MACHINE LEARNING-BASED PREDICTIVE ANALYTICS FOR AIRCRAFT ENGINE

TEAM ID: PNT2022TMID12895

Bachelor of Engineering Electronics and Communication Engineering

PSG College Of Technology

Coimbatore – 641004

Faculty Evaluator: Dr. P. Kalpana

Faculty Mentor: Dr. C. Ramya

Team Members:

19L214 – Sathyak G

19L234 – Prathish K V

19L243 – Sanjay S

19L249 – Shayith Rilwan

1. INTRODUCTION

1.1 Project Overview

Predict the possibility of an engine failing given sample data from various data. Using machine learning, that takes on previous engine sensor data and results, the end result of an engine working or failing is given by the model. Different machine learning models have been evaluated to determine the best fitting model.

1.2 Purpose

Due to the unpredictable nature of engine failure, steps must be taken to ensure the safety of passengers on every flight. Engine failure depends on a large number of parameters and not all of them can be interpreted by humans. For this purpose, a machine learning model is introduced. The model can infer pattern in the existing data that normal human beings cannot. For this reason, it is important to introduce such solutions into the market to further improve public confidence in air travel and also prevent potential catastrophic mid-air engine failures.

2. LITERATURE SURVEY

2.1. Existing Problem

The Existing solution for this purpose has been studied. It has been that inferred that, the outcome of the engine is being predicted approximately using some basic software. The result obtained is often unreliable and the margin of error is very high owing to the multiple variables affecting the output. A customizable solution is required to meet the requirements of multiple air carriers.

2.2. References

1) **Machine Learning-Based Predictive Analytics for Aircraft Engine Conceptual Design:**

Author: Michael T. Tong

Published: NASA 2020

Description: Big data and artificial intelligence/machine learning are transforming the global business environment. Data is now the most valuable asset for enterprises in every industry. Companies are using data-driven insights for competitive advantage. With that, the adoption of machine learning-based data analytics is rapidly taking hold across various industries, producing autonomous systems that support human decision-making. This work

explored the application of machine learning to aircraft engine conceptual design. Supervised machine-learning algorithms for regression and classification were employed to study patterns in an existing, open-source database of production and research turbofan engines, and resulting in predictive analytics for use in predicting performance of new turbofan designs. Specifically, the author developed machine learning-based analytics to predict cruise thrust specific fuel consumption (TSFC) and core sizes of high-efficiency turbofan engines, using engine design parameters as the input. The predictive analytics were trained and deployed in Keras, an open-source neural networks application program interface (API) written in Python, with Google's TensorFlow (an open-source library for numerical computation) serving as the backend engine. The promising results of the predictive analytics show that machine-learning techniques merit further exploration for application in aircraft engine conceptual design.

2) **Fault Tolerant Control, Artificial Intelligence and Predictive Analytics for Aerospace Systems: An Overview:**

Authors: Krishna Dev Kumar and Venkatesh Muthusamy

Published: Springer 2017

Description: An aircraft or spacecraft represents a highly complex engineering system. The requirements of high performance and increased autonomous capability have necessitated the use of fault tolerant control, artificial intelligence and predictive analytics in aerospace systems. The paper presents an overview of the current state of art in this area with a focus on the work done by the authors. Model-based fault tolerant control methods are considered for spacecraft systems while data driven prognostics is reviewed for fault prediction of aircraft engine failures. The data driven methods including artificial intelligence show promising results for applications to aerospace systems.

3) **Predictive Maintenance and Performance Optimisation in Aircrafts using Data Analytics:**

Authors: Shakthi Weerasinghe, Supunmali Ahangama.

Published: IEEE 2018

Description: Airline industry has provided a significantly conventional, faster and reliable mode of transportation for passengers and freight over the decades in which the industry has been in service despite the pressure being applied especially in maintaining operational affordability. The study critically reviews the techniques and tools, infrastructure and general application architecture for discussing the applicability of data analytics based on both batch processing and real time stream data in general aviation for health monitoring and predictive analysis in order to predict maintenance and optimize the performance of aircrafts. In this

respect, the study further evaluates the significant capability in addressing contemporary problems which are uniquely addressed by data analytics system.

4) **Applications of deep learning in big data analytics for aircraft complex system anomaly detection:**

Authors: Shungang Ning, Jianzhong Sun, Cui Liu and Yang Yi

Published: SAGE 2021

Description: Big data analytics with deep learning approach have attracted increasing attention in transportation engineering, involving operations, maintenance, and safety. In commercial aviation sectors, operational, and maintenance data produced on modern aircraft is increasing exponentially, and predictive analysis of these data is an exciting and promising field in aviation maintenance, which has a potential to revolutionize aerospace maintenance industry. This study illustrates the stateof-the-art applications of deep learning in big data analytics for predictive maintenance and a real-world case study for commercial aircraft. A Long Short-Term Memory network-based Auto-Encoders (LSTM-AE) is proposed for complex aircraft system fault detection and classification, which makes use of the raw time-series data from heterogeneous sensors. The proposed method uses nominal time-series samples corresponding to healthy behaviour of the system to learn a reconstruction model based on LSTM-AE framework. Then the system health index (HI) and fault feature vectors are derived from the reconstruction error matrix for fault detection and classification. The proposed method is demonstrated on a real-world data set from a commercial aircraft fleet. The typical PCV faults as well as the 390 F sensor and 450 F sensor faults due to sense line air leakage are successfully detected and distinguished based on the extracted features. The case study results show that the computed HI can effectively characterize the health state of the aircraft system and different fault types can be identified with high confidence, which is helpful for line fault troubleshooting.

5) **A Generic and Scalable Pipeline for Large-Scale Analytics of Continuous Aircraft Engine Data:**

Authors: Florent Forest, Jérôme Lacaille, Mustapha Lebbah and Hanene Azzag

Published: IEEE 2018

Description: A major application of data analytics for aircraft engine manufacturers is engine health monitoring, which consists in improving availability and operation of engines by leveraging operational data and past events. Traditional tools can no longer handle the increasing volume and velocity of data collected on modern aircraft. We propose a generic and scalable pipeline for large-scale analytics of operational data from a recent type of

aircraft engine, oriented towards health monitoring applications. Based on Hadoop and Spark, our approach enables domain experts to scale their algorithms and extract features from tens of thousands of flights stored on a cluster. All computations are performed using the Spark framework, however custom functions and algorithms can be integrated without knowledge of distributed programming. Unsupervised learning algorithms are integrated for clustering and dimensionality reduction of the flight features, in order to allow efficient visualization and interpretation through a dedicated web application. The use case guiding our work is a methodology for engine fleet monitoring with a self-organizing map. Finally, this pipeline is meant to be end-to-end, fully customizable and ready for use in an industrial setting.

6) **Predictive Aircraft Engine Maintenance:**

Authors: Vikas Chhikara

Published: 2020

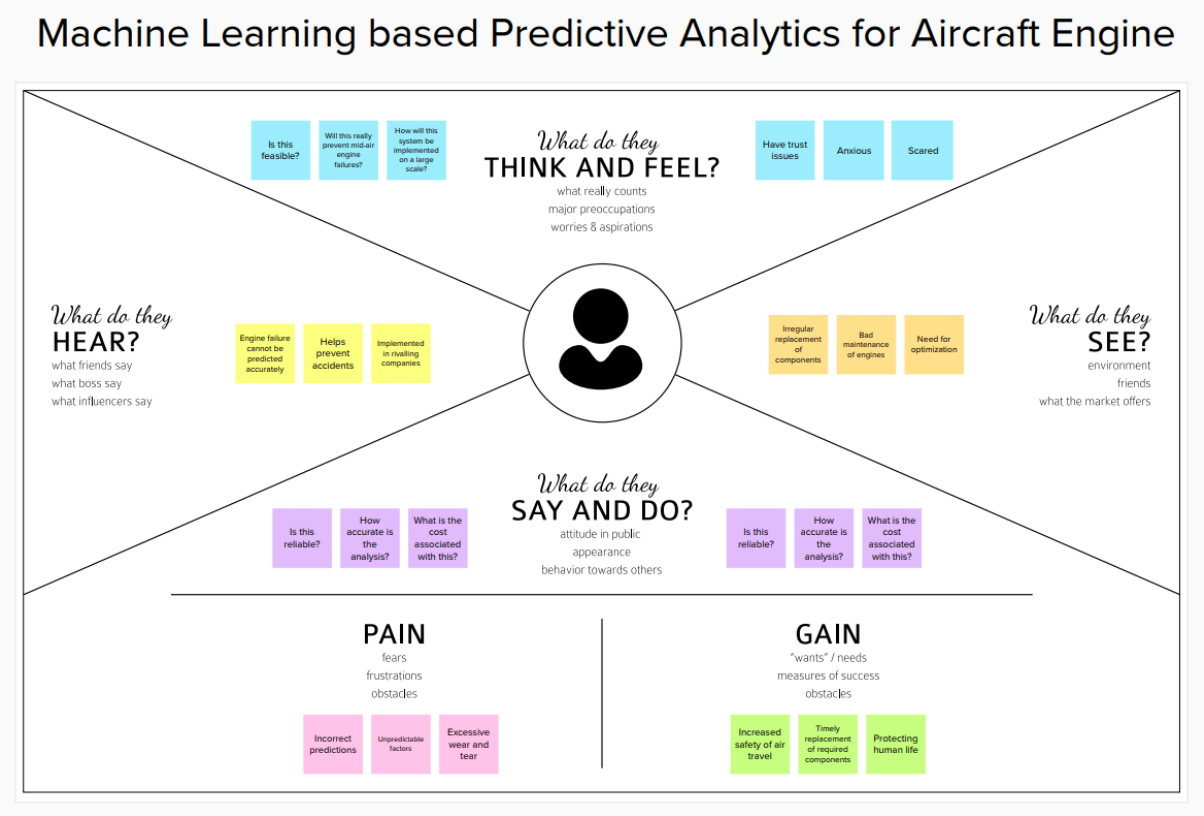
Description: For maintenance decisions and selecting a suitable operation for a machine, it's necessary to analyse the remaining useful life of the machine accurately. Machine learning techniques for RUL are usually focused as they are faster and easy to use. The existing models for RUL prediction are a single path or based on a top-down approach. For increasing the accuracy and to achieve promising results this report proposes a methodology that combines the Convolutional neural networks (CNN) and long short-term memory in order to predict the useful life of the machine. A different approach than existing models for this report CNN and LSTM model is actually combined rather than just using CNN for extracting features. But as for input single timestamp is used that can further lead to the same batch padding which could affect the model's prediction. The proposed methodology is used to overcome these issues by sliding the time one step size. For this report turbofan engine degradation data by NASA is used for training, testing, and validation of the RUL Model. By comparing the model using different Models like simple LSTM and transfer learning using the same dataset. With comparison, it will be easy to examine the performance of the proposed approach.

2.3. Problem Statement Definition

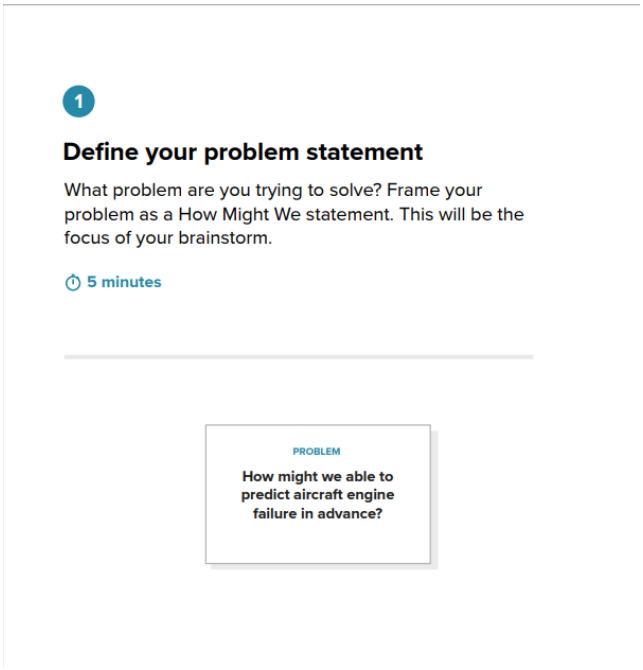
Aircraft engines suffer failure of components very frequently. Sometimes, these failures can go overlooked resulting in catastrophic mid-air engine failures. Every aircraft must take precautions to protect passenger safety because engine failure is unpredictable. There are many variables that affect engine failure, and not all of them can be understood by humans. A machine learning model is presented for this aim. Normal humans are unable to infer patterns from the existing data, but the model can. It is crucial to release such products onto the market in order to boost consumer confidence in air travel and guard against potentially disastrous mid-flight engine breakdowns.

3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas



3.2. Ideation and Brainstorming



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Sanjay

- Create an ML model to detect failure
- Use better quality components in engines
- Create web application for users to use
- Conduct full checks on all engines at one time

Prathish

- Search for datasets on Kaggle
- Use Naive Bayes classifier
- Use Logistic Regression model
- Predict the failure using only the flying hours

Rilwan

- Create dataset by ourselves
- Rigorous testing of engines before production
- Do random engine health checks
- Retire aircraft engines if one fault is detected

Sathyak

- Change all aircraft engines after some time
- Use multiple engines on all planes to prevent single point of failure
- Tracking the weather each engine flies through
- Reduce the engine weight

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

- Changes in the hardware of engines are too expensive and not ideal
- Retiring engines after some time or after a fault is not feasible
- Random checks are too dangerous as a lot of causes of failure would be overlooked
- ML models could be created using different algorithms
- A web app could serve as a platform for users to utilize
- An ML model has to consider all parameters and generalize well

3.3. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Aircraft engines suffer failure of components very frequently. Sometimes, these failures can go overlooked resulting in catastrophic mid-air engine failures.
2.	Idea / Solution description	Use Machine Learning model to accurately predict if an aircraft engine is in a safe condition to fly.
3.	Novelty / Uniqueness	This model will take a number of parameters from a variety of sensors into account before coming to a conclusion about the status of a given aircraft engine.

4.	Social Impact / Customer Satisfaction	This model helps in accurately predicting the status of an aircraft engine (safe/unsafe). If implemented properly, this model could save lives by preventing unsafe engines from being flown.
5.	Business Model (Revenue Model)	This solution in itself can be monetized and offered on a pay-as-you-go basis. Or, the model can be sold on a one-time payment.
6.	Scalability of the Solution	Since, this solution is proposed as a web application, it can be easily scaled as per the requirement to different web servers using load balancing etc. Depending the requirement of each air carrier which utilizes the solution, a number of machine learning models, trained using a diverse range of datasets using different learning models can be used.

3.4. Problem Solution Fit

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Air carriers wishing to utilize this solution to reduce catastrophic engine failures is the customer segment for this proposed solution.

6. CUSTOMER CC

Currently, the customer cannot replace engines whenever there is a single failure. Additionally, hardware modifications to the engine are expensive and not feasible.

5. AVAILABLE SOLUTIONS AS

Determining when an aircraft engine must be serviced.

No particular solution that identifies trend in underlying data to detect failures in an intuitive manner.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Inconsistent engine checks

Revenue loss

Fatality risk

9. PROBLEM ROOT CAUSE RC

Aircraft engines failures can seem unpredictable when they are manually predicted based on manual observations and decisions. The decisions on whether or not to replace certain components are made without taking into consideration any underlying pattern that could exist in the data.

7. BEHAVIOUR BE

The customer tests the model before implementing.

They study the performance and accuracy of the model, calculate the benefits and profit associated with it, and discuss the difficulties in implementing this solution.

Focus on J&P, tap into BE, understand RC

3. TRIGGERS TR	10. YOUR SOLUTION SL	8. CHANNELS OF BEHAVIOUR CH
With increasing focus on passenger safety standards, the customer is under pressure to put into place stringent safety checks and measures to ensure the safety of the passengers under their care.	Using Machine Learning that takes on previous aircraft engine data and to predict the energy output will help in reducing the risk of fatalities, and improve the general confidence that the public have on air travel.	The performance of every engine can be monitored from a central office using this solution. It would give details of the current status of the engine. Specific engines can also be manually assessed by a technician on site if manual intervention is required.
4. EMOTIONS: BEFORE / AFTER EM BEFORE: Not being able to say when an engine could fail. Had to accept the elevated risk of mid-flight engine failure. AFTER: Able to predict accurately when an engine would fail, and reduce the risk of fatality by a great extent.		

4. REQUIREMENT ANALYSIS

4.1. Functional Requirements

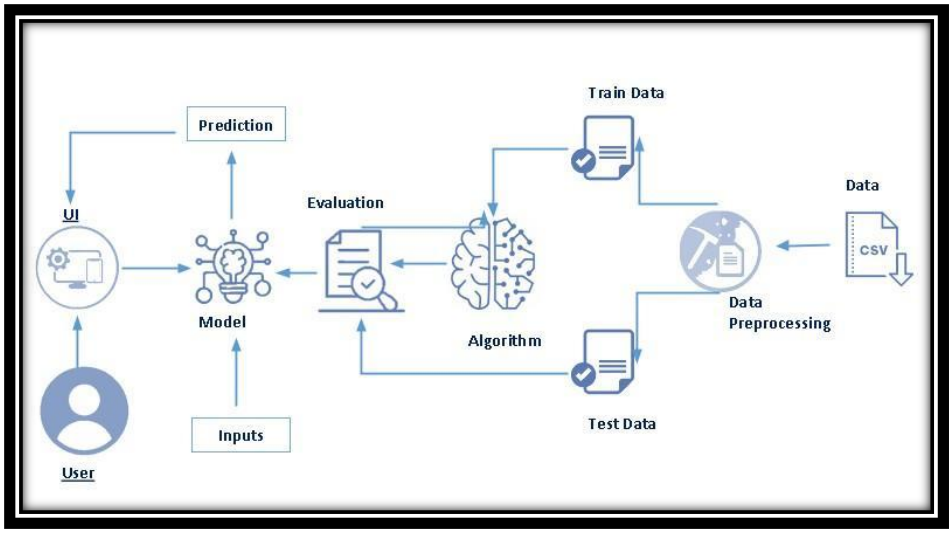
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Sending results via email
FR-3	User Infrastructure	A system to support Machine Learning A suitable GPU and CPU
FR-4	User Network	An internet connection for the web server to host the application and email the test results to the users
FR-5	User Cost	The web server running costs
FR-6	User Requirements	Knowledge on how to input the data on the application

4.2. Non-Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application is simple to use and does not require special training.
NFR-2	**Security**	The web server hosting the application is secure along with the user data.
NFR-3	**Reliability**	The predicted result is accurate and can be relied on
NFR-4	**Performance**	The Machine Learning model is fast and can predict results relatively quickly.
NFR-5	**Availability**	The web application has to be available to all users on the internet and be protected from denial-of-service attacks.
NFR-6	**Scalability**	The application must be capable of being scaled up or down across multiple servers based on demand.

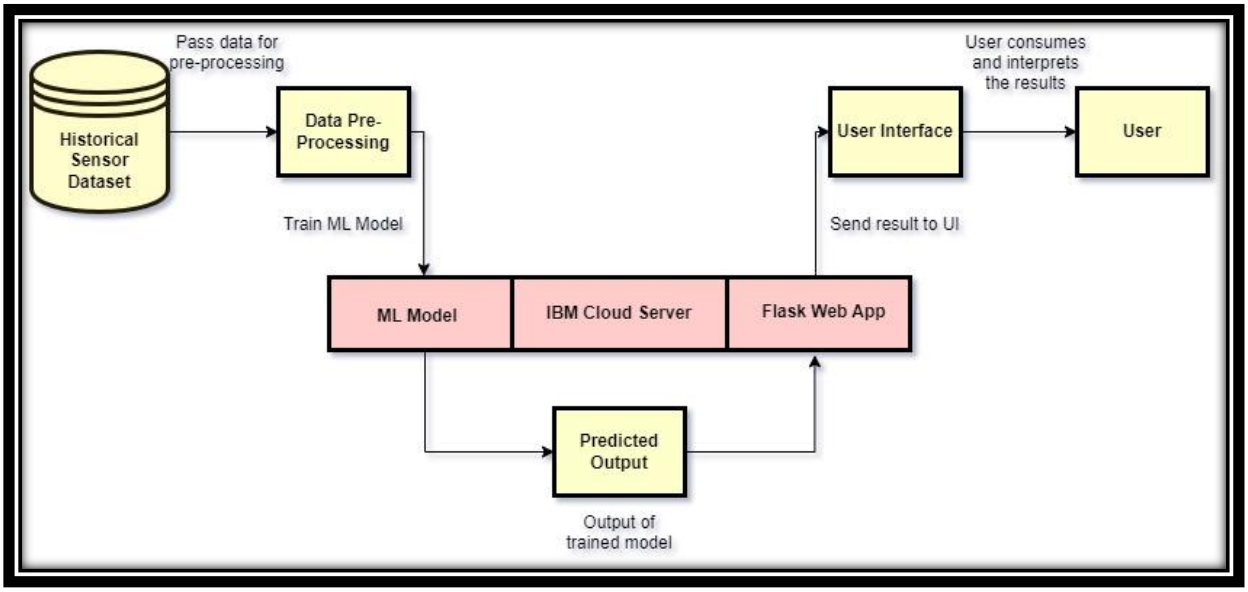
5. PROJECT DESIGN

5.1. Data Flow Diagrams



1. The dataset containing historical sensor data related to an aircraft engine is loaded from a CSV file into Python using anyone of the machine learning libraries available (sklearn).
2. This loaded data is then pre-processed and split into training and testing data in a specific proportion.
3. The model with the highest accuracy is chosen and the training data is then passed to it to train the model based on the historical sensor data to predict aircraft engine failures.
4. A web application is created using Flask, a light-weight python web framework, and is then hosted on an IBM Cloud server.
5. Sensor data entered by users in the Flask web application are then sent to the trained machine learning model to predict the chances of engine failure.
6. The classification model then classifies the engine as safe (1) or unsafe (0) based on the parameters the user enters.

5.2. Solution and Technical Architecture



S.No	Component	Description	Technology
1.	User Interface	The front-end templates	HTML, CSS, JavaScript
2.	Application Logic-1	Authentication using email and password	Python, Flask, SQL
3.	Application Logic-2	Input data to Machine Learning model to check the possibility for engine failure	Python, SKLearn Machine Learning Model
4.	Application Logic-3	Email test results to user	Python, Twilio
5.	Cloud Database	NoSQL type database to store user data on the cloud.	MongoDB Atlas
6.	External API-1	API used to send emails to the users.	Twilio
7.	External API-2	API calls to store and retrieve data from the database.	Python, pymongo
8.	Machine Learning Model	Purpose of Machine Learning Model	Aircraft Engine Failure detection model
9.	Infrastructure (Server / Cloud)	Basic server to host web application	IBM Cloud

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Flask
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Hashing to store the passwords on the database
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Model, view, controller architecture of the Flask framework
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	IBM Cloud infrastructure
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	-

5.3. Solution and Technical Architecture

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration/Login	USN-1	As a user, I can register and login to my account on the Flask web application.	I can utilize the login functionality to register & access the application.	Medium	Sprint-1
	Machine Learning Prediction	USN-1	As a user, I am able to pass data to the ML model and view the results through the app.	I can view the predicted results.	High	Sprint-2
	Receive results through e-mail	USN-2	As a user, I can receive confirmation emails of the test results.	I can receive the predicted results in the form an email.	Low	Sprint-1

6. PROJECT PLANNING AND SCHEDULING

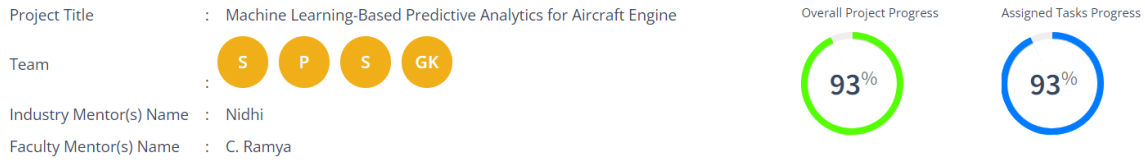
6.1. Sprint Planning and Estimation

Sprint	Milestone
Sprint 1	1. User Registers into the application through entering Email Id and Password for confirmation. 2. User Receives a confirmation mail for their registered email. 3. User can also register to the application through mobile number. 4. User logs in into the website using Email Id and password.
Sprint 2	1. User can access the dashboard. 2. User enters the required details of sensor data to get the desired Engine analytics output based on our model's prediction.
Sprint 3	1. Application stores the predictions, that can be used for future analysis. 2. The data stored has to be maintained securely.
Sprint 4	1. Administrator should properly maintain the website and update it whenever required.

6.2. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	27 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	04 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	08 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	16 Nov 2022

6.3. Workspace Progress Metric



7. CODING THE SOLUTION

```
import math
import collections
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.preprocessing import MinMaxScaler

# Visualization Libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Tuning & Splitting Libraries
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

# Loading Classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Model Evaluation Liabraries
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix, accuracy_score,
r2_score, roc_auc_score, precision_score, recall_score, f1_score

sns.set_style('darkgrid')
```

The required libraries for this project were imported at the beginning. The models are taken from the scikit-learn python package.

```
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.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='lrvvakIwcyLQUVQ12x2Wy08HkFK1Futc6qrOFHu6I2lA',
    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')

bucket = 'custommodeldeployment-donotdelete-pr-hcfp3onrensimw'
object_key = 'dataset.xlsx'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']

jet_data = pd.read_excel(body.read())
jet_data.head()
```

Importing the dataset uploaded to the IBM Cloud using the code given above. The boto3 python package is used to retrieve this.

	Unnamed: 0	id	cycle	op1	op2	op3	sensor1	sensor2	sensor3	sensor4	...	sensor14	sensor15	sensor16	sensor17	sensor18	sensor19	sensor20	sensor2
0	0	1	1	-0.0007	-0.0004	100	518.67	641.82	1589.70	1400.60	...	8138.62	8.4195	0.03	392	2388	100	39.06	23.419
1	1	1	2	0.0019	-0.0003	100	518.67	642.15	1591.82	1403.14	...	8131.49	8.4318	0.03	392	2388	100	39.00	23.423
2	2	1	3	-0.0043	0.0003	100	518.67	642.35	1587.99	1404.20	...	8133.23	8.4178	0.03	390	2388	100	38.95	23.344
3	3	1	4	0.0007	0.0000	100	518.67	642.35	1582.79	1401.87	...	8133.83	8.3682	0.03	392	2388	100	38.88	23.373
4	4	1	5	-0.0019	-0.0002	100	518.67	642.37	1582.85	1406.22	...	8133.80	8.4294	0.03	393	2388	100	38.90	23.404

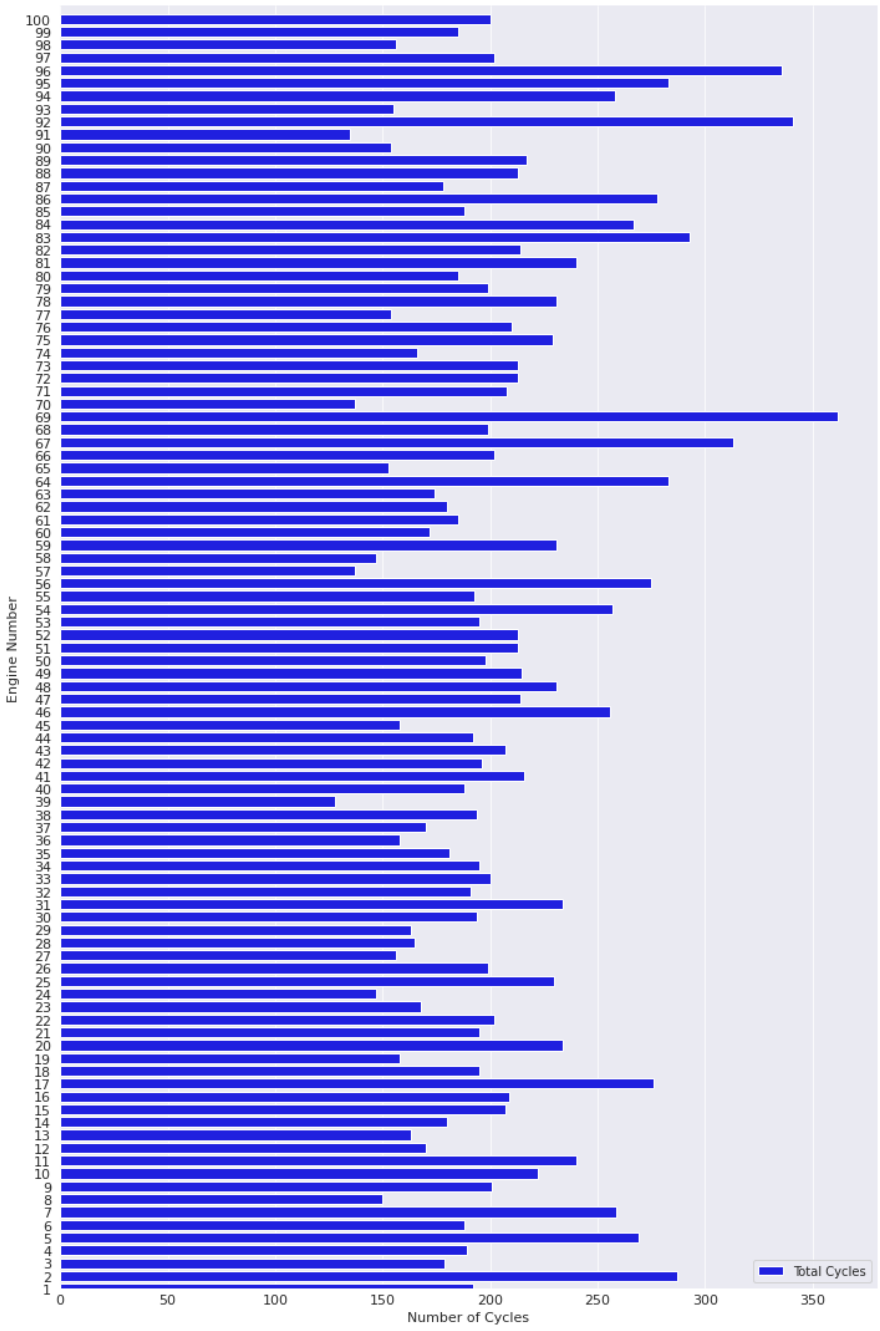
The head of the dataset is printed. This method prints only the first five records of a given dataset. As can be seen from the above image, there are many columns present in the dataset under consideration.

```
def RUL_calculator(df, df_max_cycles):
    max_cycle = df_max_cycles["cycle"]
    result_frame = df.merge(max_cycle.to_frame(name='max_cycle'), left_on='id', right_index=True)
    result_frame["RUL"] = result_frame["max_cycle"] - result_frame["cycle"]
    result_frame.drop(['max_cycle'], axis=1, inplace=True)
    return result_frame

jet_data = RUL_calculator(jet_data, jet_id_and_rul)
```

Another metric called Remaining Usage Life (RUL) is calculated for every record (engine) using proprietary logic. This method is applied to the entire data frame to calculate the RUL metric for every row.

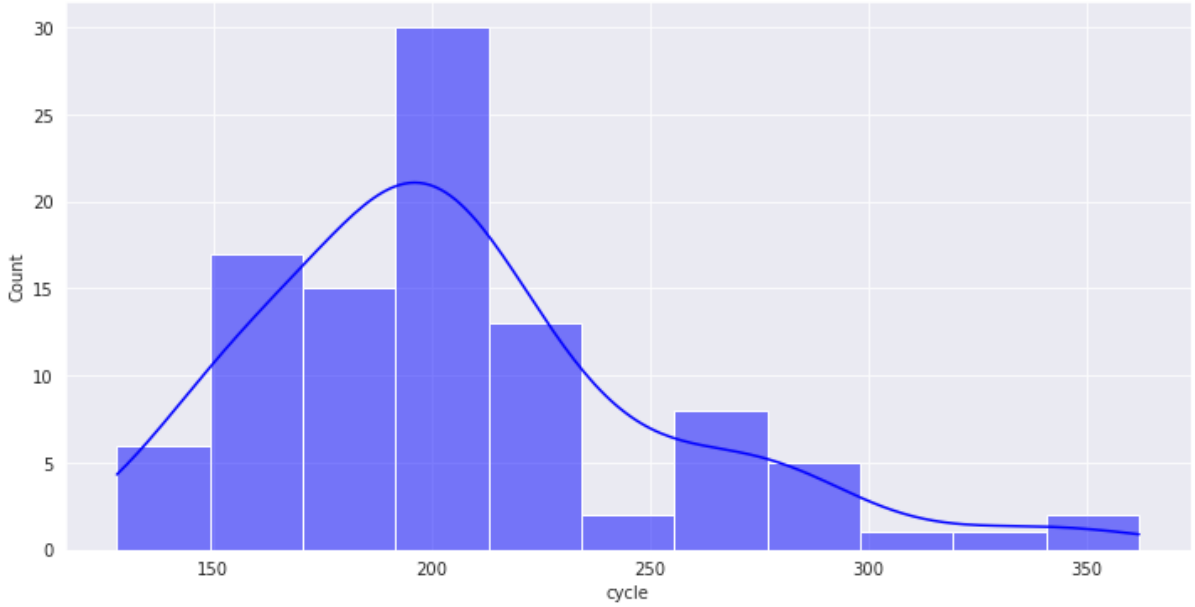
```
jet_id_and_rul = jet_data.groupby(['id'])[["id" ,"cycle"]].max()

f, ax = plt.subplots(figsize=(10, 15))
sns.set_color_codes("pastel")
sns.barplot(x="cycle", y="id", data=jet_id_and_rul, label="Total Cycles", color="blue", orient = 'h', dodge=False)
ax.legend(ncol=2, loc="lower right", frameon=True)
ax.set(ylim=(0, 100), ylabel="",xlabel="Automobile collisions per billion miles")
sns.despine(left=True, bottom=True)
ax.tick_params(labelsize=11)
ax.tick_params(length=0, axis='x')
ax.set_ylabel("Engine Number", fontsize=11)
ax.set_xlabel("Number of Cycles", fontsize=11)
plt.tight_layout()
plt.show()
```

The code given above is to visualize the average number of cycles each engine can withstand before failing. This graph is accomplished using the Matplotlib python package.



```
plt.subplots(figsize=(12, 6))
sns.histplot(jet_id_and_rul["cycle"], kde = True, color='blue');
print("Mean number of cycles after which jet engine fails is "+ str(math.floor(jet_id_and_rul["cycle"].mean())))
```

This code is to visualize the distribution of the maximum number of cycles an engine can withstand before failing. The mean of this value is also found using the graph.



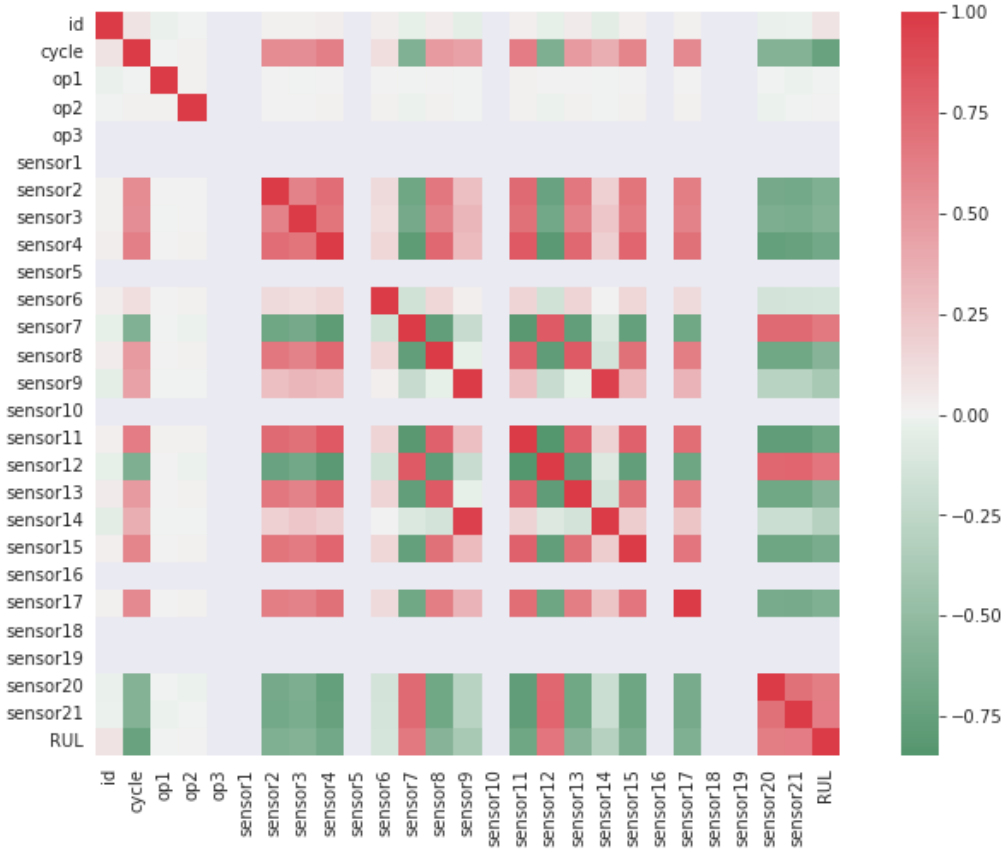
This is the distribution of the cycles after which engines fail. The mean occurs at 200 cycles. The exact value is found to be 206.

The next step is to plot a correlation matrix for all the attributes given in the dataset. This is done to prevent overfitting by reducing the number of attributes considered for the model. This correlation is plotted as a heatmap using the seaborn python visualization package.

From the given correlation matrix, some attributes are dropped from the data frame as they are deemed unnecessary for the training procedure.

```
jet_relevant_data = jet_data.drop(["cycle", "op1", "op2", "op3", "sensor1", "sensor5", "sensor6", "sensor10", "sensor16",
                                   "sensor18", "sensor19", "sensor14", "sensor13", "sensor12", "sensor11"], axis=1)
```

The above line of code shows a number of columns being dropped from the data frame.



The above image is a heatmap of the dataset before the columns are deleted. Once the deletion takes place, another heatmap of the updated dataset is drawn. This heatmap is given below.

```
scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(jet_relevant_data.drop(['id', 'RUL'], axis=1))
scaled_features = pd.DataFrame(scaled_features, columns=jet_relevant_data.drop(['id', 'RUL'], axis=1).columns)
```

Fit Transform will normalise all the values in all columns of dataframe except ‘id; and ‘RUL’.

	sensor2	sensor3	sensor4	sensor7	sensor8	sensor9	sensor15	sensor17	sensor20	sensor21	id	RUL
0	0.183735	0.406802	0.309757	0.726248	0.242424	0.109755	0.363986	0.333333	0.713178	0.724662	1	191
1	0.283133	0.453019	0.352633	0.628019	0.212121	0.100242	0.411312	0.333333	0.666667	0.731014	1	190
2	0.343373	0.369523	0.370527	0.710145	0.272727	0.140043	0.357445	0.166667	0.627907	0.621375	1	189
3	0.343373	0.256159	0.331195	0.740741	0.318182	0.124518	0.166603	0.333333	0.573643	0.662386	1	188
4	0.349398	0.257467	0.404625	0.668277	0.242424	0.149960	0.402078	0.416667	0.589147	0.704502	1	187

Normalised Values

```
cycle=30
data['label'] = data['RUL'].apply(lambda x: 1 if x <= cycle else 0)
```

The above code add label in the dataset if RUL is less than 30, it will label as 1 which means Engine Failure and if RUL is greater than 30,it will label as 0 which means no engine failure.

```
from sklearn.metrics import accuracy_score
model1 = LogisticRegression()
model1.fit(X_train, y_train)
model1_pred=model1.predict(X_test)
acc1=accuracy_score(y_test, model1_pred)
print("Accuracy for Logistic Regressipn:",acc1)
```

```
Accuracy for Logistic Regressipn: 0.9563847831354495
```

Logistic Regression Model with Accuracy 95.638%

```
model2 = KNeighborsClassifier()
model2.fit(X_train, y_train)
model2_pred=model2.predict(X_test)
acc2=accuracy_score(y_test, model2_pred)
print("Accuracy for KNN:",acc2)
```

```
Accuracy for KNN: 0.9549309425732978
```

K-Nearest Neighbour Model with Accuracy 95.493%

```
model3 = DecisionTreeClassifier()
model3.fit(X_train, y_train)
model3_pred=model3.predict(X_test)
acc3=accuracy_score(y_test, model3_pred)
print("Accuracy for Decision Tree Classifier:",acc3)
```

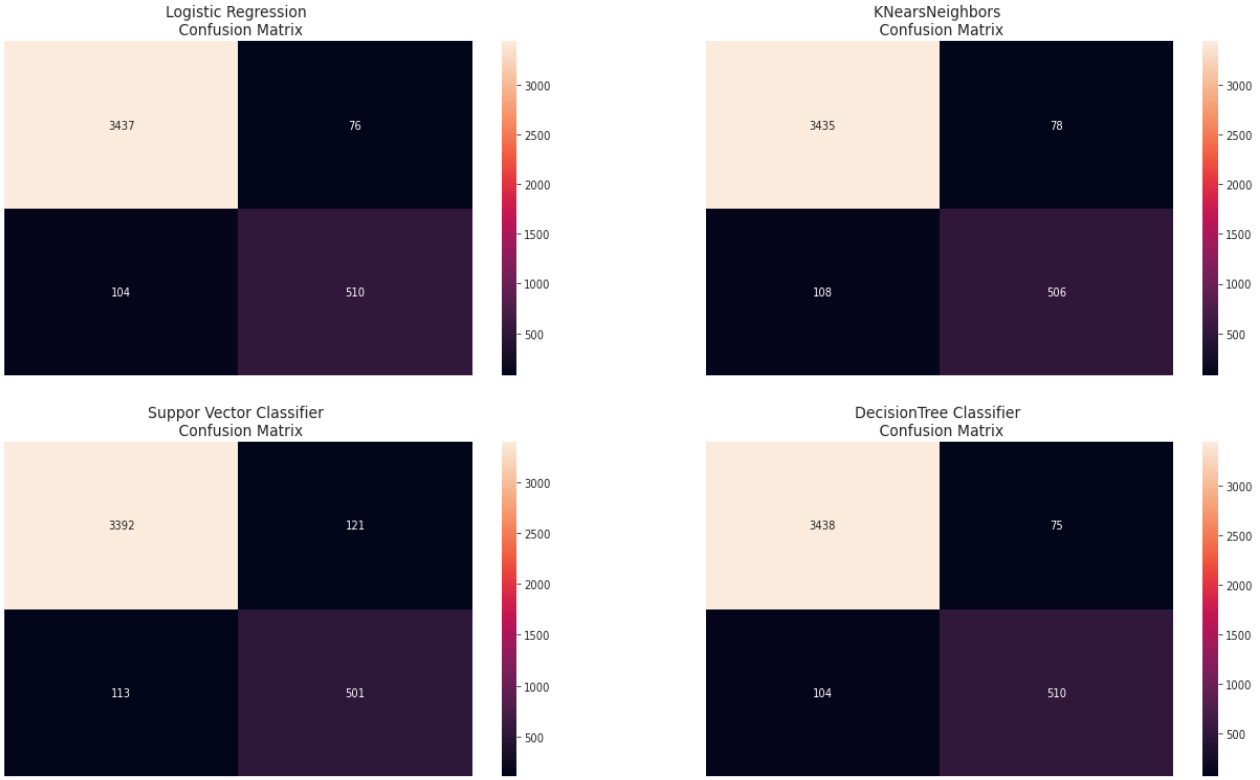
```
Accuracy for Decision Tree Classifier: 0.9433002180760843
```

Decision Tree Classifier Model with Accuracy 94.330%

```
model4 = SVC(kernel='linear')
model4.fit(X_train, y_train)
model4_pred=model4.predict(X_test)
acc4=accuracy_score(y_test, model4_pred)
print("Accuracy for Support Vector Machine:",acc4)
```

```
Accuracy for Support Vector Machine: 0.9566270898958081
```

Support Vector Machine Model with Accuracy 95.662%



Confusion Matrix for four trained Models.

8. TESTING

8.1. Test Cases

Sensor2	Sensor3	Sensor4	Sensor7	Sensor8	Sensor9	Sensor15	Sensor17	Sensor20	Sensor21	Label
643.54	1604.50	1433.58	550.86	2388.23	9065.11	8.5139	395	38.30	23.1594	0
641.82	1589.70	1400.60	554.36	2388.06	9046.19	8.4195	392	39.06	23.4190	1
642.35	1587.99	1404.20	554.26	2388.08	9052.94	8.4178	390	38.95	23.3442	1
643.85	1600.38	1432.14	550.79	2388.26	9061.48	8.5036	396	38.37	23.0511	0
643.23	1605.26	1426.53	550.68	2388.25	9073.72	8.5389	395	38.29	23.0640	0
642.37	1582.85	1406.22	554.00	2388.06	9055.15	8.4294	393	38.90	23.4044	1
643.42	1602.46	1428.18	550.94	2388.34	9065.90	8.5646	398	38.44	22.9333	1

8.2. User Acceptance Testing

The project has been tested extensively with a number of users. The users found the interface very easy to use. The Web pages were colourful and attractive. There were no unnecessary details in the web page. It was clean and simple that any new user could master it. The data input format was also simple. The user need not enter any unit. They could simply enter the value.

The prediction time is fairly low at an average time of 1.5 seconds. This delay depends on the internet connectivity. The model has been hosted in IBM cloud. Thus, with the API available, the model can be accessed remotely from any system provided IBM access key is given. The model predicts if the engine under consideration will fail in the immediate future or not. Although the prediction is not very accurate. Various inputs have been given by the users to test the consistency of the model. The model proved itself and all the users accepted the model as reliable and convenient.

9. RESULTS

9.1. Performance Metrics

The Support Vector Machine ML model that we have used here has better performance in speed and accuracy compared to other models. We have compared the performance metrics of 4 models and selected this as the best for the application. The model performed well for all the test cases. The API developed also performed good with no glitches or lag found during the testing phase.

10. ADVANTAGES & DISADVANTAGES

10.1. Advantages

This system will definitely help in saving lives and preventing mid-air engine failures, by predicting them in advance. It will also result in an increased confidence in air travel, which is beneficial to both the air carriers as well as passengers.

10.2. Disadvantages

The model may not always be accurate, and hence, could result in some false alarms. This may in turn lead to some unnecessary spending for service of an engine, but one can never put a price on human life. So, it is better to be more careful.

11. CONCLUSION

The Support Vector Machine ML model that has been used above performs well for our dataset. The model is fast and consumes less features. The API developed is also simple and userfriendly. By using this model, we could predict the Engine failure of the Aircraft provided the required input parameter. The model is not 100% accurate but it performs sufficiently. It can be concluded as the output cannot be predicted very accurately as there are several parameters that could affect the output and all those outputs cannot be taken in for training as it can result in a very complex and overtrained model. The features that have high weightage are considered in this model.

12. FUTURE SCOPE

The further works that can be done in this project is to include more features in model training to study the effect on the output. A long history of data (dataset of more than 3 years) can be used for training for increased accuracy. The application can be upgraded such that the input values from the sensors are directly fed to the model without the user entering it manually.

13. APPENDIX

13.1. Source Code

Model Training:

```
import math
import collections
```

```
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.preprocessing import MinMaxScaler
# Visualization Libraries
import seaborn as sns
import matplotlib.pyplot as plt
# Tuning & Splitting Libraries
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
# Loading Classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
# Model Evaluation Liabraries
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix, accuracy_score,
r2_score, roc_auc_score, precision_score, recall_score, f1_score
# Reading the data
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.
```

```
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='lrvvakIwcyLQUVQ12x2Wy08HkFK1Futc6qrOFHu6I2lA',
    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'
bucket = 'custommodeldeployment-donotdelete-pr-hcfp3onrensimw'
object_key = 'dataset.xlsx'
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
jet_data = pd.read_excel(body.read())
jet_data.head()
jet_data.columns =
["id","cycle","op1","op2","op3","sensor1","sensor2","sensor3","sensor4","sensor5"

,"sensor6","sensor7","sensor8","sensor9","sensor10","sensor11","sensor12","sensor1
3","sensor14","sensor15","sensor16","sensor17","sensor18","sensor19"sensor20","se
nsor21","sensor22","sensor23"]
jet_data.drop(['sensor22', 'sensor23'], axis=1, inplace=True)
jet_id_and_rul = jet_data.groupby(['id'])[["id" ,"cycle"]].max()
jet_id_and_rul.set_index('id', inplace=True)
def RUL_calculator(df, df_max_cycles):
    max_cycle = df_max_cycles["cycle"]
    result_frame = df.merge(max_cycle.to_frame(name='max_cycle'), left_on='id',
right_index=True)
    result_frame["RUL"] = result_frame["max_cycle"] - result_frame["cycle"]
    result_frame.drop(['max_cycle'], axis=1, inplace=True)
    return result_frame
jet_data = RUL_calculator(jet_data, jet_id_and_rul)
jet_id_and_rul = jet_data.groupby(['id'])[["id" ,"cycle"]].max()
f, ax = plt.subplots(figsize=(10, 15))
sns.set_color_codes("pastel")
sns.barplot(x="cycle", y="id", data=jet_id_and_rul, label="Total Cycles", color="blue",
orient = 'h', dodge=False)
ax.legend(ncol=2, loc="lower right", frameon=True)
```

```
ax.set(ylim=(0, 100), ylabel="",xlabel="Automobile collisions per billion miles")

sns.despine(left=True, bottom=True)

ax.tick_params(labelsize=11)

ax.tick_params(length=0, axis='x')

ax.set_ylabel("Engine Number", fontsize=11)

ax.set_xlabel("Number of Cycles", fontsize=11)

plt.tight_layout()

plt.show()

plt.subplots(figsize=(12, 6))

sns.histplot(jet_id_and_rul["cycle"], kde = True, color='blue');

print("Mean number of cycles after which jet engine fails is "+
str(math.floor(jet_id_and_rul["cycle"].mean())))

plt.figure(figsize=(13,8))

cmap = sns.diverging_palette(500, 10, as_cmap=True)

sns.heatmap(jet_data.corr(), cmap =cmap, center=0, annot=False, square=True);

jet_relevant_data = jet_data.drop(["cycle", "op1", "op2", "op3", "sensor1",
"sensor5", "sensor6", "sensor10", "sensor16", "sensor18", "sensor19", "sensor14",
"sensor13", "sensor12", "sensor11"], axis=1)

plt.figure(figsize=(15, 12))

cmap = sns.diverging_palette(500, 10, as_cmap=True)

sns.heatmap(jet_relevant_data.corr(), cmap =cmap, center=0, annot=True,
square=True);

scaler = MinMaxScaler()

scaled_features = scaler.fit_transform(jet_relevant_data.drop(['id', 'RUL'], axis=1))

scaled_features = pd.DataFrame(scaled_features,
columns=jet_relevant_data.drop(['id', 'RUL'], axis=1).columns)

scaled_features['id'] = jet_relevant_data['id']

scaled_features['RUL'] = jet_relevant_data['RUL']

cycle=30

data['label'] = data['RUL'].apply(lambda x: 1 if x <= cycle else 0)

y = data['label']

X = data.drop(['RUL', 'id', 'label'], axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=3)
print('X_train shape : ',X_train.shape)
print('X_test shape : ',X_test.shape)
print('y_train shape : ',y_train.shape)
print('y_test shape : ',y_test.shape)
from sklearn.metrics import accuracy_score
model1 = LogisticRegression()
model1.fit(X_train, y_train)
model1_pred=model1.predict(X_test)
acc1=accuracy_score(y_test, model1_pred)
print("Accuracy for Logistic Regressipn:",acc1)
model2 = KNeighborsClassifier()
model2.fit(X_train, y_train)
model2_pred=model2.predict(X_test)
acc2=accuracy_score(y_test, model2_pred)
print("Accuracy for KNN:",acc2)
model3 = DecisionTreeClassifier()
model3.fit(X_train, y_train)
model3_pred=model3.predict(X_test)
acc3=accuracy_score(y_test, model3_pred)
print("Accuracy for Decision Tree Classifier:",acc3)
model4 = SVC(kernel='linear')
model4.fit(X_train, y_train)
model4_pred=model4.predict(X_test)
acc4=accuracy_score(y_test, model4_pred)
print("Accuracy for Support Vector Machine:",acc4)
log_reg_cf = confusion_matrix(y_test, model1_pred)
kneighbors_cf = confusion_matrix(y_test, model2_pred)
svc_cf = confusion_matrix(y_test, model3_pred)
tree_cf = confusion_matrix(y_test, model4_pred)
fig, ax = plt.subplots(2, 2,figsize=(22,12))
```

```
sns.heatmap(log_reg_cf, ax=ax[0][0], annot=True, cmap='rocket', fmt='g')
ax[0, 0].set_title("Logistic Regression \n Confusion Matrix", fontsize=14)
ax[0, 0].set_xticklabels(['', ''], fontsize=14, rotation=90)
ax[0, 0].set_yticklabels(['', ''], fontsize=14, rotation=360)
sns.heatmap(kneighbors_cf, ax=ax[0][1], annot=True, cmap='rocket', fmt='g')
ax[0][1].set_title("KNearsNeighbors \n Confusion Matrix", fontsize=14)
ax[0][1].set_xticklabels(['', ''], fontsize=14, rotation=90)
ax[0][1].set_yticklabels(['', ''], fontsize=14, rotation=360)
sns.heatmap(svc_cf, ax=ax[1][0], annot=True, cmap='rocket', fmt='g')
ax[1][0].set_title("Suppor Vector Classifier \n Confusion Matrix", fontsize=14)
ax[1][0].set_xticklabels(['', ''], fontsize=14, rotation=90)
ax[1][0].set_yticklabels(['', ''], fontsize=14, rotation=360)
sns.heatmap(tree_cf, ax=ax[1][1], annot=True, cmap='rocket', fmt='g')
ax[1][1].set_title("DecisionTree Classifier \n Confusion Matrix", fontsize=14)
ax[1][1].set_xticklabels(['', ''], fontsize=14, rotation=90)
ax[1][1].set_yticklabels(['', ''], fontsize=14, rotation=360)
plt.show()
```

IBM Cloud Deployment:

1. Decision Tree Classifier:

```
import requests
# NOTE: you must manually set API_KEY below using information retrieved from your
  IBM Cloud account.
API_KEY = "xU7wUCiLdUHS2Y9iDdLma2qNSXQyWBzohz-ZlQCOB1Ss"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
  data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
# NOTE: manually define and pass the array(s) of values to be scored in the next line
# payload_scoring = {"input_data": [{"field":
  [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sen
```

sor20','sensor21']], "values": [[0.310241, 0.304556, 0.386226, 0.618357, 0.257576, 0.208068, 0.495575, 0.416667, 0.651163, 0.442833]]}]}

payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.183735, 0.406802, 0.309757, 0.726248,0.242424, 0.109755, 0.363986, 0.333333 ,0.713178, 0.724662]]}]}

print("Sent API call to model stored on IBM Cloud")

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/af77a4e8-8647-482b-9aa6-9ffc7c0f22dd/predictions?version=2022-11-16', json=payload_scoring,

headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

prediction = response_scoring.json()

print(prediction)

2. Support Vector Machine:

import requests

 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

API_KEY = "xU7wUCiLdUHS2Y9iDdLma2qNSXQyWBzohz-ZlQCOB1Ss"

token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":

API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token_response.json()["access_token"]header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken

 # NOTE: manually define and pass the array(s) of values to be scored in the next line

payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.310241, 0.304556, 0.386226, 0.618357, 0.257576, 0.208068, 0.495575, 0.416667, 0.651163, 0.442833]]}]}

 # payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.183735, 0.406802, 0.309757, 0.726248, 0.242424, 0.109755, 0.363986, 0.333333 ,0.713178, 0.724662]]}]}

print("Sent API call to model stored on IBM Cloud")

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/88b291c8-3850-4c4b-884c-2e617bad6870/predictions?version=2022-11-16', json=payload_scoring,

headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

prediction = response_scoring.json()

print(prediction)

3. Logistic Regression:

import requests

 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

API_KEY = "xU7wUCiLdUHS2Y9iDdLma2qNSXQyWBzohz-ZlQCOB1Ss"

token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":

API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

 # NOTE: manually define and pass the array(s) of values to be scored in the next line

 # payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.310241, 0.304556, 0.386226, 0.618357, 0.257576, 0.208068, 0.495575, 0.416667, 0.651163, 0.442833]]}]}

payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.183735, 0.406802, 0.309757, 0.726248, 0.242424, 0.109755, 0.363986, 0.333333 ,0.713178, 0.724662]]}]}

print("Sent API call to model stored on IBM Cloud")

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/d8cc712a-d330-4983-98be-ba7c0a1b2dc9/predictions?version=2022-11-16', json=payload_scoring,

headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

prediction = response_scoring.json()

print(prediction)

4. K Nearest Neighbour:

import requests

 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

API_KEY = "xU7wUCiLdUHS2Y9iDdLma2qNSXQyWBzohz-ZlQCOB1Ss"

token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":

API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

 # NOTE: manually define and pass the array(s) of values to be scored in the next line

 # payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.310241, 0.304556, 0.386226, 0.618357, 0.257576, 0.208068, 0.495575, 0.416667, 0.651163, 0.442833]]}]}

payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.183735, 0.406802, 0.309757, 0.726248, 0.242424, 0.109755, 0.363986, 0.333333 ,0.713178, 0.724662]]}]}

print("Sent API call to model stored on IBM Cloud"

response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/42a020f2-1c6e-43fc-a24d-a385ae66b45f/predictions?version=2022-11-16', json=payload_scoring,

headers={'Authorization': 'Bearer ' + mltoken})

print("Scoring response")

prediction = response_scoring.json()

print(prediction)

Flask Page:

import requests

import numpy as np

```
import os
from PIL import Image
from flask import Flask, request, render_template, url_for
from werkzeug.utils import secure_filename, redirect
from gevent.pywsgi import WSGIServer
from flask import send_from_directory
from joblib import Parallel,delayed
import joblib
import pandas as pd
from scipy.sparse import issparse
from sklearn.preprocessing import MinMaxScaler
# UPLOAD_FOLDER = 'D:/sdhi/PROJECT DEVELOPMENT PHASE/SPRINT 3/UPLOADS'
app = Flask(__name__)
# app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
print("Flask application created")
@app.route('/')
def index():
   return render_template('index.html')
@app.route('/predict', methods=['POST'])
def upload():
   if request.method == "POST":
      data =
[[request.form.get('age'),request.form.get('tb'),request.form.get('ap'),request.form.g
et('aa'),request.form.get('asa')
,request.form.get('tp'),request.form.get('a'),request.form.get('agr'),request.form.get(
'a1'),request.form.get('a2')]]
      df = pd.DataFrame(data, columns=['sensor2', 'sensor3', 'sensor4', 'sensor7',
'sensor8', 'sensor9', 'sensor15', 'sensor17', 'sensor20', 'sensor21'])
      scaler = MinMaxScaler()
      df1 = scaler.fit_transform(df)
```

```
    # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.

    API_KEY = "xU7wUCiLdUHS2Y9iDdLma2qNSXQyWBzohz-ZlQCOB1Ss"

    token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":

    API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})

    mltoken = token_response.json()["access_token"]

    header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

    # NOTE: manually define and pass the array(s) of values to be scored in the next line

    # payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.310241, 0.304556, 0.386226, 0.618357, 0.257576, 0.208068, 0.495575, 0.416667, 0.651163, 0.442833]]}]}

    # payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": [[0.183735, 0.406802, 0.309757, 0.726248, 0.242424, 0.109755, 0.363986, 0.333333 ,0.713178, 0.724662]]}]}

    payload_scoring = {"input_data": [{"field": [['sensor2','sensor3','sensor4','sensor7','sensor8','sensor9','sensor15','sensor17','sensor20','sensor21']], "values": df1.tolist()}]}

    print("Sent API call to model stored on IBM Cloud")

    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/d8cc712a-d330-4983-98be-ba7c0a1b2dc9/predictions?version=2022-11-16', json=payload_scoring,

    headers={'Authorization': 'Bearer ' + mltoken})

    print("Scoring response")

    prediction = response_scoring.json()

    pred = prediction['predictions'][0]['values'][0][0]

    msg = "There is less chance of this engine failing in the immediate future" if pred == 0 else "There is a good chance that this engine will fail soon!"

    return render_template('predict.html', num=msg)

if __name__ == '__main__':
```

```
app.run(debug=True, threaded=False)
```

HTML Page:

1. Input Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="static/CSS/index.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@200&display=swap"
rel="stylesheet">
  <title>Aircraft</title>
</head>
<body>
  <h2>Aircraft Engine Failure Prediction</h2>
  <div id="positive" hidden><h4 id="pos_data">Positive</h4></div>
  <div id="negative" hidden><h4 id="neg_data">Negative</h4></div>
  <table>
    <form action="/predict" method="POST" enctype="multipart/form-data">
    <tr>
      <td>sensor2</td>
      <td><input name="age" type="number" min="0" step="0.01" placeholder=""
required></td>
    </tr>
    <tr>
      <td>sensor3</td>
      <td><input name="tb" type="number" min="0" step="0.01" placeholder=""
required></td>
    </tr>
    <tr>
      <td>sensor4</td>
      <td><input name="ap" type="number" min="0" step="0.01" placeholder=""
required></td>
    </tr>
    <tr>
      <td>sensor7</td> <td><input name="aa" type="number" min="0" step="0.01"
placeholder="" required></td>
```

```
      </tr>
      <tr>
        <td>sensor8</td>
        <td><input name="asa" type="number" min="0" step="0.01" placeholder="" required></td>
      </tr>
      <tr>
        <td>sensor9</td>
        <td><input name="tp" type="number" min="0" step="0.01" placeholder="" required></td>
      </tr>
      <tr>
        <td>sensor15</td>
        <td><input name="a" type="number" min="0" step="0.01" placeholder="" required></td>
      </tr>
      <tr>
        <td>sensor17</td>
        <td><input name="agr" type="number" min="0" step="0.01" placeholder="" required></td>
      </tr>
      <tr>
        <td>sensor20</td>
        <td><input name="a1" type="number" min="0" step="0.01" placeholder="" required></td>
      </tr>
      <tr>
        <td>sensor21</td>
        <td><input name="a2" type="number" min="0" step="0.01" placeholder="" required></td>
      </tr>
      <tr>
        <td></td><td id="btn"><div id="btndiv"><input id="submit" type="submit" value="Predict"></div> <div id="btndiv"><input id="reset" type="reset" value="Clear"></div></td>
      </tr>

      </form>
    </table>
</body>
</html>
```

2. Output Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Demo</title>
   <link rel="stylesheet" href="static/CSS/index.css">
   <link rel="preconnect" href="https://fonts.googleapis.com">
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
   <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@200&display=swap"
rel="stylesheet">
</head>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Prediction</title>
</head>
<body>
   <h1 id="h">{{num}}</h1>
</body>
</html>
```

CSS:

```
*{
   background-color: #292929;
   color: whitesmoke;
   font-variant: small-caps;
   font-size: large;
   font-family: 'Poppins', sans-serif;
   margin: 0;
   padding: 0;
}
body{
   min-width: 200px;
   min-height: 600px;
```

```
    align-items: center;
    justify-content: center;
    font-weight: bold;
}
#ans{
    text-align: center;
}
h2{
    display: block;
    font-size: 1.5cm;
    margin: 0.7em;
    text-align: center;
}
table{
    margin-left: auto;
    margin-right: auto;
    vertical-align: middle;
}
td{
    margin: 1em;
    padding: 0.5em;
    text-align: justify;
}
input{
    border-radius: 2px;
    border: solid;
    border-color: whitesmoke;
    border-width: 2px;
    color: #00ccff;
```

```
    padding: 0.2em 0.7em;
    -moz-appearance: textfield; /*For Firefox*/
}
/*For Chrome and safari*/
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}
input::placeholder{
    color: #00ccff;
}
#btndiv{
    padding: 2px;
    width: max-content;
    border-radius: 1em;
    background: #1aff22;
    display: inline;
}
#submit,#reset{
    /*border-radius: 1em;
    border-style: solid;
    border-color: whitesmoke;*/
    border: none;
    background: #0e1538;
    color: whitesmoke;
    padding: 0.2em 0.7em;
    border-radius: 1em;
}
```

```
#btn{
    display: flex;
}
#submit:hover{
    cursor: pointer;
}
#btndiv:hover{
    background: linear-gradient(45deg,#1aff22,#0e1538,#ff075b);
}

#positive{
    background: linear-gradient(60deg,#1aff22,#0e1538,#0e1538,#0e1538,#1aff22);
    padding: 5px;
    margin: 1em;
    border-radius: 3rem;
}
#negative{
    background: linear-gradient(60deg,#ff075b,#0e1538,#0e1538,#0e1538,#ff075b);
    padding: 5px;
    margin: 1em;
    border-radius: 3rem;
}
#positive:hover{
    background: linear-gradient(300deg,#1aff22,#0e1538,#0e1538,#0e1538,#1aff22);
}
#negative:hover{
    background: linear-gradient(300deg,#ff075b,#0e1538,#0e1538,#0e1538,#ff075b);
}
h4{
```

```
    display: block;
    font-size: 0.8cm;
    margin: auto;
    padding: 1em;
    border-radius: 3rem;
    text-align: center;
    background: #0e1538;
}
```

13.2. GitHub & Project Demo Link:

GitHub Repo: https://github.com/IBM-EPBL/IBM-Project-37376-1660306488
Demo Link:
https://drive.google.com/file/d/1YsGMJjmWqriNn3Q36trpFpbEMBG6QcVC/view?usp=sharing