# Flight Delays Prediction Using Machine Learning

## 1 Introduction

### a. 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 the air. These delays are responsible for large economic and environmental losses The main objective of the model is to predict flight delays accurately 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 vectors like departure date, departure delay, the distance between the two airports, scheduled arrival time, etc. We then use a decision tree classifier to predict if the flight arrival will be delayed or not. A flight is considered to be delayed when the difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare the 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 be able to analyze 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.

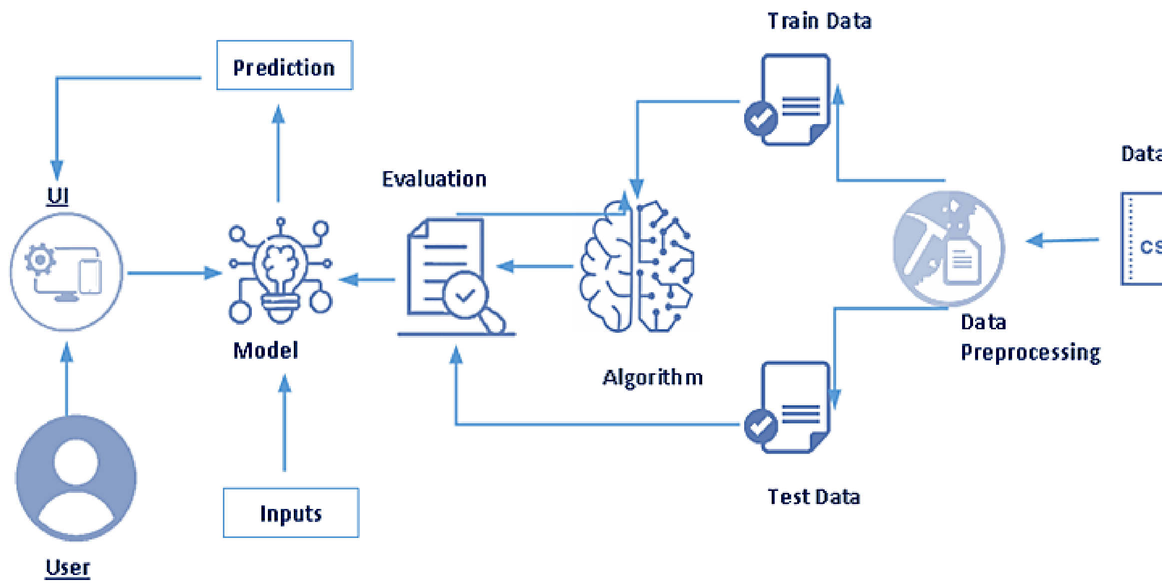
## 2.2 Proposed solution

- **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.

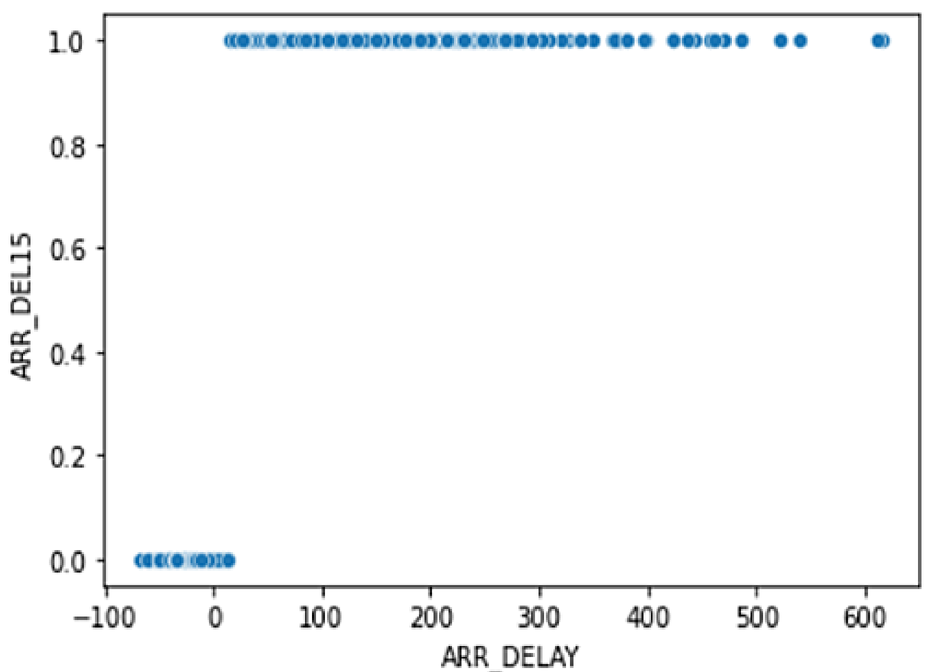
## 4. Analysis or the investigation made while working on the solution.

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.

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

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



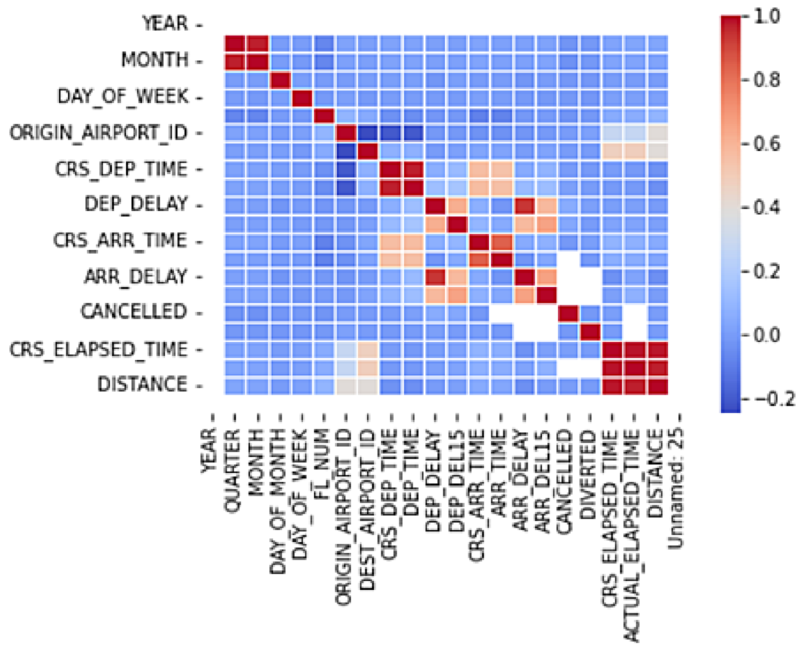

From this scatterplot, comparing the two columns we can see many flights were delayed from their arrival time.

Heatmap

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

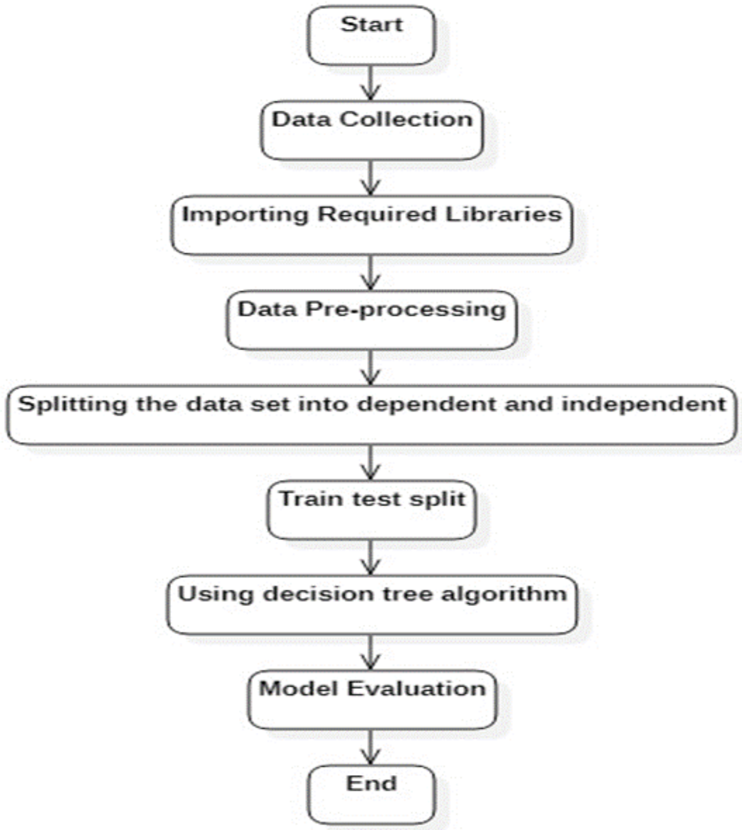
```
In [14]: sns.heatmap(dataset.corr(),cmap='coolwarm',linecolor='white',linewidths=1)

Out[14]: <AxesSubplot:>
```




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

## 5 Flow Chart




## 6. Result

## Result final findings (output) of the project along with screenshots.

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

```
y_pred
```

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

### Decision  Tree Model Accuracy

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

```
desacc
```

```
0.8682688028482421
```

### Confusion Matrix

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

```
cm
```

```
array([[1783,  153],
       [ 143,  168]], dtype=int64)
```

## 7 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 the incomes of airline agencies.

## 8 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. On the prediction page, the user will get the output based on the inputs they were given on the prediction page.

## 9 Future Scope

There have been many kinds of research on modeling 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 the massive volume of data, dependencies, and an extreme number of parameters. This paper proposes a model for predicting flight delays based on Deep Learning (DL).DL is one of the newest methods employed in solving problems with a high level of complexity and a massive amount of data. Moreover, DL is capable of automatically extracting the important features from data. Furthermore, because most flight delay data are noisy, a technique based on stack denoising autoencoder is designed and added to the proposed model. Also, the Levenberg-Marquart algorithm is applied to find weight and bias proper values, and finally, the output has been optimized to produce highly accurate results. To study the effect of stack denoising autoencoder and LM algorithm on the model structure, two other structures are also designed. The 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 to the U.S flight dataset which is the imbalanced dataset. To create a balanced dataset, the undersampling method is used. We measured the precision, accuracy, sensitivity, recall, and F-measure of the three models in two cases. The accuracy of the proposed prediction model was analyzed and compared to the previous prediction method. results of three models on both imbalanced and balanced datasets show that precision, accuracy, sensitivity, recall, and F-measure of the SDA-LM model with the imbalanced and balanced dataset is an improvement in SAE-LM and SDA models. The results also show that the accuracy of the proposed model in forecasting flight delay on imbalanced and balanced datasets respectively has greater than the previous model called RNN.

## 10 Bibliography

## References

E.R. Mueller and G.B. Chatterji. “Analysis of aircraft arrival anddeparture 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 InternationalAirport”. In: 4th USA/Europe Air Traffic Management R&D Seminar. 2001.

Lu Zonglei, Wang Jiandong, and Zheng Guansheng. “A new method toalarm large scale of flights delay based on machine learning”. In: Knowledge Acquisition and Modeling, 2008.

Y. Tu, M.O. Ball, and W.S. Jank. “Estimating flight departure delaydistributions – a statistical approach with the 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.

## Appendix

# IMPORTING LIBRARIES

```
import numpy
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

# IMPORTING THE DATASET

```
dataset=pd.read_csv(r'E:\ML\New folder\flightdata.csv')
```

```
dataset.head()
```

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

5 rows × 26 columns

```
dataset.tail()
```

| | YEAR | QUARTER | MONTH | DAY_OF_MONTH | DAY_OF_WEEK | UNIQUE_CARRIER | TAIL_NUM | FL_NUM | ORIGIN_AIRPORT_ID | ORIGIN | ... | CRS_ARR_TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **11226** | 2016 | 4 | 12 | 30 | 5 | DL | N940DL | 1715 | 11433 | DTW | ... | 1223 |
| **11227** | 2016 | 4 | 12 | 30 | 5 | DL | N836DN | 1770 | 14747 | SEA | ... | 2046 |
| **11228** | 2016 | 4 | 12 | 30 | 5 | DL | N583NW | 1823 | 11433 | DTW | ... | 2210 |
| **11229** | 2016 | 4 | 12 | 30 | 5 | DL | N554NW | 1901 | 10397 | ATL | ... | 1806 |
| **11230** | 2016 | 4 | 12 | 30 | 5 | DL | N843DN | 2005 | 10397 | ATL | ... | 925 |

5 rows × 26 columns

# ANALYSE THE DATA

```
dataset.info()
```

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

```
dataset.describe()
```

| | YEAR | QUARTER | MONTH | DAY_OF_MONTH | DAY_OF_WEEK | FL_NUM | ORIGIN_AIRPORT_ID | DEST_AIRPORT_ID | CRS_DEP_TIME | DEP_ |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 11231.0 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11231.000000 | 11124.0C |
| mean | 2016.0 | 2.544475 | 6.628973 | 15.790758 | 3.960199 | 1334.325617 | 12334.516695 | 12302.274508 | 1320.798326 | 1327.1E |
| std | 0.0 | 1.090701 | 3.354678 | 8.782056 | 1.995257 | 811.875227 | 1595.026510 | 1601.988550 | 490.737845 | 500.3C |
| min | 2016.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 7.000000 | 10397.000000 | 10397.000000 | 10.000000 | 1.0C |
| 25% | 2016.0 | 2.000000 | 4.000000 | 8.000000 | 2.000000 | 624.000000 | 10397.000000 | 10397.000000 | 905.000000 | 905.0C |
| 50% | 2016.0 | 3.000000 | 7.000000 | 16.000000 | 4.000000 | 1267.000000 | 12478.000000 | 12478.000000 | 1320.000000 | 1324.0C |
| 75% | 2016.0 | 3.000000 | 9.000000 | 23.000000 | 6.000000 | 2032.000000 | 13487.000000 | 13487.000000 | 1735.000000 | 1739.0C |
| max | 2016.0 | 4.000000 | 12.000000 | 31.000000 | 7.000000 | 2853.000000 | 14747.000000 | 14747.000000 | 2359.000000 | 2400.0C |

8 rows × 22 columns

```
dataset.describe().transpose()
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **YEAR** | 11231.0 | 2016.000000 | 0.000000 | 2016.0 | 2016.0 | 2016.0 | 2016.0 | 2016.0 |
| **QUARTER** | 11231.0 | 2.544475 | 1.090701 | 1.0 | 2.0 | 3.0 | 3.0 | 4.0 |
| **MONTH** | 11231.0 | 6.628973 | 3.354678 | 1.0 | 4.0 | 7.0 | 9.0 | 12.0 |
| **DAY_OF_MONTH** | 11231.0 | 15.790758 | 8.782056 | 1.0 | 8.0 | 16.0 | 23.0 | 31.0 |
| **DAY_OF_WEEK** | 11231.0 | 3.960199 | 1.995257 | 1.0 | 2.0 | 4.0 | 6.0 | 7.0 |
| **FL_NUM** | 11231.0 | 1334.325617 | 811.875227 | 7.0 | 624.0 | 1267.0 | 2032.0 | 2853.0 |
| **ORIGIN_AIRPORT_ID** | 11231.0 | 12334.516695 | 1595.026510 | 10397.0 | 10397.0 | 12478.0 | 13487.0 | 14747.0 |
| **DEST_AIRPORT_ID** | 11231.0 | 12302.274508 | 1601.988550 | 10397.0 | 10397.0 | 12478.0 | 13487.0 | 14747.0 |
| **CRS_DEP_TIME** | 11231.0 | 1320.798326 | 490.737845 | 10.0 | 905.0 | 1320.0 | 1735.0 | 2359.0 |
| **DEP_TIME** | 11124.0 | 1327.189410 | 500.306462 | 1.0 | 905.0 | 1324.0 | 1739.0 | 2400.0 |
| **DEP_DELAY** | 11124.0 | 8.460266 | 36.762969 | -16.0 | -3.0 | -1.0 | 4.0 | 645.0 |
| **DEP_DEL15** | 11124.0 | 0.142844 | 0.349930 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **CRS_ARR_TIME** | 11231.0 | 1537.312795 | 502.512494 | 2.0 | 1130.0 | 1559.0 | 1952.0 | 2359.0 |
| **ARR_TIME** | 11116.0 | 1523.978499 | 512.536041 | 1.0 | 1135.0 | 1547.0 | 1945.0 | 2400.0 |
| **ARR_DELAY** | 11043.0 | -2.573123 | 39.232521 | -67.0 | -19.0 | -10.0 | 1.0 | 615.0 |
| **ARR_DEL15** | 11043.0 | 0.124513 | 0.330181 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **CANCELLED** | 11231.0 | 0.010150 | 0.100241 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **DIVERTED** | 11231.0 | 0.006589 | 0.080908 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **CRS_ELAPSED_TIME** | 11231.0 | 190.652124 | 78.386317 | 93.0 | 127.0 | 159.0 | 255.0 | 397.0 |
| **ACTUAL_ELAPSED_TIME** | 11043.0 | 179.661233 | 77.940399 | 75.0 | 117.0 | 149.0 | 236.0 | 428.0 |
| **DISTANCE** | 11231.0 | 1161.031965 | 643.683379 | 509.0 | 594.0 | 907.0 | 1927.0 | 2422.0 |

## HANDLING MISSING VALUES

```
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
Unnamed: 25            11231
dtype: int64
```

```
dataset.isnull().any()
```

```
YEAR                   False
QUARTER                False
MONTH                  False
DAY_OF_MONTH           False
DAY_OF_WEEK            False
UNIQUE_CARRIER         False
TAIL_NUM               False
FL_NUM                 False
ORIGIN_AIRPORT_ID      False
ORIGIN                 False
DEST_AIRPORT_ID        False
DEST                   False
CRS_DEP_TIME           False
DEP_TIME                True
DEP_DELAY               True
DEP_DEL15               True
CRS_ARR_TIME           False
ARR_TIME                True
ARR_DELAY               True
ARR_DEL15               True
CANCELLED              False
DIVERTED               False
CRS_ELAPSED_TIME       False
ACTUAL_ELAPSED_TIME     True
DISTANCE               False
Unnamed: 25             True
dtype: bool
```

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

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

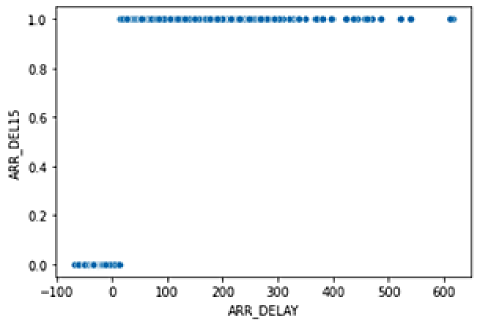
```
dataset.drop('Unnamed: 25', axis=1)
dataset.isnull().sum()
```

```
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
Unnamed: 25            11231
dtype: int64
```

# DATA VISUALIZATION

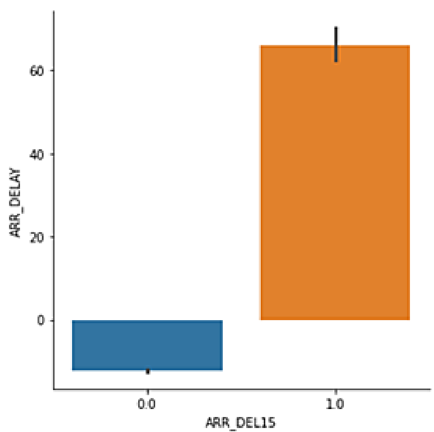
```
sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=dataset)
```

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



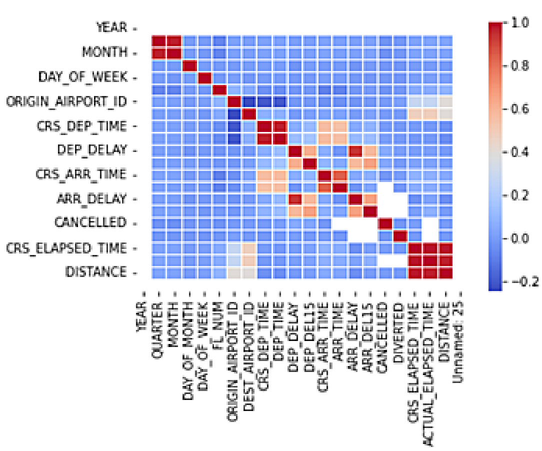

```
sns.catplot(x='ARR_DEL15',y='ARR_DELAY',kind='bar',data=dataset)
```

```
<seaborn.axisgrid.FacetGrid at 0x134fbae6f10>
```




```
sns.heatmap(dataset.corr(),cmap='coolwarm',linecolor='white',linewidths=1)
```

```
<AxesSubplot:>
```



# FILTER THE DATASET

```
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()
```

| | 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 |

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

```
0    0.0
dtype: float64
```

```
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)
```

```
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 |

## LABEL ENCODER

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

```
dataset.head()
```

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

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

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

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

```
x.shape
```

```
(11231, 8)
```

```
y.shape
```

```
(11231, 1)
```

## ONE HOT ENCODER

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
a=ohe.fit_transform(x[:,4:5]).toarray()
b=ohe.fit_transform(x[:,5:6]).toarray()
```

```
a
```

```
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.]])
```

```
b
```

```
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)
```

```
x.shape
```

```
(11231, 6)
```

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

```
x.shape
```

```
(11231, 6)
```

```
x=np.concatenate((a,b,x),axis = 1)
```

```
x.shape
```

```
(11231, 16)
```

# TRAIN TEST SPLIT

```
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)
```

```
x_test.shape
```

```
(2247, 16)
```

```
x_train.shape
```

```
(8984, 16)
```

```
y_test.shape
```

```
(2247, 1)
```

```
y_train.shape
```

```
(8984, 1)
```

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

# ALGORITHM

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

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

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

```
y_pred
```

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

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

```
desacc
```

```
0.8682688028482421
```

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

```
cm
```

```
array([[1783,  153],
       [ 143,  168]], dtype=int64)
```

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


index.html
C:/Users/prana/Downloads/Flask/Flask/templates/index.html
Prediction of Flight Delay
Enter the Flight Number : 1399
Month : 2
Day of Month : 4
Day of Week : 5
origin JFK
destination SEA
Scheduled Departure Time : 1
Scheduled Arrival Time : 22
Actual Departure Time : 1
SUBMIT
{{showcase}}


localhost:5000/prediction
localhost:5000/prediction
Prediction of Flight Delay
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 on time