Flight Delay Prediction For Aviation Industry Using Machine Learning

1 .INTRODUCTION

1.1 Overview

As people increasingly choose to travel by air, the amount of flights that fail to take off on time also increases. This growth exacerbates the crowded situation at airports and causes financial difficulties within the airline industry. Air transportation delay indicates the lack of efficiency of the aviation system.

It is a high cost to both airline companies and their passengers. According to the estimation by the Total Delay Impact Study, the total cost of air transportation delay to air travelers and the airline industry in 2007 was $32.9 billion in the US, resulting in a $4 billion reduction in GDP [**1**]. Therefore, predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy.

In this study, the main goal is to compare the performance of machine learning classification algorithms when predicting flight delays. The airport used in the study is John F. Kennedy International Airport that located in New York City. The information of flights leaving JFK airport between one-year periods was being analyzed. The study made use of several algorithms, and their predictions were evaluated using a number of measures.

The theoretical aspects of selected machine learning models and performance evaluation methods are explained in Section 3. In Section 2, related works by past researchers are discussed. The empirical processes and results of different models are presented and compared in Section 4. The conclusion of the comparative analysis and directions for future research are presented in Section 5.

1.2 Purpose

Therefore, predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy. In this study, the main goal is to compare the performance of machine learning classification algorithms when predicting flight delays.

2.Problem Definition & Design Thinking

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

2.1 Empathy Map

2.2 Ideation & Brainstroming Map

3. RESULT :

The model is designed using Python in Tensor flow and is installed on a system of 40 core CPU at a frequency of 2.6 hz, 80 G RAM and 250 G Hard. The flight info data is an open dataset collected by the Bureau of Transportation Statistics of United State Department of Transportation [163] where, the reason for delay is due to canceled or flight delay, and time duration of each flight. Model testing and training employs these data that include 18 million records.Model, uses 80% of data for training and the remaining 20% for testing [164]. Finally, the model evaluation considers two analysis which are studied in the following section.

First analysis

In order to evaluate the model, the number of denoising autoencodersand neurons must be determined based on the values for precision, accuracy and time consuming. In order to do this, at first, the model is trained using one stack and 64 neurons, and the precision and accuracy values are calculated. By adding another denoising autoencoder, the values for precision and accuracy are increased; therefore, another stack was added to the model's structure. On the other hand, by adding each stack denoising autoencoder to the structure, the processing time is also risen. Therefore, denoising autoencoder increment process should consider excellence between processing time and number of denoising autoencoder. As a result, adding denoising autoencoder addition is continued until differences of precision and accuracy for previous and newer structure exceed the threshold limit. Figure 6 shows the amount of accuracy based on number of denoising autoencoders and computation time.

4. Trail-head profile link

Team Lead - https://trailblazer.me/id/spkaran
Team Member 1- https://trailblazer.me/id/ranjithsrk10
Team Member 2- https://trailblazer.me/id/mmuthukumar8
Team Member 3- https://trailblazer.me/id/naresh2605

5. ADVANTAGES & DISADVANTAGES

Advantages:

- predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy.

Disadvantages:

- Flight delays not only irritate air passengers and disrupt their schedules but also cause a decrease in efficiency, an increase in capital costs, reallocation of flight crews and aircraft, and additional crew expenses.
- Inclement weather — such as thunderstorms, strong winds, and snow — can lead to flight delays or cancellations. Airlines must prioritize the safety of passengers and crew. So if the weather is too severe, they may have to delay or cancel flight

6. APPLICATIONS

Machine learning can be applied to predict flight delays and improve passanger retention in various ways. Here are some areas where machine learning can beapplied to enhance prediction of flight delays.

- 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.

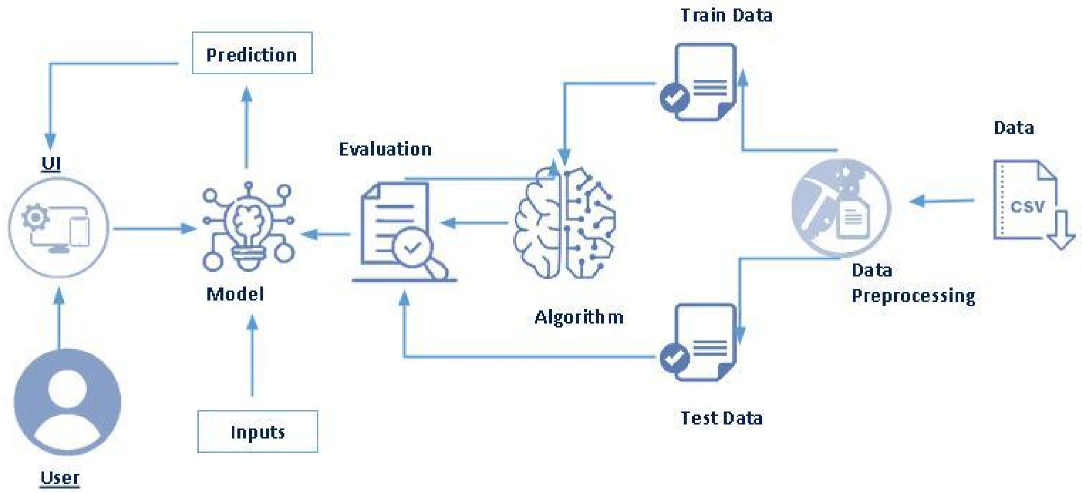
7. CONCLUSION

The paper performed a prediction of the occurrence of flight delays by adapting it into a machine learning problem. A supervised machine learning approach in the form of binary classification was used for the prediction. Seven algorithms were used for delay prediction, and four measures were used for algorithms performance evaluation. Due to the imbalanced nature of the data set, evaluation measures were weighted to eliminate the dominant effect of non-delayed flights over delayed flights. After applying classifiers to the delay prediction, the values of their four measures were compared to evaluate the performance of each model. The result shows that the highest values of accuracy, precision, recall, and f1- score are generated by the Decision Tree model (accuracy: 0.9778; precision: 0.9777; recall: 0.9778; f1-score: 0.9778). Such high values indicate that the Decision Tree performs well when predicting flight delays in the data set. Other tree-based ensemble classifiers also show good performance. Random Forest and Gradient Boosted Tree have an accuracy of 0.9240 and 0.9334, significantly higher than the rest of the models. The other four base classifiers Logistic Regression, KNN, Gaussian Naïve Bayes, and SVM, are not tree-based and did not show good performance. The KNN model is the worst performed since its precision and f1-score are the lowest among the seven models. The data set selected for this paper is imbalanced distributed, which may cause significant variation in the performance of each algorithm. In this paper, this problem was solved by the use of weighted evaluation measures. For future studies, using techniques such as SMOTE can better resolve this imbalance and improve the prediction. The result of algorithm comparison shows that tree-based ensemble algorithms tend to better predict flight delays of this data set. It will be valuable to repeat similar experimental processes using more tree-based ensemble algorithms to discover their significance in flight delay prediction.

8. FUTURE SCOPE

predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy. In this study, the main goal is to compare the performance of machine learning classification algorithms when predicting flight delays.The results show that adverse weatherconditions, low ceilings, and low visibility conditions strongly influence flight delays. Similarly, Asfe et al. [2] investigated the major causal factors of flight delays by ranking different factors using the analytical hierarchical process.

Technical Architecture :



Importing The Libraries :

```
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
```

Read the Dataset :

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

Python

```
dataset.head()
```

Python

	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

Handling Missing Values :

```
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
```

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

[18]

```
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
```

```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)
```

	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

Handling Categorical Values :

```
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]])
```

```python
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)
```

```python
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.]])
```

```python
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.]])
```

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

Descriptive Statistical :

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

Python

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_NUM	ORIGIN_AIRPORT_ID	DEST_AIRPORT_ID	CRS_DEP_TIME	DEP_TIME	...
count	11231.0	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11124.000000	...
mean	2016.0	2.544475	6.628973	15.790758	3.960199	1334.325617	12334.516695	12302.274508	1320.798326	1327.189410	...
std	0.0	1.090701	3.354678	8.782056	1.995257	811.875227	1595.026510	1601.988550	490.737845	500.306462	...
min	2016.0	1.000000	1.000000	1.000000	1.000000	7.000000	10397.000000	10397.000000	10.000000	1.000000	...
25%	2016.0	2.000000	4.000000	8.000000	2.000000	624.000000	10397.000000	10397.000000	905.000000	905.000000	...
50%	2016.0	3.000000	7.000000	16.000000	4.000000	1267.000000	12478.000000	12478.000000	1320.000000	1324.000000	...
75%	2016.0	3.000000	9.000000	23.000000	6.000000	2032.000000	13487.000000	13487.000000	1735.000000	1739.000000	...
max	2016.0	4.000000	12.000000	31.000000	7.000000	2853.000000	14747.000000	14747.000000	2359.000000	2400.000000	...

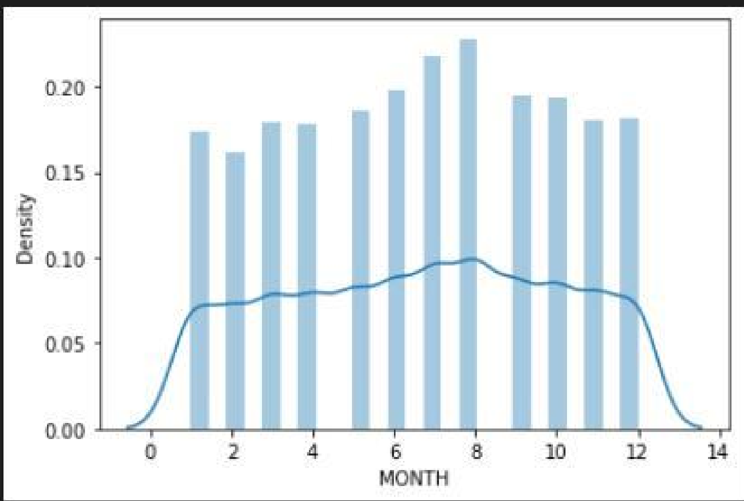
	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	DISTANCE	Unnamed: 25
count	11231.000000	11116.000000	11043.000000	11043.000000	11231.000000	11231.000000	11231.000000	11043.000000	11231.000000	0.0
mean	1537.312795	1523.978499	-2.573123	0.124513	0.010150	0.006589	190.652124	179.661233	1161.031965	NaN
std	502.512494	512.536041	39.232521	0.330181	0.100241	0.080908	78.386317	77.940399	643.683379	NaN
min	2.000000	1.000000	-67.000000	0.000000	0.000000	0.000000	93.000000	75.000000	509.000000	NaN
25%	1130.000000	1135.000000	-19.000000	0.000000	0.000000	0.000000	127.000000	117.000000	594.000000	NaN
50%	1559.000000	1547.000000	-10.000000	0.000000	0.000000	0.000000	159.000000	149.000000	907.000000	NaN
75%	1952.000000	1945.000000	1.000000	0.000000	0.000000	0.000000	255.000000	236.000000	1927.000000	NaN
max	2359.000000	2400.000000	615.000000	1.000000	1.000000	1.000000	397.000000	428.000000	2422.000000	NaN

rows × 22 columns

Univariate Analysis :

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

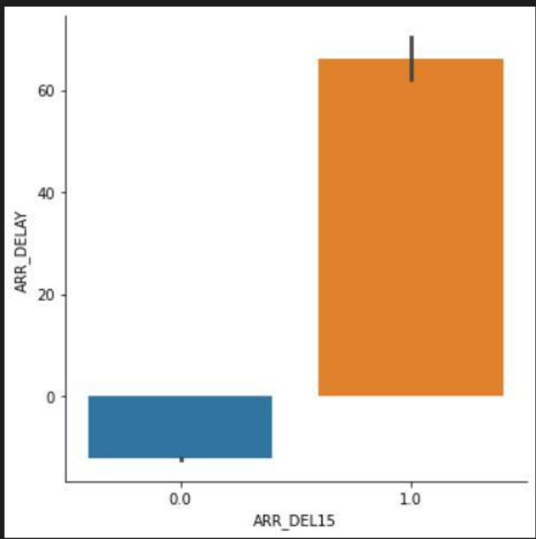
```
C:\Users\Saumya\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
figure-level function with similar flexibility) or `histplot` (an axes-lev
  warnings.warn(msg, FutureWarning)

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



BivariateAnalysis :

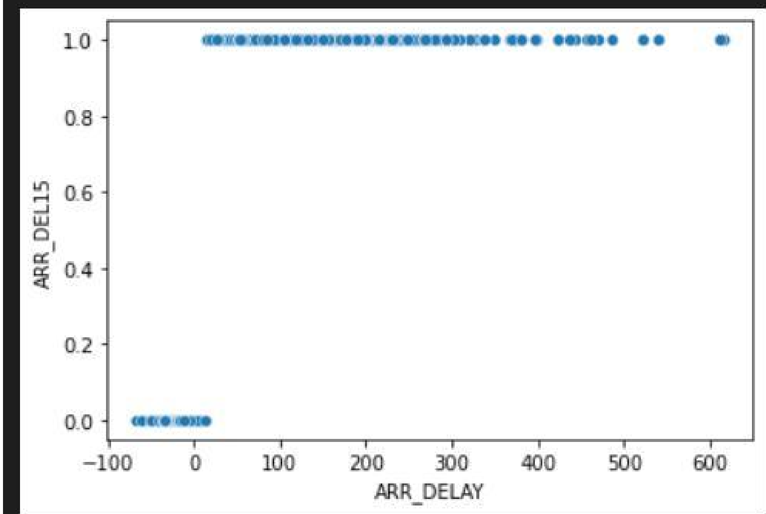
```
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=flight_data)
```

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




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

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

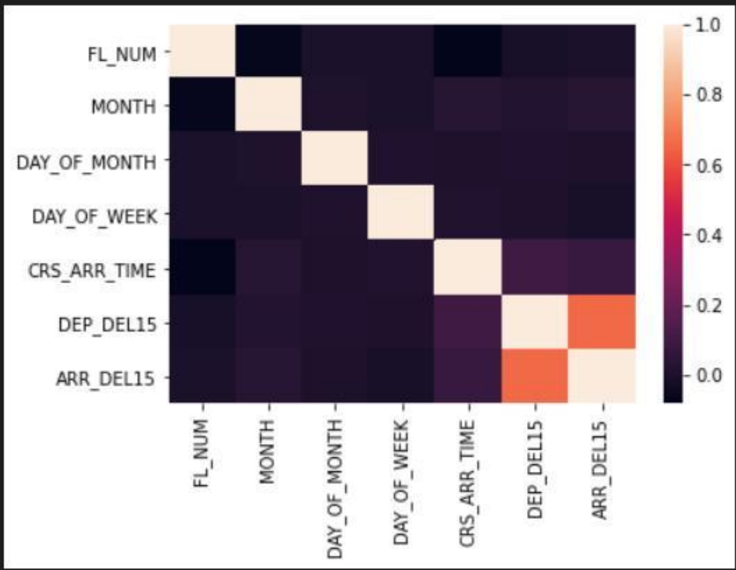


Multiivariate Analysis :

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

[32]

```
<AxesSubplot:>
```




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

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

```
x_test.shape
```

```
(2247, 16)
```

```
x_train.shape
```

```
(8984, 16)
```

```
y_test.shape
```

```
(2247, 1)
```

+ Code + Markdown

```
y_train.shape
```

```
(8984, 1)
```

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

Decision Tree Model :

```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)
```

Random Forest Model :

```
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)
```

ANN Model :

```
# 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
```

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.])
```

Compare The Model :

```
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
# 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
    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
```

Build Python Code :

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

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

app = Flask(__name__)#initializing the app

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


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

```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)
```

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

Run The Web Application :