FLIP ROBO

Flight Price Prediction

Submitted by:

ASHISH YADAV

ACKNOWLEDGMENT

We would like to express our deep and sincere gratitude to FlipnWork for giving us the opportunity to do this project. As a great bridge between academic and industry, this program educated us how to perform theoretical methodology in real life. We would like to express our sincere thankfulness to my mentor Sapna Verma for the continuous support , for their patience, enthusiasm, motivation and immense knowledge. As our academic mentor, Sapna Verma supported and helped in this project. Additionally, we would also like to thank all our friends who offered us some help, such as Divyanshu who helped me during this project

INTRODUCTION

Problem Statement:

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on - 1. Time of purchase patterns (making sure last-minute purchases are expensive) 2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases) So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

Business Goal:

We are required to model the price of flights with the available independent variables. This model will then be used by the management to understand flight fares. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Domain Understanding :

• The terms *'aviation industry'* and *'airline industry'* are sometimes thought of as being interchangeable, but they do actually describe different things. An airline is a business offering air transportation services for people or cargo, with the airline industry being the collective term used to describe these companies.

- The aviation industry has also been a key contributor to global economic prosperity, not only as a result of the tourism industry boosting local economies, but also because it has allowed for improvements to global trade.
- Meanwhile, the aviation industry also directly provides millions of jobs for people around the world, with examples including everything from pilots and cabin crew, through to air traffic controllers and aerospace engineers. On top of this, the aviation industry has helped to create many jobs in the wider travel and tourism industry too.

LITERATURE :

The main steps in my research were the following.

• Exploratory Data Analysis (EDA): By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.

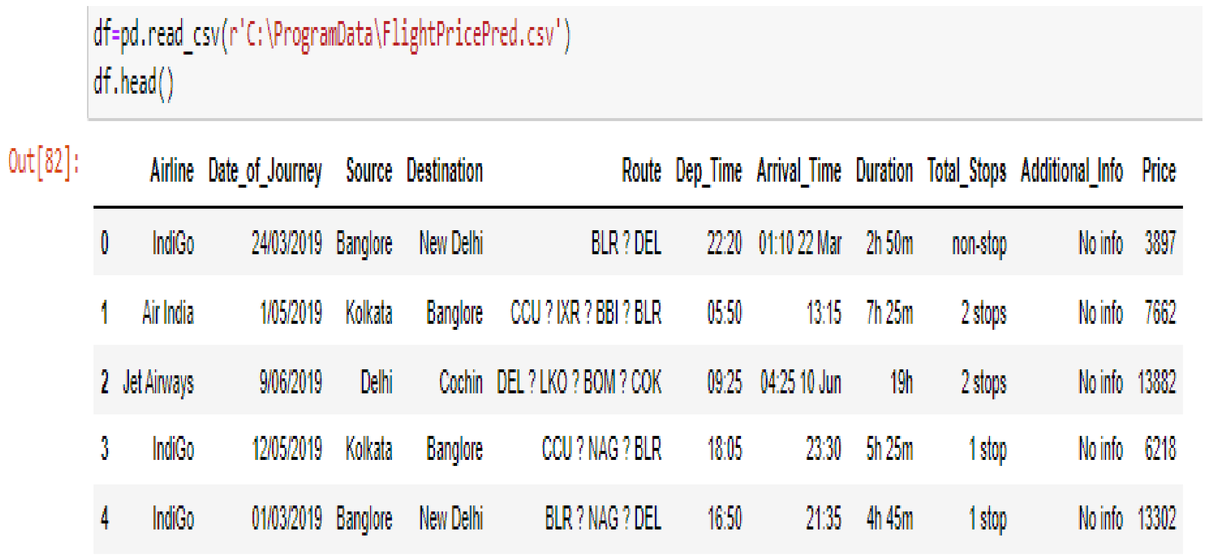
• Feature Selection: We select the best features out of available features to bring out the best model for the problem using feature selection

• Modeling: We apply Decision Tree , Random Forest , Linear Regression , K-Neighbours,Decision Tree,Gradient Boost models for prediction of the sale price

Analytical Problem Framing

• Mathematical/ Analytical Modeling of the Problem

Dataset :

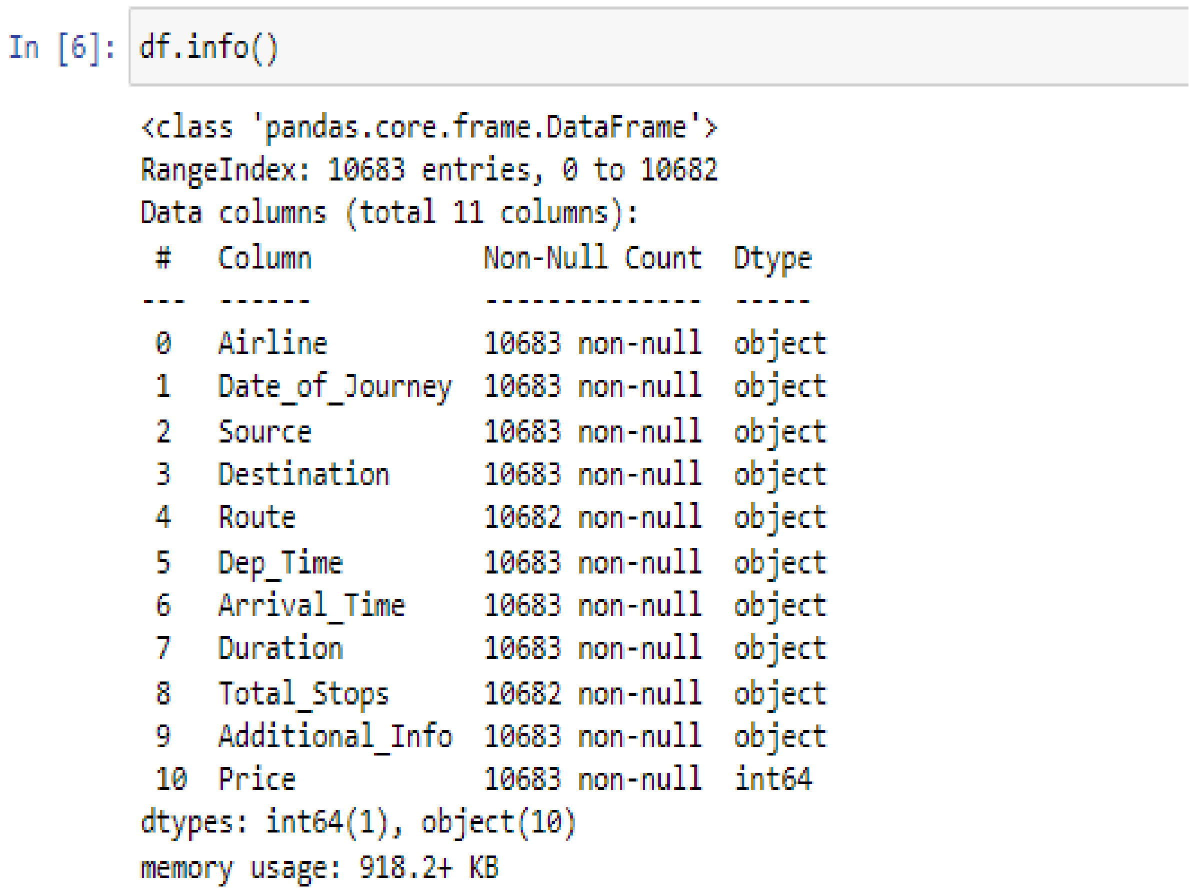


```
df=pd.read_csv(r'C:\ProgramData\FlightPricePred.csv')
df.head()
```

```
Out[82]:
     Airline  Date_of_Journey    Source  Destination                 Route  Dep_Time   Arrival_Time  Duration  Total_Stops  Additional_Info  Price
0     IndiGo       24/03/2019  Banglore    New Delhi             BLR ? DEL     22:20   01:10 22 Mar    2h 50m     non-stop          No info   3897
1  Air India        1/05/2019   Kolkata     Banglore   CCU ? IXR ? BBI ? BLR     05:50          13:15    7h 25m      2 stops          No info   7662
2  Jet Airways      9/06/2019     Delhi       Cochin   DEL ? LKO ? BOM ? COK     09:25   04:25 10 Jun       19h      2 stops          No info  13882
3     IndiGo       12/05/2019   Kolkata     Banglore       CCU ? NAG ? BLR     18:05          23:30    5h 25m       1 stop          No info   6218
4     IndiGo       01/03/2019  Banglore    New Delhi       BLR ? NAG ? DEL     16:50          21:35    4h 45m       1 stop          No info  13302
```

EXPLORATORY DATA ANALYSIS :

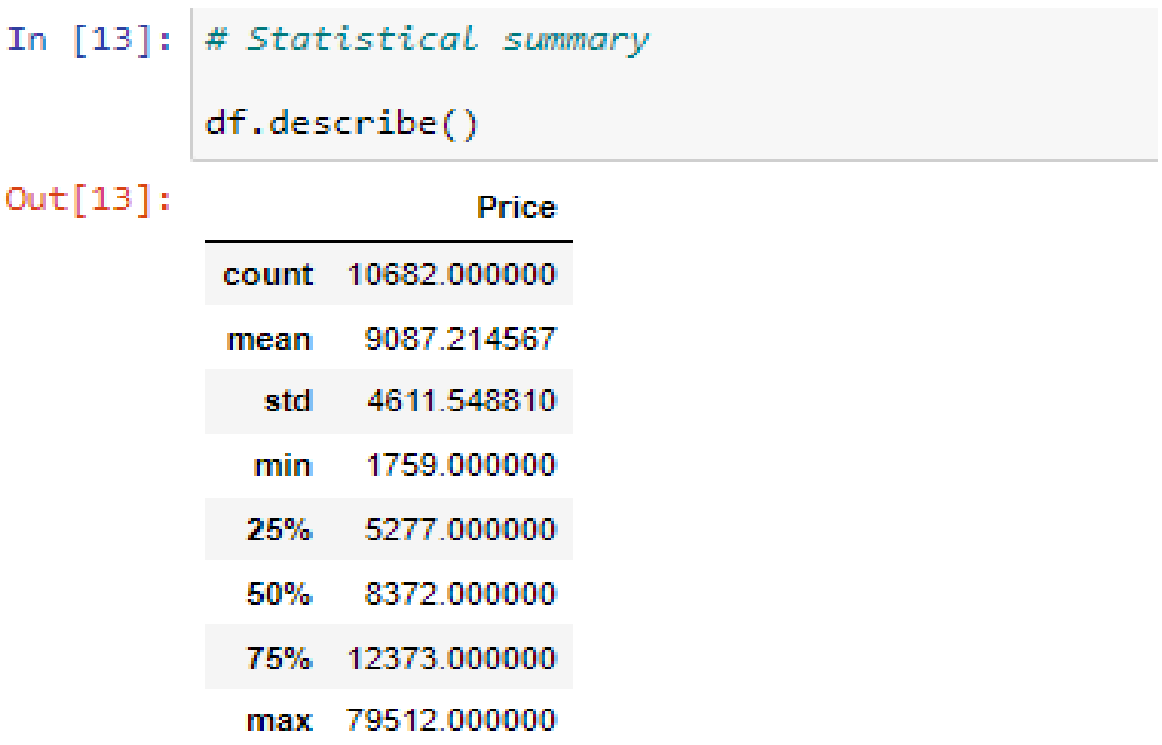
Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, Python. Before it can conduct analysis on data collected by multiple data sources and stored in data warehouses, an organization must know how many cases are in a data set, what variables are included, how many missing values there are and what general hypotheses the data is likely to support. An initial exploration of the data set can help answer these questions by familiarizing analysts with the data with which they are working.



```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

Statistical Summary:

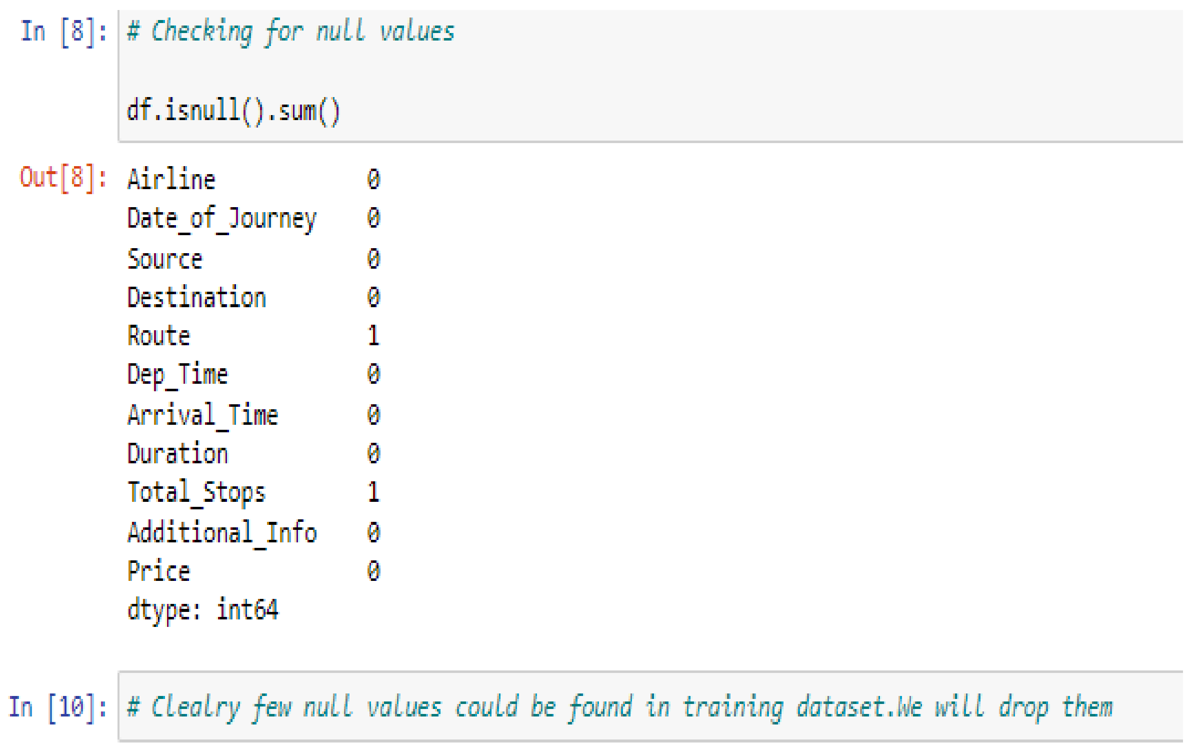


```
In [13]: # Statistical summary

         df.describe()
```

Out[13]:

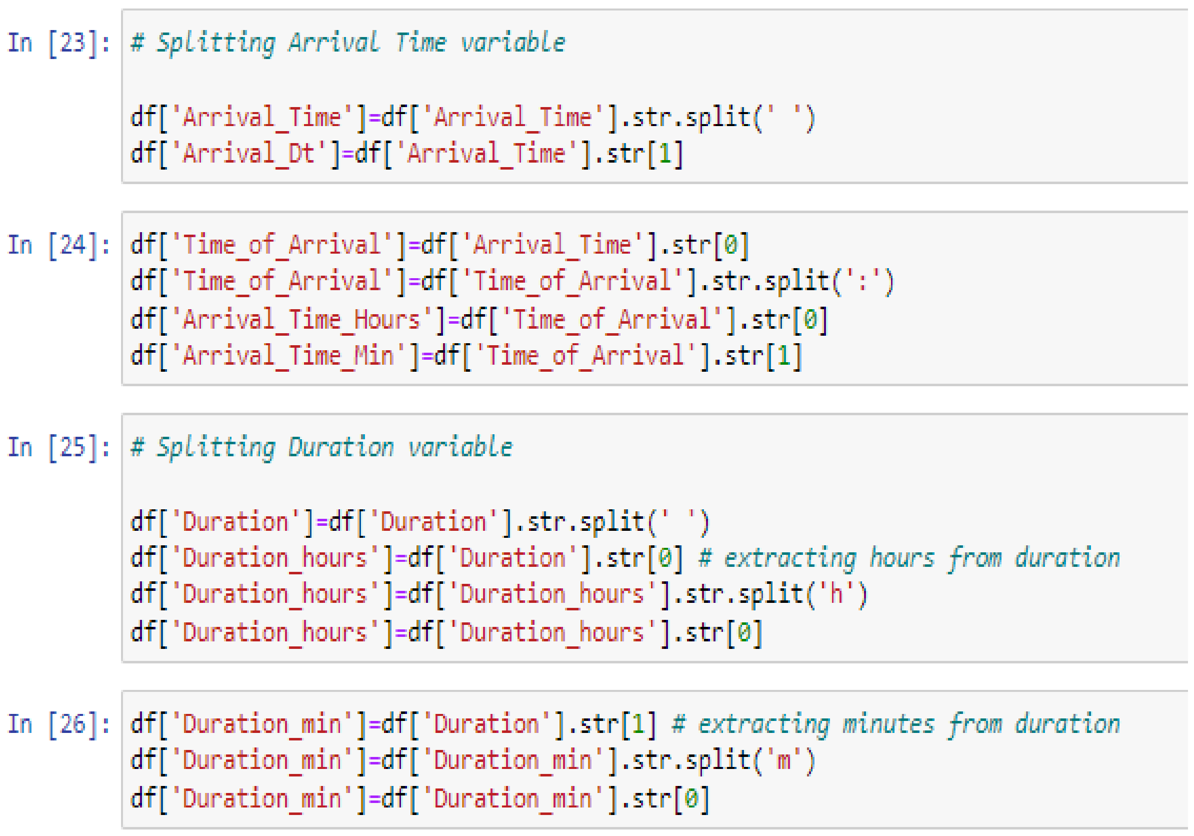
	Price
count	10682.000000
mean	9087.214567
std	4611.548810
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

Checking null values in dataset:



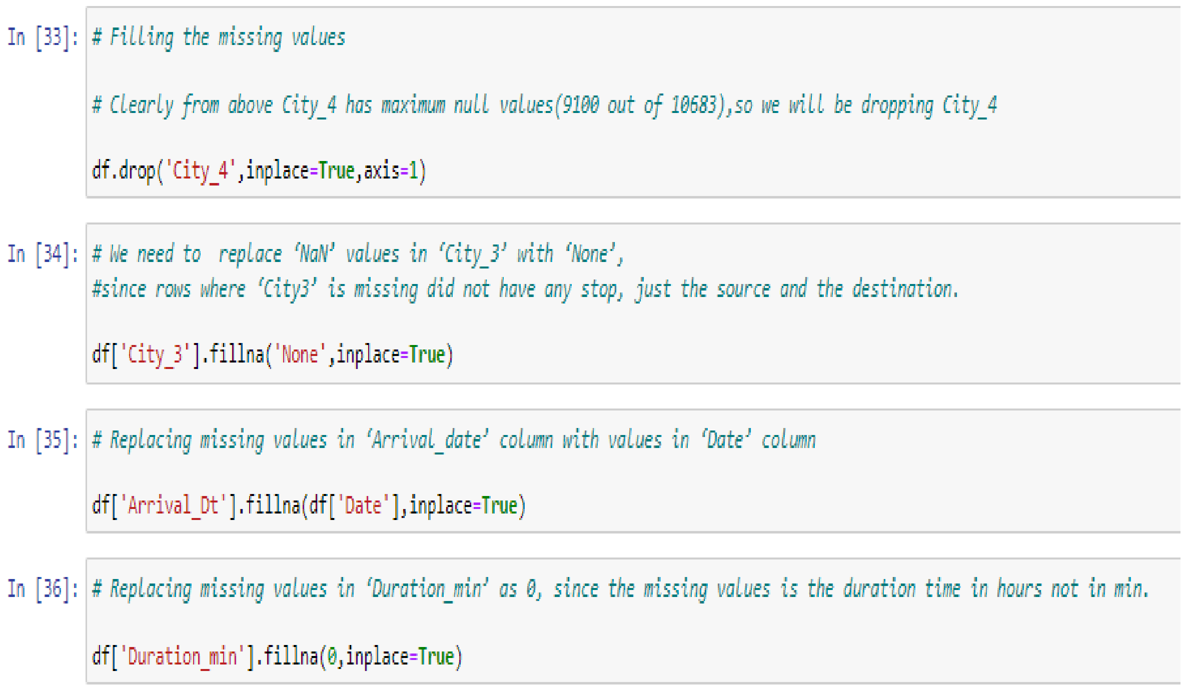
```
In [8]: # Checking for null values

        df.isnull().sum()

Out[8]: Airline            0
        Date_of_Journey    0
        Source             0
        Destination        0
        Route              1
        Dep_Time           0
        Arrival_Time       0
        Duration           0
        Total_Stops        1
        Additional_Info    0
        Price              0
        dtype: int64

In [10]: # Clealry few null values could be found in training dataset.We will drop them
```

Splitting Variables :



```
In [23]: # Splitting Arrival Time variable

         df['Arrival_Time']=df['Arrival_Time'].str.split(' ')
         df['Arrival_Dt']=df['Arrival_Time'].str[1]

In [24]: df['Time_of_Arrival']=df['Arrival_Time'].str[0]
         df['Time_of_Arrival']=df['Time_of_Arrival'].str.split(':')
         df['Arrival_Time_Hours']=df['Time_of_Arrival'].str[0]
         df['Arrival_Time_Min']=df['Time_of_Arrival'].str[1]

In [25]: # Splitting Duration variable

         df['Duration']=df['Duration'].str.split(' ')
         df['Duration_hours']=df['Duration'].str[0] # extracting hours from duration
         df['Duration_hours']=df['Duration_hours'].str.split('h')
         df['Duration_hours']=df['Duration_hours'].str[0]

In [26]: df['Duration_min']=df['Duration'].str[1] # extracting minutes from duration
         df['Duration_min']=df['Duration_min'].str.split('m')
         df['Duration_min']=df['Duration_min'].str[0]
```

Filling Null Values :



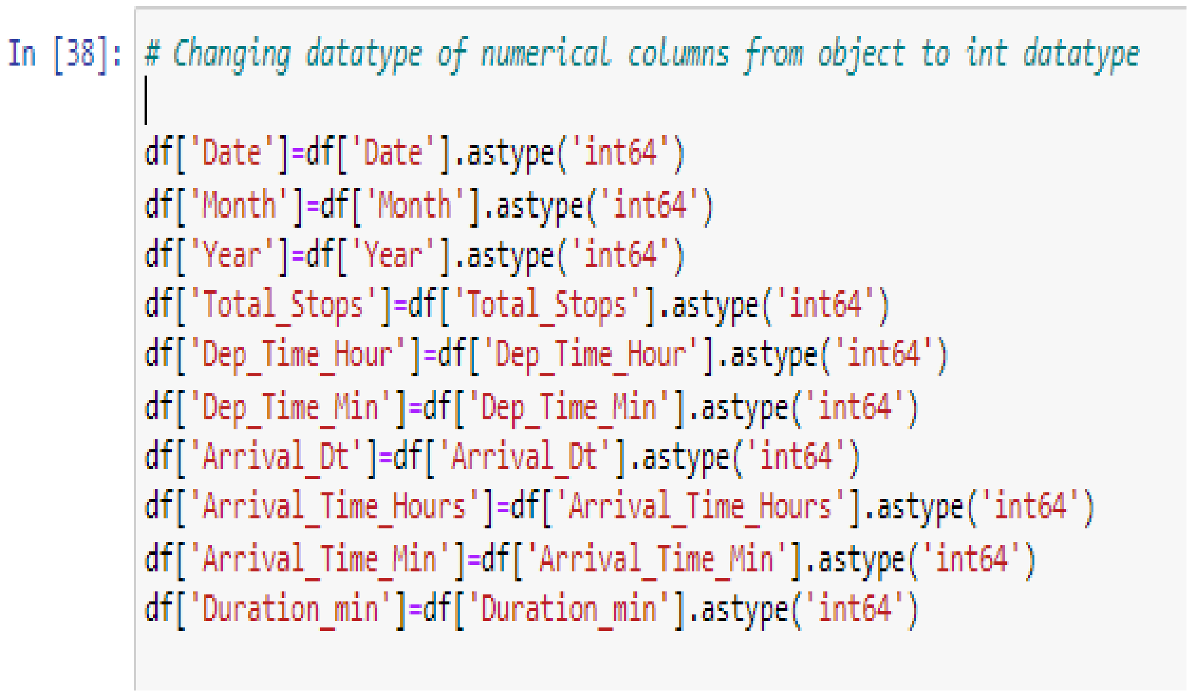
```
In [33]: # Filling the missing values

         # Clearly from above City_4 has maximum null values(9100 out of 10683),so we will be dropping City_4

         df.drop('City_4',inplace=True,axis=1)
```

```
In [34]: # We need to  replace 'NaN' values in 'City_3' with 'None',
         #since rows where 'City3' is missing did not have any stop, just the source and the destination.

         df['City_3'].fillna('None',inplace=True)
```

```
In [35]: # Replacing missing values in 'Arrival_date' column with values in 'Date' column

         df['Arrival_Dt'].fillna(df['Date'],inplace=True)
```

```
In [36]: # Replacing missing values in 'Duration_min' as 0, since the missing values is the duration time in hours not in min.

         df['Duration_min'].fillna(0,inplace=True)
```

Changing datatype of numerical columns from object to int datatype :

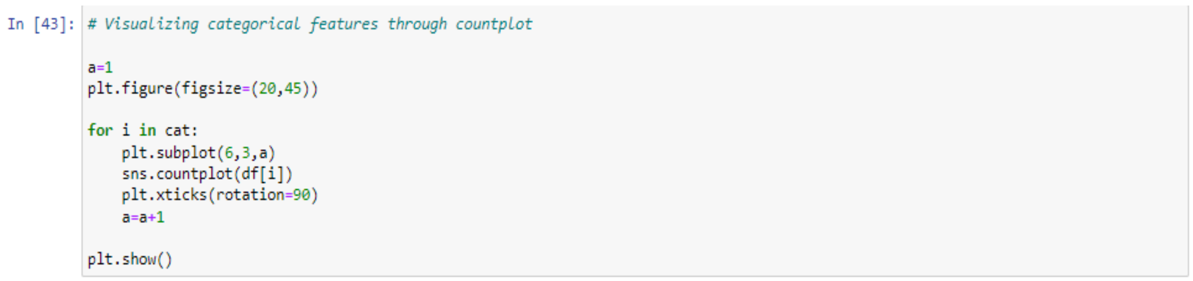


```
In [38]: # Changing datatype of numerical columns from object to int datatype

         df['Date']=df['Date'].astype('int64')
         df['Month']=df['Month'].astype('int64')
         df['Year']=df['Year'].astype('int64')
         df['Total_Stops']=df['Total_Stops'].astype('int64')
         df['Dep_Time_Hour']=df['Dep_Time_Hour'].astype('int64')
         df['Dep_Time_Min']=df['Dep_Time_Min'].astype('int64')
         df['Arrival_Dt']=df['Arrival_Dt'].astype('int64')
         df['Arrival_Time_Hours']=df['Arrival_Time_Hours'].astype('int64')
         df['Arrival_Time_Min']=df['Arrival_Time_Min'].astype('int64')
         df['Duration_min']=df['Duration_min'].astype('int64')
```

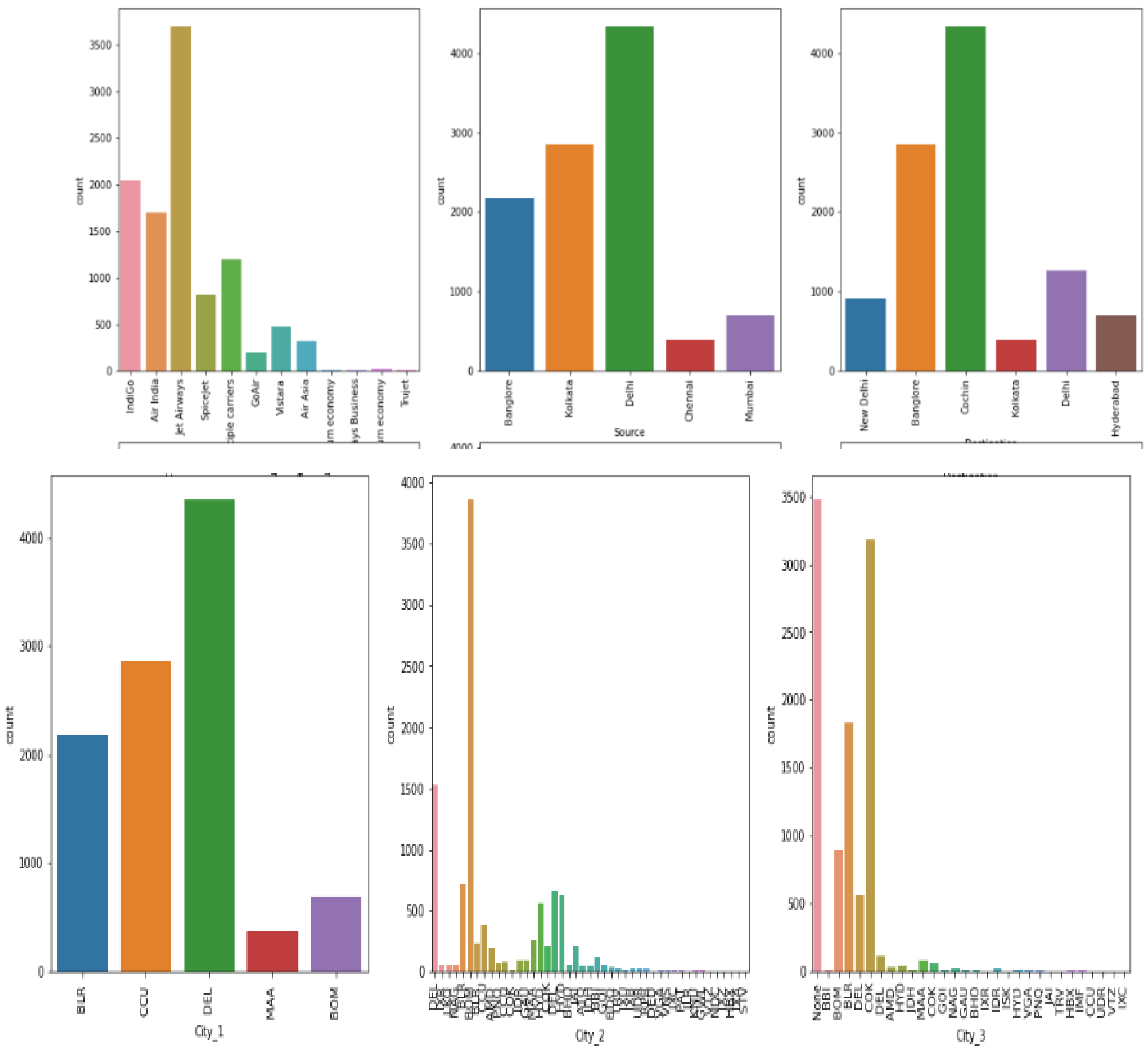
DATA VISUALIZATION :

- Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

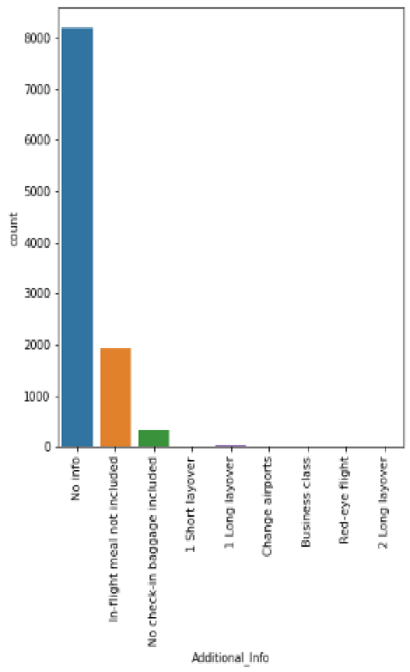
HISTOGRAM PLOTS :



```
In [43]: # Visualizing categorical features through countplot

         a=1
         plt.figure(figsize=(20,45))

         for i in cat:
             plt.subplot(6,3,a)
             sns.countplot(df[i])
             plt.xticks(rotation=90)
             a=a+1

         plt.show()
```



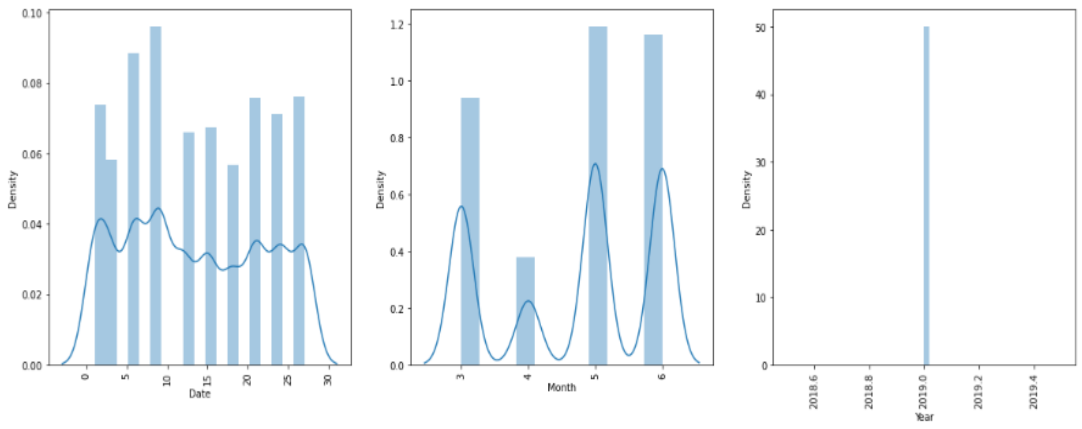
HISTOGRAM PLOTS:



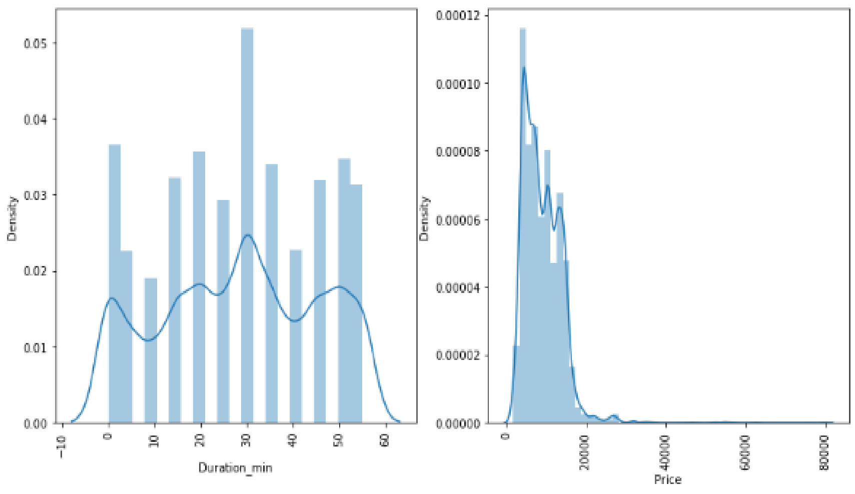
```
In [44]: # Important observation from above plots:

         # 1) Jet Airways is the most preferred airline with the highest row count, followed by Indigo and AirIndia
         # 2) Maximum flights take off from Delhi,least from Chennai
         # 3) Maximum flights land in Cochin,least in Kolkatta
         # 4) Majority of the flights have a stop in Bombay.
```

DISTRIBUTION PLOT :

```
In [45]: # Distibution plots of numerical columns:

         a=1
         plt.figure(figsize=(20,45))

         for i in num:
             plt.subplot(6,3,a)
             sns.distplot(df[i])
             plt.xticks(rotation=90)
             a=a+1

         plt.show()
```



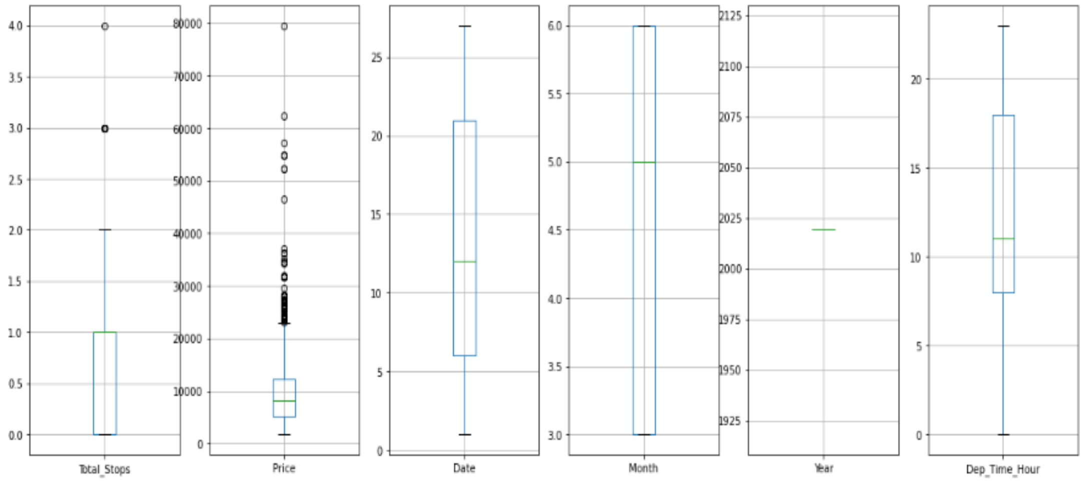
DISTRIBUTION PLOTS :



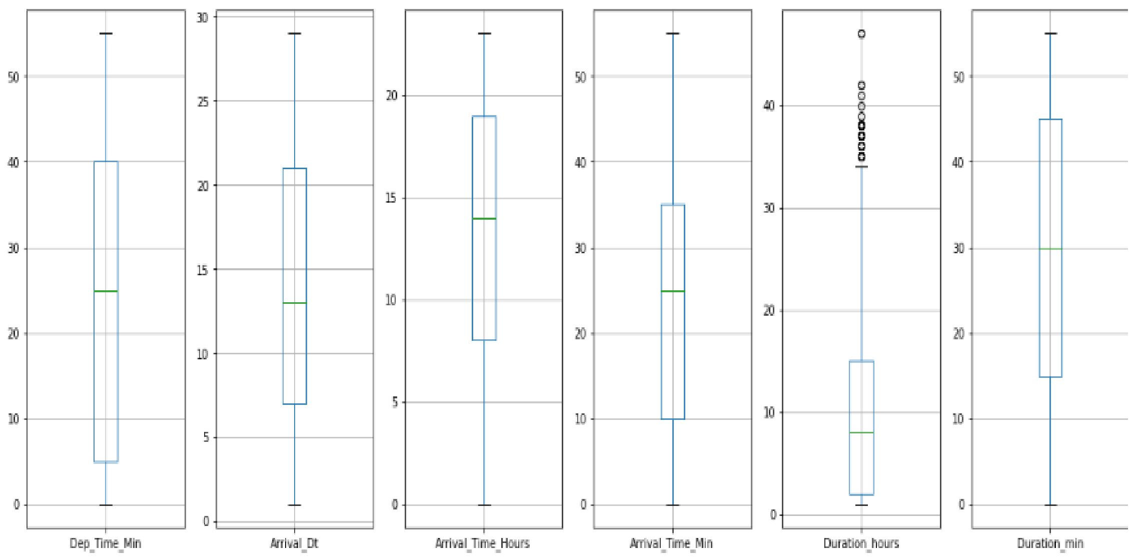
```
n [46]: # Important observations from above plots :

        # 1) Maximum flights have stops as 1, flights with 3 and 4 stops are quite low
        # 2) Majority of the flights  fly in the early morning time
        # 3) Majority of the flights take off and land on the same day
        # 4) Price column is rightly skewed and maximum flights are under 20000 price
```

OUTLIERS :

```
In [47]: # plotting boxplot to check outliers

         df.plot(kind='box', subplots=True, layout=(3,6), figsize=(20,25), grid=True)
         plt.show

Out[47]: <function matplotlib.pyplot.show(close=None, block=None)>
```

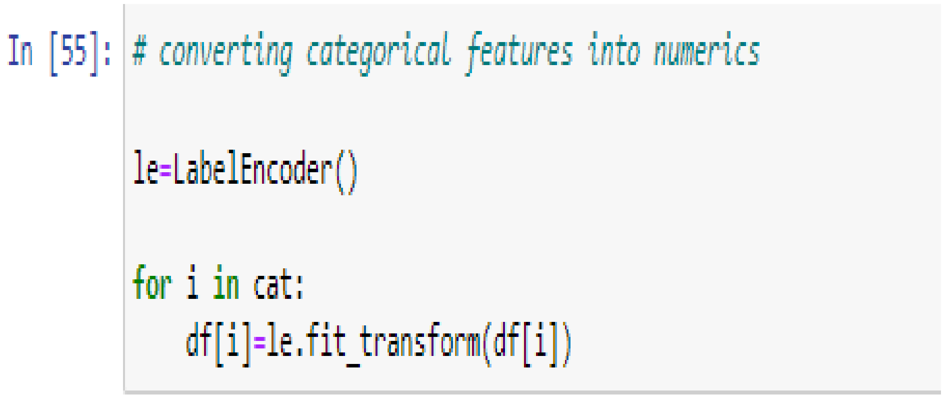


OUTLIERS :



```
n [48]: # Clearly outliers could be seen in Price,Duration_hours and Total_Stops columns
        # Making changes in Duration_hours and Total_Stops column would affect Price ,so we will not remove outliers
```

LABEL ENCODING :

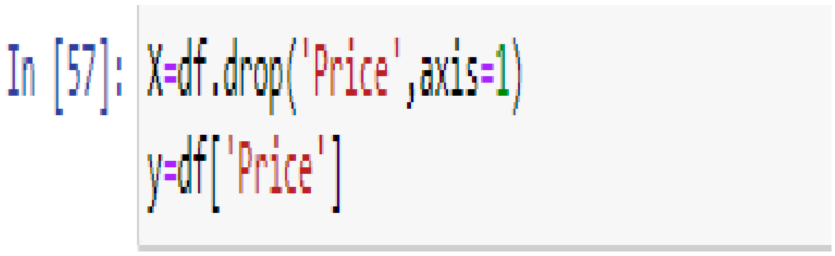


```
In [55]: # converting categorical features into numerics

         le=LabelEncoder()

         for i in cat:
             df[i]=le.fit_transform(df[i])
```

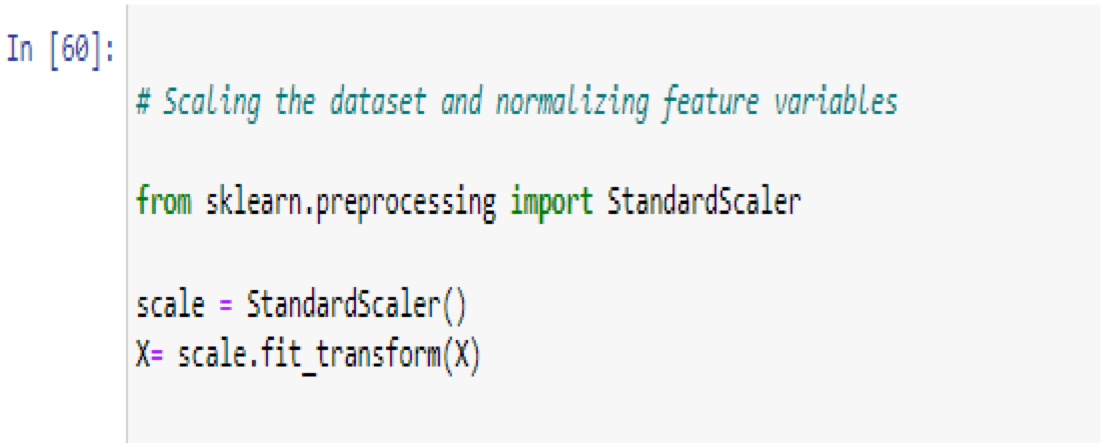
SKEWNESS :

```
In [49]: # checking skewness in dataset

         df.skew().sort_values()

Out[49]: Arrival_Time_Hours   -0.378863
         Month                -0.377753
         Duration_min         -0.086035
         Year                  0.000000
         Dep_Time_Hour         0.108461
         Arrival_Time_Min      0.110791
         Date                  0.124448
         Arrival_Dt            0.125494
         Dep_Time_Min          0.171488
         Total_Stops           0.332400
         Duration_hours        0.892611
         Price                 1.858221
         dtype: float64
```

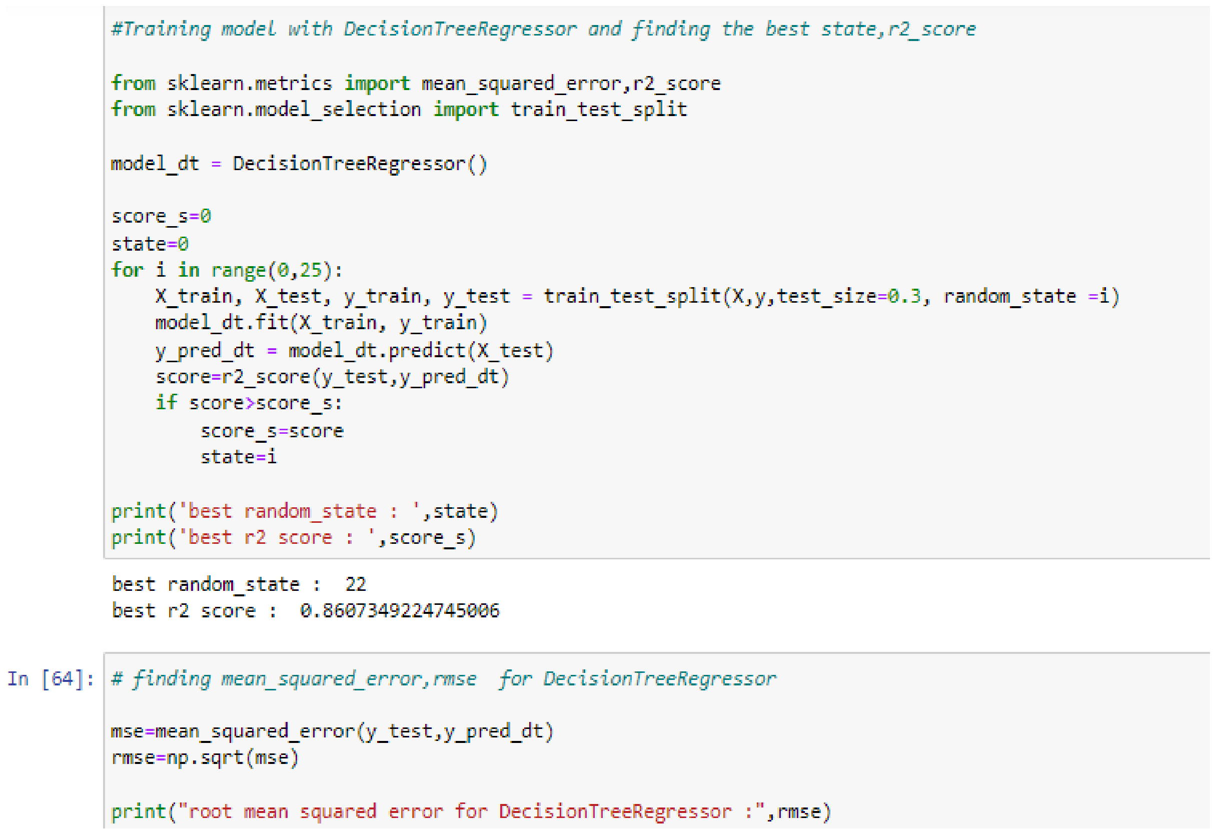
SPLITTING DATASET INTO X,Y :


```
In [57]: X=df.drop('Price',axis=1)
         y=df['Price']
```

SCALING OF DATASET :


```
In [60]: 
         # Scaling the dataset and normalizing feature variables

         from sklearn.preprocessing import StandardScaler

         scale = StandardScaler()
         X= scale.fit_transform(X)
```

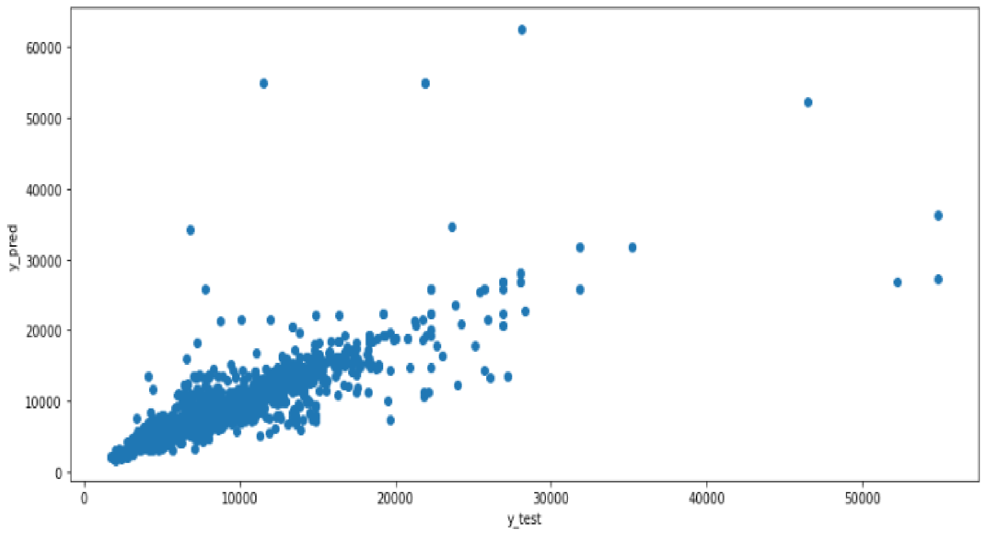
EVALUATION OF MODELS :


```
#Training model with DecisionTreeRegressor and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_dt = DecisionTreeRegressor()

score_s=0
state=0
for i in range(0,25):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_dt.fit(X_train, y_train)
    y_pred_dt = model_dt.predict(X_test)
    score=r2_score(y_test,y_pred_dt)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  22
best r2 score :  0.8607349224745006
```

```
In [64]: # finding mean_squared_error,rmse  for DecisionTreeRegressor

         mse=mean_squared_error(y_test,y_pred_dt)
         rmse=np.sqrt(mse)

         print("root mean squared error for DecisionTreeRegressor :",rmse)
```

In [66]:
```
# # plotting original training data wth predicted values for DecisionTreeRegressor model

plt.figure(figsize=(14,6))
plt.scatter(x=y_test,y=y_pred_dt)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



In [67]:
```
# The predicted values looks close enough to original values with We are getting a best fit line for DecisionTreeRegressor
# and we are getting a best fit line.So model seems to be doing good
```

```
#Training model with RandomForestRegressor and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_rfr = RandomForestRegressor()

score_s=0
state=0
for i in range(0,10):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_rfr.fit(X_train, y_train)
    y_pred_rfr = model_rfr.predict(X_test)
    score=r2_score(y_test,y_pred_rfr)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  4
best r2 score :  0.9197948797323932
```

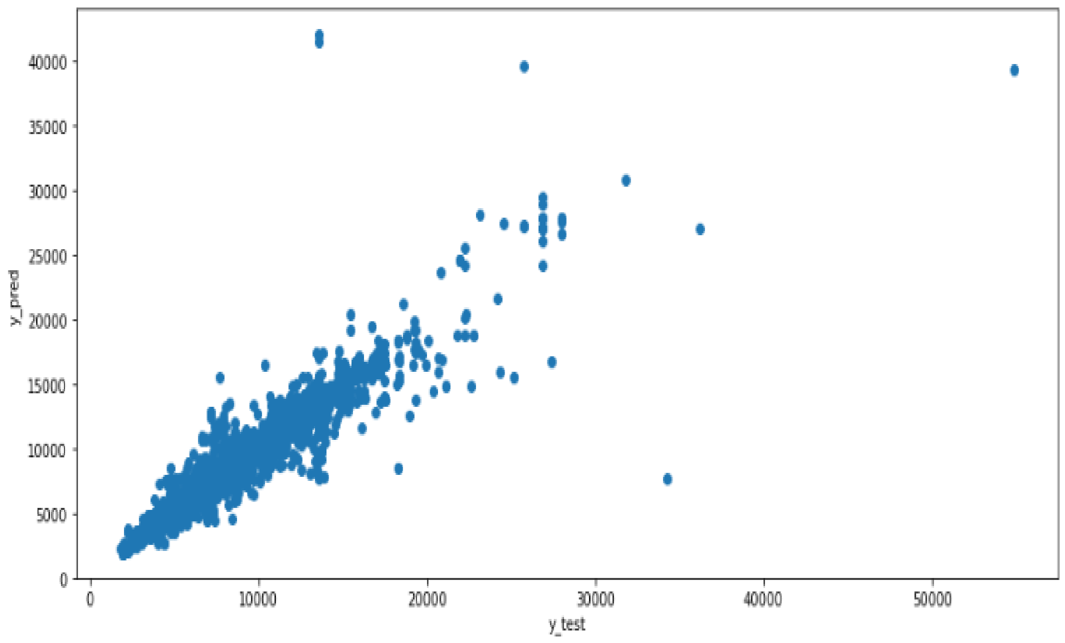
In [77]:

```
# finding mean_squared_error,rmse  for RandomForestRegressor

mse=mean_squared_error(y_test,y_pred_rfr)
rmse=np.sqrt(mse)

print("mean squared error for RandomForestRegressor :",mse)
print("root mean squared error for RandomForestRegressor :",rmse)
```

```
mean squared error for RandomForestRegressor : 2077386.0521827196
root mean squared error for RandomForestRegressor : 1441.3140019380646
```

In [78]:

```
# # plotting original training data wth predicted values for RandomForestRegressor model

plt.figure(figsize=(14,6))
plt.scatter(x=y_test,y=y_pred_rfr)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



In [91]:
```
# Evaluation of models

#Training model with GradientBoostingRegressor and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_gbr = GradientBoostingRegressor()

score_s=0
state=0
for i in range(0,10):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_gbr.fit(X_train, y_train)
    y_pred_gbr = model_gbr.predict(X_test)
    score=r2_score(y_test,y_pred_gbr)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  4
best r2 score :  0.8496985086459006
```

In [84]:
```
score=r2_score(y_train,model_gbr.predict(X_train))
print(score)
```

```
0.8455071246981563
```

In [ ]:
```
#RandomForestRegressor model gives us the best accuracy, with an R2 score of 91%, but the model is overfitting on training data.
#Gradient boosting also gives a score of 84%, which is better than K-Neighbors and the model is not overfitting as well.
```

HYPERPARAMETER TUNING :

In [92]:
```
#  HYPERPARAMETER TUNING OF GRADIENTBOOSTINGREGRESSOR MODEL


search_grid={'alpha':[0.9,0.09,0.1],'n_estimators':[100,50,10],'learning_rate':[0.1,0.01],'max_depth':[3,4,5],'min_samples_split'
search=GridSearchCV(model_gbr,search_grid,cv=3)
```

In [93]:
```
# Fitting the model

search.fit(X_train,y_train)
```

Out[93]:
```
GridSearchCV(cv=3, estimator=GradientBoostingRegressor(),
             param_grid={'alpha': [0.9, 0.09, 0.1],
                         'learning_rate': [0.1, 0.01], 'max_depth': [3, 4, 5],
                         'min_samples_leaf': [1, 2, 3],
                         'min_samples_split': [2, 3, 4],
                         'n_estimators': [100, 50, 10]})
```

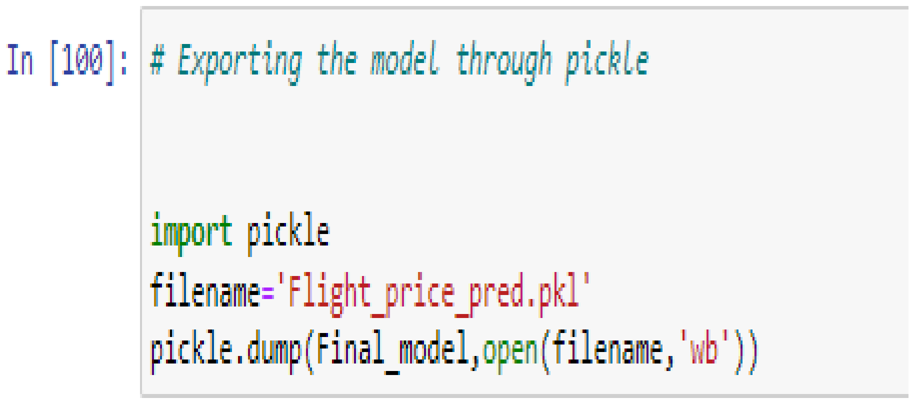
In [94]:
```
search.best_params_
```

Out[94]:
```
{'alpha': 0.9,
 'learning_rate': 0.1,
 'max_depth': 5,
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 100}
```

In [95]:
```
print("Best score", search.best_score_)
```

```
Best score 0.8707683106526228
```

EXPORTING THE MODEL :



```
In [100]: # Exporting the model through pickle


          import pickle
          filename='Flight_price_pred.pkl'
          pickle.dump(Final_model,open(filename,'wb'))
```

CONCLUSION:

# Clearly from above best score,we can conclude that Gradient Boosting Regression is a good choice for predicting Flight prices.