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

Team Members

1. Ashish Santoshkumar Nair 19BCG10028
2. Satvik Shetty 19BAI10051
3. Sairam Pimple 19BCY10174
4. Ranjan Kudesia 19BCG10037

1. Introduction

1.1 Overview

The project uses a Machine Learning Approach for predictive Maintenance Aircraft Engine Using IBM Watson Studio.

For this project a combination of Python, Data Analysis, Exploratory Data Analysis, Data Preprocessing Techniques, Classification and Regression Algorithms.

1.2 Purpose

This project can be used to predict aircraft engine failures before it happens. This can save a lot of lives, money and time.

Finding out the possibility of a system failure can be achieved with this project.

2. Literature Survey

2.1 Existing Problem:

An engine failure occurs when an engine unexpectedly stops producing power due to a malfunction other than fuel exhaustion. It often applies for aircraft, but other turbine engines can fail, like ground-based turbines used in power plants or combined diesel and gas vessels and vehicles.

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.

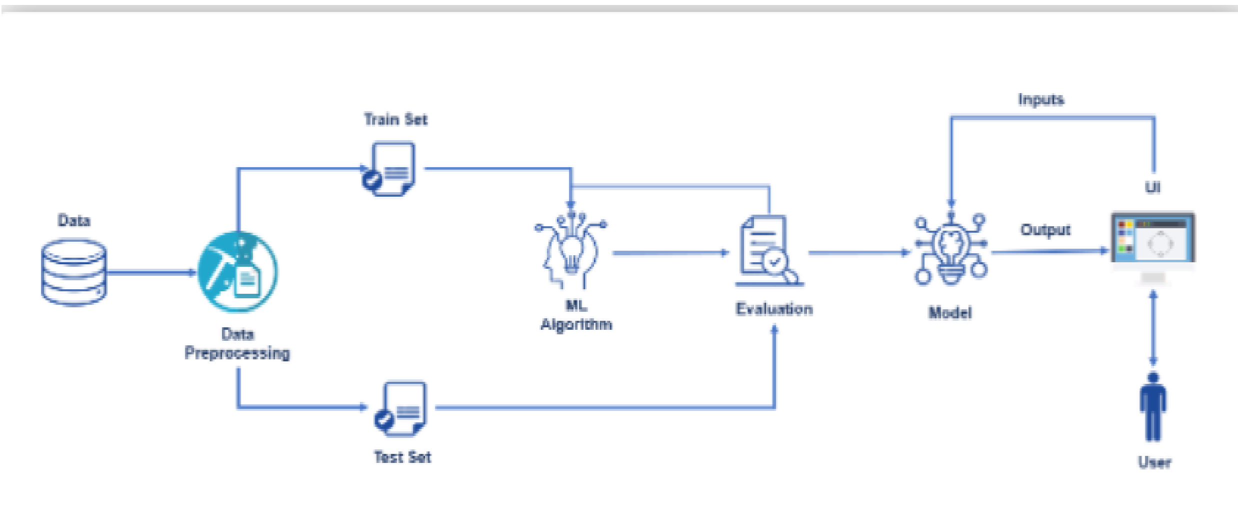
2.2 Proposed Solution:

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

3. Theoretical Analysis

3.1 Block Diagram

The below diagram is the technical architecture of the whole project



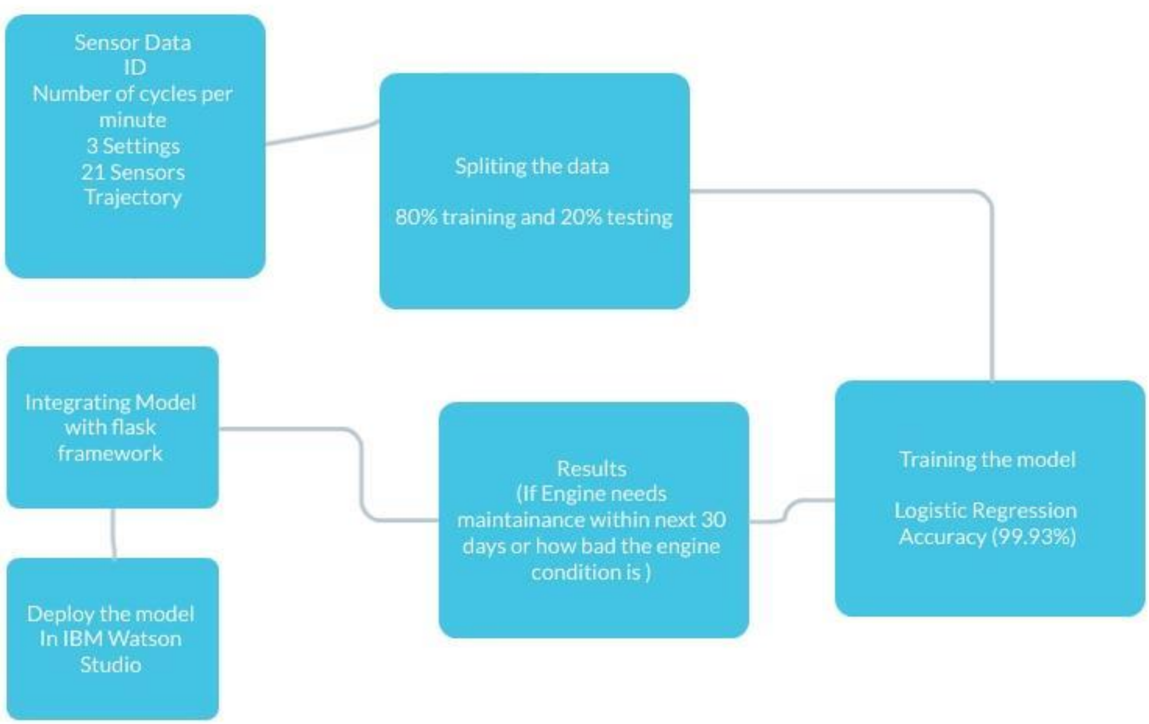
3.2 Hardware/Software requirements:

1) Jupyter Notebook
2) Spyder
3) Visual studios
4) IBM Cloud
    i) IBM Watson studios
5) Flask
6) Packages:
    a) Pandas
    b) numpy
    c) scikit -learn
    d) matplotlib & seaborn
    e) Pickle

4. Experimental Investigations:

1) Here, we are considering a dataset which has operational setting data and the data came from different sensors of an engine
2) The dataset contains 20631 data points inside the train dataset and we have 3 different datasets for train, test and truth.
3) We first built a model in a jupyter notebook.
4) The data is split into training and testing set in 8:2 ratio.
5) The model is trained using Logistic Regression.
6) Logistic Regression yields the most accurate result with around 99.93 percent accuracy.
7) Then we integrate the machine learning model with the flask to build the web application.
8) The model is then deployed using IBM Watson Studio.

5. Flowchart:



6. Result

Once the proposed– model is implemented, you will be welcomed by a UI where the input received from 21 different engine sensors and 3 different settings can be either manually typed in and see if the engine needs a maintenance or not or we also have automatic system from which data from all the sensors can be collected which are available on our dataset and predict the condition of the engine and see if the engine needs a maintenance or not.

7. Advantages and Disadvantages

7.1 Advantages:

1. Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software’s; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML Algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data, you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multidimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multidimensional and multi-variety, and they can do this in dynamic or uncertain environments.

7.2 Disadvantages:

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

8. Applications

This solution can be applied anywhere there is a use of an aircraft engine. Commercial planes, passenger planes or even fighter planes-this prediction system can be used. Its application depends on the fact that the machine uses an aircraft engine.

9. Conclusion

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

This solution helps save so much time and money by simply just entering the values we get from the different sensors which are already in place for a commercial plane and this data simply putting in our website can give you a 99.93 percent accuracy about the condition of your aircraft engine. Whether the engine is in good condition or not or if it needs a maintenance in the next 30 days to get the engine back in good condition and this can help save lives of countless people by preventing any engine malfunction midflight.

10. Future Scope:

Currently our model is using sensor data received from airplanes engine and training the model using these datasets to predict if the engine is in good condition or not. We have to manually enter all the sensor data received from all these sensors in our website for manual prediction. Although this is very accurate and trustworthy, in future we can even make it better by configuring all the different sensors in such a way that all the data is directly sent to our website and we can monitor the engine constantly at particular intervals of time and we will immediately come to know if some problem is there with the engine and get it fixed as soon as possible.

11. Bibliography:

https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/

https://analyticsindiamag.com/machine-learning-for-predictive-maintenance-key-approaches-techniques-to-consider/

https://ieeexplore.ieee.org/document/9289466

https://aerospace.honeywell.com/us/en/learn/about-us/blogs/moving-beyond-the-hype-of-predictive-maintenance

https://www.infoq.com/articles/machine-learning-techniques-predictive-maintenance/

Appendix

a. Source Code

In [80]:
```
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 [81]:
```
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.

if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
    endpoint_6d7dd7cba2f840e2914ab3d8acae12bf = 'https://s3.us.cloud-object-storage.appdomain.cloud'
else:
    endpoint_6d7dd7cba2f840e2914ab3d8acae12bf = 'https://s3.private.us.cloud-object-storage.appdomain.cloud'

client_6d7dd7cba2f840e2914ab3d8acae12bf = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='xpdaDRAhmF4tEIgAO9Pn-N8AZEJvF2mp9lBIheRUXfrq',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_6d7dd7cba2f840e2914ab3d8acae12bf)

streaming_body_5 = client_6d7dd7cba2f840e2914ab3d8acae12bf.get_object(Bucket='smartinternzprojectdeployment-donotdelete-pr-jhhfs

# 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_5,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()
```

In [83]:
```
streaming_body_6 = client_6d7dd7cba2f840e2914ab3d8acae12bf.get_object(Bucket='smartinternzprojectdeployment-donotdelete-pr-jhhfs

# 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_6,sep=' ',header=None).drop([26,27],axis=1)
dataset_test.columns=col_names
#dataset_test.head()
print('Shape of Test dataset: ',dataset_test.shape)
dataset_test.head()
```

```
Shape of Test dataset:  (13096, 26)
```

Out[83]:

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	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	...	521.72	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	...	522.16	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	...	521.97	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	...	521.38	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	...	522.15	2388.03	8129.54	8.4031	0.03	390	2388	100.0	38.99	23.4130

In [84]:

```
streaming_body_7 = client_6d7dd7cba2f840e2914ab3d8acae12bf.get_object(Bucket='smartinternzprojectdeployment-donotdelete-pr-jhhfs

# 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_7,sep=' ',header=None).drop([1],axis=1)
pm_truth.columns=['more']
pm_truth['id']=pm_truth.index+1
pm_truth.head()
```

Out[84]:

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

In [85]:

```
pm_truth.shape
```

Out[85]:

```
(100, 2)
```

In [86]:

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

Out[86]:

	id	max
0	1	31
1	2	49

In [95]:

```
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[95]:

	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

5 rows × 28 columns

In [96]:

```
df_train['label_bc'].value_counts()
```

Out[96]:

```
0    17531
1     3100
Name: label_bc, dtype: int64
```

In [97]:

```
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 [98]:

```
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 [105]:

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

In [106]:

```
from sklearn.linear_model import LogisticRegression
model_log = LogisticRegression()
model_log.fit(x_train,y_train)
```

Out[106]:

```
LogisticRegression()
```

In [107]:
```python
import joblib
```

In [108]:
```python
joblib.dump(model_log, "engine_model.sav")
```

Out[108]:
```
['engine_model.sav']
```

In [109]:
```python
x_test = df_test.iloc[:,:-1].values
y_test = df_test.iloc[:,-1:].values
```

In [110]:
```python
y_predlog = model_log.predict(x_test)
```

In [111]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_predlog,y_test)
```

Out[111]:
```
0.9993127672571778
```

In [112]:
```python
df_test['label_bc'].value_counts()
```

Out[112]:
```
0    12764
1      332
Name: label_bc, dtype: int64
```

In [113]:
```python
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test,y_predlog)
cm1
```

Out[113]:
```
array([[12763,     1],
       [    8,   324]])
```

In [123]:
```python
from ibm_watson_machine_learning import APIClient
wml_credentials = {
        "url":"https://us-south.ml.cloud.ibm.com",
        "apikey":"lr0H_CgBPBvkI7tnG0zpVKqTkNYwFnTOMRMfGM6GsF_f"
}
client = APIClient(wml_credentials)
```

In [127]:
```python
def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()

    return(next(item for item in space['resources'] if item['entity']['name'] == space_name)['metadata']['id'])
```

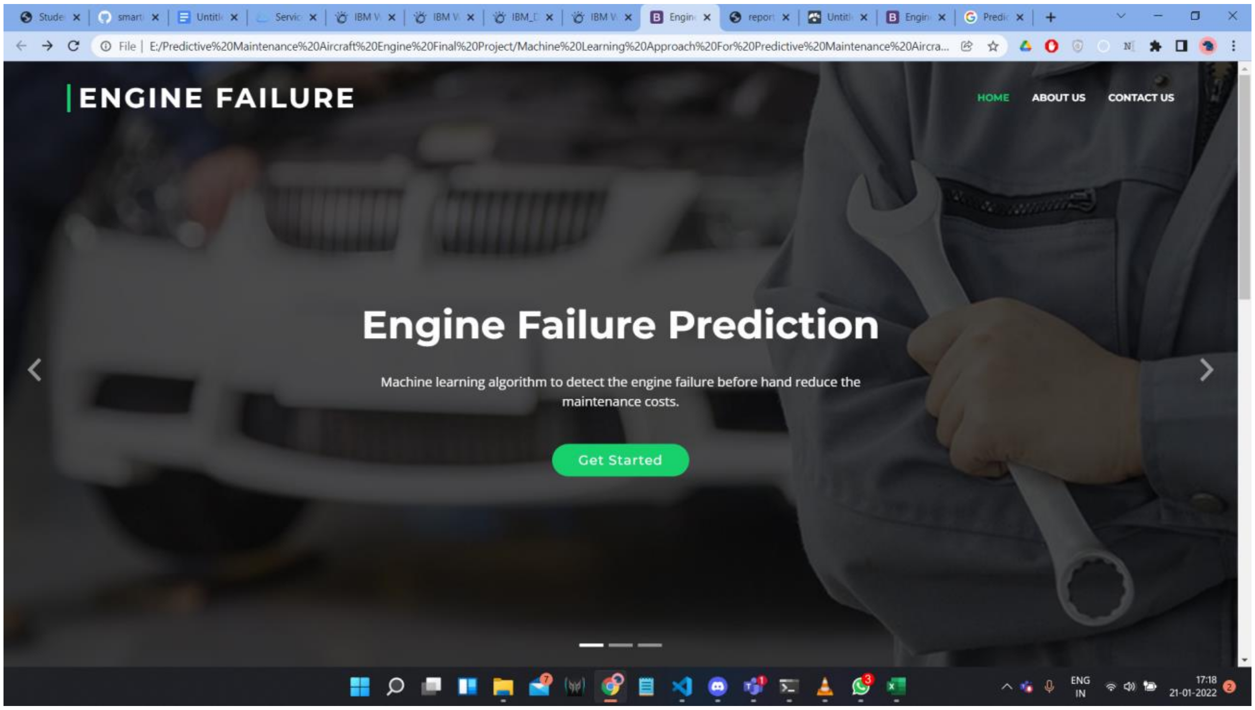
In [128]:
```python
space_uid = guid_from_space_name(client,'NewDeploymentSpace')
print("Space UID = " + space_uid)
```

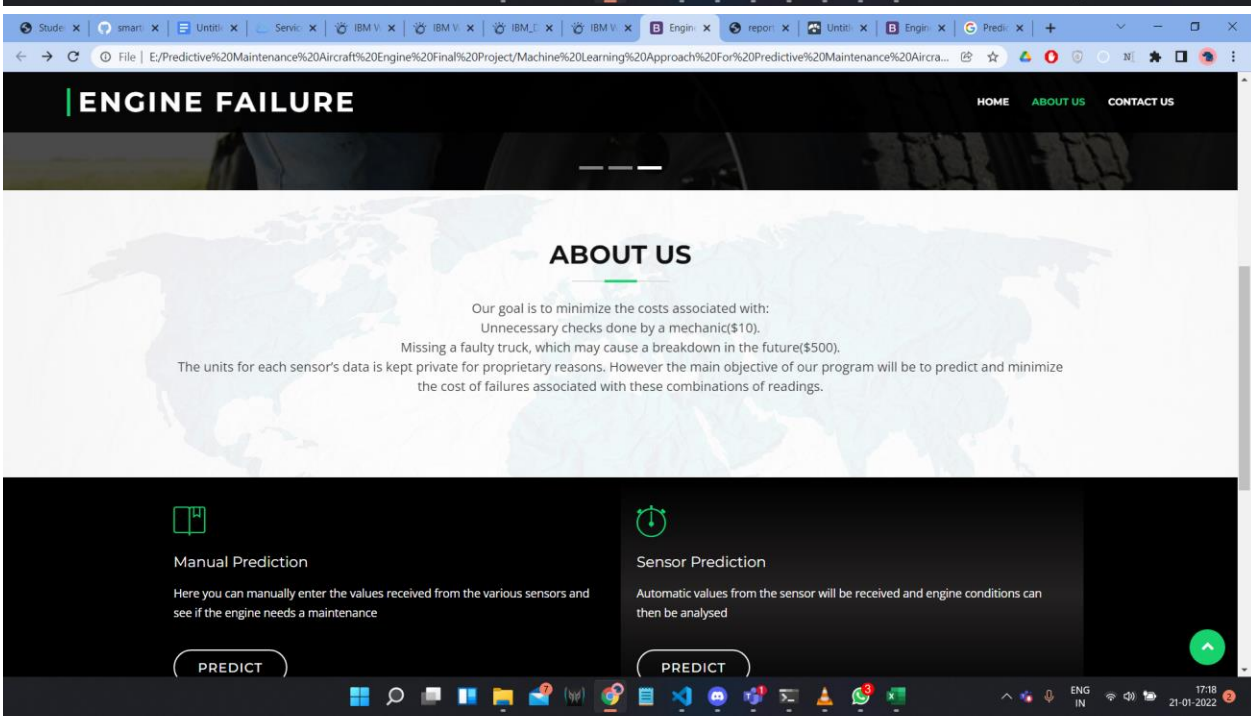
```
Space UID = a75f11cb-aebb-4db3-a829-19eabbb674de
```

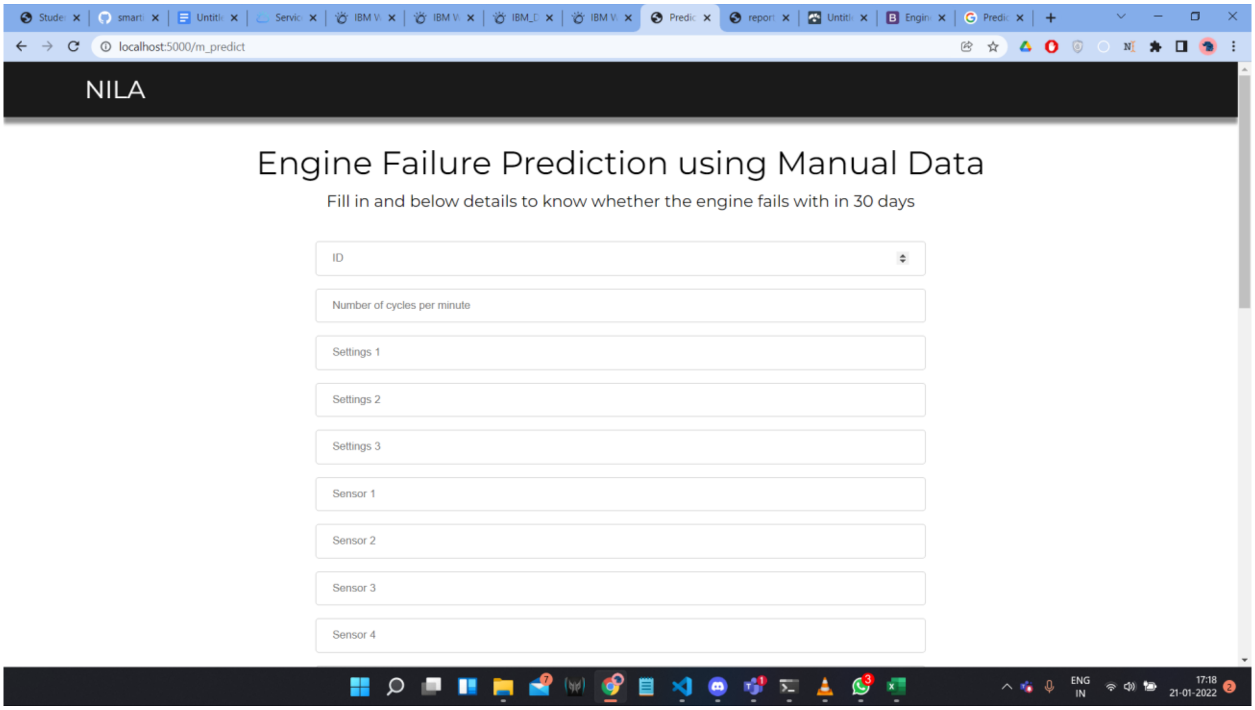
In [129]:
```python
client.set.default_space(space_uid)
```

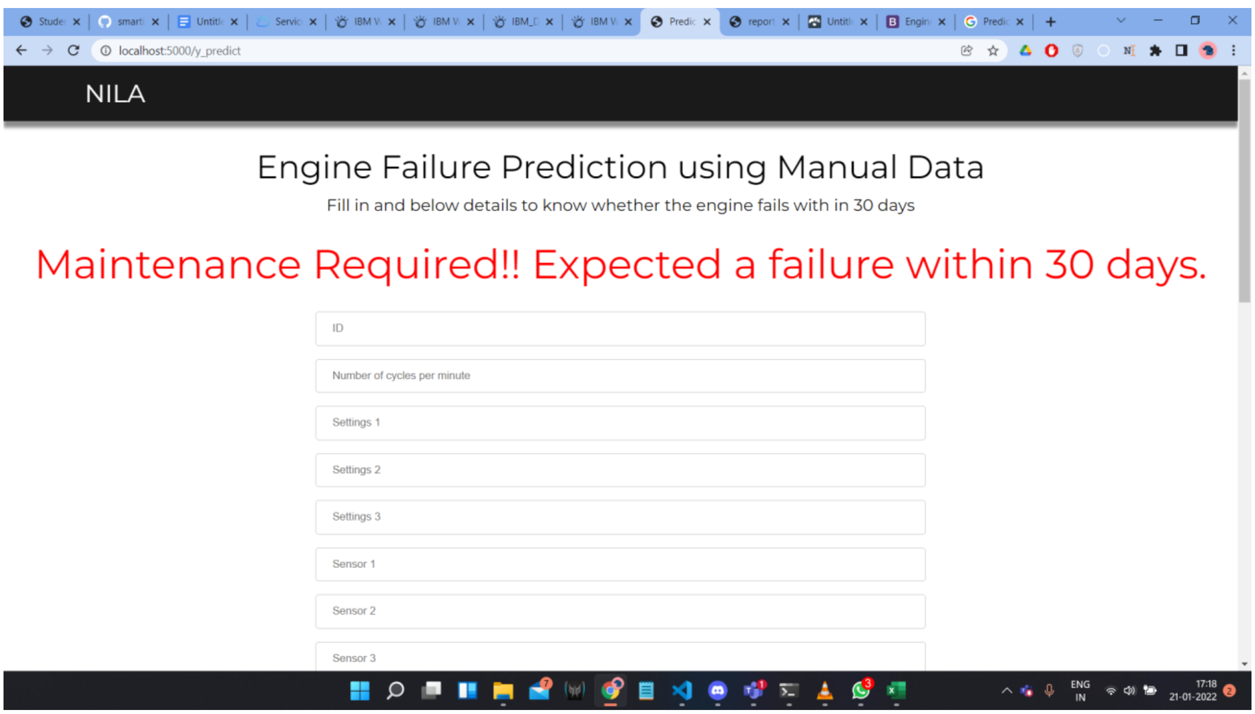
Out[129]:
```
'SUCCESS'
```

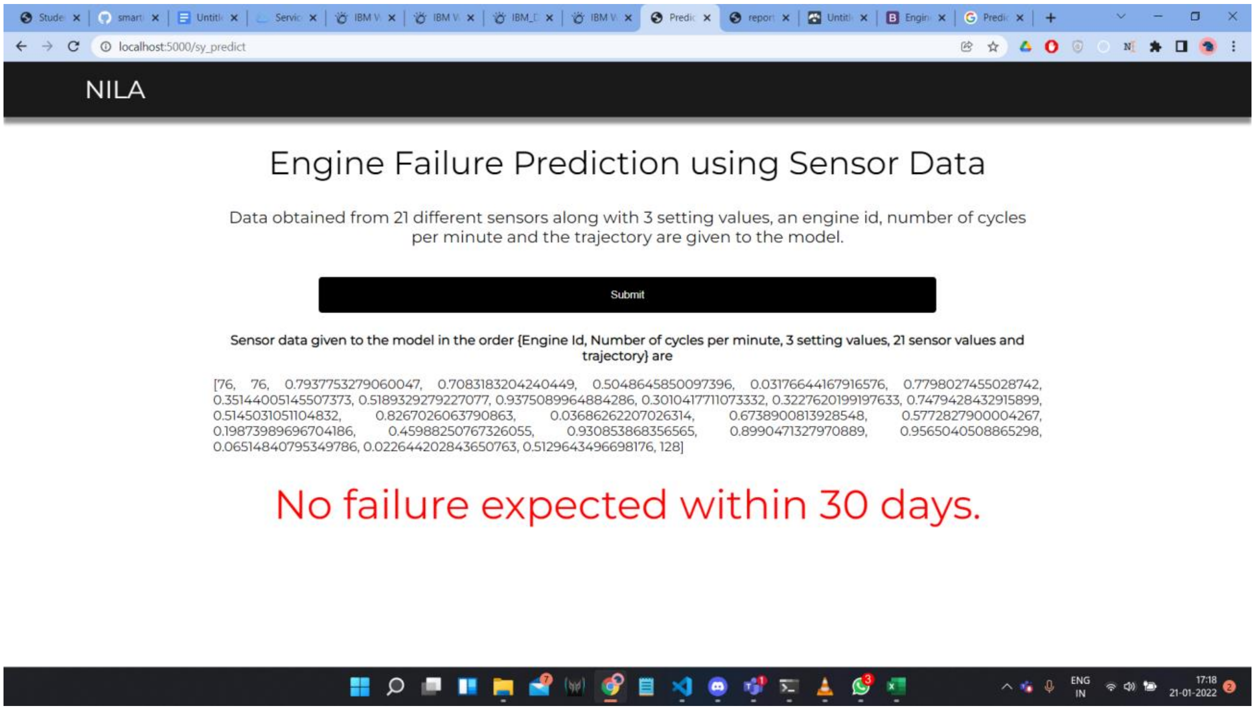
Output Page:


ENGINE FAILURE
HOME
ABOUT US
CONTACT US
Engine Failure Prediction
Machine learning algorithm to detect the engine failure before hand reduce the maintenance costs.
Get Started


ENGINE FAILURE
HOME
ABOUT US
CONTACT US
ABOUT US
Our goal is to minimize the costs associated with:
Unnecessary checks done by a mechanic($10).
Missing a faulty truck, which may cause a breakdown in the future($500).
The units for each sensor's data is kept private for proprietary reasons. However the main objective of our program will be to predict and minimize the cost of failures associated with these combinations of readings.
Manual Prediction
Here you can manually enter the values received from the various sensors and see if the engine needs a maintenance
PREDICT
Sensor Prediction
Automatic values from the sensor will be received and engine conditions can then be analysed
PREDICT


localhost:5000/m_predict
NILA
Engine Failure Prediction using Manual Data
Fill in and below details to know whether the engine fails with in 30 days
ID
Number of cycles per minute
Settings 1
Settings 2
Settings 3
Sensor 1
Sensor 2
Sensor 3
Sensor 4
17:18
21-01-2022


localhost:5000/y_predict
NILA
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
Sensor 1
Sensor 2
Sensor 3
17:18
21-01-2022



localhost:5000/sy_predict
NILA
Engine Failure Prediction using Sensor Data
Data obtained from 21 different sensors along with 3 setting values, an engine id, number of cycles per minute and the trajectory are given to the model.
Submit
Sensor data given to the model in the order {Engine Id, Number of cycles per minute, 3 setting values, 21 sensor values and trajectory} are
[76, 76, 0.7937753279060047, 0.7083183204240449, 0.5048645850097396, 0.03176644167916576, 0.7798027455028742, 0.35144005145507373, 0.5189329279227077, 0.9375089964884286, 0.3010417711073332, 0.3227620199197633, 0.7479428432915899, 0.5145031051104832, 0.8267026063790863, 0.03686262207026314, 0.6738900813928548, 0.5772827900004267, 0.19873989696704186, 0.45988250767326055, 0.930853868356565, 0.8990471327970889, 0.9565040508865298, 0.06514840795349786, 0.022644202843650763, 0.5129643496698176, 128]
No failure expected within 30 days.
ENG
IN
17:18
21-01-2022