FLIGHT DELAY PREDICTION FOR AVIATION INDUSTRY USING MACHINE LEARNING

DONE BY,
Sabith .S
Sandhiya .G
Selvi .M
Sujitha .S

INTRODUCTION

1.1 Overview

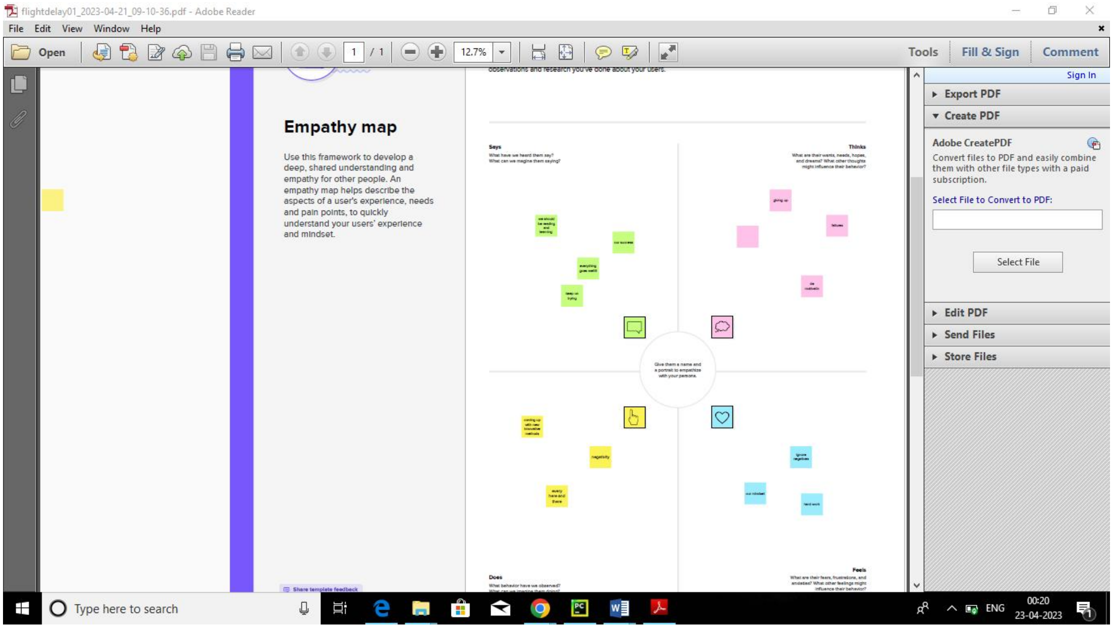
Over the last twenty years, air travel has been increasingly preferred among travelers, mainly
because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also resulted in massive levels of aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses. The main objective of the model is to predict flight delays accurately in order to optimize flight operations and minimize delays.

1.2 Purpose

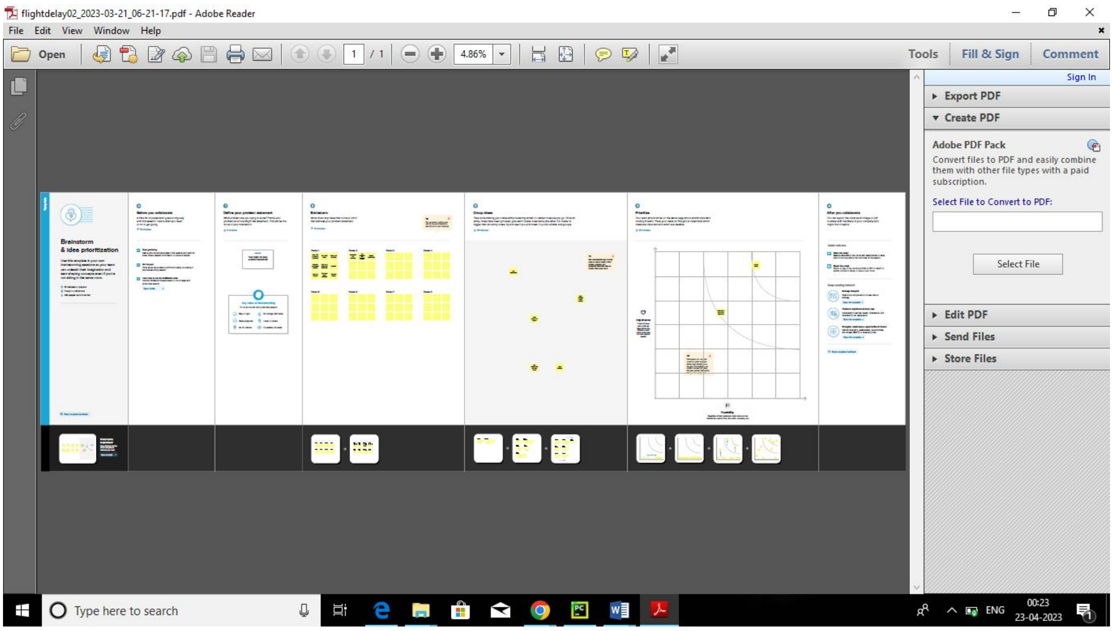
Using a machine learning model, we can predict flight arrival delays. The input to our algorithm. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is considered to be delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit.

2 PROBLEM DEFINITION & DESIGN THINKING

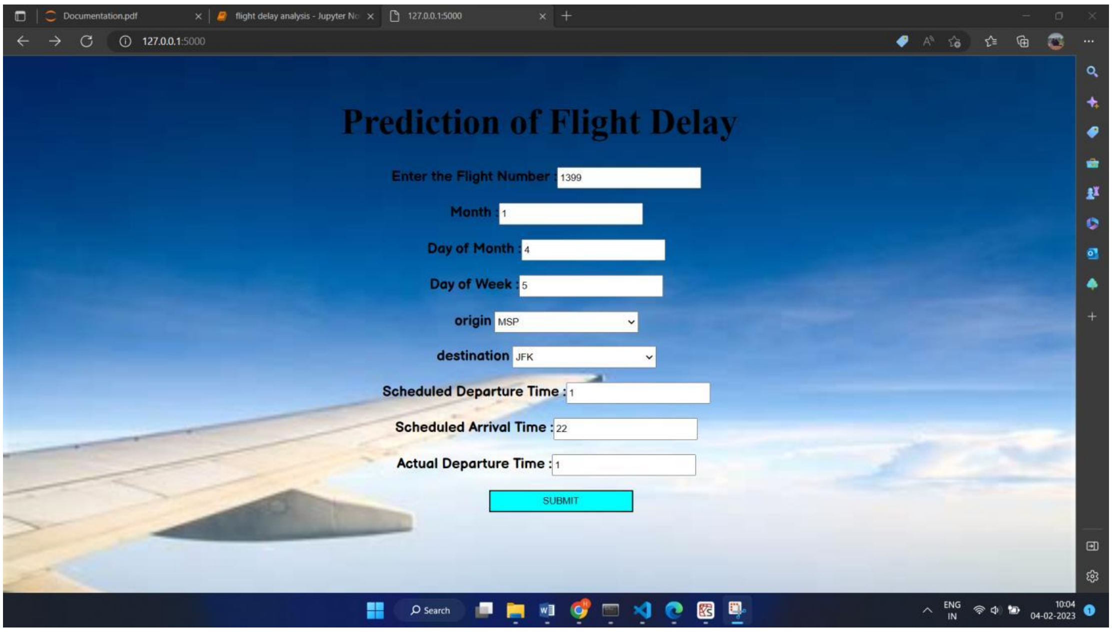
2.1 EMPATHY MAP

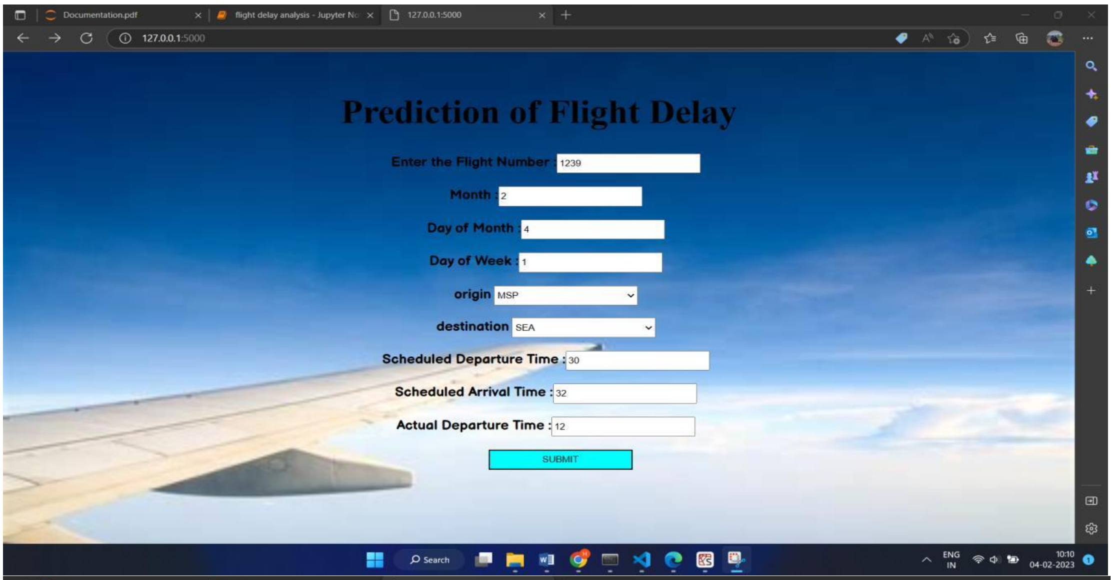


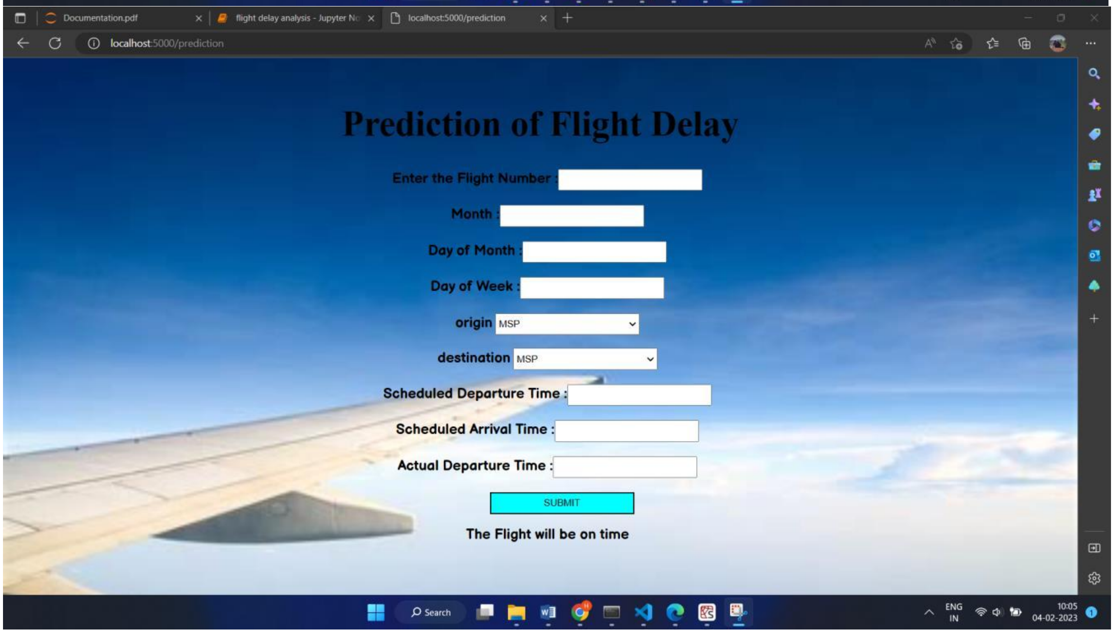
3. BRAIN STROME



6. RESULT


Documentation.pdf
flight delay analysis - Jupyter No
127.0.0.1:5000
127.0.0.1:5000
Prediction of Flight Delay
Enter the Flight Number : 1399
Month : 1
Day of Month : 4
Day of Week : 5
origin MSP
destination JFK
Scheduled Departure Time : 1
Scheduled Arrival Time : 22
Actual Departure Time : 1
SUBMIT
Search
ENG
IN
10:04
04-02-2023





7. ADVANTAGES AND DISADVANTAGES

Advantage: Using the flight delay system we can predict whether the flight will departure late
when compared to the scheduled departure time.

Disadvantage: To use this system we need both scheduled departure time and actual departure
time to calculate the delay.

8. APPLICATIONS

This can be applied for customers who wait for confirmation if the flight will arrive or will get

delayed through customer service for a long time. Customers will get to know their answer
pretty quick also.

9. CONCLUSION

Following this project, it is likely that the choice of approaches that can be utilised to produce
notable results will be heavily influenced by the dataset's balance. Many machine learning
models, such as Decision Tree Classifier, have been used to predict airplane arrival and delays.
We were able to acquire a quick answer about the flight status thanks to IBM Cloud and the
Flask application.

10. F UTURE SCOPE

Many machinelearning models canbeusedto forecast airline arrivaldelays, including Logistic
Regression, Random Forest Regression, Linear Regression, and its variation Boosted Linear
Regression. Even these algorithms will be able to forecast delays with excellent accuracy when
given the proper combination of input parameters. We can forecast arrival delay even without
including departure delay as an attribute if weather and air traffic control information are made
available. We can also estimate whether a flight will be delayed or cancelled depending on
weather elements such as snow, rain, or storms.

APPENDIX
Source code:
Jupyter notebook

In [1]:
```
import sys
import numpy
import pandas as pd
import numpy as np
```

In [2]:
```
dataset= pd.read_csv("flightdata.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_ARR_TIME	AI
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	2143	
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	1435	
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	1215	
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	1335	
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	607	

5 rows × 26 columns

In [4]:
```
dataset.isnull().any()
```

In [6]:
```
dataset['DEST'].unique()
```

Out[6]:
```
array(['SEA', 'MSP', 'DTW', 'ATL', 'JFK'], dtype=object)
```

In [7]:
```
dataset = dataset.drop('Unnamed: 25', axis=1)
dataset.isnull().sum()
```

Out[7]:
```
YEAR                   0
QUARTER                0
MONTH                  0
DAY_OF_MONTH           0
DAY_OF_WEEK            0
UNIQUE_CARRIER         0
TAIL_NUM               0
FL_NUM                 0
ORIGIN_AIRPORT_ID      0
ORIGIN                 0
DEST_AIRPORT_ID        0
DEST                   0
CRS_DEP_TIME           0
DEP_TIME             107
DEP_DELAY            107
DEP_DEL15            107
CRS_ARR_TIME           0
ARR_TIME             115
ARR_DELAY            188
```

In [8]:
```
import seaborn as sns
%matplotlib inline
```

In [9]:
```
flight_data = pd.read_csv('flightdata.csv')
flight_data.describe()
```


In [8]:
```
import seaborn as sns
%matplotlib inline
```

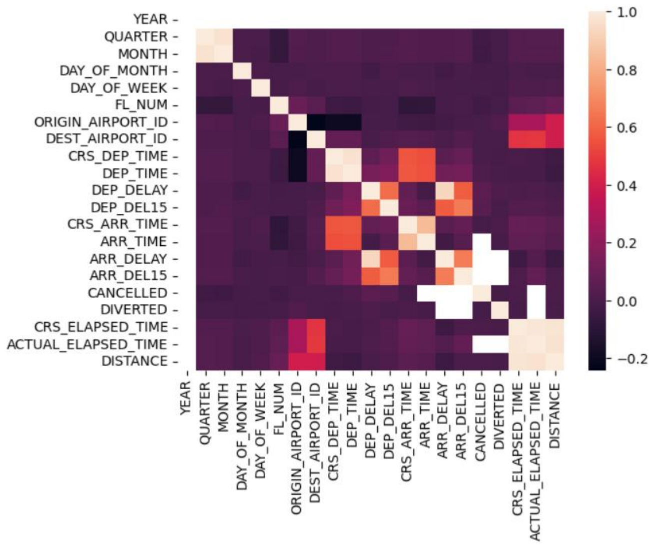
In [9]:
```
flight_data = pd.read_csv('flightdata.csv')
flight_data.describe()
```


Out[9]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.00
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.18
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.30
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.00
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.00
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.00
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.00
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.00

8 rows × 22 columns

In [12]:
```
sns.heatmap(dataset.corr())
```
Out[12]: <AxesSubplot:>



In [32]:
```
#filter the dataset to eliminate columns that aren't relevant to a predictive model.
dataset = dataset[["FL_NUM", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST", "CRS_ARR_TIME","DEP_DEL15", "ARR_DEL15"]]
dataset.isnull().sum()
```
Out[32]:
```
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
```
In [ ]:
```
dataset[dataset.isnull().any(axis=1)].head(10)
```
In [ ]:
```
dataset['DEP_DEL15'].mode()
```
In [ ]:
```
#replace the missing values with 1s.
dataset = dataset.fillna({'ARR_DEL15': 1})
dataset = dataset.fillna({'DEP_DEL15': 0})
dataset.iloc[177:185]
```
In [ ]:
```
import math

for index, row in dataset.iterrows():
    dataset.loc[index, 'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME'] / 100)
dataset.head()
```
In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['DEST'] = le.fit_transform(dataset['DEST'])
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```
In [ ]:
```
dataset.head(5)
```
In [ ]:
```
dataset['ORIGIN'].unique()
```
```
dataset = pd.get_dummies(dataset, columns=['ORIGIN', 'DEST'])
dataset.head()
```
In [14]:
```
x = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 8:9].values
```
In [15]:
```
x
```
Out[15]:
```
array([[2016, 1, 1, ..., 'DL', 'N836DN', 1399],
       [2016, 1, 1, ..., 'DL', 'N964DN', 1476],
       [2016, 1, 1, ..., 'DL', 'N813DN', 1597],
       ...,
       [2016, 4, 12, ..., 'DL', 'N583NW', 1823],
       [2016, 4, 12, ..., 'DL', 'N554NW', 1901],
       [2016, 4, 12, ..., 'DL', 'N843DN', 2005]], dtype=object)
```
In [16]:
```
y
```

```
In [17]: x.shape
Out[17]: (11231, 8)

In [18]: y.shape
Out[18]: (11231, 1)

In [19]: from sklearn.preprocessing import 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)

In [20]: z
Out[20]: array([[0., 0., 0., ..., 1., 0., 0.],
               [0., 0., 0., ..., 1., 0., 0.],
               [0., 0., 0., ..., 1., 0., 0.],
               ...,
               [0., 0., 0., ..., 1., 0., 0.],
               [0., 0., 0., ..., 1., 0., 0.],
               [0., 0., 0., ..., 1., 0., 0.]])

In [21]: t
Out[21]: array([[1.],
               [1.],
               [1.],
               ...,
               [1.],
               [1.],
               [1.]])

In [22]: x=np.delete(x,[4,5],axis=1)
```

```
In [52]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)

         from sklearn.model_selection import train_test_split
         train_x, test_x, train_y, test_y = train_test_split(dataset.drop('ARR_DEL15', axis=1), df['ARR_DEL15'], test_size=0.2,
         random_state=0)

In [53]: x_test.shape
Out[53]: (2247, 16)

In [54]: x_train.shape
Out[54]: (8984, 16)

In [55]: y_test.shape
Out[55]: (2247, 1)

In [56]: y_train.shape
Out[56]: (8984, 1)

In [57]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.transform(x_test)

In [58]: from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(random_state = 0)
         classifier.fit(x_train,y_train)
Out[58]: DecisionTreeClassifier(random_state=0)
```

```
In [59]: decisiontree = classifier.predict(x_test)

In [60]: decisiontree
Out[60]: array([1., 0., 0., ..., 0., 0., 1.])

In [61]: from sklearn.metrics import accuracy_score
         desacc = accuracy_score(y_test,decisiontree)

In [62]: desacc
Out[62]: 0.8673787271918113

In [63]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,decisiontree)

In [64]: cm
Out[64]: array([[1777,  159],
                [ 139,  172]], dtype=int64)

In [65]: import sklearn.metrics as metrics
         fpr1 ,tpr1 ,threshold1 =metrics.roc_curve(y_test,decisiontree)
         roc_auc1 = metrics.auc(fpr1,tpr1)

In [66]: fpr1
Out[66]: array([0.        , 0.0821281, 1.        ])

In [67]: tpr1
Out[67]: array([0.        , 0.55305466, 1.        ])
```

```
In [68]: threshold1
Out[68]: array([2., 1., 0.])

In [69]: import matplotlib.pyplot as plt
         plt.title("roc")
         plt.plot(fpr1,tpr1,'b',label = 'Auc = %0.2f'% roc_auc1)
         plt.legend(loc = 'lower right')
         plt.plot([0,1],[0,1],'r--')
         plt.xlim([0,1])
         plt.ylim([0,1])
         plt.xlabel('tpr')
         plt.ylabel('fpr')
         plt.show()
```

```
In [70]: import pickle
         pickle.dump(classifier,open('flight.pkl','wb'))
```

app.py

```
from flask import Flask,render_template,request

import pickle
import numpy as np
model = pickle.load(open('flight.pkl','rb'))
app = Flask(__name__)

app = Flask(__name__)
return render_template("index.html")


@app.route('/prediction',methods =['POST'])
def predict():
```

```
name = request.form['name']
month = request.form['month']
dayofmonth = request.form['dayofmonth']
dayofweek = request.form['dayofweek']
origin = request.form['origin']
if(origin == "msp"):
    origin1,origin2,origin3,origin4,orgin5 = 0,0,0,0,1
if(origin == "dtw"):
    origin1,origin2,origin3,origin4,orgin5 = 1,0,0,0,0
if(origin == "jfk"):
    origin1,origin2,origin3,origin4,orgin5 = 0,0,1,0,0
if(origin == "sea"):
    origin1,origin2,origin3,origin4,orgin5 = 0,1,0,0,0
if(origin == "alt"):
    origin1,origin2,origin3,origin4,orgin5 = 0,0,0,1,0

destination = request.form['destination']
if(destination == "msp"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,0,0,1
if(destination == "dtw"):
    destination1,destination2,destination3,destination4,destination5 = 1,0,0,0,0
if(destination == "jfk"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,1,0,0
if(destination == "sea"):
    destination1,destination2,destination3,destination4,destination5 = 0,1,0,0,0
if(destination == "alt"):
    destination1,destination2,destination3,destination4,destination5 = 0,0,0,1,0

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,origin4,orgin5,destinati
on1,destination2,destination3,destination4,destination5,int(arrtime),int(dept15)]]

#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)
```

Here the actual and scheduled departure time is same the flight will be on time.
Now giving values as the flight will be get delayed the output will be,