Logistics Regression has done quite a good performance than expected with 0.99832accuracy . This leads to conclusion that how much important is feature selection and feature transformation is. Our results showed that the most predictive features are EMPLOYER SUCCESS RATE and PREVAILING WAGE.

10. FUTURESCOPE

By adopting a predictive-maintenance (PDM) strategy, you can mine your critical asset data and identify anomalies or deviations from their standard performance. In this paper we are just predicting whether engine fail or not within 30 days. 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 f maintenance overspend, and mitigate safety and environmental risk.

11. BIBILOGRAPHY.

Hermawan, A. P., Kim, D-S., and Lee, M-J. (2020). Predictive Maintenance of Aircraft Engine using Deep

Learning Technique. In 2020 International Conference on Information and Communication Technology

Convergence (ICTC), pp. 1296-1298.

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

Amruthnath, N., and Gupta, T. (2018). A research study on unsupervised machine learning algorithms for

early fault detection in predictive maintenance. In 2018 5th International Conference on Industrial

Engineering and Applications (ICIEA). pp. 355-361.

APPENDIX

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

        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, LSTM, Activation
        from tensorflow.keras.callbacks import EarlyStopping

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

```
In [2]: import os, types
        import pandas as pd
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        client_f50763f7c1f14adfb97b4fbbbc23830b = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='NsjyYpsqxATF6q6gOdgbG0yVNK3RiMUYtyUJDzEyFrIG',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

        streaming_body_1 = client_f50763f7c1f14adfb97b4fbbbc23830b.get_object(Bucket='machinelearningapproachforpredict-donotdelete-pr-d
        9duqt0m66f6dx', Key='PM_train.txt')['Body']

        # Your data file was loaded into a botocore.response.StreamingBody object.
        # Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
        # ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
        # pandas documentation: http://pandas.pydata.org/
        dataset_train=pd.read_csv(streaming_body_1,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','s1
        3','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[2]:

	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	2
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
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	2
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	2
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	2

5 rows × 26 columns

```
In [3]: streaming_body_3 = client_f50763f7c1f14adfb97b4fbbbc23830b.get_object(Bucket='machinelearningapproachforpredict-donotdelete-pr-d
        9duqt0m66f6dx', Key='PM_test.txt')['Body']

        # Your data file was loaded into a botocore.response.StreamingBody object.
        # Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
        # ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
        # pandas documentation: http://pandas.pydata.org/
        dataset_test=pd.read_csv(streaming_body_3,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[3]:

	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	2
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
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	2
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	2
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	2

5 rows × 26 columns

IBM Watson Studio | All | Search | Buy | ATHIRA P J's Account

Projects / Aircraftengine / Aircraft

In [4]:
```
streaming_body_5 = client_f50763f7c1f14adfb97b4fbbbc23830b.get_object(Bucket='machinelearningapproachforpredict-donotdelete-pr-d9duqt0m66f6dx', Key='PM_truth.txt')['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
pm_truth=pd.read_csv(streaming_body_5,sep=' ',header=None).drop([1],axis=1)
pm_truth.columns=['more']
pm_truth['id']=pm_truth.index+1
pm_truth.head()
```

Out[4]:

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

In [5]:
```
pm_truth.shape
```

Out[5]: (100, 2)

In [6]:
```
rul = pd.DataFrame(dataset_test.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id', 'max']
```

Activate Windows
Go to Settings to activate Windows

Type here to search | 28°C Partly cloudy | ENG | 11:56 07-03-2022

IBM Watson Studio | All | Search | Buy | ATHIRA P J's Account

Projects / Aircraftengine / Aircraft

In [5]:
```
pm_truth.shape
```

Out[5]: (100, 2)

In [6]:
```
rul = pd.DataFrame(dataset_test.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id', 'max']
rul.head()
```

Out[6]:

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

In [7]:
```
rul.shape
```

Out[7]: (100, 2)

In [8]:
```
pm_truth['rtf']=pm_truth['more']+rul['max']
pm_truth.head()
```

Activate Windows
Go to Settings to activate Windows

Out[8]:

IBM Watson Studio | All | Search | Buy | ATHIRA P J's Account

Projects / Aircraftengine / Aircraft

Out[8]:

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

In [9]:
```
pm_truth.shape
```

Out[9]: (100, 3)

In [10]:
```
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[10]:

	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

In [11]:
```
dataset_test.shape
```
Out[11]: (13096, 27)

In [12]:
```
dataset_train['ttf'] = dataset_train.groupby(['id'])['cycle'].transform(max)-dataset_train['cycle']
dataset_train.head()
```
Out[12]:

	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 [13]:
```
dataset_train.shape
```
Out[13]: (20631, 27)

In [14]:
```
dataset_train['ttf'].value_counts()
```
Out[14]:
```
0      100
87     100
118    100
```



```
346      1
347      1
348      1
351      1
Name: ttf, Length: 362, dtype: int64
```

In [15]:
```
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[15]:

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

5 rows × 28 columns

In [16]:
```
df_train['label_bc'].value_counts()
```


In [16]:
```
df_train['label_bc'].value_counts()
```
Out[16]:
```
0    17531
1     3100
Name: label_bc, dtype: int64
```

In [17]:
```
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 [18]:
```
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 [19]:
```
df_train.head()
```
Out[19]:

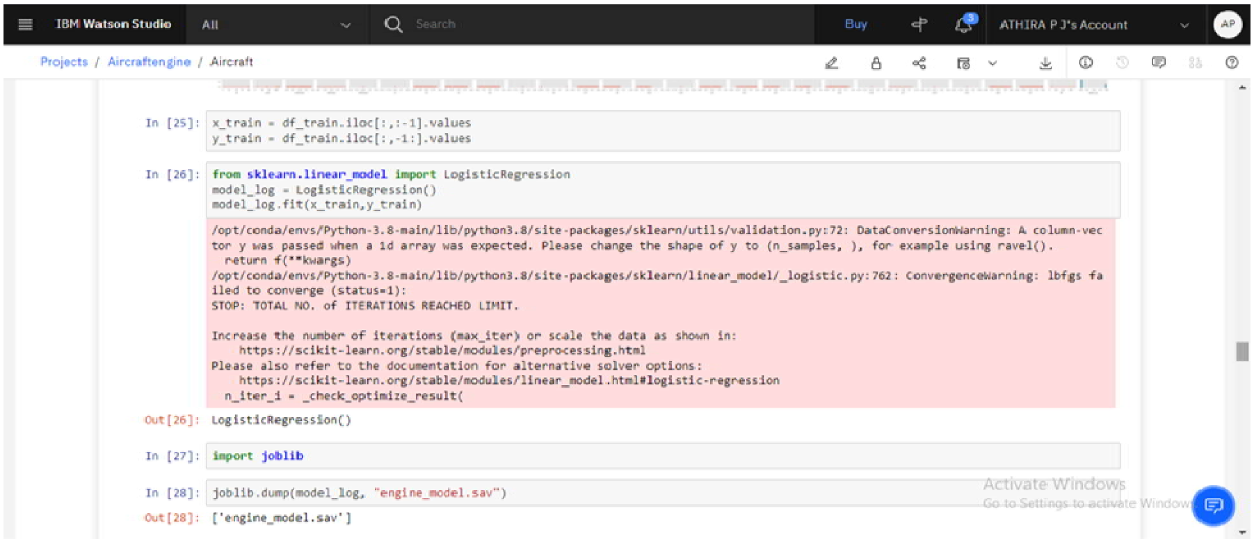
	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	0.459770	0.166667	0.0	0.0	0.183735	0.406802	0.309757	0.0	...	0.199608	0.363986	0.0	0.333333	0.0	0.0	0.713178	0.724662
1	1	2	0.609195	0.250000	0.0	0.0	0.283133	0.453019	0.352633	0.0	...	0.162813	0.411312	0.0	0.333333	0.0	0.0	0.666667	0.731014
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
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
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

5 rows × 28 columns

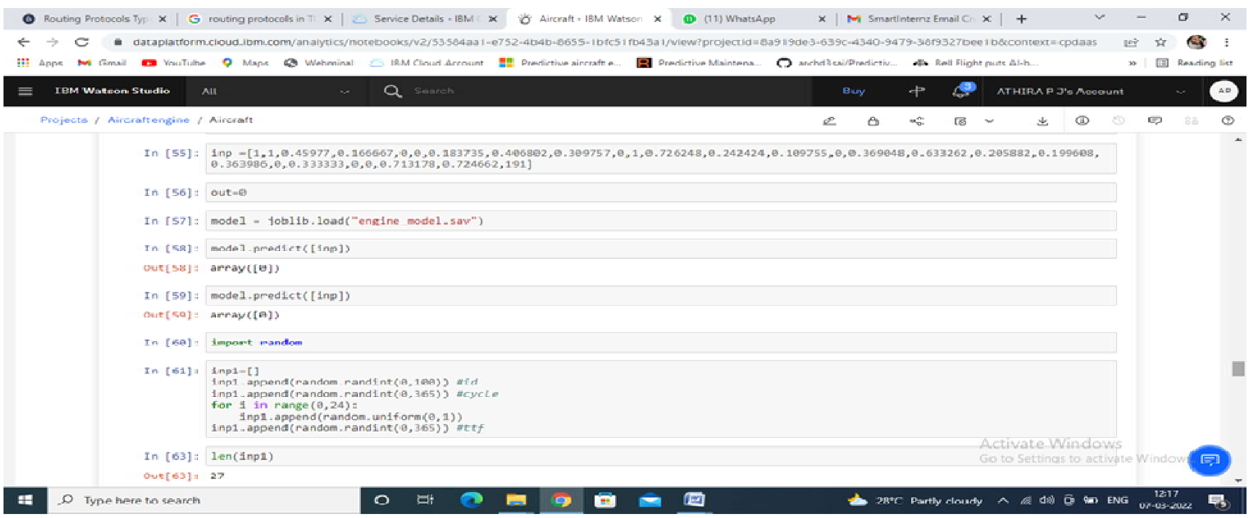
```
In [20]: df_train.info()

<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
```

```
 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 [21]: df_train['ttf'].min()
Out[21]: 0

In [22]: df_train['ttf'].max()
Out[22]: 361

In [23]: df_train.iloc[0,:]
Out[23]: 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
```

```
In [24]: import seaborn as sns
         sns.pairplot(df_train)
Out[24]: <seaborn.axisgrid.PairGrid at 0x7f7438048cd0>
```



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

In [26]: from sklearn.linear_model import LogisticRegression
         model_log = LogisticRegression()
         model_log.fit(x_train,y_train)

/opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages/sklearn/utils/validation.py:72: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
/opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs fa
iled to converge (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[26]: LogisticRegression()

In [27]: import joblib

In [28]: joblib.dump(model_log, "engine_model.sav")

Out[28]: ['engine_model.sav']
```



```
In [55]: inp =[1,1,0.45977,0.166667,0,0,0.183735,0.406802,0.309757,0,1,0.726248,0.242424,0.109755,0,0.369048,0.633262,0.205882,0.199608,
         0.363986,0,0.333333,0,0,0.713178,0.724662,191]

In [56]: out=0

In [57]: model = joblib.load("engine_model.sav")

In [58]: model.predict([inp])

Out[58]: array([0])

In [59]: model.predict([inp])

Out[59]: array([0])

In [60]: import random

In [61]: 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

In [63]: len(inp1)

Out[63]: 27
```

```
In [29]: x_test = df_test.iloc[:,:-1].values
         y_test = df_test.iloc[:,-1:].values

In [30]: y_predlog = model_log.predict(x_test)

In [31]: from sklearn.metrics import accuracy_score
         accuracy_score(y_predlog,y_test)

Out[31]: 0.9993127672571778

In [52]: df_test['label_bc'].value_counts()

Out[52]: 0    12764
         1      332
         Name: label_bc, dtype: int64

In [53]: from sklearn.metrics import confusion_matrix
         cm1 = confusion_matrix(y_test,y_predlog)
         cm1

Out[53]: array([[12763,     1],
                [    8,   324]])

In [54]: import numpy as np
         from sklearn.linear_model import LogisticRegression
         import joblib
```

Projects / Aircraftengine / Aircraft

```
In [64]: model.predict([inp1])
Out[64]: array([0])

In [65]: !pip install ibm_watson_machine_learning
Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.189)
Requirement already satisfied: pandas<1.4.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.3.4)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.26.0)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (21.3)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (4.8.2)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2021.10.8)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (0.10.0)
```

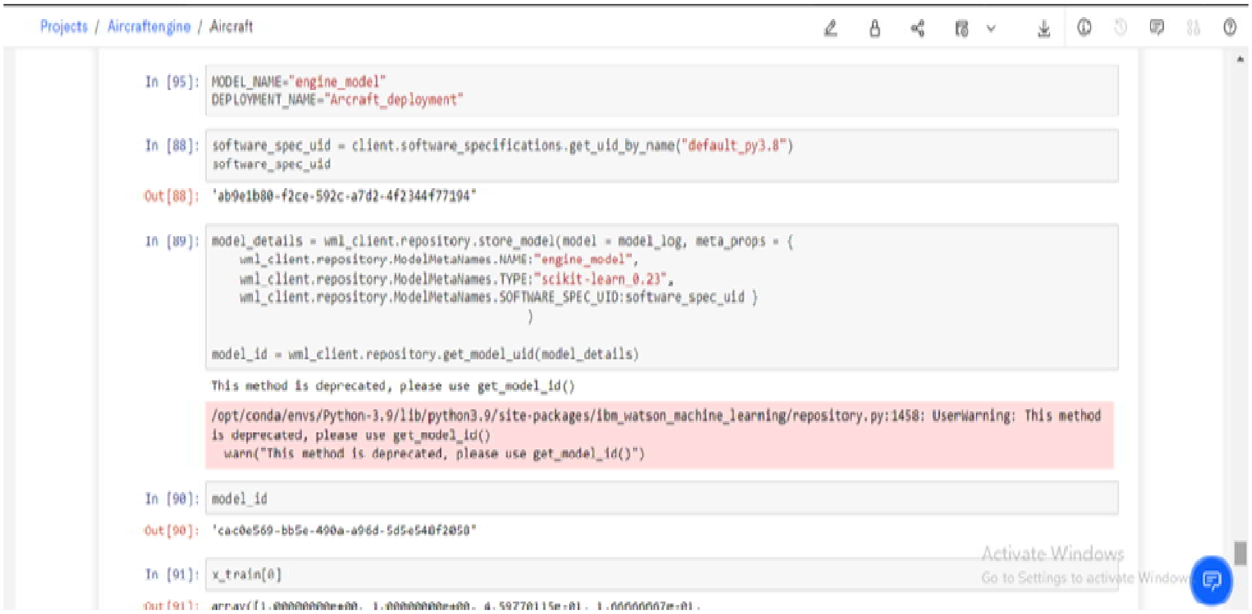
Activate Windows
Go to Settings to activate Windows.

IBM Watson Studio | All | Search | Buy | ATHIRA P J's Account | AP

Projects / Aircraftengine / Aircraft

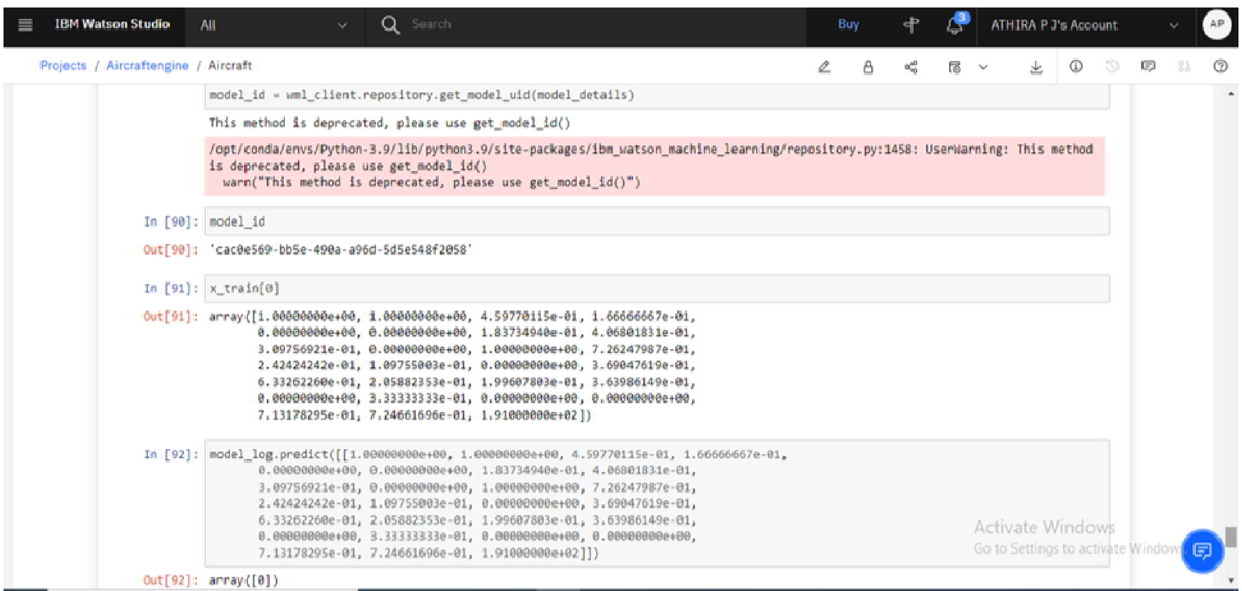
```
In [66]: from ibm_watson_machine_learning import APIClient
         wml_credentials = {
                 "url":"https://us-south.ml.cloud.ibm.com",
                 "apikey":"jqja7xuBXX5cTYoUhTtd_g3PTUQppwhOF0IzQ5eixeBl"
         }
         client = APIClient(wml_credentials)

In [ ]:

In [78]: wml_client = APIClient(wml_credentials)
         wml_client.spaces.list()

Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  -------------  ------------------------
ID                                    NAME           CREATED
1682ef0d-1b09-40e5-9f18-474561193aae  Aircraft_space  2022-03-05T10:27:35.561Z
------------------------------------  -------------  ------------------------

In [84]: SPACE_ID = "1682ef0d-1b09-40e5-9f18-474561193aae"

In [86]: wml_client.set.default_space(SPACE_ID)
Out[86]: 'SUCCESS'

In [87]: wml_client.software_specifications.list()

-----------------------------  ------------------------------------  ----
NAME                           ASSET_ID                              TYPE
```

Activate Windows
Go to Settings to activate Windows.



Projects / Aircraftengine / Aircraft

```
In [95]: MODEL_NAME="engine_model"
         DEPLOYMENT_NAME="Arcraft_deployment"

In [88]: software_spec_uid = client.software_specifications.get_uid_by_name("default_py3.8")
         software_spec_uid

Out[88]: 'ab9e1b80-f2ce-592c-a7d2-4f2344f77194'

In [89]: model_details = wml_client.repository.store_model(model = model_log, meta_props = {
             wml_client.repository.ModelMetaNames.NAME:"engine_model",
             wml_client.repository.ModelMetaNames.TYPE:"scikit-learn_0.23",
             wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid }
                                                                              )

         model_id = wml_client.repository.get_model_uid(model_details)

         This method is deprecated, please use get_model_id()

         /opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_machine_learning/repository.py:1458: UserWarning: This method
         is deprecated, please use get_model_id()
           warn("This method is deprecated, please use get_model_id()")

In [90]: model_id

Out[90]: 'cac0e569-bb5e-490a-a96d-5d5e548f2058'

In [91]: x_train[0]
```



IBM Watson Studio | All | Search | Buy | ATHIRA P J's Account | AP

Projects / Aircraftengine / Aircraft

```
         model_id = wml_client.repository.get_model_uid(model_details)

         This method is deprecated, please use get_model_id()

         /opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_machine_learning/repository.py:1458: UserWarning: This method
         is deprecated, please use get_model_id()
           warn("This method is deprecated, please use get_model_id()")

In [90]: model_id

Out[90]: 'cac0e569-bb5e-490a-a96d-5d5e548f2058'

In [91]: x_train[0]

Out[91]: array([1.00000000e+00, 1.00000000e+00, 4.59770115e-01, 1.66666667e-01,
                0.00000000e+00, 0.00000000e+00, 1.83734940e-01, 4.06801831e-01,
                3.09756921e-01, 0.00000000e+00, 1.00000000e+00, 7.26247987e-01,
                2.42424242e-01, 1.09755003e-01, 0.00000000e+00, 3.69047619e-01,
                6.33262260e-01, 2.05882353e-01, 1.99607803e-01, 3.63986149e-01,
                0.00000000e+00, 3.33333333e-01, 0.00000000e+00, 0.00000000e+00,
                7.13178295e-01, 7.24661696e-01, 1.91000000e+02])

In [92]: model_log.predict([[1.00000000e+00, 1.00000000e+00, 4.59770115e-01, 1.66666667e-01,
                0.00000000e+00, 0.00000000e+00, 1.83734940e-01, 4.06801831e-01,
                3.09756921e-01, 0.00000000e+00, 1.00000000e+00, 7.26247987e-01,
                2.42424242e-01, 1.09755003e-01, 0.00000000e+00, 3.69047619e-01,
                6.33262260e-01, 2.05882353e-01, 1.99607803e-01, 3.63986149e-01,
                0.00000000e+00, 3.33333333e-01, 0.00000000e+00, 0.00000000e+00,
                7.13178295e-01, 7.24661696e-01, 1.91000000e+02]])

Out[92]: array([0])
```