Flight Delays Prediction Using Machine Learning

1. **Introduction**

   a. **Overview**

Over the last twenty years, air travel has been increasingly preferred among travellers 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.

   b. **Purpose The use of this project.**

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 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 Literature Survey

2.1 Existing problem

- You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- You will able to analyse or get insights into data through visualization.
- Applying different algorithms according to the dataset and based on visualization.

- You will be able to know how to build a web application using the Flask framework.

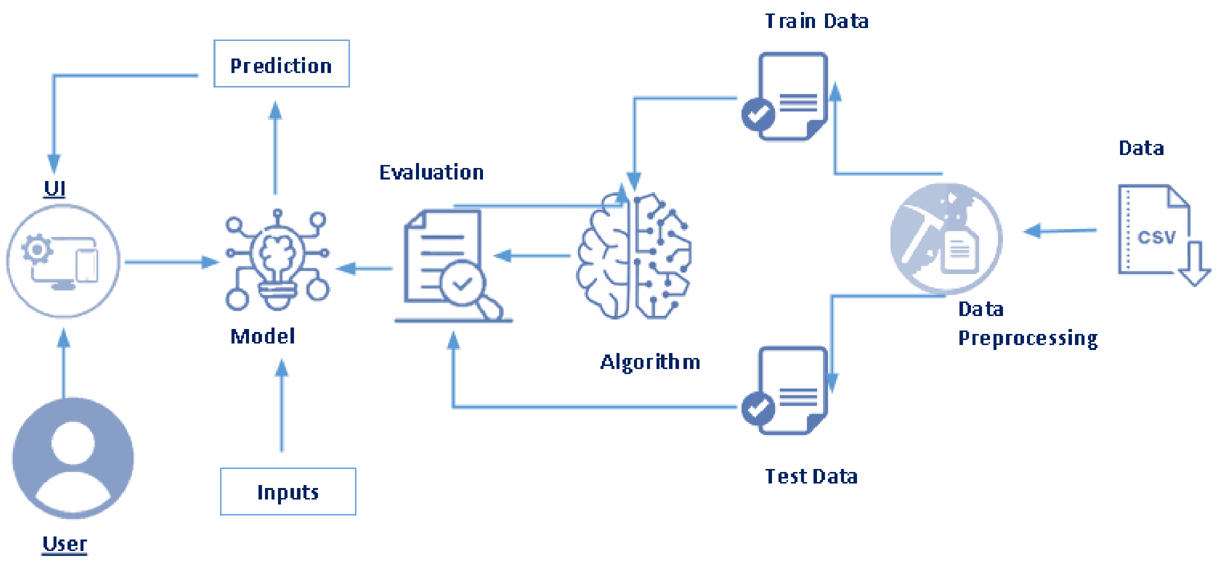
2.2 Proposed solution

- **Data Collection.**
    - Collect the dataset
- **Data Pre-processing.**
    - Import the Libraries.
    - Importing the dataset.
    - Checking for Null Values.
    - Data Visualization.
    - Taking care of Missing Data.
    - Label encoding.
    - One Hot Encoding.
    - Feature Scaling.
    - Splitting Data into Train and Test.
- **Model Building**
    - Training and testing the model
    - Evaluation of Model (Decision Tree Classification)

- **Application Building**
    - Create an HTML file
    - Build a Python Code

3 Theoretical Analysis

3.1 Block diagram Diagrammatic overview of the project.



3.2 Software designing

we will be using

- **Jupyter** notebook
- **Spyder**
  because it is a free and open-source distribution of the Python for data science and machine learning related applications

- Flask (Web applications)

Hardware:

1. **Processor:** Processor Intel CORE i5 and above Internet
2. **System architecture:** Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit aarch64 (AWS Graviton2 / arm64), 64-bit Power8/Power9, s390x (Linux on IBM Z & Linux ONE).
3. **Ram** 4 GB or above.

4 Analysis or the investigation made while working on the solution.

- flightdata.csv is the dataset used for training our model.

- flight delay analysis ipynb is the python file where the ML algorithm is applied to the dataset for testing and training. Finally, the model is saved for future use.

- Flight pkl is the saved model used in the flask to predict the output instantly for the given inputs.

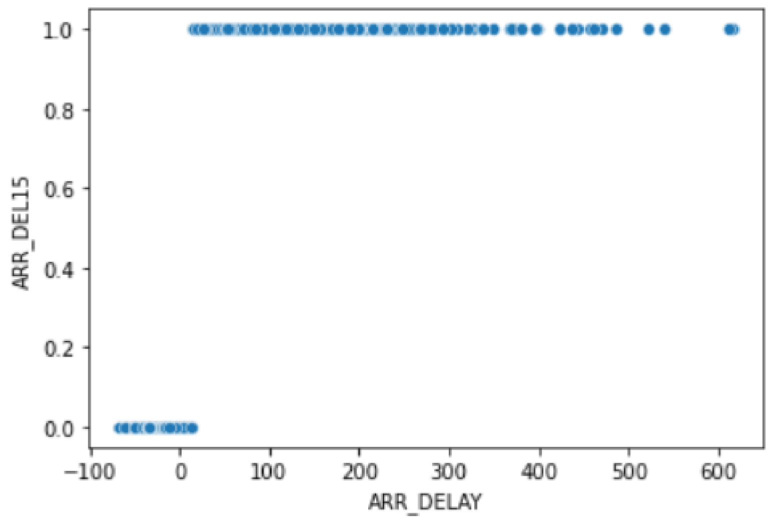
- To build a Flask Application, save HTML pages in the templates folder and a python script app.py for server-side scripting.

Scatterplot:

- A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

```
In [12]: sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=dataset)
```

```
Out[12]: <AxesSubplot:xlabel='ARR_DELAY', ylabel='ARR_DEL15'>
```

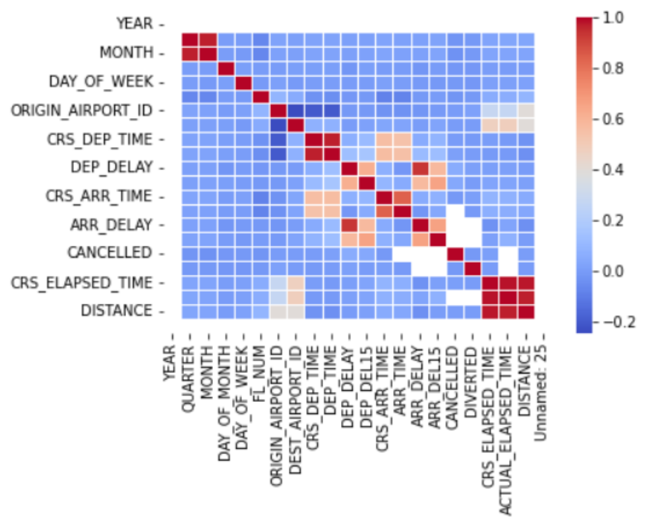


From this scatterplot, comparing the two columns we can see many flights were delayed from their arrival time.

Heatmap:

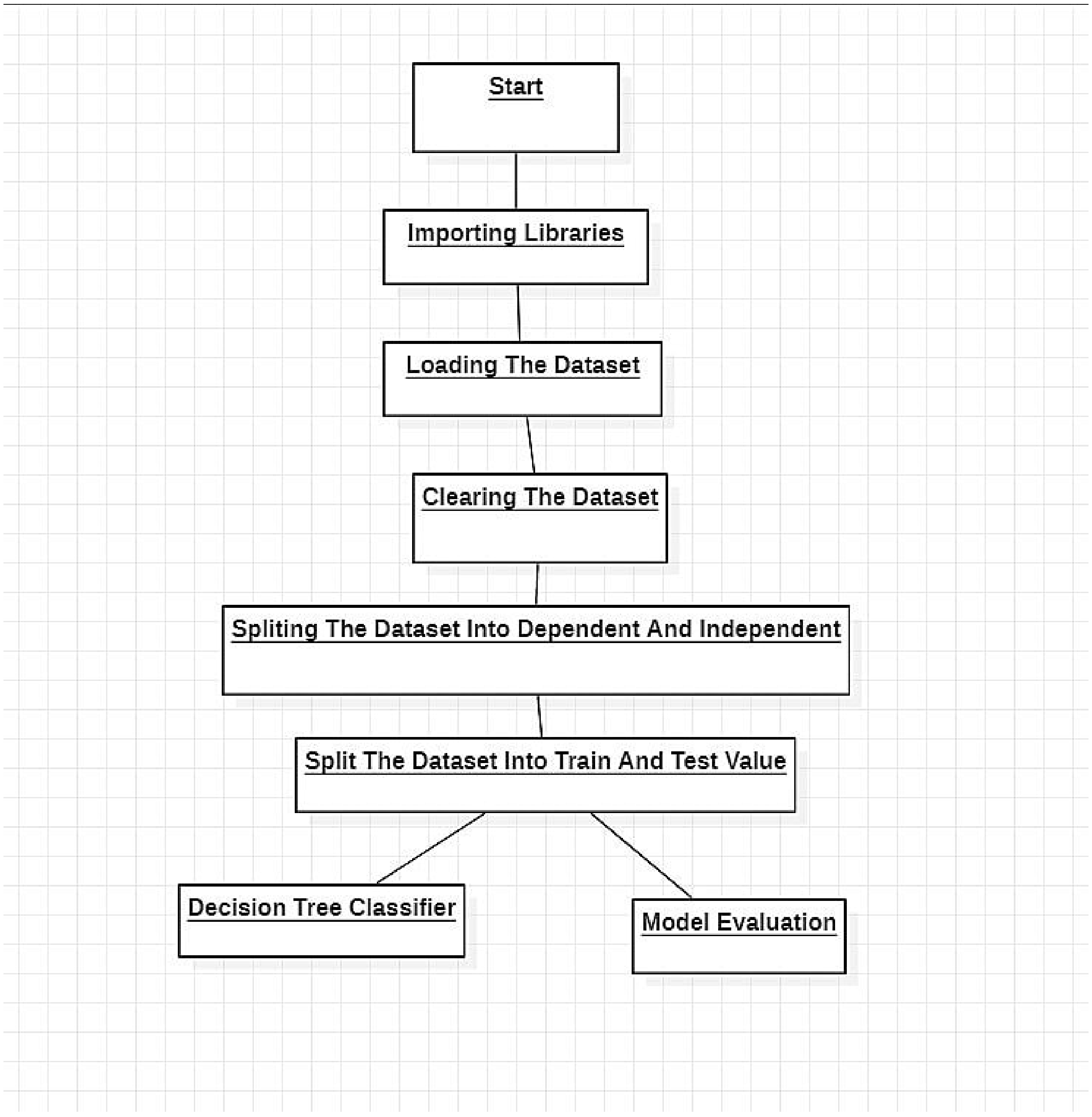
Heatmap is defined as a graphical representation of data using colours to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colours basically reddish colours are used and to represent less common or activity values, darker colours are preferred.

```
In [14]: sns.heatmap(dataset.corr(),cmap='coolwarm',linecolor='white',linewidths=1)
Out[14]: <AxesSubplot:>
```



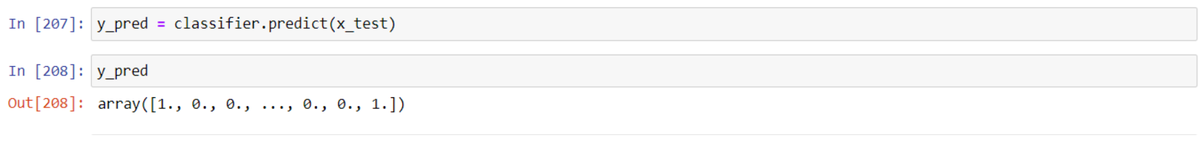
If you observe the heatmap, lighter the colour the correlation between those two variables will be high. And correlation plays a very important role for extracting the correct features for building our model.

5 Flow Chart


Start
Importing Libraries
Loading The Dataset
Clearing The Dataset
Spliting The Dataset Into Dependent And Independent
Split The Dataset Into Train And Test Value
Decision Tree Classifier
Model Evaluation

6 Result:

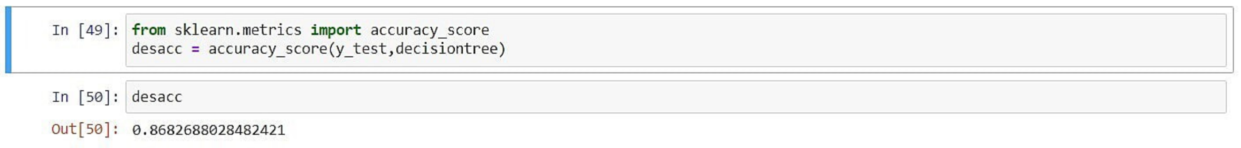
Predicted values



```
In [207]: y_pred = classifier.predict(x_test)

In [208]: y_pred
Out[208]: array([1., 0., 0., ..., 0., 0., 1.])
```

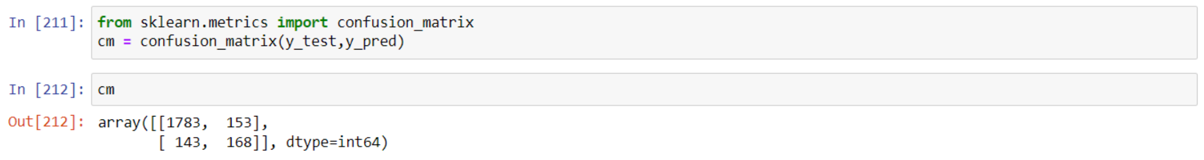
Decision Tree Model Accuracy



```
In [49]: from sklearn.metrics import accuracy_score
         desacc = accuracy_score(y_test,decisiontree)

In [50]: desacc
Out[50]: 0.8682688028482421
```

Confusion Matrix



```
In [211]: from sklearn.metrics import confusion_matrix
          cm = confusion_matrix(y_test,y_pred)

In [212]: cm
Out[212]: array([[1783,  153],
                 [ 143,  168]], dtype=int64)
```

8 Application

Flight delay is inevitable and it plays an important role in both profits and loss of the airlines. An accurate estimation of flight delay is critical for airlines because the results can be applied to increase customer satisfaction and incomes of airline agencies.

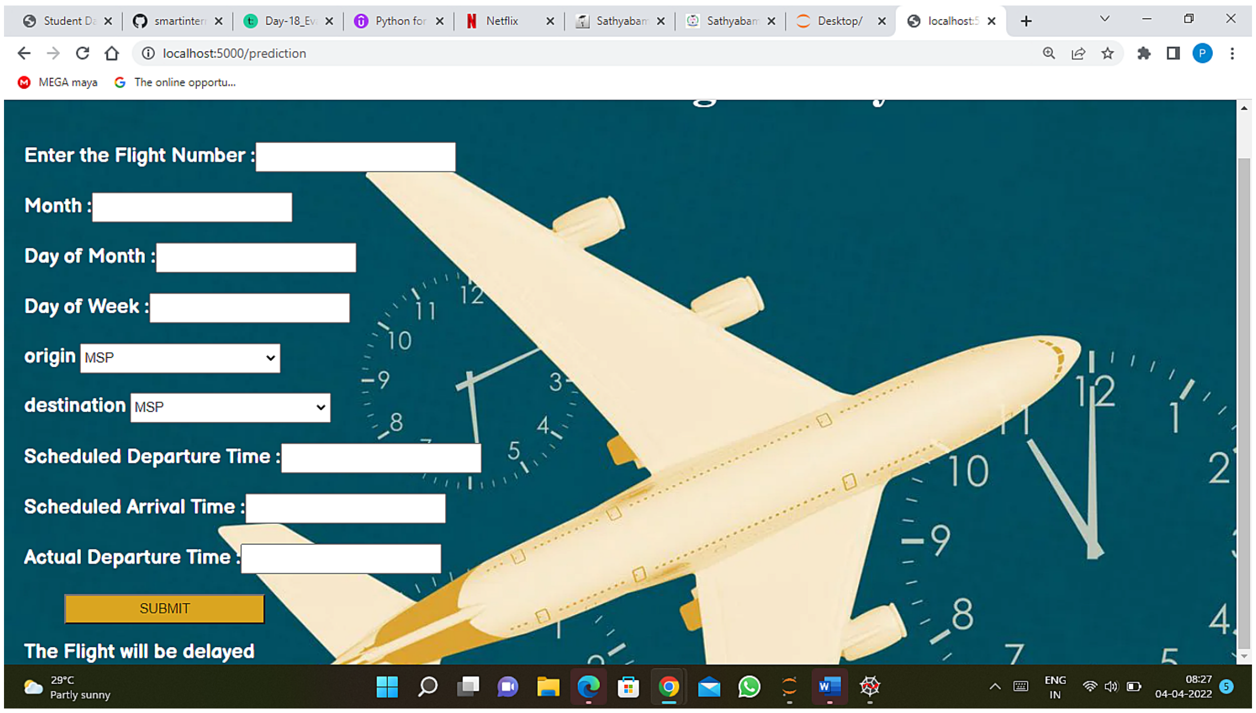
9 Conclusion:

1. By measuring the performance of the models using real data, we have seen interesting results on the predictability of the delays.

2. This is the main page of Prediction of Flight Delay. where you may know about the inputs.

3. The prediction page user gives the input for predicting the output where they can give input as Flight Number, Month, Day of Month, Week, Origin, Destination, Schedule Departure Time, Schedule Arrival Time, Actual Departure Time then click to submit the output.

4. In the prediction page user will get the output based on the inputs they given in the prediction page.


Enter the Flight Number :

Month :

Day of Month :

Day of Week :

origin MSP

destination MSP

Scheduled Departure Time :

Scheduled Arrival Time :

Actual Departure Time :

SUBMIT

The Flight will be delayed

10 Future Scope

There have been many researches on modelling and predicting flight delays, where most of them have been trying to predict the delay through extracting important characteristics and most related features. However, most of the proposed methods are not accurate enough because of massive volume data, dependencies and extreme number of parameters. This paper proposes a model for predicting flight delay based on Deep Learning (DL). DL is one of the newest methods employed in solving problems with high level of complexity and massive amount of data. Moreover, DL is capable to automatically extract the important features from data. Furthermore, due to the fact that most of flight delay data are noisy, a technique based on stack denoising autoencoder is designed and added to the proposed model. Also, Levenberg-Marquart algorithm is applied to find weight and bias proper values, and finally the output has been optimized to produce high accurate results. In order to study effect of stack denoising autoencoder and LM algorithm on the model structure, two other structures are also designed. First structure is based on autoencoder and LM algorithm (SAE-LM), and the second structure is based on denoising autoencoder only (SDA). To investigate the three models, we apply the proposed model on U.S flight dataset that it is imbalanced dataset. In order to create balance dataset, under sampling method are used. We measured precision, accuracy, sensitivity, recall and F-measure of the three models on two cases. Accuracy of the proposed prediction model analysed and compared to previous prediction method. results of three models on both imbalanced and balanced datasets shows that precision, accuracy, sensitivity, recall and F-measure of SDA-LM model with imbalanced and balanced dataset is improvement than SAE-LM and SDA models. The results also show that accuracy of the proposed model in forecasting flight delay on imbalanced and balanced dataset respectively has greater than previous model called RNN.

11 Bibliography

References

E.R. Mueller and G.B. Chatterjee. “Analysis of aircraft arrival and departure delay characteristics”. In: Proceedings of the AIAA Aircraft Technology, Integration, and Operations (ATIO) Conference, Los Angeles, CA. 2002.

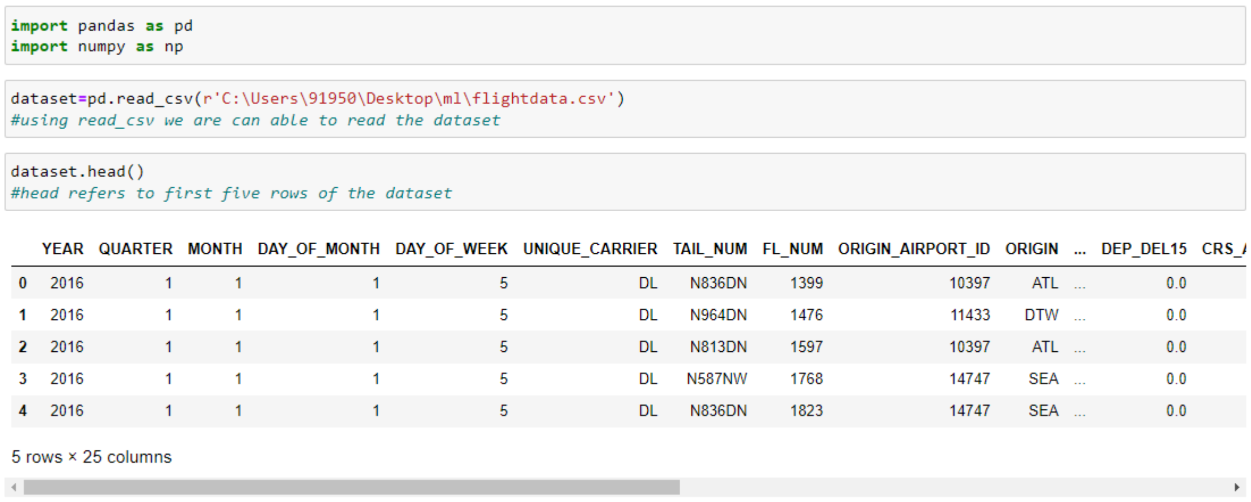
SS Allan et al. “Analysis of delay causality at Newark International Airport”. In: 4th USA/Europe Air Traffic Management R&D Seminar. 2001.

Lu Jonglei, Wang Jindong, and Zheng Guan sheng. “A new method to alarm large scale of flights delay based on machine learning”. In: Knowledge Acquisition and Modelling, 2008.

Y. Tu, M.O. Ball, and W.S. Janks. “Estimating flight departure delay distributions – a statistical approach with long-term trend and short-term pattern”. In: Journal of the American Statistical Association 103.481 (2008), pp. 112–125.

B.W. Silverman. Density estimation for statistics and data analysis. Vol. 26. Chapman & Hall/CRC, 1986.

Source Code



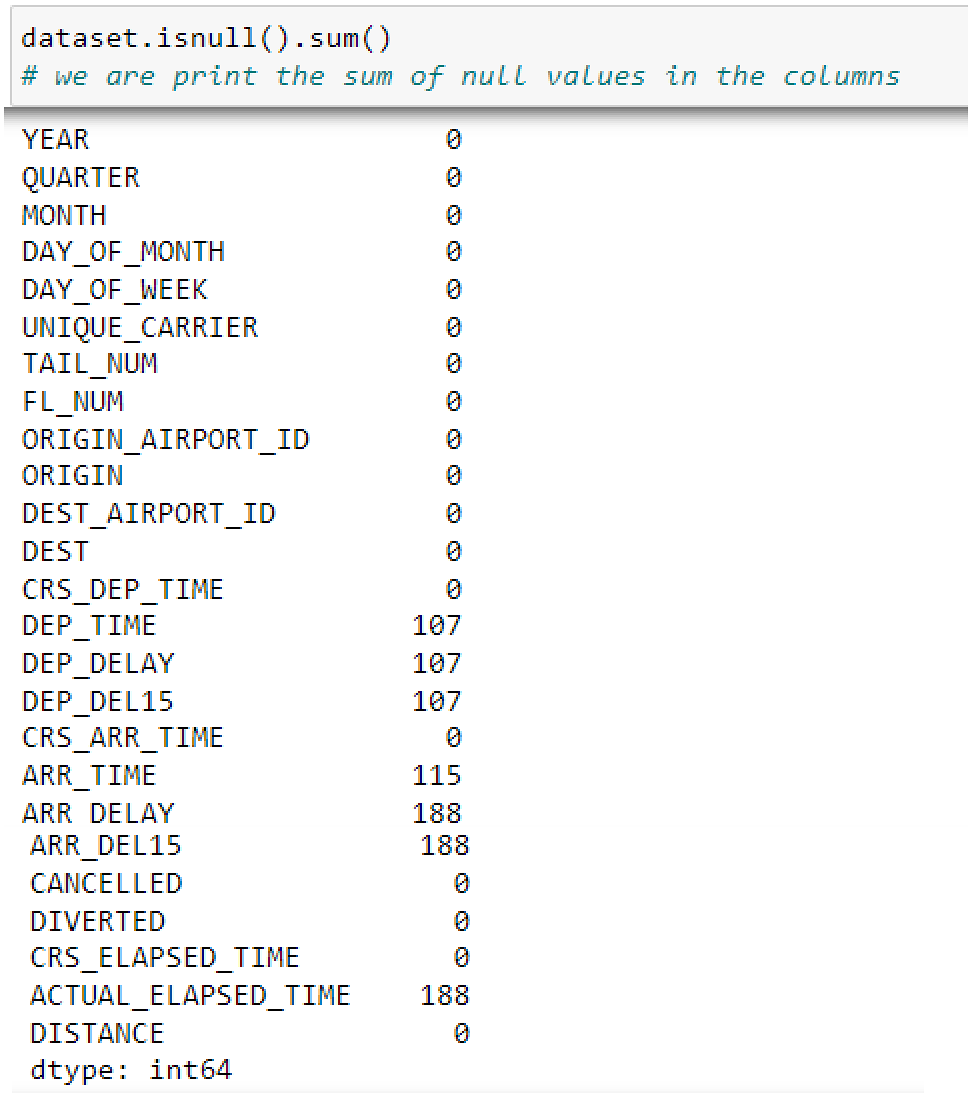
```
import pandas as pd
import numpy as np
```

```
dataset=pd.read_csv(r'C:\Users\91950\Desktop\ml\flightdata.csv')
#using read_csv we are can able to read the dataset
```

```
dataset.head()
#head refers to first five rows of the dataset
```

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	DEP_DEL15	CRS_
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	0.0	
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	0.0	
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	0.0	
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	0.0	
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	0.0	

5 rows × 25 columns



```
dataset.isnull().sum()
# we are print the sum of null values in the columns
```

```
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
```

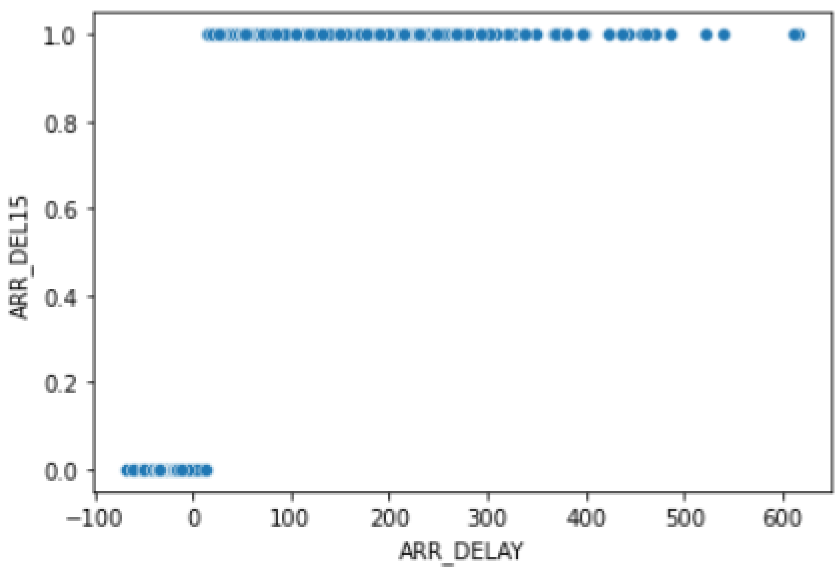
```
dataset['DEST'].unique()
#unique values in DEST column
```

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

```
import seaborn as sns # used for visiualization
%matplotlib inline
```

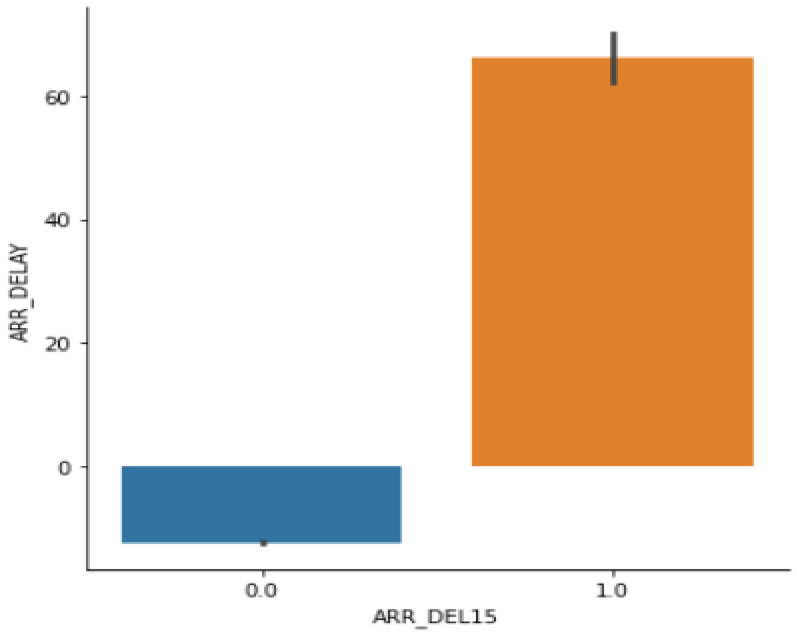
```
sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=dataset)
```

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



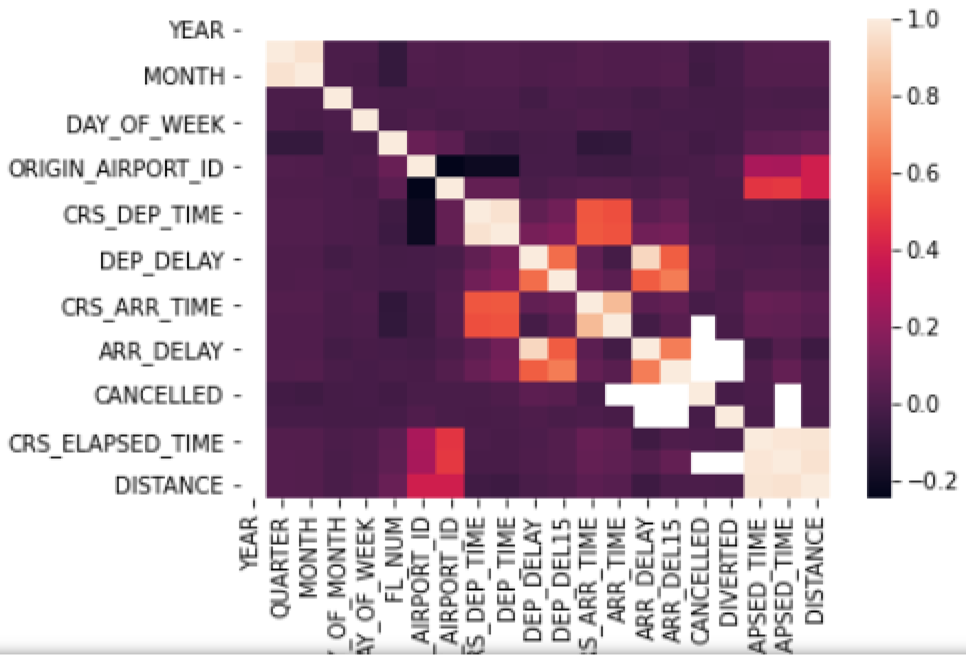
```
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=dataset)
```

```
<seaborn.axisgrid.FacetGrid at 0x21583a4f850>
```



```
sns.heatmap(dataset.corr())
# it shows the inter relationship between the numerical columns
```

```
<AxesSubplot:>
```



```python
#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
```

```python
dataset[dataset.isnull().any(axis=1)].head(10)
#we are print only the null values rows
```

	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

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

```
0    0.0
dtype: float64
```

```python
#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

```
import math

for index, row in dataset.iterrows():
    dataset.loc[index, 'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME'] / 100)
    # we are reducing the values of the CRS_ARR_TIME by floor division
dataset.head()
```

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

```
from sklearn.preprocessing import LabelEncoder
# it is used to change the string values into numerical values
le = LabelEncoder()
dataset['DEST'] = le.fit_transform(dataset['DEST'])
dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```

```
dataset.head(5)
# After using labelEncoder , printing the first 5 values
```

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

```
dataset['ORIGIN'].unique()
# This is the numerical unique values after using labelEncoder
```

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

```
x = dataset.iloc[:, 0:8].values
# x is independent values
y = dataset.iloc[:, 8:9].values
# y is dependent values
```

```
x # printing x values
```

```
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]])
```

```
y # printing y values
```

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

```
x.shape
# shape of x
```

```
(11231, 8)
```

```
y.shape
# sahpe of y
```

```
(11231, 1)
```

```
from sklearn.preprocessing import OneHotEncoder
# After using LabelEncoder the numerical values cannot able to understand what is 0 and 1
# so we are using OneHotEncoder to one column values into different column values
oh = OneHotEncoder()
z=oh.fit_transform(x[:,4:5]).toarray() # column ORIGIN
t=oh.fit_transform(x[:,5:6]).toarray() # column DEST
```

```
z
# this is column ORIGIN and doing OneHotEncoder
```

```
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
# this is column DEST and doing OneHotEncoder
```

```
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)
# we are deleting the ORIGIN and DEST in the dataset
```

```
x.shape
# shape after deleteing the 2 columns
```

```
(11231, 6)
```

```
x=np.concatenate((t,z,x),axis = 1)
# we are concatenating the new origin and dest in the dataset
```

```
x.shape
# this is the present shape of x
```

```
(11231, 16)
```

```
from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
# to find the accuracy of the predicited values
desacc
```

```
0.8682688028482421
```

```
from sklearn.metrics import confusion_matrix
# shows the predicited values in matrix format tt tf ft ff
cm = confusion_matrix(y_test,decisiontree)
cm
```

```
array([[1780,  156],
       [ 140,  171]], dtype=int64)
```

```
from sklearn.model_selection import train_test_split
# it is used to split the dataaset for train and test
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```
print(x_test.shape)
print(x_train.shape)
print(y_test.shape)
print(y_train.shape)
```

```
(2247, 16)
(8984, 16)
(2247, 1)
(8984, 1)
```

```
import sklearn.metrics as metrics
# shows AUC_ROC curve
fpr1 ,tpr1 ,threshold1 =metrics.roc_curve(y_test,decisiontree)
roc_auc1 = metrics.auc(fpr1,tpr1)
```

```
fpr1
```

```
array([0.        , 0.08057851, 1.        ])
```

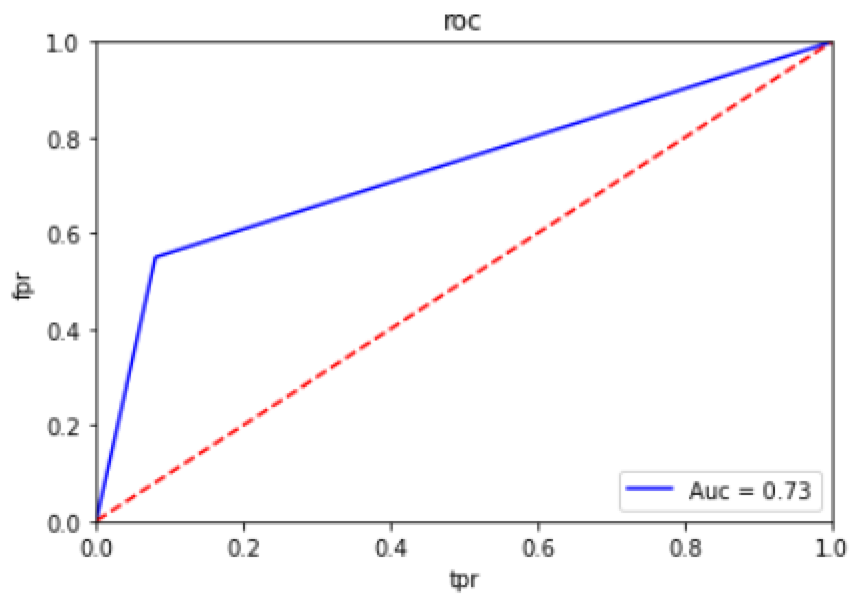
```
tpr1
```

```
array([0.        , 0.54983923, 1.        ])
```

```
threshold1
```

```
array([2., 1., 0.])
```

```
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()
```



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