ARIGNAR ANNA GOVERNMENT ARTS COLLEGE

VILLUPURAM – 605 602.

DEPARTMENT OF COMPUTER APPLICATIONS

MACHINE LEARNING WITH PYTHON

Project Title: Flight Delay Prediction for aviation Industry using Machine Learning

Team Id: NM2023TMID18764

Team Leader: SURYA M (C15359E2905BE3D92514DDB1AA7EAC7B)

Team member: SWETHA M (9D9C9AC70A4B2DA93474B76057B2B919)

Team member: SANJAY G (D89AB4DC90C47CCAAF3437EEC7B91A2F)

Team member: RAMACHANDRAN R (9CF4622B6924B45AEA64B5BB67BB80D8)

Abstract

The project aims to develop a machine learning model for predicting flight delays. The model will be trained on historical flight data to identify patterns and factors that contribute to delays, such as weather conditions, air traffic congestion, and airport operations. The model will then use this information to predict the likelihood and duration of flight delays for future flights. The project aims to improve the accuracy of flight delay predictions, which can help airlines and passengers better plan their travel schedules and minimize disruptions caused by delays.

1. INTRODUCTION

1.1 Overview:

The flight delay prediction project using machine learning involves the development of a model that can accurately predict flight delays. This project utilizes historical flight data and machine learning techniques to identify patterns and factors that contribute to flight delays.

The model is trained on a large dataset of flight information, which includes various features such as departure and arrival times, weather conditions, airline and airport operations, and other relevant factors that may impact flight delays.

Once the model is trained, it can be used to predict the likelihood and duration of flight delays for future flights. This information can be used by airlines and passengers to better plan their travel schedules, reduce the impact of flight delays, and improve overall flight efficiency.

The project aims to improve the accuracy of flight delay predictions by utilizing advanced machine learning techniques such as neural networks and decision trees. By doing so, the model can identify complex relationships and interactions between various factors that may impact flight delays.

Overall, the flight delay prediction project using machine learning has the potential to greatly improve the efficiency and reliability of air travel by providing more accurate and timely information on flight delays.

1.2 Purpose:

The purpose of the flight delay prediction using machine learning project is to develop a model that can accurately predict flight delays. Flight delays can have significant impacts on both airlines and passengers, leading to reduced efficiency, increased costs, and lost revenue.

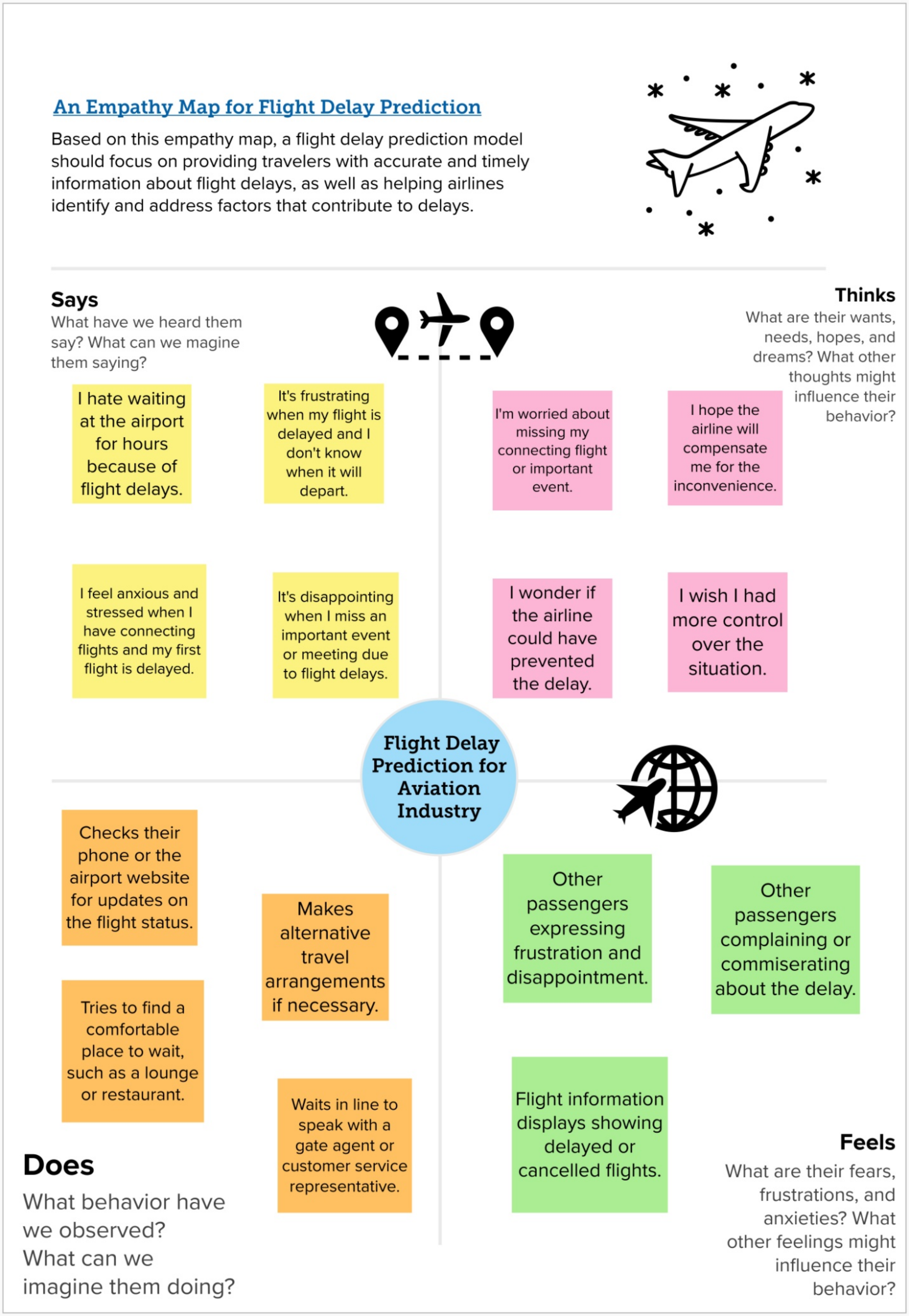
By predicting flight delays, airlines can take proactive measures to minimize their impact, such as adjusting flight schedules, re-routing flights, or rescheduling crew and equipment. Passengers can also benefit from more accurate and timely information on flight delays, allowing them to plan their travel schedules more effectively and avoid unnecessary waiting times at the airport.

The project aims to leverage the power of machine learning to identify patterns and factors that contribute to flight delays, and use this information to develop a predictive model. The model can then be used to provide real-time predictions of flight delays for airlines and passengers, helping to improve the overall efficiency and reliability of air travel.

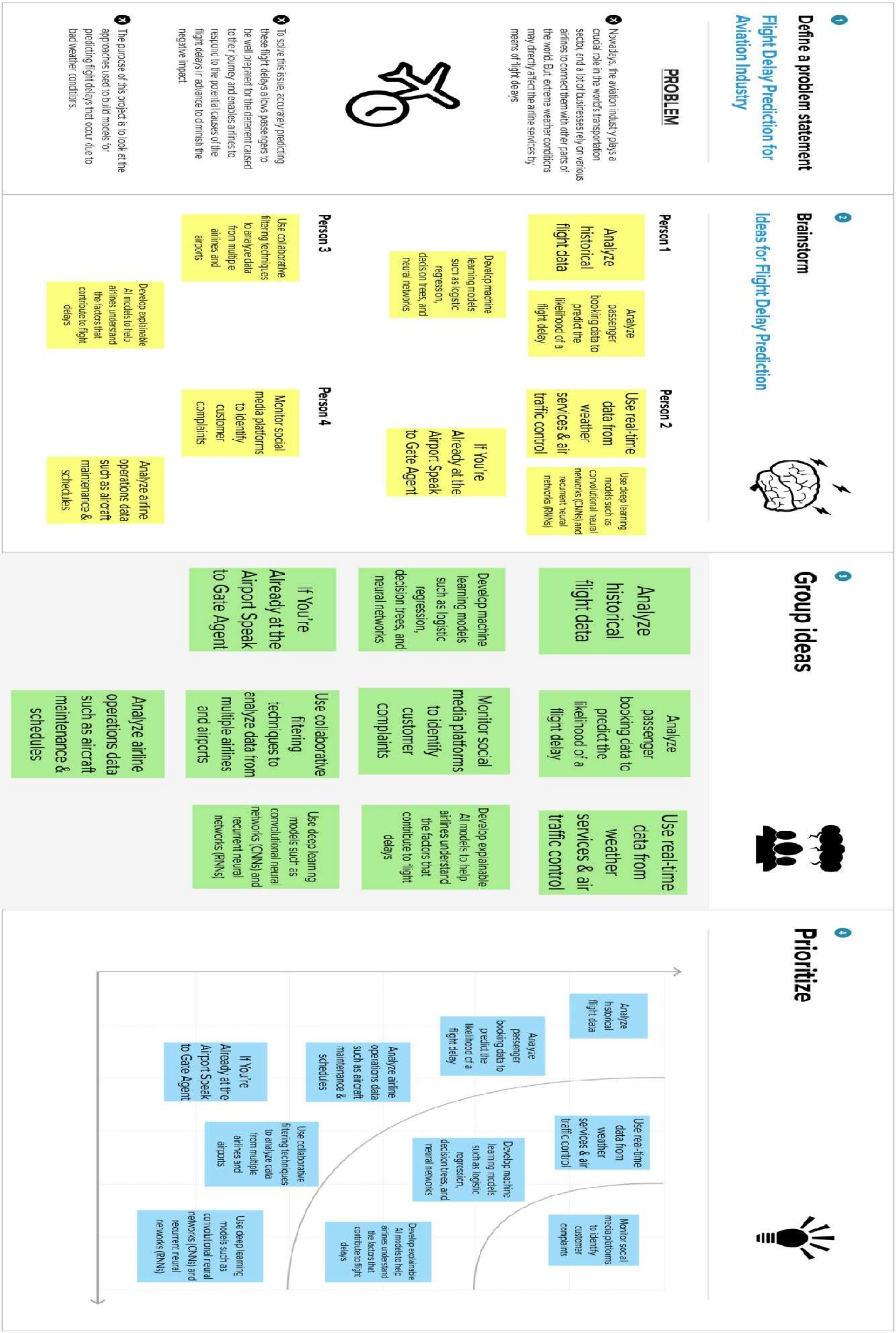
In summary, the purpose of the flight delay prediction using machine learning project is to improve the accuracy and timeliness of flight delay predictions, which can help airlines and passengers to better plan their travel schedules and minimize the impact of flight delays on their operations and experiences.

2. PROBLEM DEFINITION AND DESIGN THINKING

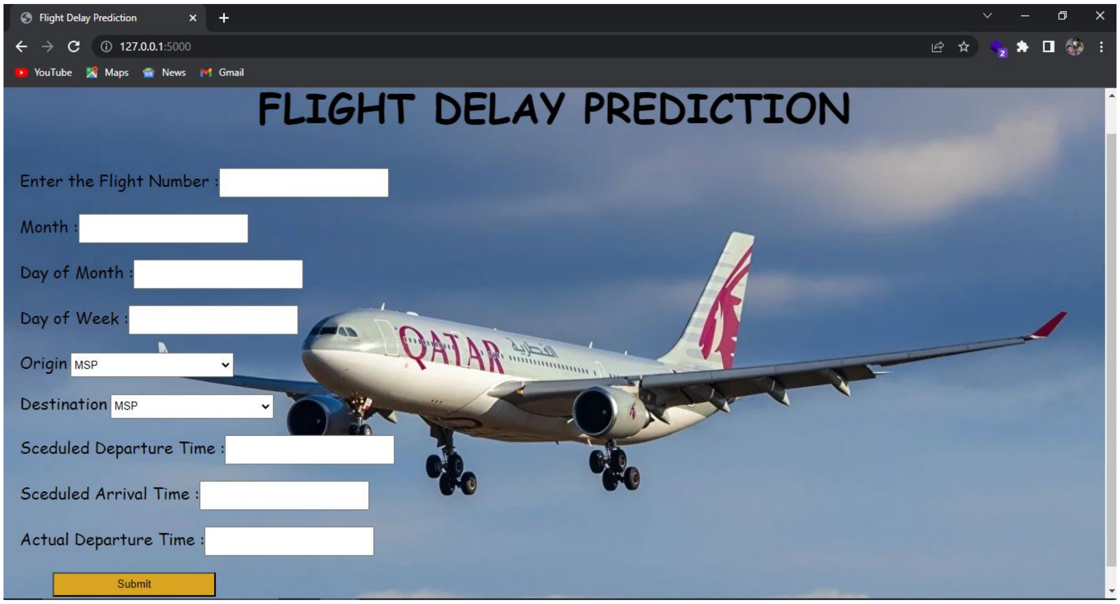
2.1 Empathy map:

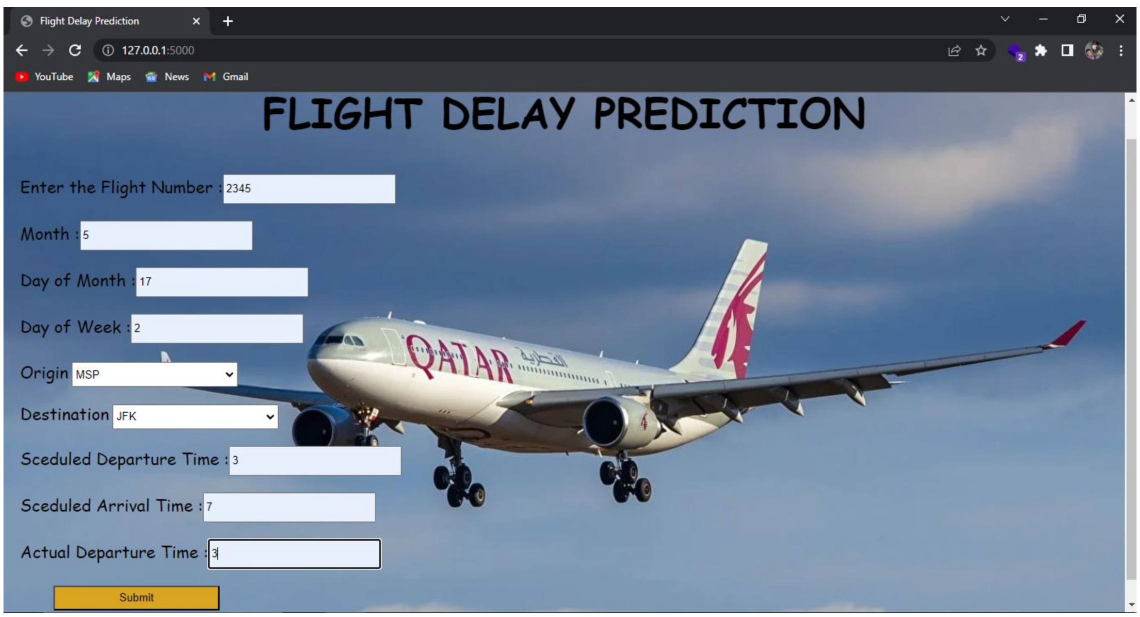


2.2 Ideation & Brainstorming Map:



3. RESULT


Flight Delay Prediction
127.0.0.1:5000
YouTube
Maps
News
Gmail
FLIGHT DELAY PREDICTION
Enter the Flight Number :
Month :
Day of Month :
Day of Week :
Origin MSP
Destination MSP
Sceduled Departure Time :
Sceduled Arrival Time :
Actual Departure Time :
Submit


Flight Delay Prediction
127.0.0.1:5000
YouTube
Maps
News
Gmail
FLIGHT DELAY PREDICTION
Enter the Flight Number : 2345
Month : 5
Day of Month : 17
Day of Week : 2
Origin MSP
Destination JFK
Sceduled Departure Time : 3
Sceduled Arrival Time : 7
Actual Departure Time : 3
Submit


Flight Delay Prediction
localhost:5000/prediction
YouTube
Maps
News
Gmail
Enter the Flight Number :
Month :
Day of Month :
Day of Week :
Origin MSP
Destination MSP
Sceduled Departure Time :
Sceduled Arrival Time :
Actual Departure Time :
Submit
The Flight will be On Time

4. ADVANTAGES AND DISADVANTAGES

Advantages:

- **Improved accuracy:** Machine learning models can identify complex patterns and relationships in data that humans may not be able to detect. This means that machine learning models can often make more accurate predictions about flight delays than traditional methods.
- **Real-time predictions:** Machine learning models can provide real-time predictions of flight delays, allowing airlines and passengers to make informed decisions about flight schedules and travel plans.
- **Reduced costs:** By predicting flight delays in advance, airlines can take proactive measures to minimize their impact, such as adjusting flight schedules or rescheduling crew and equipment. This can help to reduce costs associated with flight delays, such as lost revenue and additional expenses.
- **Improved customer experience:** By providing accurate and timely information on flight delays, passengers can better plan their travel schedules and avoid unnecessary waiting times at the airport. This can help to improve the overall customer experience and satisfaction.
- **Enhanced safety:** Predicting flight delays in advance can help airlines to better manage their operations and ensure that flights are operating safely and efficiently.

Disadvantages:

- **Data quality:** The accuracy and effectiveness of machine learning models rely heavily on the quality and quantity of data available for training. Incomplete or inaccurate data can lead to unreliable predictions and reduced performance of the model.
- **Limited scope:** Machine learning models are designed to make predictions based on historical data and may not account for unexpected events or changes in the operating environment. Therefore, the model's effectiveness may be limited in situations where there are significant changes in conditions or external factors that affect flight operations.
- **Complex implementation:** Machine learning models can be complex and require specialized skills and knowledge to develop, implement, and maintain. This can lead to increased costs and technical challenges for organizations that are not well-equipped to handle these types of projects.
- **Privacy concerns:** Machine learning models may require access to sensitive data such as passenger information, flight schedules, and airport operations, which can raise privacy and security concerns. It is essential to ensure that appropriate measures are taken to protect this data and maintain confidentiality.
- **Regulatory challenges:** There may be regulatory challenges associated with the implementation of machine learning models for flight delay prediction. This may include compliance with data privacy regulations, safety regulations, and other legal considerations that vary by region and jurisdiction.

5. APPLICATIONS

- **Airline operations:** Airlines can use machine learning models to predict flight delays and take proactive measures to adjust schedules, allocate resources, and optimize their operations. This can help to reduce costs associated with delays and improve overall efficiency.
- **Passengers:** Passengers can benefit from machine learning models that predict flight delays, enabling them to plan their travel schedules better, avoid unnecessary waiting times at airports, and reduce the stress associated with delays.
- **Airport operations:** Airports can use machine learning models to predict flight delays and improve the management of airport resources, such as gates, runways, and ground handling equipment. This can help to optimize airport operations, reduce congestion, and improve safety.
- **Air traffic management:** Machine learning models can help air traffic controllers to predict flight delays, reduce congestion in the air, and optimize flight paths. This can help to improve safety, reduce fuel consumption, and minimize the impact of flight delays on the environment.
- **Weather forecasting:** Weather conditions can have a significant impact on flight delays. Machine learning models can be used to predict weather patterns and conditions, enabling airlines and airports to prepare for potential disruptions and minimize their impact on flight operations.

6. CONCLUSION

- In this project, we use flight data, weather, and demand data to predict flight departure delay. Our result shows that the Random Forest method yields the best performance compared to the ANN model.

- Somehow the ANN model is very time consuming and does not necessarily produce better results. In the end, our model correctly predicts 91% of the non-delayed flights.

- However, the delayed flights are only correctly predicted 41% of time. As a result, there can be additional features related to the causes of flight delay that are not yet discovered using our existing data sources.

- In the second part of the project, we can see that it is possible to predict flight delay patterns from just the volume of concurrently published tweets, and their sentiment and objectivity.

- This is not unreasonable; people tend to post about airport delays on Twitter; it stands to reason that these posts would become more frequent, and more profoundly emotional, as the delays get worse. Without more data, we cannot make a robust model and find out the role of related factors and chance on these results.

- However, as a proof of concept, there is potential for these results. It may be possible to routinely use tweets to ascertain an understanding of concurrent airline delays and traffic patterns, which could be useful in a variety of circumstances.

7. FUTURE SCOPE

- This project is based on data analysis from year 2008. A large dataset is available from 1987-2008 but handling a bigger dataset requires a great amount of preprocessing and cleaning of the data.

- Therefore, the future work of this project includes incorporating a larger dataset. There are many different ways to preprocess a larger dataset like running a Spark cluster over a server or using a cloud based services like AWS and Azure to process the data.

- With the new advancement in the field of deep learning, we can use Neural Networks algorithm on the flight and weather data. Neural Network works on the pattern matching methodology. It is divided into three basic parts for data modeling that includes feed forward networks, feedback networks, and self organization network.

- Feed-forward and feedback networks are generally used in the areas of prediction, pattern recognition, associative memory, and optimization calculation, whereas self organization networks are generally used in cluster analysis. Neural Network offers distributed computer architecture with important learning abilities to represent nonlinear relationships.

8. APPENDIX

Source Code:

Milestone 2:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
import tensorflow
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn import model_selection
from sklearn.neural_network import MLPClassifier
```

```python
[2] #Read the dataset
dataset = pd.read_csv("flightdata.csv")
dataset.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_AR
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	

5 rows × 26 columns

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   YEAR                 11231 non-null  int64
 1   QUARTER              11231 non-null  int64
 2   MONTH                11231 non-null  int64
 3   DAY_OF_MONTH         11231 non-null  int64
 4   DAY_OF_WEEK          11231 non-null  int64
 5   UNIQUE_CARRIER       11231 non-null  object
 6   TAIL_NUM             11231 non-null  object
 7   FL_NUM               11231 non-null  int64
 8   ORIGIN_AIRPORT_ID    11231 non-null  int64
 9   ORIGIN               11231 non-null  object
 10  DEST_AIRPORT_ID      11231 non-null  int64
 11  DEST                 11231 non-null  object
 12  CRS_DEP_TIME         11231 non-null  int64
 13  DEP_TIME             11124 non-null  float64
 14  DEP_DELAY            11124 non-null  float64
 15  DEP_DEL15            11124 non-null  float64
 16  CRS_ARR_TIME         11231 non-null  int64
 17  ARR_TIME             11116 non-null  float64
 18  ARR_DELAY            11043 non-null  float64
 19  ARR_DEL15            11043 non-null  float64
 20  CANCELLED            11231 non-null  float64
 21  DIVERTED             11231 non-null  float64
 22  CRS_ELAPSED_TIME     11231 non-null  float64
 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64
 24  DISTANCE             11231 non-null  float64
 25  Unnamed: 25          0 non-null      float64
dtypes: float64(12), int64(10), object(4)
memory usage: 2.2+ MB
```

```
#removing a last column
dataset = dataset.drop('Unnamed: 25', axis=1)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 25 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   YEAR                 11231 non-null  int64
 1   QUARTER              11231 non-null  int64
 2   MONTH                11231 non-null  int64
 3   DAY_OF_MONTH         11231 non-null  int64
 4   DAY_OF_WEEK          11231 non-null  int64
 5   UNIQUE_CARRIER       11231 non-null  object
 6   TAIL_NUM             11231 non-null  object
 7   FL_NUM               11231 non-null  int64
 8   ORIGIN_AIRPORT_ID    11231 non-null  int64
 9   ORIGIN               11231 non-null  object
 10  DEST_AIRPORT_ID      11231 non-null  int64
 11  DEST                 11231 non-null  object
 12  CRS_DEP_TIME         11231 non-null  int64
 13  DEP_TIME             11124 non-null  float64
 14  DEP_DELAY            11124 non-null  float64
 15  DEP_DEL15            11124 non-null  float64
 16  CRS_ARR_TIME         11231 non-null  int64
 17  ARR_TIME             11116 non-null  float64
 18  ARR_DELAY            11043 non-null  float64
 19  ARR_DEL15            11043 non-null  float64
 20  CANCELLED            11231 non-null  float64
 21  DIVERTED             11231 non-null  float64
 22  CRS_ELAPSED_TIME     11231 non-null  float64
 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64
 24  DISTANCE             11231 non-null  float64
dtypes: float64(11), int64(10), object(4)
```

```
#retrieve need dataset
dataset = dataset[["FL_NUM", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CR
dataset.isnull().sum()
```

```
FL_NUM           0
MONTH            0
DAY_OF_MONTH     0
DAY_OF_WEEK      0
ORIGIN           0
DEST             0
CRS_ARR_TIME     0
DEP_DEL15      107
ARR_DEL15      188
dtype: int64
```

```
dataset[dataset.isnull().any(axis=1)].head(10)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	NaN
179	86	1	10	7	MSP	DTW	1632	NaN	NaN
184	557	1	10	7	MSP	DTW	912	0.0	NaN
210	1096	1	10	7	DTW	MSP	1303	NaN	NaN
478	1542	1	22	5	SEA	JFK	723	NaN	NaN
481	1795	1	22	5	ATL	JFK	2014	NaN	NaN
491	2312	1	22	5	MSP	JFK	2149	NaN	NaN
499	423	1	23	6	JFK	ATL	1600	NaN	NaN
500	425	1	23	6	JFK	ATL	1827	NaN	NaN
501	427	1	23	6	JFK	SEA	1053	NaN	NaN

```
[78] dataset['DEP_DEL15'].mode()
```

```
0    0.0
Name: DEP_DEL15, dtype: float64
```

```
[79] #changing null values
dataset = dataset.fillna({"ARR_DEL15" : 1})
dataset = dataset.fillna({"DEP_DEL15" : 0})
dataset.iloc[177:185]
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	1.0
178	2839	1	9	6	DTW	JFK	1724	0.0	0.0
179	86	1	10	7	MSP	DTW	1632	0.0	1.0
180	87	1	10	7	DTW	MSP	1649	1.0	0.0
181	423	1	10	7	JFK	ATL	1600	0.0	0.0
182	440	1	10	7	JFK	ATL	849	0.0	0.0
183	485	1	10	7	JFK	SEA	1945	1.0	0.0

```
[80] #convert CRS_ARR_TIME
for index, row in dataset.iterrows():
    dataset.loc[index, 'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME'] / 100)
dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	ATL	SEA	21	0.0	0.0
1	1476	1	1	5	DTW	MSP	14	0.0	0.0
2	1597	1	1	5	ATL	SEA	12	0.0	0.0
3	1768	1	1	5	SEA	MSP	13	0.0	0.0
4	1823	1	1	5	SEA	DTW	6	0.0	0.0

```
#convert DEST & ORIGIN using LabelEncoder
le = LabelEncoder()
dataset['DEST'] = le.fit_transform(dataset['DEST'])
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

```
[82] dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	0	4	21	0.0	0.0
1	1476	1	1	5	1	3	14	0.0	0.0
2	1597	1	1	5	0	4	12	0.0	0.0
3	1768	1	1	5	4	3	13	0.0	0.0
4	1823	1	1	5	4	1	6	0.0	0.0

```
[83] dataset["ORIGIN"].unique()
array([0, 1, 4, 3, 2])
```

```
[84] dataset.iloc[:,8:9]
```

	ARR_DEL15
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
11226	0.0
11227	0.0
11228	0.0
11229	0.0
11230	0.0

11231 rows × 1 columns

Milestone 3:

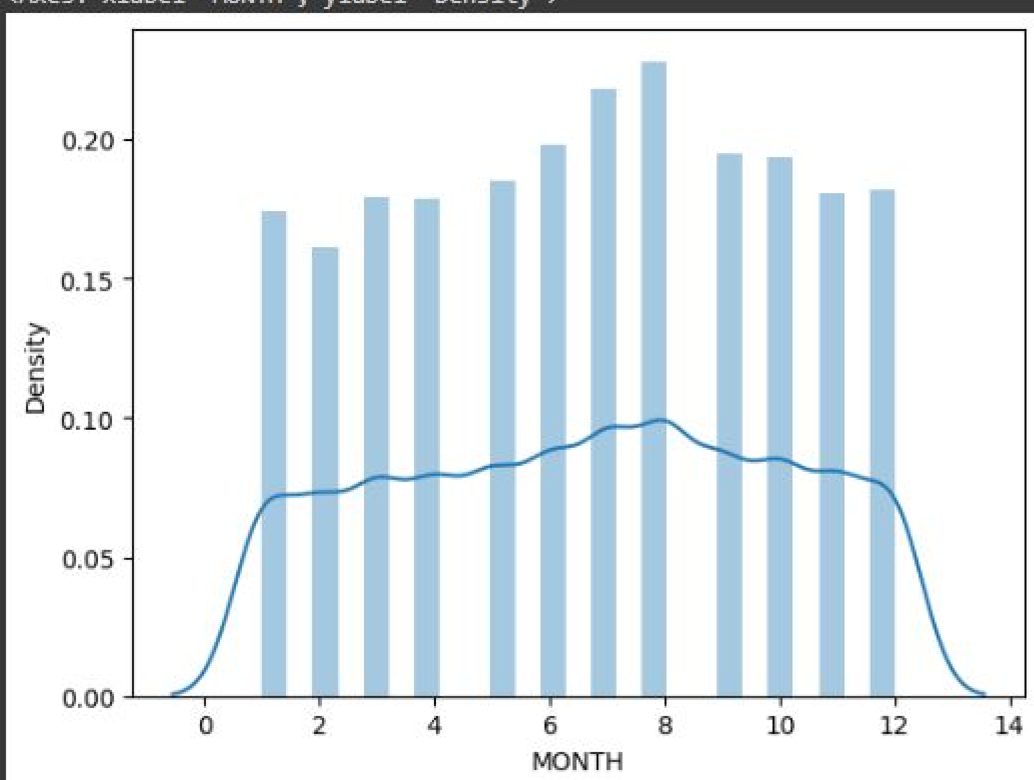
```
[85] #TASK 3
#descriptive analysis
data = pd.read_csv("flightdata.csv")
data.describe()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	112
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	13
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	4
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	9
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	13
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	17
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	23

8 rows × 22 columns

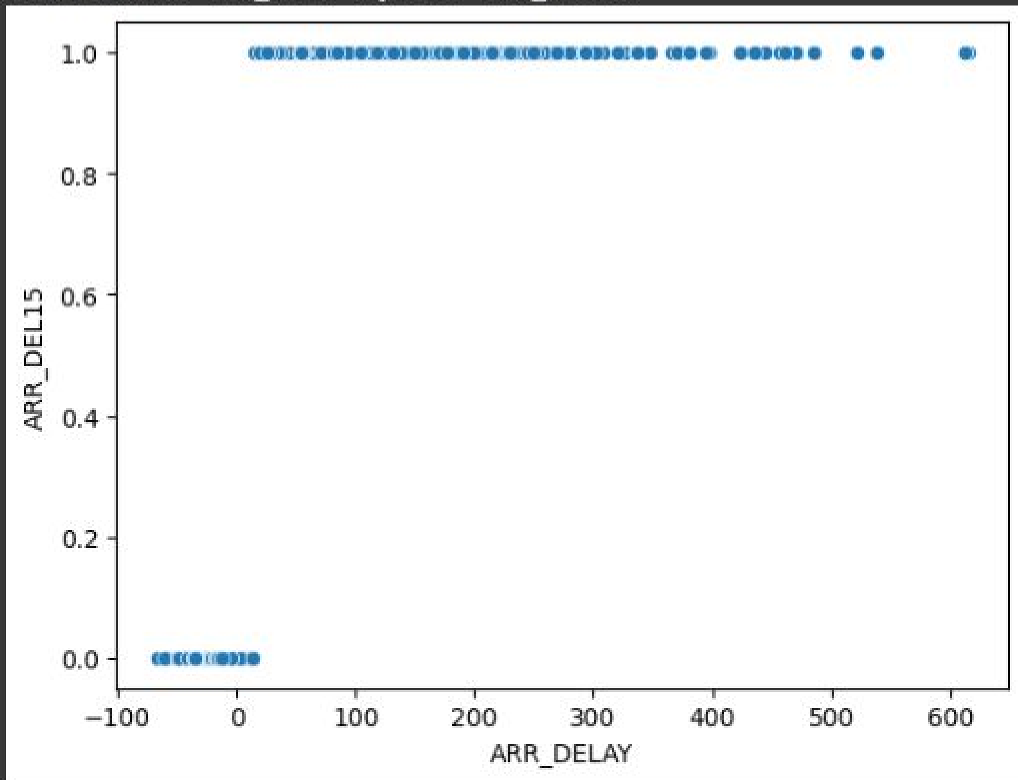
```
#univariate analysis
sns.distplot(data.MONTH)
```

<Axes: xlabel='MONTH', ylabel='Density'>



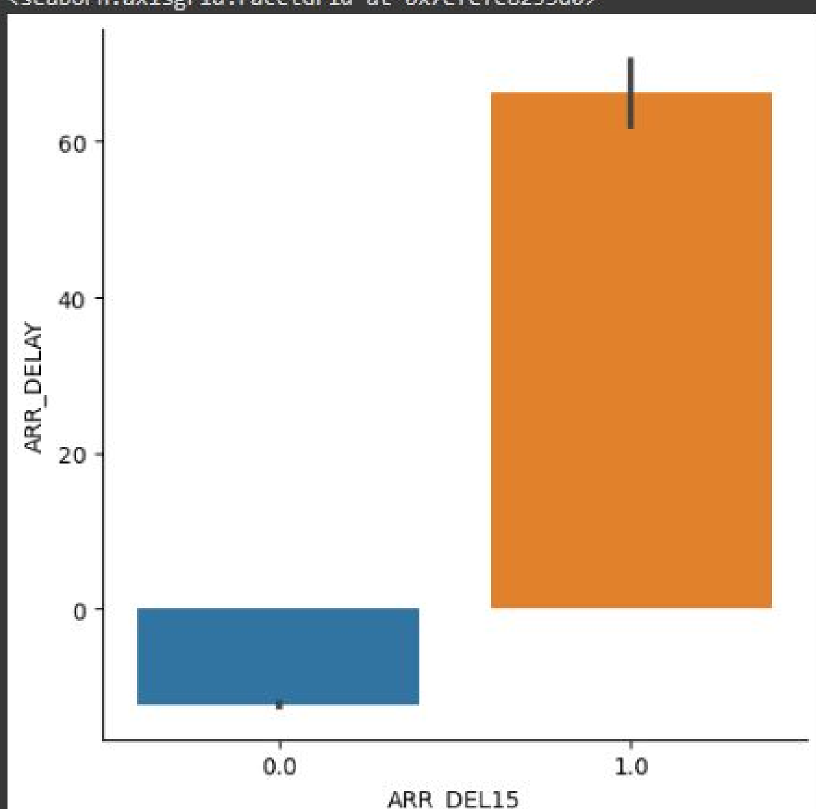
```python
#bivariate analysis
sns.scatterplot(x='ARR_DELAY', y='ARR_DEL15', data=data)
```

<Axes: xlabel='ARR_DELAY', ylabel='ARR_DEL15'>




```python
sns.catplot(x='ARR_DEL15', y='ARR_DELAY', kind='bar', data=data)
```

<seaborn.axisgrid.FacetGrid at 0x7efefe8253d0>



```
[228] #splitting data into dependent & independent
      x = dataset.iloc[:, 0:8].values
      y = dataset.iloc[:, 8:9].values
      x
```

```
array([[1.399e+03, 1.000e+00, 1.000e+00, ..., 4.000e+00, 2.100e+01,
        0.000e+00],
       [1.476e+03, 1.000e+00, 1.000e+00, ..., 3.000e+00, 1.400e+01,
        0.000e+00],
       [1.597e+03, 1.000e+00, 1.000e+00, ..., 4.000e+00, 1.200e+01,
        0.000e+00],
       ...,
       [1.823e+03, 1.200e+01, 3.000e+01, ..., 4.000e+00, 2.200e+01,
        0.000e+00],
       [1.901e+03, 1.200e+01, 3.000e+01, ..., 4.000e+00, 1.800e+01,
        0.000e+00],
       [2.005e+03, 1.200e+01, 3.000e+01, ..., 1.000e+00, 9.000e+00,
        0.000e+00]])
```

```
[229] y
```

```
array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [0.]])
```

```
[232] #OneHotEncoder
      oh = OneHotEncoder()
      z = oh.fit_transform(x[:,4:5]).toarray()
      t = oh.fit_transform(x[:,5:6]).toarray()
      #x = np.delete(x,[4,7],axis=1)
```

```
[233] z
```

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       ...,
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.]])
```

```
[234] t
```

```
array([[0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.],
       ...,
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0.]])
```

```
[235] x = np.delete(x, [4,5], axis=1)
```

```
[236] x.shape
```

```
(11231, 6)
```

```
[237] x = np.concatenate((t,z,x), axis=1)
```

```
[238] x.shape
```

```
(11231, 16)
```

```
[239] dataset=pd.get_dummies(dataset,columns=['ORIGIN','DEST'])
      dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	ORIGIN_4	DEST_0	DEST_1	DEST_2	DEST_3	DEST_4
0	1399	1	1	5	21	0.0	0.0	1	0	0	0	0	0	0	0	0	1
1	1476	1	1	5	14	0.0	0.0	0	1	0	0	0	0	0	0	1	0
2	1597	1	1	5	12	0.0	0.0	1	0	0	0	0	0	0	0	0	1
3	1768	1	1	5	13	0.0	0.0	0	0	0	0	1	0	0	0	1	0
4	1823	1	1	5	6	0.0	0.0	0	0	0	0	1	0	1	0	0	0

```
[240] #Splitting data into train and test
      x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=0)
```

```
[241] x_test.shape
```

```
(2247, 16)
```

```
[242] x_train.shape
```

```
(8984, 16)
```

```
[243] y_test.shape
```

```
(2247, 1)
```

```
[244] y_train.shape
```

```
(8984, 1)
```

```
[245] #Scalling the data
      sc = StandardScaler()
      x_train = sc.fit_transform(x_train)
      x_test = sc.fit_transform(x_test)
```

Milestone 4:

```
[246] #TASK 4 - Model building
      #Decision tree model
      classifier = DecisionTreeClassifier(random_state = 0)
      classifier.fit(x_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

```
[247] decisiontree = classifier.predict(x_test)
      decisiontree
```

```
array([1., 0., 0., ..., 0., 0., 1.])
```

```
[248] #check accuracy
      desacc = accuracy_score(y_test, decisiontree)
      desacc
```

```
0.8673787271918113
```

```
[249] #Random forest model
      rfc=RandomForestClassifier(n_estimators=10,criterion='entropy')
      rfc.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

```
[250] y_predict=rfc.predict(x_test)
      y_predict
```

```
array([1., 0., 0., ..., 0., 0., 1.])
```

```
[251] #ANN model
      from keras.api._v2.keras import activations
      #creating ANN skleton view
      classification = Sequential()
      classification.add(Dense(30,activation='relu'))
      classification.add(Dense(128,activation='relu'))
      classification.add(Dense(64,activation='relu'))
      classification.add(Dense(32,activation='relu'))
      classification.add(Dense(1,activation='sigmoid'))
```

```
#compilling the ANN model
classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

#Training the model
classification.fit(x_train, y_train, batch_size=4, validation_split=0.2, epochs=100)
```

```
Epoch 1/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.2876 - accuracy: 0.8945 - val_loss: 0.2731 - val_accuracy: 0.9060
Epoch 2/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.2685 - accuracy: 0.9047 - val_loss: 0.2778 - val_accuracy: 0.8987
Epoch 3/100
1797/1797 [==============================] - 6s 3ms/step - loss: 0.2654 - accuracy: 0.9062 - val_loss: 0.2913 - val_accuracy: 0.8915
Epoch 4/100
```

```
Epoch 96/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0642 - accuracy: 0.9754 - val_loss: 1.0650 - val_accuracy: 0.8703
Epoch 97/100
1797/1797 [==============================] - 4s 2ms/step - loss: 0.0637 - accuracy: 0.9745 - val_loss: 1.1772 - val_accuracy: 0.8587
Epoch 98/100
1797/1797 [==============================] - 3s 2ms/step - loss: 0.0555 - accuracy: 0.9784 - val_loss: 1.2497 - val_accuracy: 0.8715
Epoch 99/100
1797/1797 [==============================] - 3s 2ms/step - loss: 0.0640 - accuracy: 0.9759 - val_loss: 1.0644 - val_accuracy: 0.8709
Epoch 100/100
1797/1797 [==============================] - 3s 2ms/step - loss: 0.0619 - accuracy: 0.9758 - val_loss: 1.1291 - val_accuracy: 0.8614
<keras.callbacks.History at 0x7efefe82b6d0>
```

```
[253] #Activity 2: Test the model
      #Decision tree
      y_pred=classifier.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])
      y_pred
```

```
array([0.])
```

```
[254] #RandomForest
      y_pred=rfc.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])
      y_pred
```

```
array([0.])
```

```
[255] classification.save('flight.h5')
```

```
[256] #Testing the model
      y_pred=classification.predict(x_test)
      y_pred
```

```
71/71 [==============================] - 0s 1ms/step
array([[8.3022517e-01],
       [1.8294374e-21],
       [2.5718522e-01],
       ...,
       [3.2494348e-17],
       [4.7324735e-01],
       [8.7308043e-01]], dtype=float32)
```

```
[257] y_pred = (y_pred>0.5)
      y_pred
```

```
array([[ True],
       [False],
       [False],
       ...,
       [False],
       [False],
       [ True]])
```

```
[258] def predict_exit(sample_value):
        #convert list to numpy array
        sample_value=np.array(sample_value)
        #Reshape because sample_value is contains only 1 value
        sample_value=sample_value.reshape(1,-1)
        #Feature scaling
        sample_value=sc.transform(sample_value)
        return classifier.predict(sample_value)
```

```
[259] test=classification.predict([[1,1,121.000000,36.0,0,0,1,0,1,1,1,1,1,1,1,1,]])
      if test==1:
        print('Prediction: Chance Of Delay')
      else:
        print('Prediction: No Chance Of Delay')
```

```
1/1 [==============================] - 0s 38ms/step
Prediction: No Chance Of Delay
```

Milestone 5:

```
[260] #Task-5 Performance Testing & Hyperparameter Tuning
      #Compare the model

      def classification_report():
        dfs=[]
        models=[
              ('RF',RandomForestClassifier()),
              ('DecisionTree',DecisionTreeClassifier()),
              ('ANN',MLPClassifier())
            ]
        results=[]
        names=[]
        scoring=['accuracy','precision_weighted','recall_weighted','f1_weighted','roc_auc']
        target_names=['no delay','delay']
        for name,model in models:
          kfold=model_selection.KFold(n_splits=5, shuffle=True, random_state=90210)
          cv_results=model_selection.cross_validate(model,x_train, y_train, cv=kfold, scoring=scoring)
          clf=model.fit(x_train, y_train)
          y_pred=clf.predict(x_test)
          print(name)
          print(classification_report(y_test,y_pred,target_names=target_names))
          results.append(cv_results)
          names.append(name)
          this_df=pd.DataFrame(cv_results)
          this_df['models']=name
          dfs.append(this_df)
          final=pd.concat(dfs,ignore_index=True)
        return final
```

```
[261] #RandomForest Accuracy
      print('Training Accuracy: ',accuracy_score(y_pred, y_predict))
      print('Testing Accuracy: ',accuracy_score(y_test, y_predict))
```

```
Training Accuracy:  0.9141076991544281
Testing Accuracy:  0.8945260347129506
```

```
[262] # Making the Confusion matrix
      from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_test, y_predict)
      cm
```

```
array([[1873,   63],
       [ 174,  137]])
```

```
[263] # Accuracy score of decision tree
      desacc = accuracy_score(y_test,decisiontree)
      desacc
```

```
0.8673787271918113
```

```
[264] from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_test,decisiontree)
      cm
```

```
array([[1778,  158],
       [ 140,  171]])
```

```
[265] from sklearn.metrics import accuracy_score,classification_report
      score = accuracy_score(y_pred,y_test)
      print('The accuracy for ANN model is: {}%'.format(score*100))
```

```
The accuracy for ANN model is: 87.18291054739653%
```

```
[266] # Making the Confusion matrix
      from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_test, y_pred)
      cm
```

```
array([[1797,  139],
       [ 149,  162]])
```

```
[267] # Activity 2: Hyperparameter tuning
      # giving some parameters
      parameters = {
          'n_estimators' : [1,20,30,55,68,74,90,120,115],
          'criterion' : ['gini', 'entropy'],
          'max_features' : ['auto', 'sqrt', 'log2'],
          'max_depth' : [2,5,8,10], 'verbose' : [1,2,3,4,6,8,9,10]
      }
```

```
[268] # Performing the randomized cv
      from sklearn.model_selection import RandomizedSearchCV
      RCV = RandomizedSearchCV(estimator = rfc, param_distributions = parameters, cv = 10, n_iter = 4)
```

```
[269] RCV.fit(x_train, y_train)

building tree 1 of 115
building tree 2 of 115
building tree 3 of 115
building tree 4 of 115
building tree 5 of 115
building tree 6 of 115
building tree 7 of 115
building tree 8 of 115
building tree 9 of 115
building tree 10 of 115
building tree 11 of 115
building tree 12 of 115
building tree 13 of 115
building tree 14 of 115
building tree 15 of 115
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
building tree 16 of 115
```

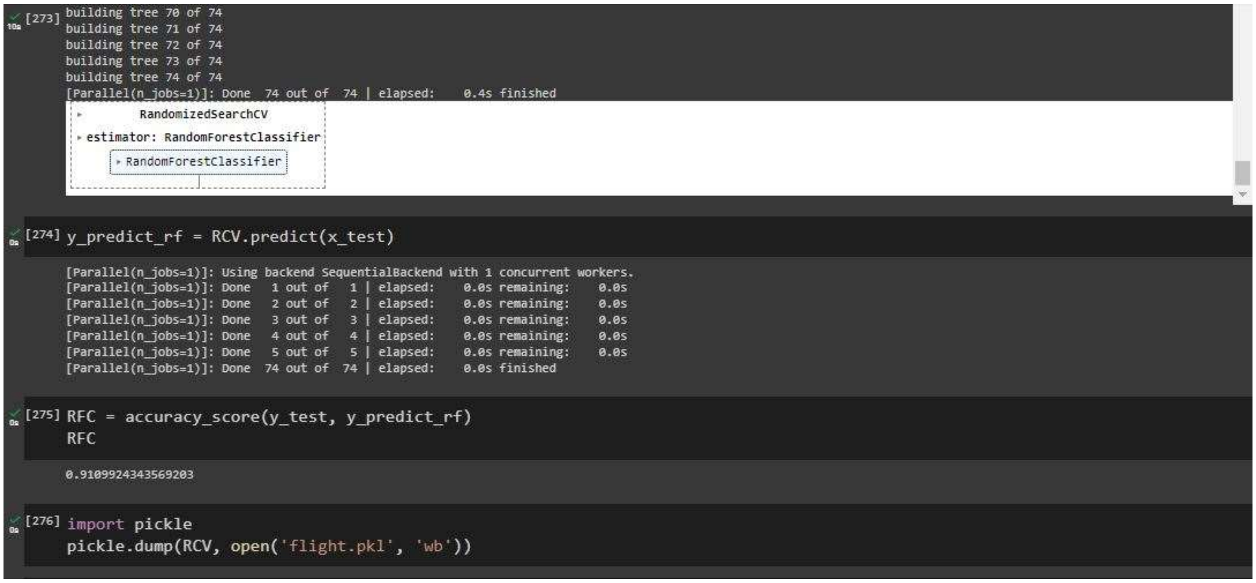
```
[269] building tree 50 of 55
building tree 51 of 55
building tree 52 of 55
building tree 53 of 55
building tree 54 of 55
building tree 55 of 55
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:    0.4s finished
RandomizedSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

```
[270] # Getting the best parameters

bt_params = RCV.best_params_
bt_score = RCV.best_score_
```

```
[271] bt_params

{'verbose': 2,
 'n_estimators': 55,
 'max_features': 'auto',
 'max_depth': 10,
 'criterion': 'entropy'}
```

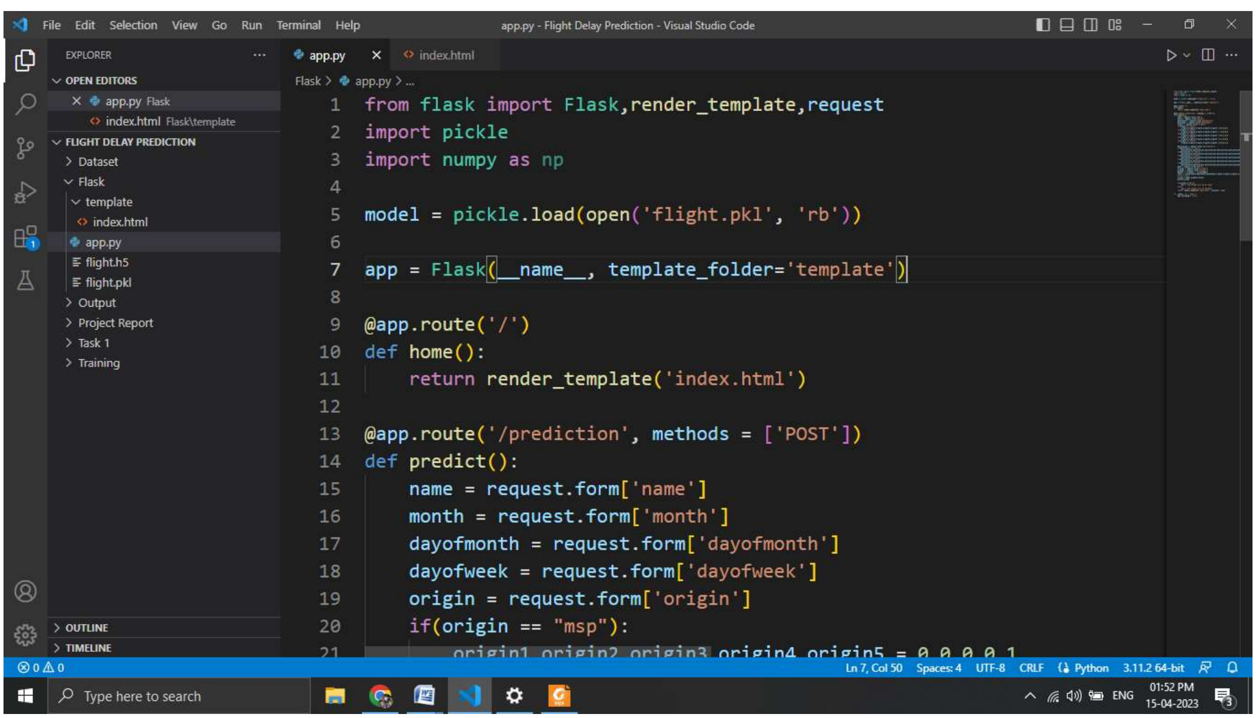
```
[272] bt_score

0.9036063331937738
```

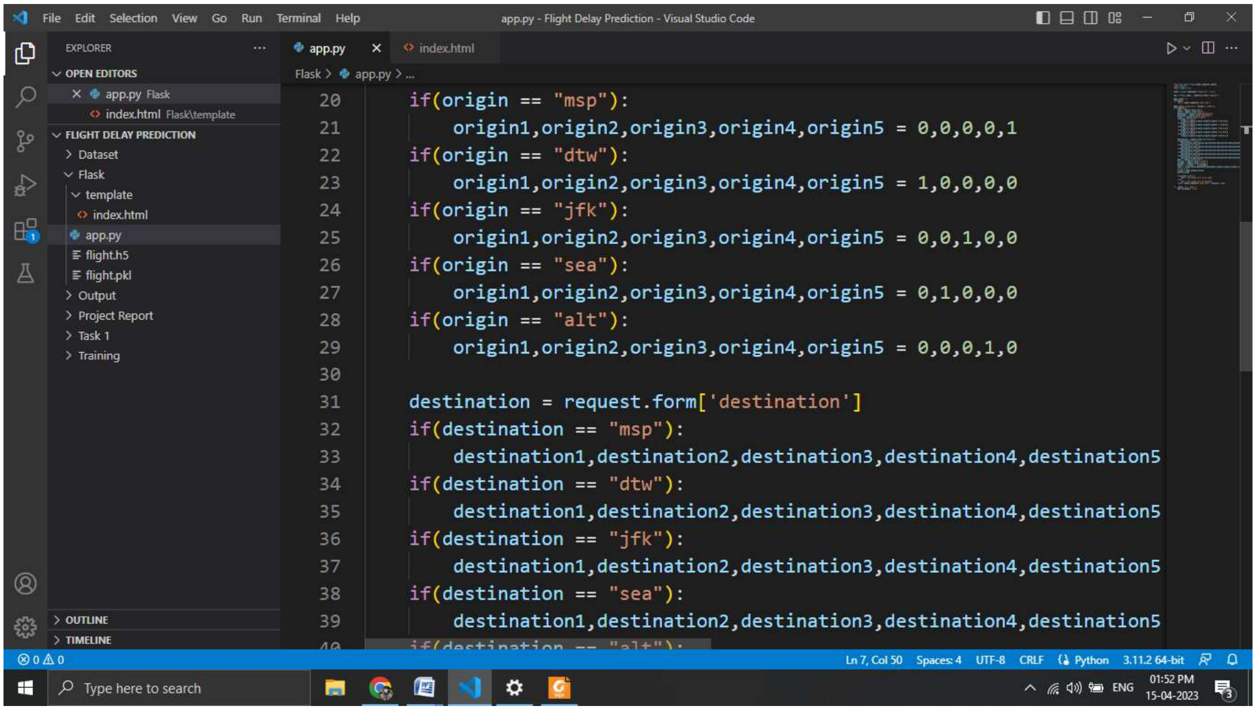
```
[273] model = RandomForestClassifier(verbose = 10, n_estimators = 120, max_features = 'log2', max_depth = 10, criterion = 'entro
RCV.fit(x_train, y_train)

building tree 1 of 30
building tree 2 of 30
building tree 3 of 30
building tree 4 of 30
building tree 5 of 30
building tree 6 of 30
building tree 7 of 30
building tree 8 of 30
building tree 9 of 30
building tree 10 of 30
```

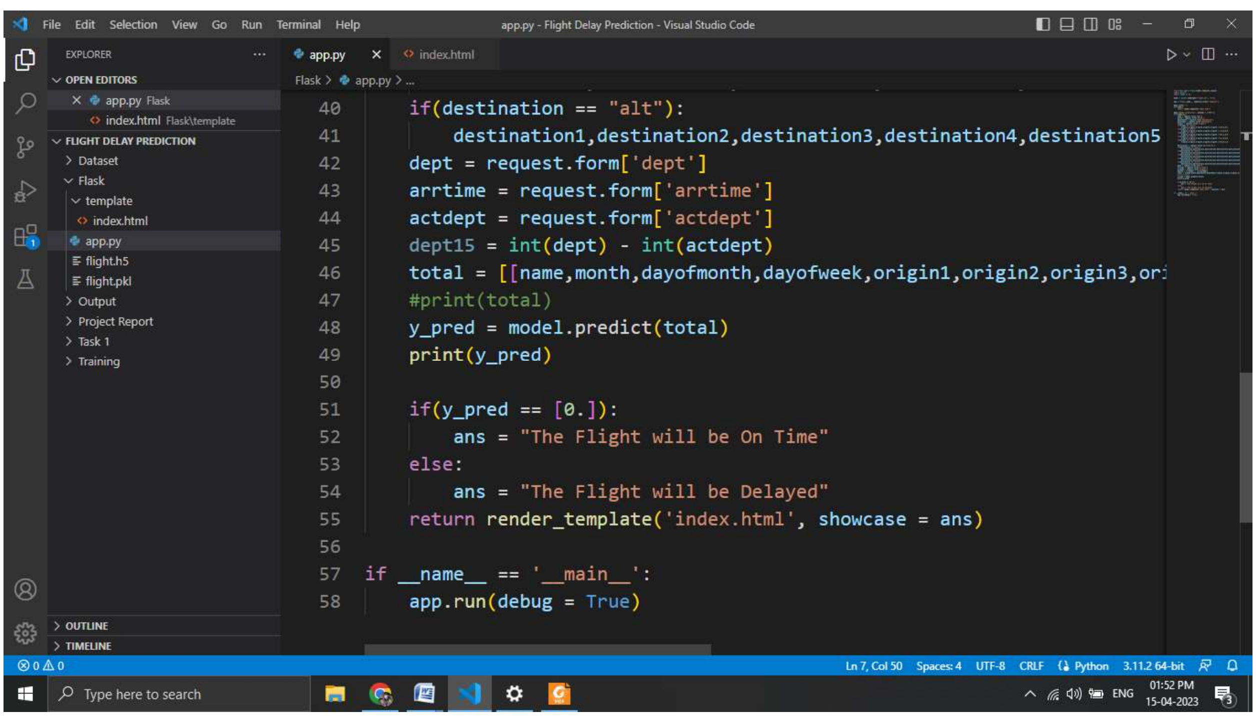


Milestone 6:

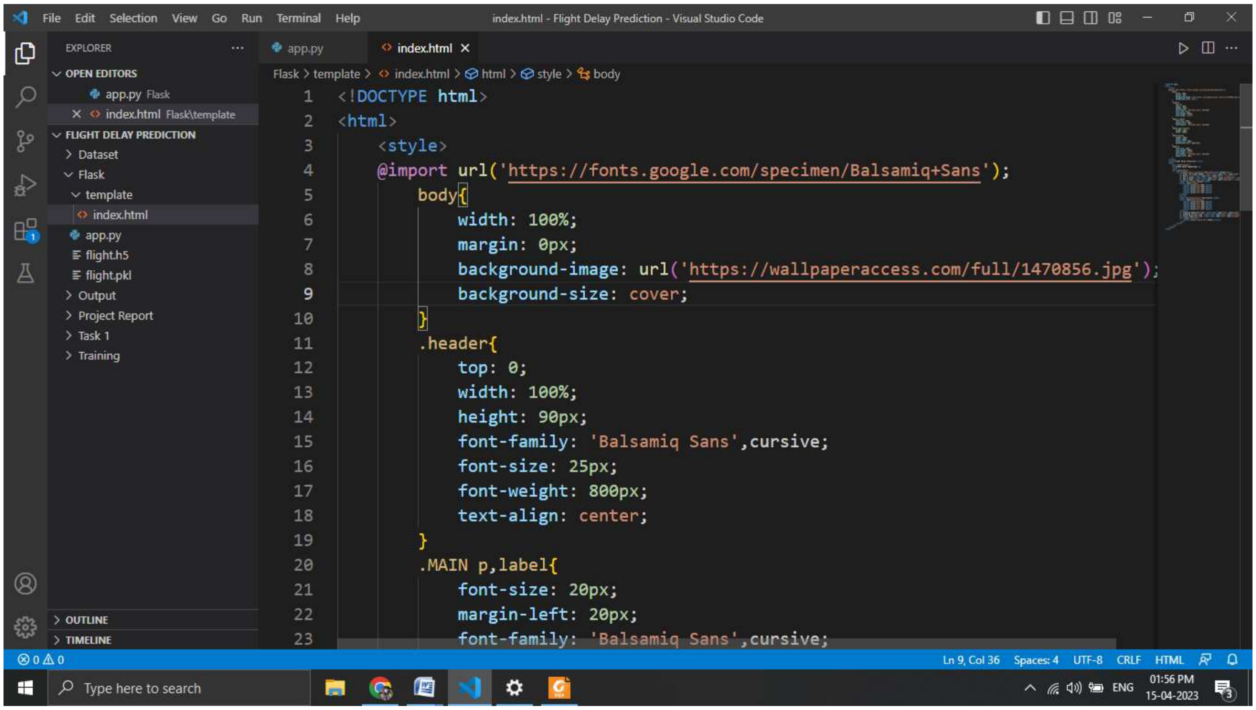
Flask file (app.py):

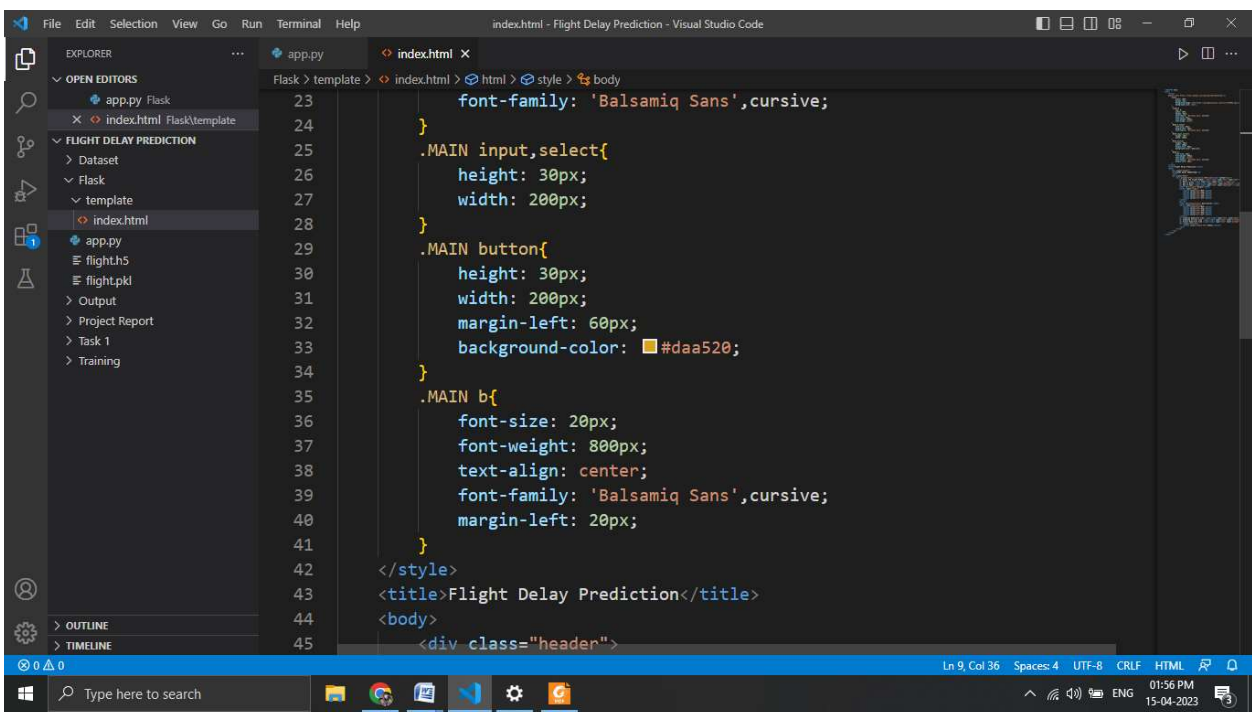


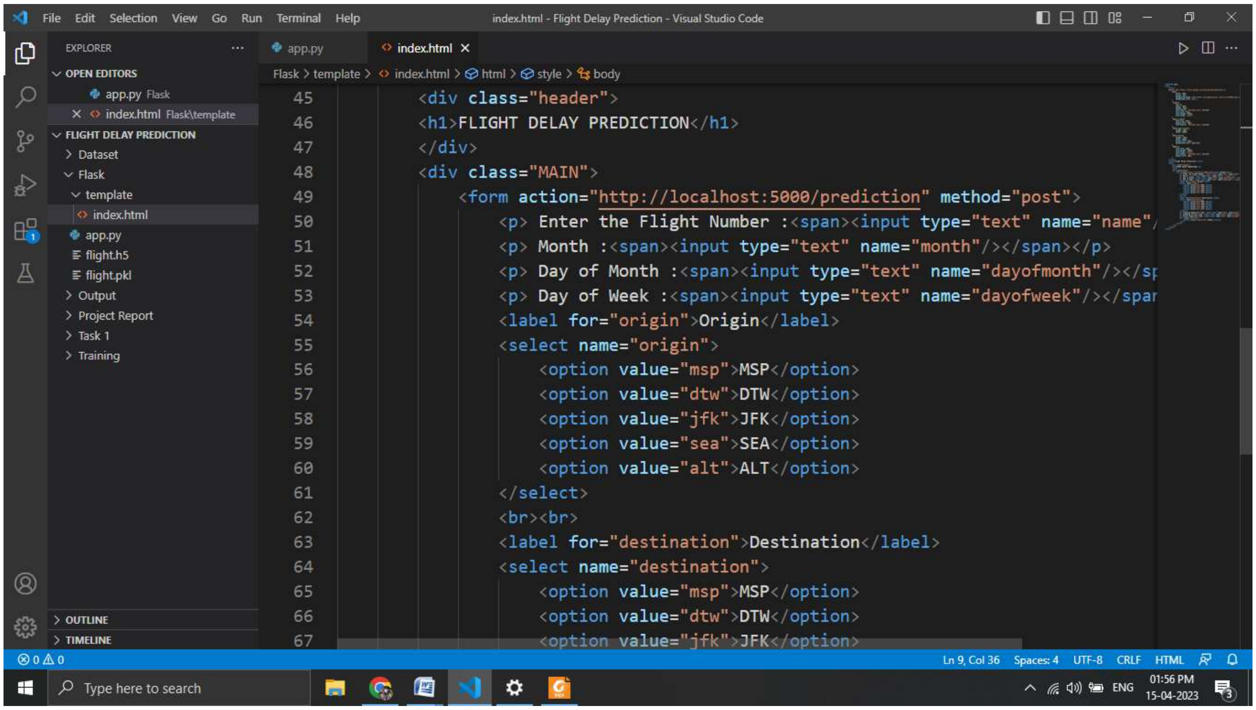

File Edit Selection View Go Run Terminal Help
app.py - Flight Delay Prediction - Visual Studio Code
EXPLORER
OPEN EDITORS
app.py Flask
index.html Flask\template
FLIGHT DELAY PREDICTION
Dataset
Flask
template
index.html
app.py
flight.h5
flight.pkl
Output
Project Report
Task 1
Training
OUTLINE
TIMELINE
app.py
index.html
Flask > app.py > ...
if(origin == "msp"):
origin1,origin2,origin3,origin4,origin5 = 0,0,0,0,1
if(origin == "dtw"):
origin1,origin2,origin3,origin4,origin5 = 1,0,0,0,0
if(origin == "jfk"):
origin1,origin2,origin3,origin4,origin5 = 0,0,1,0,0
if(origin == "sea"):
origin1,origin2,origin3,origin4,origin5 = 0,1,0,0,0
if(origin == "alt"):
origin1,origin2,origin3,origin4,origin5 = 0,0,0,1,0
destination = request.form['destination']
if(destination == "msp"):
destination1,destination2,destination3,destination4,destination5
if(destination == "dtw"):
destination1,destination2,destination3,destination4,destination5
if(destination == "jfk"):
destination1,destination2,destination3,destination4,destination5
if(destination == "sea"):
destination1,destination2,destination3,destination4,destination5
Ln 7, Col 50 Spaces: 4 UTF-8 CRLF Python 3.11.2 64-bit
Type here to search
01:52 PM
15-04-2023


File Edit Selection View Go Run Terminal Help
app.py - Flight Delay Prediction - Visual Studio Code
EXPLORER
OPEN EDITORS
app.py Flask
index.html Flask\template
FLIGHT DELAY PREDICTION
Dataset
Flask
template
index.html
app.py
flight.h5
flight.pkl
Output
Project Report
Task 1
Training
OUTLINE
TIMELINE
app.py
index.html
Flask > app.py > ...
if(destination == "alt"):
destination1,destination2,destination3,destination4,destination5
dept = request.form['dept']
arrtime = request.form['arrtime']
actdept = request.form['actdept']
dept15 = int(dept) - int(actdept)
total = [[name,month,dayofmonth,dayofweek,origin1,origin2,origin3,ori
#print(total)
y_pred = model.predict(total)
print(y_pred)
if(y_pred == [0.]):
ans = "The Flight will be On Time"
else:
ans = "The Flight will be Delayed"
return render_template('index.html', showcase = ans)
if __name__ == '__main__':
app.run(debug = True)
Ln 7, Col 50 Spaces: 4 UTF-8 CRLF Python 3.11.2 64-bit
Type here to search
01:52 PM
15-04-2023

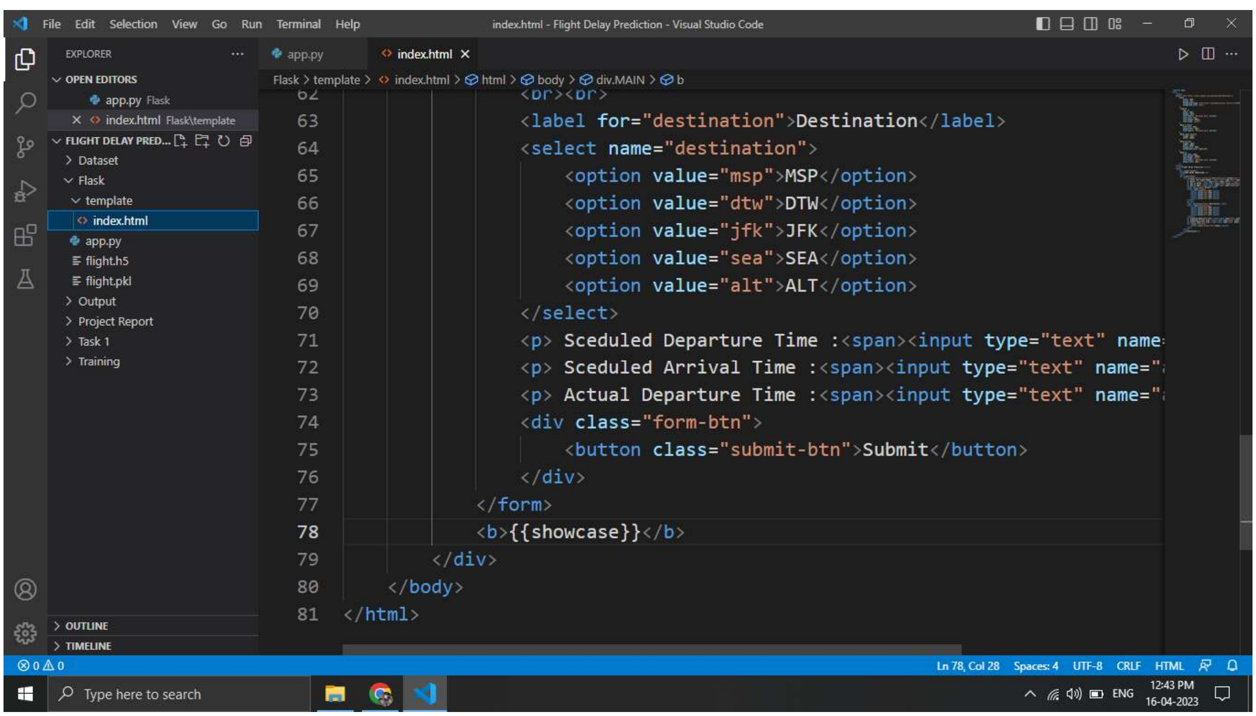
User Interface (index.html):





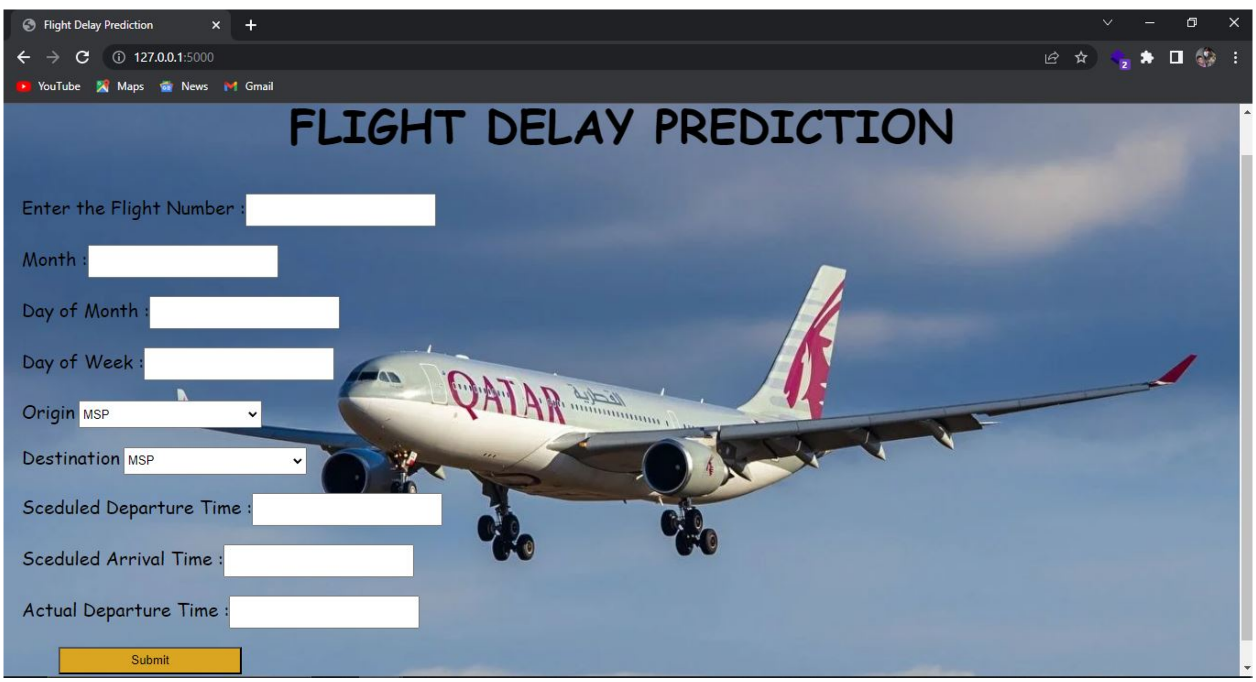

index.html - Flight Delay Prediction - Visual Studio Code
<div class="header">
<h1>FLIGHT DELAY PREDICTION</h1>
</div>
<div class="MAIN">
<form action="http://localhost:5000/prediction" method="post">
<p> Enter the Flight Number :<input type="text" name="name"
<p> Month :<input type="text" name="month"/></p>
<p> Day of Month :<span><input type="text" name="dayofmonth"/></sp
<p> Day of Week :<span><input type="text" name="dayofweek"/></spar
<label for="origin">Origin</label>
<select name="origin">
<option value="msp">MSP</option>
<option value="dtw">DTW</option>
<option value="jfk">JFK</option>
<option value="sea">SEA</option>
<option value="alt">ALT</option>
</select>

<label for="destination">Destination</label>
<select name="destination">
<option value="msp">MSP</option>
<option value="dtw">DTW</option>
<option value="jfk">JFK</option>


index.html - Flight Delay Prediction - Visual Studio Code
<label for="destination">Destination</label>
<select name="destination">
<option value="msp">MSP</option>
<option value="dtw">DTW</option>
<option value="jfk">JFK</option>
<option value="sea">SEA</option>
<option value="alt">ALT</option>
</select>
<p> Sceduled Departure Time :<span><input type="text" name
<p> Sceduled Arrival Time :<span><input type="text" name="
<p> Actual Departure Time :<span><input type="text" name="
<div class="form-btn">
<button class="submit-btn">Submit</button>
</div>
</form>
{{showcase}}
</div>
</body>
</html>

Output:

Now, Go the web browser and write the localhost URL (http://127.0.0.1:5000) to get the below result



Input – Now, the user will give inputs to get the predicted result after clicking onto the submit button.