SMARTBRIDGE
Let's Bridge the Gap

Smart
Internz

# Flight Delay Prediction for aviation Industry using Machine Learning

Project Based Experiential Learning Program

Flight Delay Prediction for aviation Industry using Machine Learning

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

Technical Architecture:
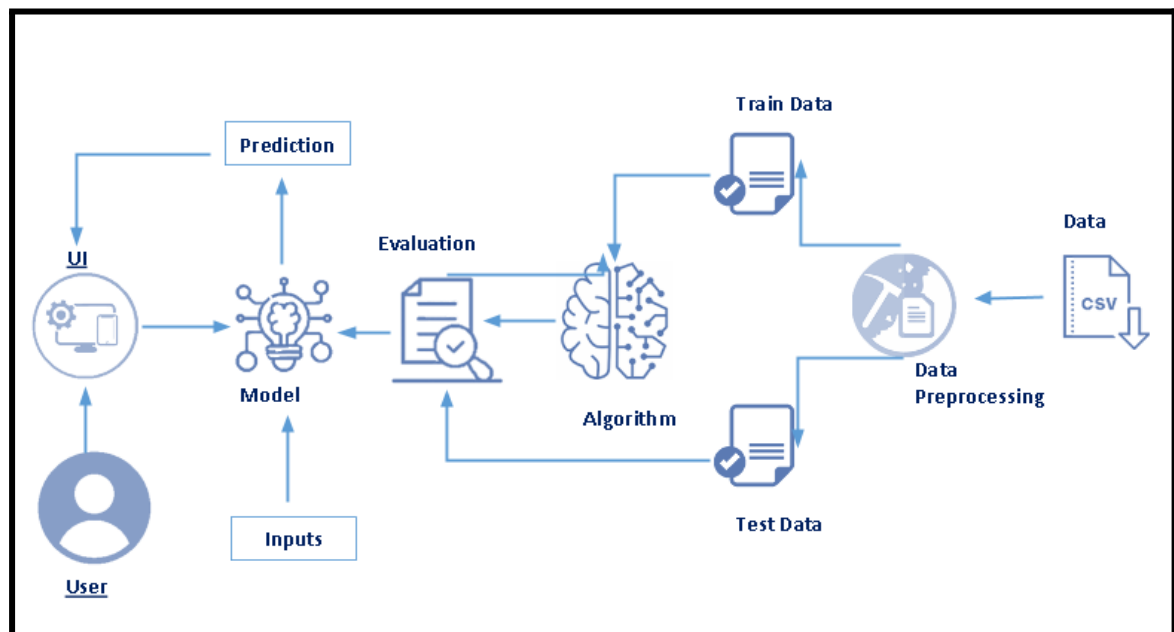
Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Project Structure:

Create the Project folder which contains files as shown below
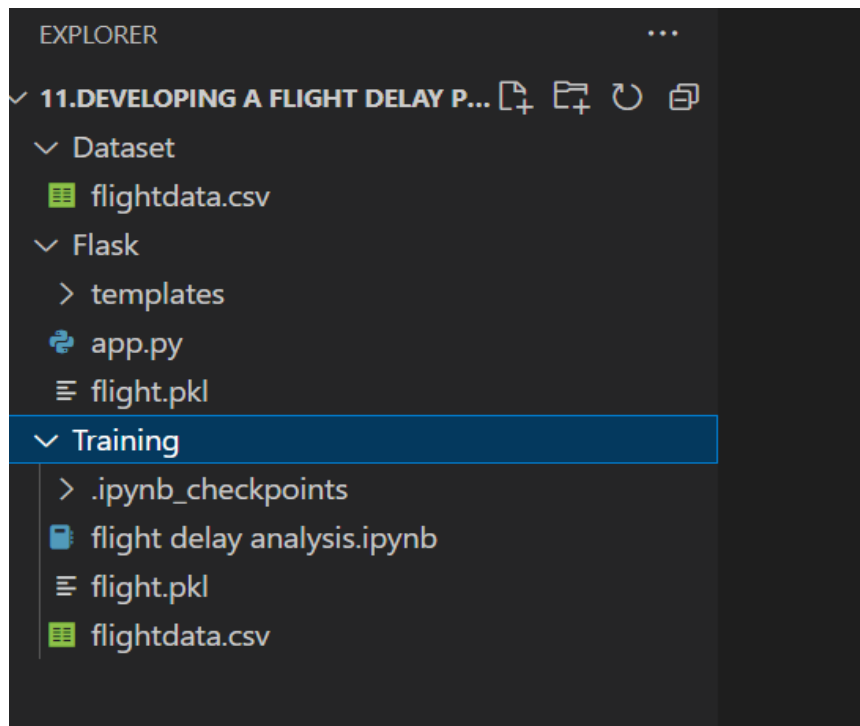
- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- flight.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

Activity 2: Business 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.

Activity 3: Literature Survey (Student Will Write)

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.

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.

Activity 4: Social or Business Impact.

The social and business impact of flight delay prediction using machine learning (ML) can be significant.

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.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: flightdata.csv - Google Drive

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: 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.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

```python
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
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
```

Activity 1.2: 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.

```python
dataset= pd.read_csv("flightdata.csv")
```
```python
dataset.head()
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	DISTANCE	Unnamed: 25
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	2143	2102.0	-41.0	0.0	0.0	0.0	338.0	295.0	2182.0	NaN
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	1435	1439.0	4.0	0.0	0.0	0.0	110.0	115.0	528.0	NaN
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	1215	1142.0	-33.0	0.0	0.0	0.0	335.0	300.0	2182.0	NaN
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	1335	1345.0	10.0	0.0	0.0	0.0	196.0	205.0	1399.0	NaN
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	607	615.0	8.0	0.0	0.0	0.0	247.0	259.0	1927.0	NaN

rows × 26 columns

Activity 2: Data Preparation

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

The download data set is not suitable for training the machine learning model as it might have so much randomness so 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

Note: 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.

Activity 2.1: Handling missing values

- Let's find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.

```
       dataset.info()
172]
```

```
Output exceeds the size limit. Open the full output data in a text editor
<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, df.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.

```
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       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

```
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

Activity 2.2: 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)
```

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```python
flight_data.describe()
```

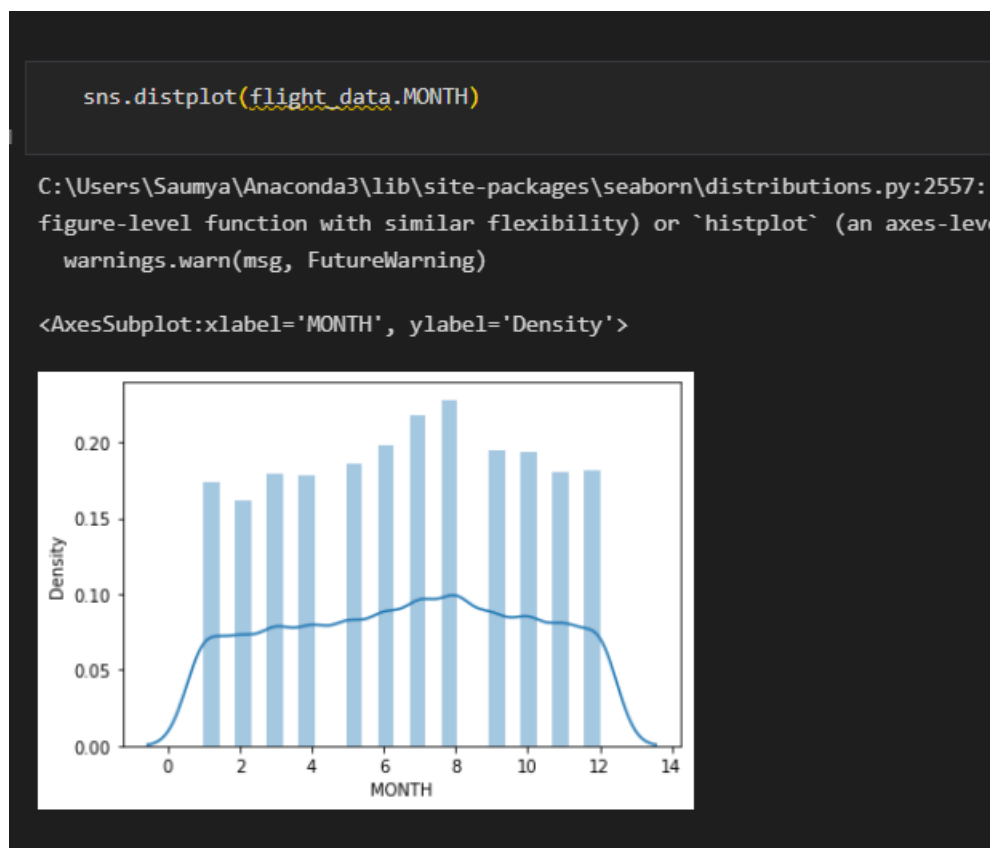|	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_TIME	...	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	DISTANCE	Unnamed: 25
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.000000	...	11231.000000	11116.000000	11043.000000	11043.000000	11231.000000	11231.000000	11231.000000	11043.000000	11231.000000	0.0
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.189410	...	1537.312795	1523.978499	-2.573123	0.124513	0.010150	0.008589	190.652124	179.661233	1161.031965	NaN
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	480.737845	500.306462	...	502.512494	512.536041	39.232521	0.330181	0.100241	0.080908	78.386317	77.940399	643.683379	NaN
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.000000	...	2.000000	1.000000	-67.000000	0.000000	0.000000	0.000000	93.000000	75.000000	509.000000	NaN
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.000000	...	1130.000000	1135.000000	-19.000000	0.000000	0.000000	0.000000	127.000000	117.000000	594.000000	NaN
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.000000	...	1559.000000	1547.000000	-10.000000	0.000000	0.000000	0.000000	159.000000	149.000000	907.000000	NaN
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.000000	...	1952.000000	1945.000000	1.000000	0.000000	0.000000	0.000000	255.000000	236.000000	1927.000000	NaN
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.000000	...	2359.000000	2400.000000	615.000000	1.000000	1.000000	1.000000	397.000000	428.000000	2422.000000	NaN

rows × 22 columns

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

- 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.


```python
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-leve
  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
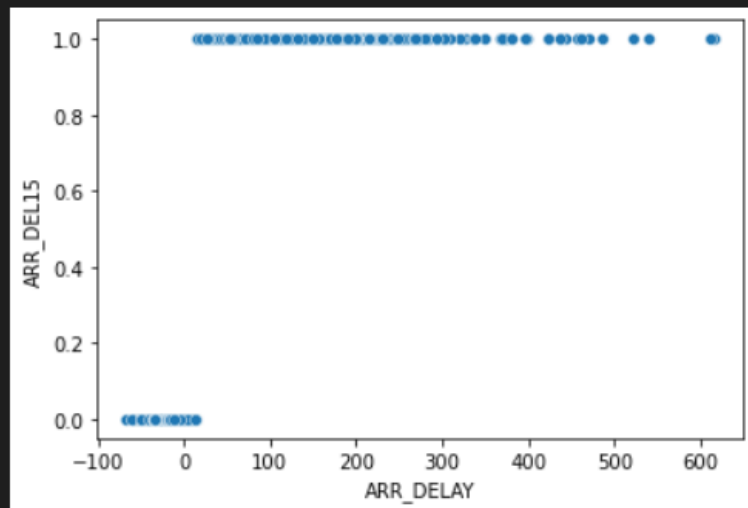
Countplot:-

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

From the graph we can infer that , gender and education is a categorical variables with 2 categories , from gender column we can infer that 0-category is having more weightage than category-1,while education with 0,it means no education is a underclass when compared with category -1, which means educated .

Activity 2.2: Bivariate analysis


```
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>
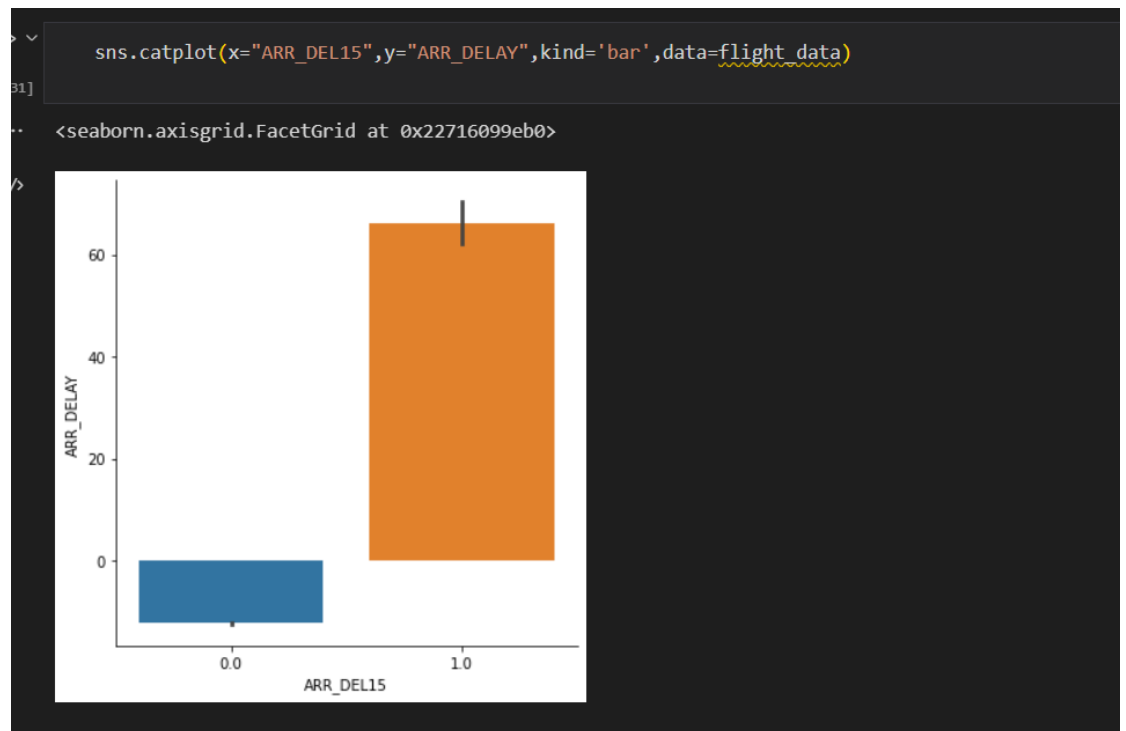
Activity 2.3: Multivariate analysis

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

```
sns.heatmap(dataset.corr())
```

<AxesSubplot:>
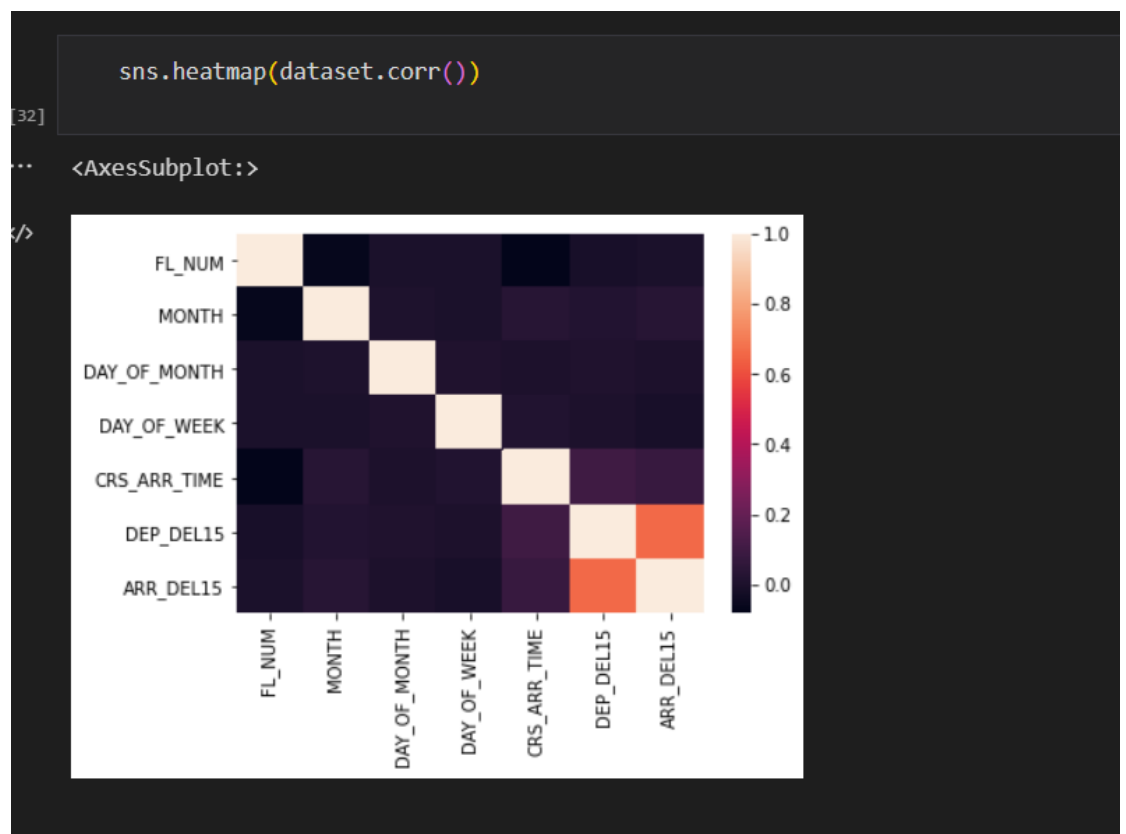
From the above graph we are plotting the relationship all the features.

Splitting data into dependent and independent variables

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

```python
x = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 8:9].values
```

Splitting data into train and test

Now let's split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```python
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)
```

```python
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)
```

```python
x_test.shape
```

```
(2247, 16)
```

```python
x_train.shape
```

```
(8984, 16)
```

```python
y_test.shape
```

```
(2247, 1)
```

+ Code   + Markdown

```python
y_train.shape
```

```
(8984, 1)
```

Scaling the data

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

We are going to use x_train and y_train obtained above in train_test_split section to train our **Decision Tree Classifier** model. We're using the fit method and passing the parameters as shown below.

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

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

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

```python
decisiontree
```

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

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

Activity 1.2: Random forest model

A function named random Forest is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with. predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
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)
```

Activity 1.3: ANN model

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.

```
# 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>
```

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]])

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.])

In ANN we first have to save the model to the test the inputs.

```
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
```

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

This code defines a function named "predict_exit" which takes in a sample_value as an input. The function then converts the input sample_value from a list to a numpy array. It reshapes the sample_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample_value array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample_value.

```
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.
```

Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the above three models

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

```python
    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
```

```python
# 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
```

```python
# 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)
```

```python
# Accuracy score of desicionTree

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

```python
desacc
```

```
0.8673787271918113
```

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

```python
cm
```

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

```
# 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%
```

```
# 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)
```

Activity 2: Comparing model accuracy before & after applying hyperparameter tuning

Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross validation, refer to this link

```
# 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]


}
```

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

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

```
building tree 89 of 90
building tree 90 of 90

[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:    1.1s finished

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]})
```

```
#getting the best paarmets from the giving list and best score from them
bt_params = RCV.best_params_
bt_score = RCV.best_score_
```

```
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
```

Milestone 6: Model Deployment

Activity 1: 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.

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

Activity 2: 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 is showcased on the UI.

This section has the following tasks

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

Activity 2.1: Building Html Pages:

For this project create one HTML files namely

- index.html

and save them in the templates folder.

Activity 2.2: Build Python code:

Import the libraries

```python
# 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.

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

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

Render HTML page:

```python
@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:

```python
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
```

```python
    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:

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

Activity 2.3: 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.

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1: - Record explanation Video for project end to end solution

Activity 2: - Project Documentation-Step by step project development procedure

Create document as per the template provided