Flight Delay Prediction for aviation Industry Using Machine learning

DONE BY

G.Padmaja
R.Elakkiya
A.Sakthi Ganesh
S.Achuthan
M.Pravin Kumar

Flight Delay Prediction for aviation Industry using Machine Learning

1 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. According to, taxi-out operations are responsible for 4,000 tons of hydrocarbons, 8,000 tons of nitrogen oxides and 45,000 tons of carbon monoxide emissions in the United States in 2007. Moreover, the economic impact of flight delays for domestic flights in the US is estimated to be more than $19 Billion per year to the airlines and over $41 Billion per year to the national economy In response to growing concerns of fuel emissions and their negative impact on health, there is active research in the aviation industry for finding techniques to predict flight delays accurately in order to optimize flight operations and minimize delays. Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is 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. Finally, it will be integrated to web based application.

Requirements

To predict flight delays using machine learning, you will need to collect and process a large amount of data on past flight delays. This data should include information such as the flight's departure and arrival times, the airline, the aircraft type, and the weather conditions at the departure and arrival airports. Once you have collected and cleaned the data, you can use a variety of machine learning techniques such as regression, decision trees, or neural networks to train a model that can predict flight delays based on this data. It is important to note that flight delay prediction is a highly complex task and requires a lot of data, but it is possible with the right resources.

It is important to note that flight delay prediction is a highly complex task and requires a lot of data. The literature suggests that ML models, specifically decision tree, ANN and random forest models, have been used to predict flight delays with varying degrees of accuracy. Commonly used features include historical flight data, weather conditions, and airport operations. It also shows that a combination of data mining techniques can be used to identify the factors that contribute to flight delays. One of the major business problems that airlines face is the significant costs that are associated with flights being delayed due to natural occurrences and operational shortcomings, which is an expensive affair for the airlines, creating problems in scheduling and operations for the end-users thus causing bad reputation and customer dissatisfaction. In our paper, a two-stage predictive model was developed employing supervised machine learning algorithms for the prediction of flight ontime performance. The first stage of the model performs binary classification to predict the occurrence of flight delays and the second stage does regression to predict the value of the delay in minutes. The dataset used for evaluating the model was obtained from historical data which contains flight schedules and weather data for 5 years.

Impact

From a social perspective, flight delay prediction can help improve the travel experience for passengers. By providing accurate and timely predictions of flight delays, passengers can make more informed decisions about their travel plans and potentially avoid delays or missed connections. This can lead to a reduction in travel-related stress and inconvenience. From a business perspective, flight delay prediction can help airlines and airports improve their operations and reduce costs. By identifying and addressing the factors that contribute to flight delays, airlines and airports can take proactive measures to mitigate the impact of delays. This can lead to improved on-time performance, which can help airlines and airports attract and retain customers and increase revenue. Additionally, flight delay prediction can help airlines and airports optimize their staffing and resource allocation, resulting in cost savings.

Data Collection & Preparation

In this project we have used .csv data. This data is downloaded from kaggle.com.

Link: flightdata.csv - Google Drive

There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing the libraries

Import the necessary libraries as shown in the image.

```
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

We are input the raw string(‘r’) to solve the Unicode error “unicodeescape”.

```
import pandas as pd

dataset= pd.read_csv(r"C:\Users\TAMIL BOYS\PycharmProjects\flightdata.csv")

dataset.head()

print(dataset)
```

```
       YEAR  QUARTER  MONTH  ...  ACTUAL_ELAPSED_TIME  DISTANCE Unnamed: 25
0      2016        1      1  ...                295.0    2182.0         NaN
1      2016        1      1  ...                115.0     528.0         NaN
2      2016        1      1  ...                300.0    2182.0         NaN
3      2016        1      1  ...                205.0    1399.0         NaN
4      2016        1      1  ...                259.0    1927.0         NaN
...     ...      ...    ...  ...                  ...       ...         ...
11226  2016        4     12  ...                105.0     594.0         NaN
11227  2016        4     12  ...                181.0    1399.0         NaN
11228  2016        4     12  ...                295.0    1927.0         NaN
11229  2016        4     12  ...                332.0    2182.0         NaN
11230  2016        4     12  ...                110.0     594.0         NaN

[11231 rows x 26 columns]
```

Data Preparation

As we have understood how the data is, let's pre-process the collected data

We need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data

These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Handling missing values

- Let’s find the shape of our dataset first. To find the shape of our

data,the dataset.shape method is used. To find the data type, dataset.info() function is used.

```
dataset.info()
```

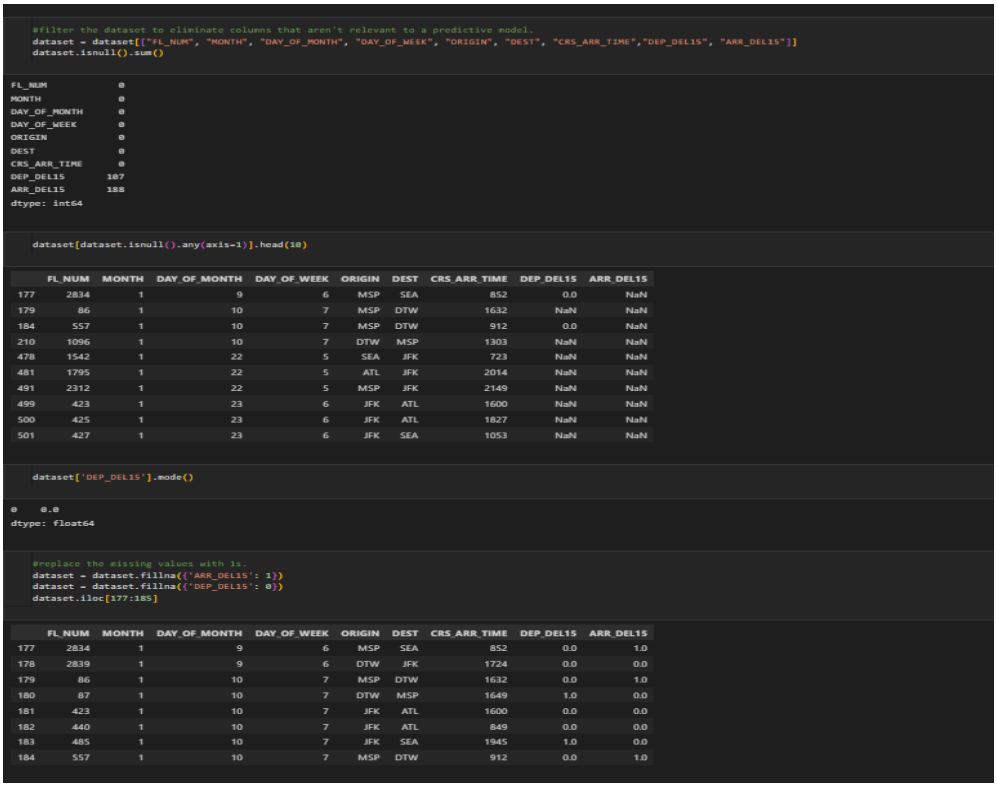
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11231 entries, 0 to 11230
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   YEAR                 11231 non-null  int64
 1   QUARTER              11231 non-null  int64
 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
```

- For checking the null values, dataset.isnull() function is used. To sum those null values we use. Sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
[18]
    dataset = dataset.drop('Unnamed: 25', axis=1)
    dataset.isnull().sum()
```

```
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
ARR_DEL15             188
CANCELLED               0
DIVERTED                0
CRS_ELAPSED_TIME        0
ACTUAL_ELAPSED_TIME   188
DISTANCE                0
dtype: int64
```

- We will fill in the missing values in the numeric data type using the mean value of that particular column and categorical data type using the most repeated value.



```
#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()
```

```
FL_NUM            0
MONTH             0
DAY_OF_MONTH      0
DAY_OF_WEEK       0
ORIGIN            0
DEST              0
CRS_ARR_TIME      0
DEP_DEL15       187
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

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

```
0    0.0
dtype: float64
```

```
#replace the missing values with 1s.
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
184	557	1	10	7	MSP	DTW	912	0.0	1.0

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

```
import math

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
1399	1	1	5	ATL	SEA	21	0.0	0.0
1476	1	1	5	DTW	MSP	14	0.0	0.0
1597	1	1	5	ATL	SEA	12	0.0	0.0
1768	1	1	5	SEA	MSP	13	0.0	0.0
1823	1	1	5	SEA	DTW	6	0.0	0.0

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['DEST'] = le.fit_transform(dataset['DEST'])
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

```
dataset.head(5)
```

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

```
    dataset['ORIGIN'].unique()
```

```
array([0, 1, 4, 3, 2])
```

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

```
    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]])
```

```
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)
```

```
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.]])
```

```
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.]])
```

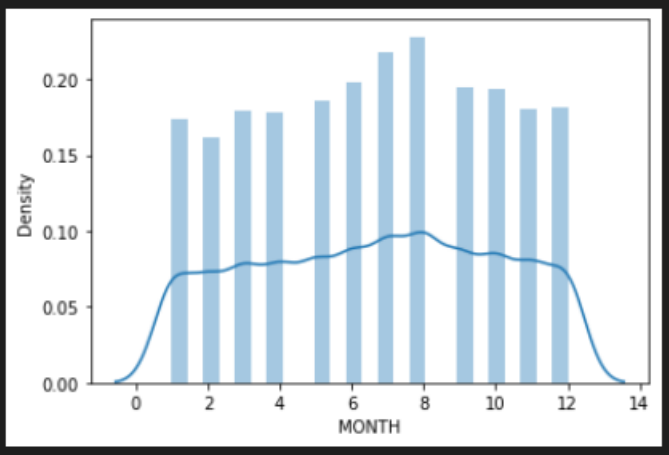
```
x=np.delete(x,[4,5],axis=1)
```

ARCHITECTURE

The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
sns.distplot(flight_data.MONTH)
```

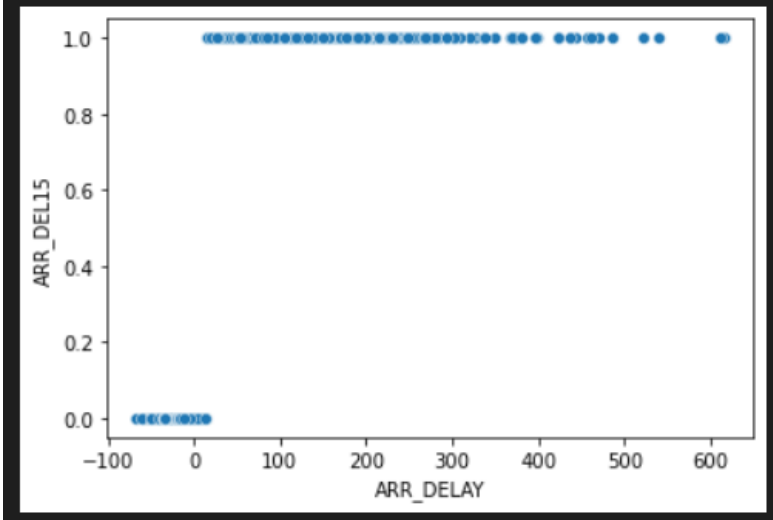
```
C:\Users\Saumya\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
figure-level function with similar flexibility) or `histplot` (an axes-lev
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='MONTH', ylabel='Density'>
```



- In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
- From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

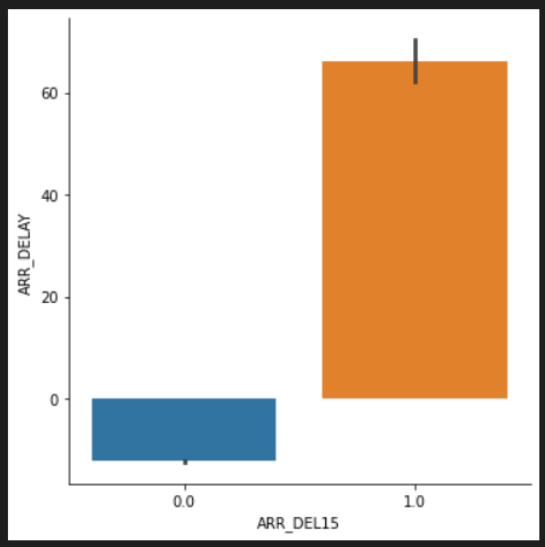
-
```
sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=flight_data)
```

```
<AxesSubplot:xlabel='ARR_DELAY', ylabel='ARR_DEL15'>
```



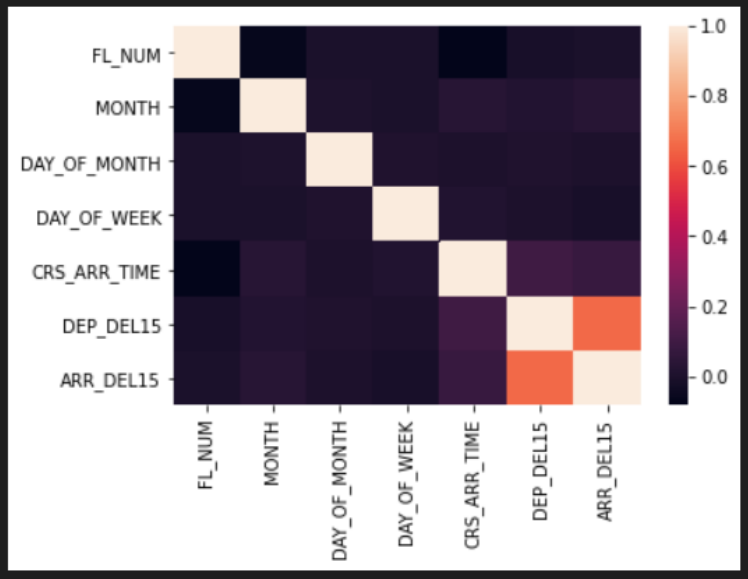

```
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=flight_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x22716099eb0>
```



- In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.

```
[32]  sns.heatmap(dataset.corr())
...   <AxesSubplot:>
```



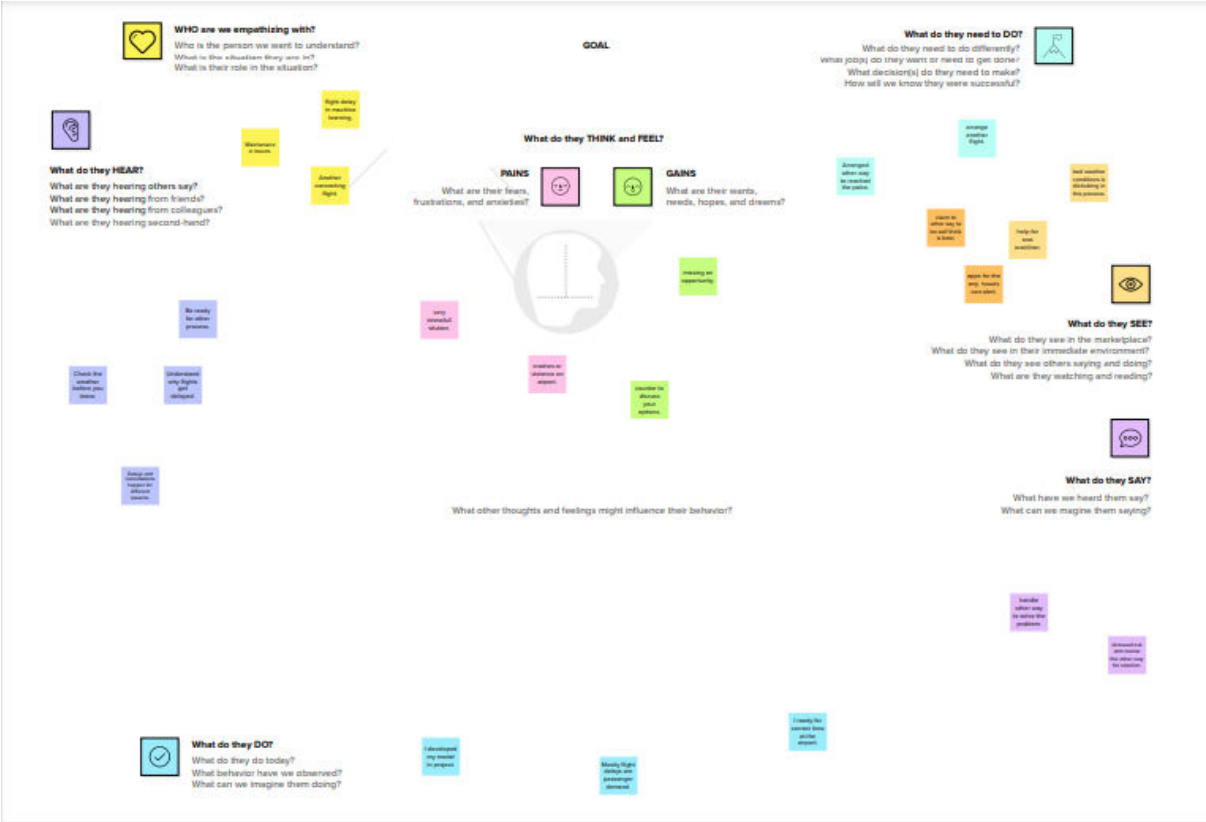
1.2 PURPOSE

Flight delays are gradually increasing and bring more financial difficulties and customer dissatisfaction to airline companies. To resolve this situation, supervised machine learning models were implemented to predict flight delays. Average aircraft delay is regularly referred to as an indication of airport capacity. Flight delay is a prevailing problem in this world. It's very tough to explain the reason for a delay. A few factors responsible for the flight delays like runway construction to excessive traffic are rare, but bad weather seems to be a common cause. Some flights are delayed because of the reactionary delays,

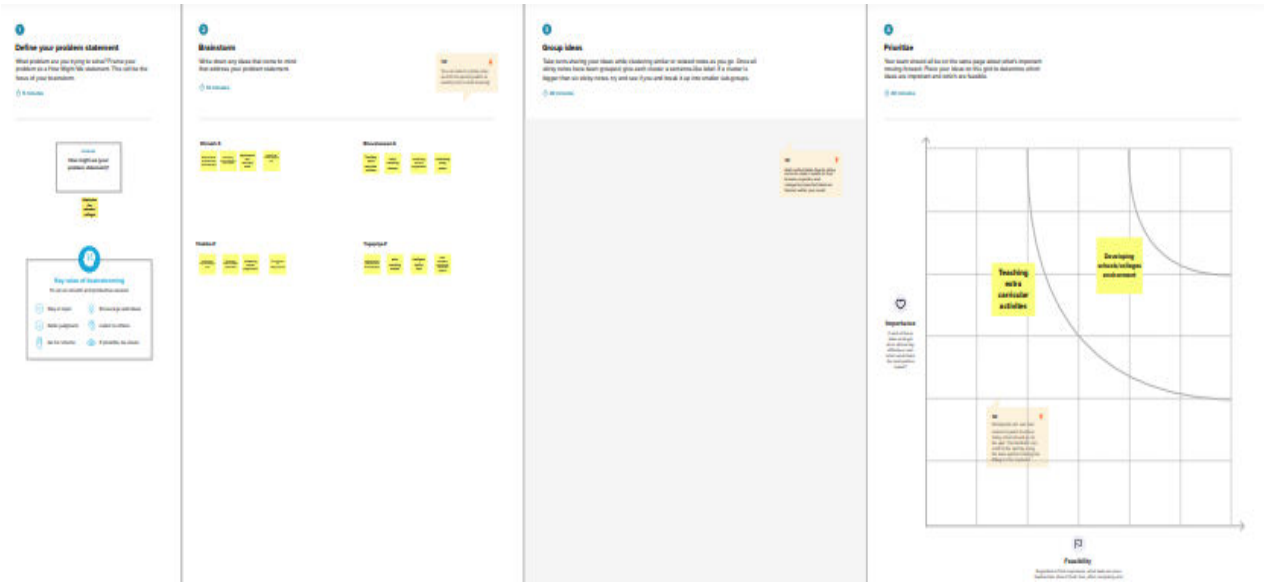
due to the late arrival of the previous flight. It hurts airports, airlines, and affects a company's marketing strategies as companies rely on customer loyalty to support their frequent flying programs.

2 PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy map



2.2 Ideation & Brainstorming Map



3 RESULT

Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle
pickle.dump(RCV,open('flight.pkl','wb'))
```

Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction

This section has the following tasks

- Building HTML Pages
- Building server side script
- Run the web application

Building Html Pages

For this project create one HTML files namely

- index.html

and save them in the templates folder.

Build Python Code

Import the libraries

```
# importing the necessary dependencies
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
model = pickle.load(open('flight.pkl','rb'))

app = Flask(__name__)#initializing the app
```

Render HTML page:

```
@app.route('/')
def home():
    return render_template("index.html")


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

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
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,destination1,destination2,destination3,destination4,destination5,i
#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)
```

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

Main Function:

```
if __name__ == '__main__':
    app.run(debug = True)
```

Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

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

Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.

4 ADVANTAGES & DISADVANTAGES

Advantages

● You can enjoy your holiday destinations for longer.

● You have a right to care.

● You have time to sort out your holiday photos

Disadvantages

● Arriers attribute flight delays to several causes such as bad weather conditions, airport congestion, airspace congestion

● Use of smaller aircraft by airlines. These delays and cancellations tarnish the airlines' reputation, often resulting in loss of demand by passengers.

5 APPLICATIONS

Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is 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. Finally, it will be integrated to web based application

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 SVM model. Somehow the SVM 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.

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 2016. A large dataset is available from 2016 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 modelling that includes feed forward networks, feedback networks, and selforganization 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

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state = 0)
classifier.fit(x_train,y_train)
```

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

```
decisiontree = classifier.predict(x_test)
```

```
decisiontree
```

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

```
from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

```
rfc.fit(x_train,y_train)
```

```
<ipython-input-125-b87bb2ba9825>:1: DataConversionWarning: A column-vector y wa
ravel().
  rfc.fit(x_train,y_train)

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

```
y_predict = rfc.predict(x_test)
```

```
# Importing the Keras libraries and packages
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
# 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'))
```

```
# Compiling 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)
```

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
1797/1797 [==============================] - 6s 2ms/step - loss: 0.2873 - accuracy: 0.8988 - val_loss: 0.2722 - val_accuracy: 0.9071
…
Epoch 99/100 1797/1797 [==============================] - 4s 2ms/step - loss: 0.0586 - accuracy: 0.9789 - val_loss: 1.1199 - val_accuracy: 0.8676 Epoch 100/100 1797/1797 [==============================] - 5s 3ms/step - loss: 0.0517 - accuracy: 0.9811 - val_loss: 1.1271 - val_accuracy: 0.8648

<tensorflow.python.keras.callbacks.History at 0x22721bdb7c0>
```

```
## Decision tree

y_pred = classifier.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])

print(y_pred)
(y_pred)
```

```
[0.]

array([0.])
```

```
## RandomForest

y_pred = rfc.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1]])

print(y_pred)
(y_pred)
```

```
[0.]

array([0.])
```

```
classification.save('flight.h5')
```

```
# Testing the model

y_pred = classification.predict(x_test)
```

```
y_pred
```

```
array([[3.1306639e-01],
       [4.3961532e-19],
       [8.1048012e-03],
       ...,
       [1.5726548e-10],
       [3.8635731e-04],
       [9.9994898e-01]], dtype=float32)
```

```
    y_pred = (y_pred > 0.5)
    y_pred
166]
```

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

```
def predict_exit(sample_value):

  # Convert list to numpy array
    sample_value = np.array(sample_value)

  # Reshape because sample_value contains only 1 record
    sample_value = sample_value.reshape(1, -1)

  # Feature Scaling
    sample_value = sc.transform(sample_value)

    return classifier.predict(sample_value)
```

```
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.')
```

```
Prediction: No chance of delay.
```

```
from sklearn import model_selection
from sklearn.neural_network import MLPClassifier
```

```
    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['model'] = name
        dfs.append(this_df)
final = pd.concat(dfs, ignore_index=True)
return final
```

RF

	precision	recall	f1-score	support
no delay	0.93	0.96	0.95	1936
delay	0.72	0.58	0.64	311
accuracy			0.91	2247
macro avg	0.82	0.77	0.79	2247
weighted avg	0.90	0.91	0.91	2247

DecisionTree

	precision	recall	f1-score	support
no delay	0.93	0.93	0.93	1936
delay	0.56	0.55	0.55	311
accuracy			0.88	2247
macro avg	0.74	0.74	0.74	2247
weighted avg	0.88	0.88	0.88	2247

ANN

	precision	recall	f1-score	support
no delay	0.93	0.96	0.95	1936
delay	0.70	0.58	0.63	311
accuracy			0.91	2247
macro avg	0.82	0.77	0.79	2247
weighted avg	0.90	0.91	0.90	2247

```
    # RandomForest Accuracy
    print('Training accuracy: ',accuracy_score(y_train,y_predict_train))
    print('Testing accuracy: ',accuracy_score(y_test,y_predict))
```

```
Training accuracy:  0.9892030276046304
Testing accuracy:  0.89942145082332
```

```
    # Making the Confusion Matrix
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_predict)
    cm
```

```
array([[1874,   62],
       [ 161,  150]], dtype=int64)
```

```
    # Accuracy score of desicionTree

    from sklearn.metrics import accuracy_score
    desacc = accuracy_score(y_test,decisiontree)
```

```
    desacc
```

```
0.8673787271918113
```

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

```
    cm
```

```
array([[1777,  159],
       [ 139,  172]], dtype=int64)
```

```python
# Calculate the Accuracy of ANN
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: 87.2719181130396%
```

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1812,  124],
       [ 162,  149]], dtype=int64)
```

```python
# giving some parameters that can be used in randized search cv
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]


}
```

```python
#performing the randomized cv
RCV  = RandomizedSearchCV(estimator=rf,param_distributions=parameters,cv=10,n_iter=4)
```

```python
RCV.fit(x_train,y_train)
```

```
bt_params
```

```
{'verbose': 10,
 'n_estimators': 90,
 'max_features': 'log2',
 'max_depth': 10,
 'criterion': 'entropy'}
```

```
bt_score
```

```
0.905498809615237
```

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

```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(), n_iter=4,
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': [2, 5, 8, 10],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'n_estimators': [1, 20, 30, 55, 68, 74,
                                                         90, 120, 115],
                                        'verbose': [1, 2, 3, 4, 6, 8, 9, 10]})
```

```
y_predict_rf = RCV.predict(x_test)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 115 out of 115 | elapsed:    0.0s finished
```

```
RFC=accuracy_score(y_test,y_predict_rf)
RFC
```

```
0.9096573208722741
```

```
if __name__ == '__main__':
    app.run(debug = True)
```

```
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,destination1,destination2,destination3,destination4,destination5,i
#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)
```