FLIP ROBO

NAME OF THE PROJECT

Flight Price Prediction

Submitted by:

Shubham Nalawade

ACKNOWLEDGMENT

This is to acknowledge that i have done all my work with great sincerity and enthusiasm. I would like to thank madam Sapna Verma who has given me the opportunity to work on **Flight Price Prediction** project. While working on the project i experienced lot of problems and hurdles but i am glad that i completed my project on time. In this project i have learned a lot of things and i think that this project will prove to bring a breakthrough in my future career.

A special thank to the company FLIPROBO to provide me such an exciting data which excite me a lot and got to learn many things from the data.

Problem Statement:-

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, and it will be a different story.

To solve this problem, we have scrapped with prices of flight tickets for various airlines between the months of January and march of 2022 and between various cities, using which we aim to build a model which predicts the prices of the flights using various input features.

Objective—

The objective of this article is to predict flight prices given the various parameters. Data used in this article is scraped from various websites for ex:- yatra.com,easymytrip.com,travolook.com. This will be a regression problem since the target or dependent variable is the price (continuous numeric value).

INTRODUCTION :-

Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices.

Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

- **The Dataset:-**
  The dataset contains 10780 rows and 10 columns scraped from various websites. The output 'Price' column needs to be predicted in this set. We will use Regression techniques here, since the predicted output will be a continuous value.

Following is the description of features available in the dataset –

1. Flight Name : Name of the Flight
2. Departure time: The time when the journey starts from the source.
3. Date of Journey:- The date of the journey
4. Source: The source from which the service begins.
5. Destination: The destination where the service ends.
6. Arrival Time: Time of arrival at the destination.
7. Duration: Total duration of the flight.
8. No of Stop: Total stops between the source and destination.
9. Duration: Total duration of the flight.
10. Price: Fare of Flights

- **Contents of the article :-**

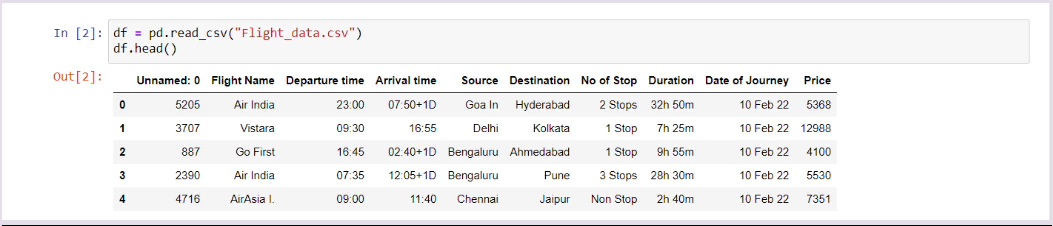
This article explains the complete process to build a machine learning model. Below mentioned are the various phases that we will go through, throughout the project: -

1. Exploratory data analysis and Data Modeling
2. Outlier detection and skewness treatment
3. Encoding the data — Label Encoder
4. Fitting the machine learning models
5. Cross-validation of the selected model
6. Model hyper Parameter-tuning
7. Saving the final model and prediction using saved model

So let’s begin exploring our data set and start building a prediction model.

- **Exploratory Data Analysis and Data Modeling:-**

We load the dataset using Pandas library



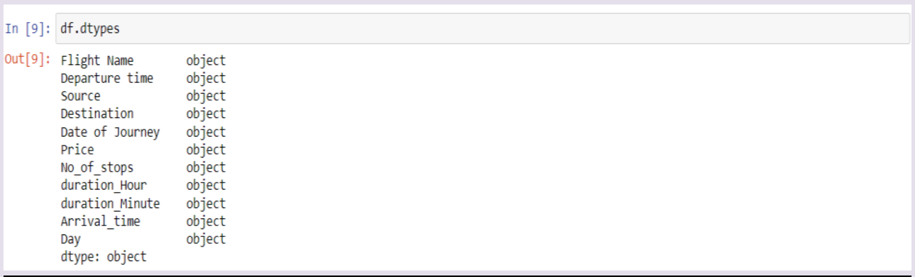
```
In [2]: df = pd.read_csv("Flight_data.csv")
        df.head()
```

Out[2]:

	Unnamed: 0	Flight Name	Departure time	Arrival time	Source	Destination	No of Stop	Duration	Date of Journey	Price
0	5205	Air India	23:00	07:50+1D	Goa In	Hyderabad	2 Stops	32h 50m	10 Feb 22	5368
1	3707	Vistara	09:30	16:55	Delhi	Kolkata	1 Stop	7h 25m	10 Feb 22	12988
2	887	Go First	16:45	02:40+1D	Bengaluru	Ahmedabad	1 Stop	9h 55m	10 Feb 22	4100
3	2390	Air India	07:35	12:05+1D	Bengaluru	Pune	3 Stops	28h 30m	10 Feb 22	5530
4	4716	AirAsia I.	09:00	11:40	Chennai	Jaipur	Non Stop	2h 40m	10 Feb 22	7351

1. we can see that the dataset have a unnecessary column which is Unknown:0, we have to delete that column.
2. Arrival time has extra +1D which has to be separated from that column.
3. No of stops has strings as well as integer type data so we need to separate string and integer data.
4. Date of journey is not in date time format so, we need to format it into date time format.
5. We need to separate hour and minutes from departure time as well as arrival time.
6. We can also see that price has , in between so we need to replace “,” with empty .

<u>We further proceed to explore the dataset:-</u>

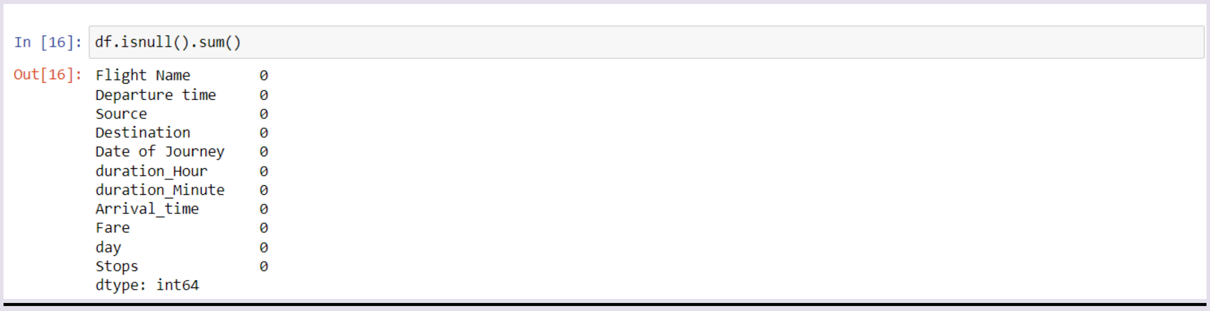


```
In [9]: df.dtypes

Out[9]: Flight Name        object
        Departure time     object
        Source             object
        Destination        object
        Date of Journey    object
        Price              object
        No_of_stops        object
        duration_Hour      object
        duration_Minute    object
        Arrival_time       object
        Day                object
        dtype: object
```

We run the data.info() command, which gives us the information about number of values present in each column, and data types of each column.

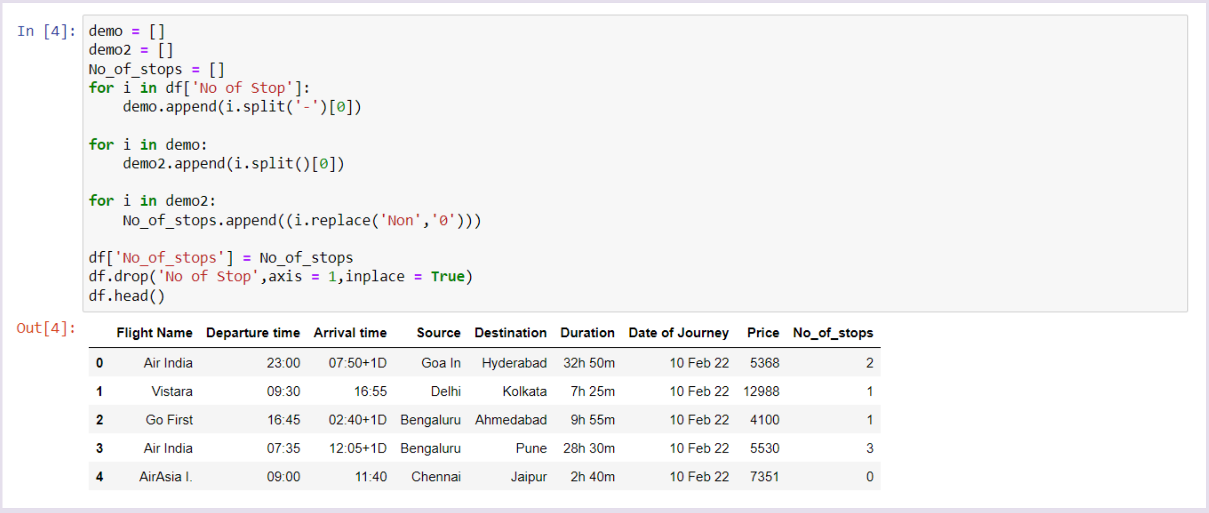
We observed that every column has object type data so we need to convert departure time and arrival time and also date of journey into date time format. Day and price into integer and Flight name, source into strings.

<u>We now check the count of NaN (null) values in our dataset, which turns out to give the following result –</u>



```
In [16]: df.isnull().sum()

Out[16]: Flight Name        0
         Departure time     0
         Source             0
         Destination        0
         Date of Journey    0
         duration_Hour      0
         duration_Minute    0
         Arrival_time       0
         Fare               0
         day                0
         Stops              0
         dtype: int64
```

We can observe that the dataset has no any null values. So, we can move ahead for further exploration.

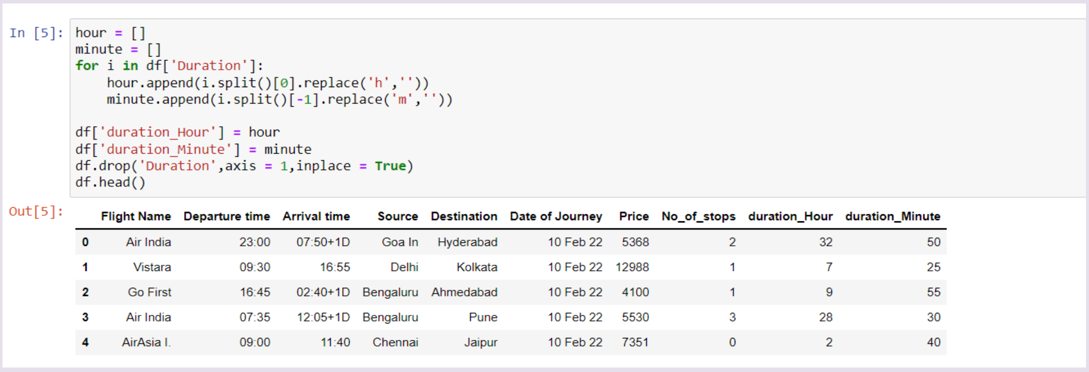


```
In [4]: demo = []
        demo2 = []
        No_of_stops = []
        for i in df['No of Stop']:
            demo.append(i.split('-')[0])

        for i in demo:
            demo2.append(i.split()[0])

        for i in demo2:
            No_of_stops.append((i.replace('Non','0')))

        df['No_of_stops'] = No_of_stops
        df.drop('No of Stop',axis = 1,inplace = True)
        df.head()
```

Out[4]:

	Flight Name	Departure time	Arrival time	Source	Destination	Duration	Date of Journey	Price	No_of_stops
0	Air India	23:00	07:50+1D	Goa In	Hyderabad	32h 50m	10 Feb 22	5368	2
1	Vistara	09:30	16:55	Delhi	Kolkata	7h 25m	10 Feb 22	12988	1
2	Go First	16:45	02:40+1D	Bengaluru	Ahmedabad	9h 55m	10 Feb 22	4100	1
3	Air India	07:35	12:05+1D	Bengaluru	Pune	28h 30m	10 Feb 22	5530	3
4	AirAsia I.	09:00	11:40	Chennai	Jaipur	2h 40m	10 Feb 22	7351	0

Here we have created a new column no of stops and separated the integer values from the original column and deleted the original col

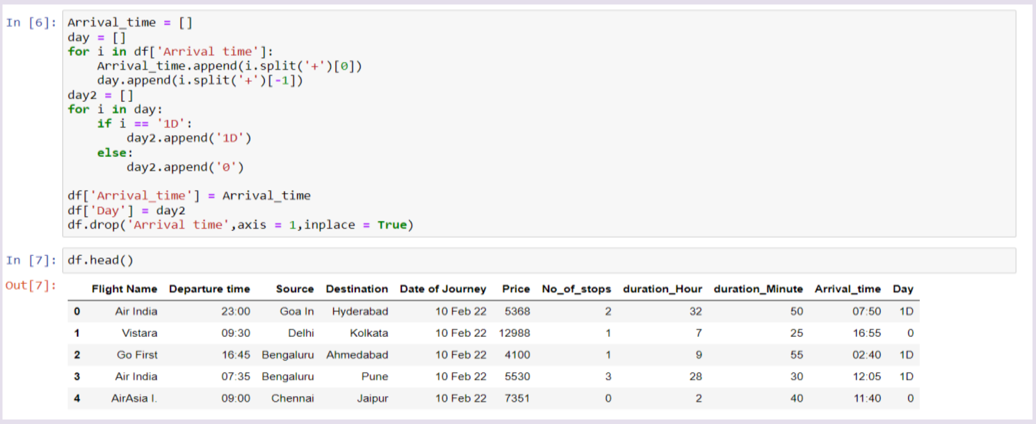


```
In [5]: hour = []
        minute = []
        for i in df['Duration']:
            hour.append(i.split()[0].replace('h',''))
            minute.append(i.split()[-1].replace('m',''))

        df['duration_Hour'] = hour
        df['duration_Minute'] = minute
        df.drop('Duration',axis = 1,inplace = True)
        df.head()
```

Out[5]:

	Flight Name	Departure time	Arrival time	Source	Destination	Date of Journey	Price	No_of_stops	duration_Hour	duration_Minute
0	Air India	23:00	07:50+1D	Goa In	Hyderabad	10 Feb 22	5368	2	32	50
1	Vistara	09:30	16:55	Delhi	Kolkata	10 Feb 22	12988	1	7	25
2	Go First	16:45	02:40+1D	Bengaluru	Ahmedabad	10 Feb 22	4100	1	9	55
3	Air India	07:35	12:05+1D	Bengaluru	Pune	10 Feb 22	5530	3	28	30
4	AirAsia I.	09:00	11:40	Chennai	Jaipur	10 Feb 22	7351	0	2	40

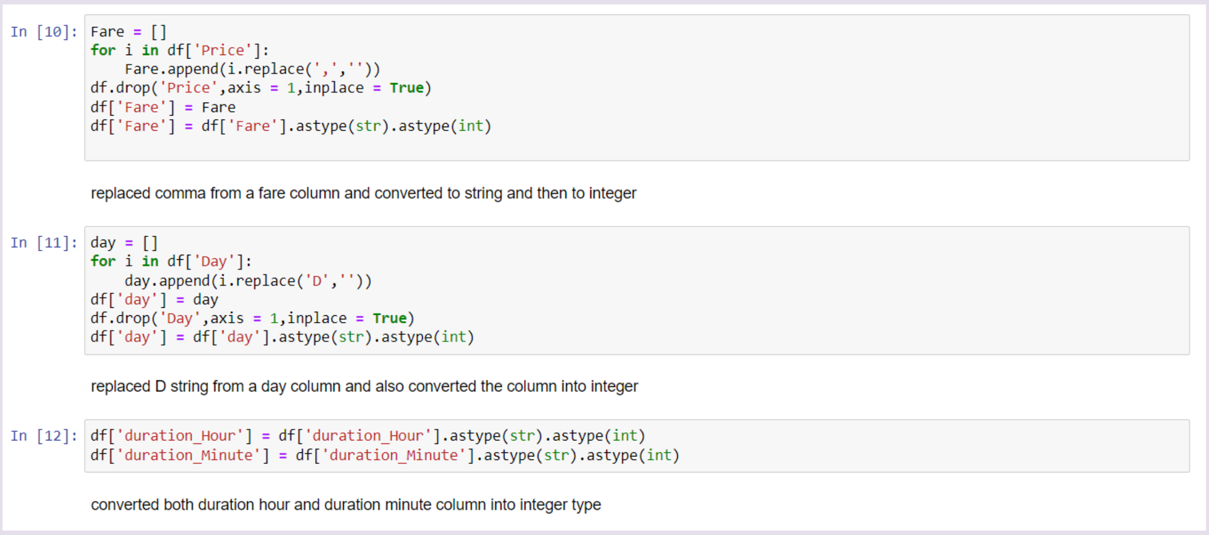
We have separated hour and minute from duration column and created new column as duration_hour and duration_minute and deleted the original column.



```
In [6]: Arrival_time = []
        day = []
        for i in df['Arrival time']:
            Arrival_time.append(i.split('+')[0])
            day.append(i.split('+')[-1])
        day2 = []
        for i in day:
            if i == '1D':
                day2.append('1D')
            else:
                day2.append('0')

        df['Arrival_time'] = Arrival_time
        df['Day'] = day2
        df.drop('Arrival time',axis = 1,inplace = True)

In [7]: df.head()

Out[7]:
```

	Flight Name	Departure time	Source	Destination	Date of Journey	Price	No_of_stops	duration_Hour	duration_Minute	Arrival_time	Day
0	Air India	23:00	Goa In	Hyderabad	10 Feb 22	5368	2	32	50	07:50	1D
1	Vistara	09:30	Delhi	Kolkata	10 Feb 22	12988	1	7	25	16:55	0
2	Go First	16:45	Bengaluru	Ahmedabad	10 Feb 22	4100	1	9	55	02:40	1D
3	Air India	07:35	Bengaluru	Pune	10 Feb 22	5530	3	28	30	12:05	1D
4	AirAsia I.	09:00	Chennai	Jaipur	10 Feb 22	7351	0	2	40	11:40	0

We have separated 1D from Arrival time and created new column as Day.



```
In [10]: Fare = []
         for i in df['Price']:
             Fare.append(i.replace(',',''))
         df.drop('Price',axis = 1,inplace = True)
         df['Fare'] = Fare
         df['Fare'] = df['Fare'].astype(str).astype(int)
```

replaced comma from a fare column and converted to string and then to integer

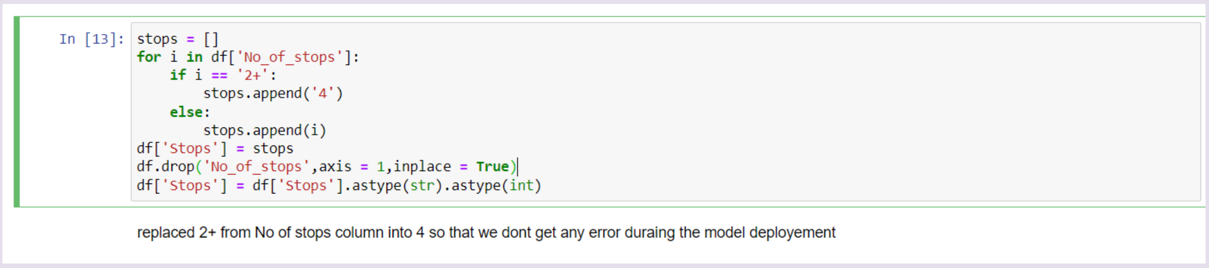
```
In [11]: day = []
         for i in df['Day']:
             day.append(i.replace('D',''))
         df['day'] = day
         df.drop('Day',axis = 1,inplace = True)
         df['day'] = df['day'].astype(str).astype(int)
```

replaced D string from a day column and also converted the column into integer

```
In [12]: df['duration_Hour'] = df['duration_Hour'].astype(str).astype(int)
         df['duration_Minute'] = df['duration_Minute'].astype(str).astype(int)
```

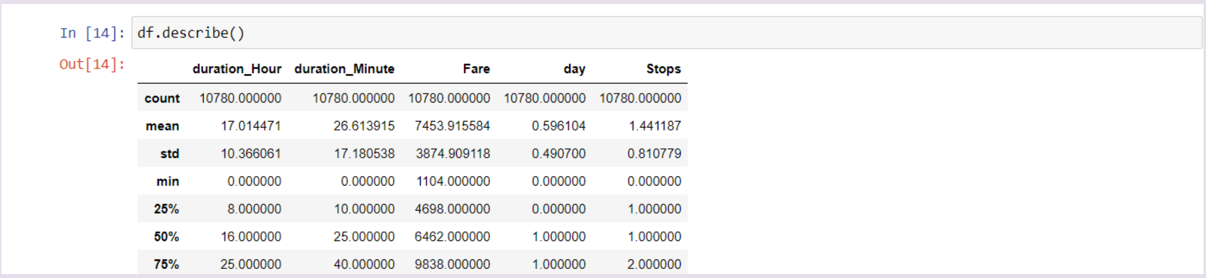
converted both duration hour and duration minute column into integer type

We have replaced comma from a price column and converted object type data into integer type and replaced 'D' from a space and converted Day column into integer data type also converted duration hour and duration minute into integer type.



```
In [13]: stops = []
         for i in df['No_of_stops']:
             if i == '2+':
                 stops.append('4')
             else:
                 stops.append(i)
         df['Stops'] = stops
         df.drop('No_of_stops',axis = 1,inplace = True)
         df['Stops'] = df['Stops'].astype(str).astype(int)
```

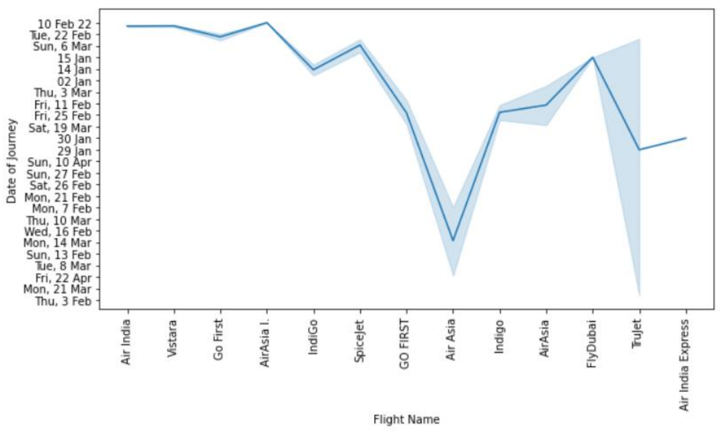
replaced 2+ from No of stops column into 4 so that we dont get any error duraing the model deployement



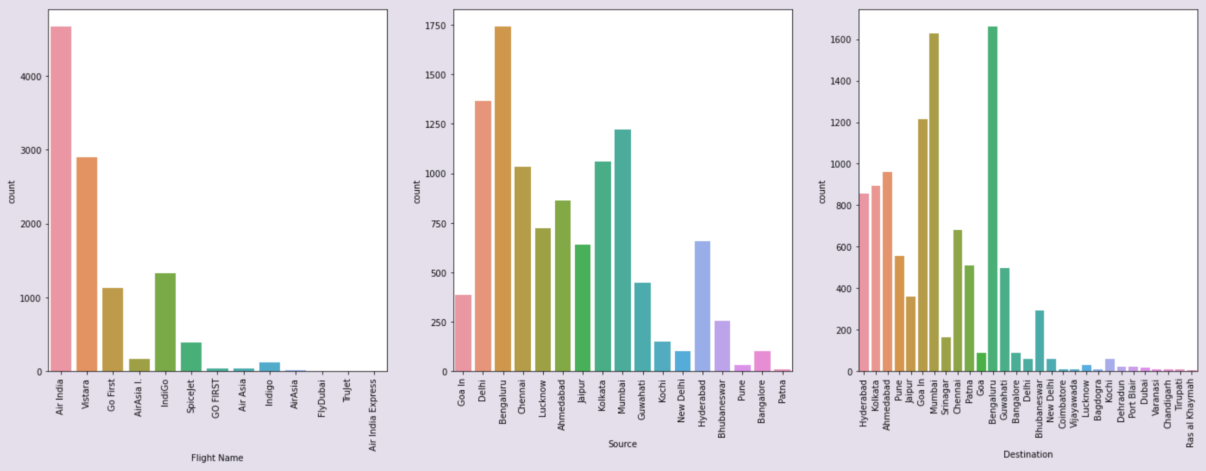
```
In [14]: df.describe()

Out[14]:
       duration_Hour  duration_Minute          Fare           day         Stops
count   10780.000000     10780.000000  10780.000000  10780.000000  10780.000000
mean       17.014471        26.613915   7453.915584      0.596104      1.441187
std        10.366061        17.180538   3874.909118      0.490700      0.810779
min         0.000000         0.000000   1104.000000      0.000000      0.000000
25%         8.000000        10.000000   4698.000000      0.000000      1.000000
50%        16.000000        25.000000   6462.000000      1.000000      1.000000
75%        25.000000        40.000000   9838.000000      1.000000      2.000000
```

We used df.describe() to check all the statistical measures across the data.

- **<u>Visualization:-</u>**



The above graph shows that the which flight is availabel on which date.



<u>From the above graph we could conclude that:</u>

Airlines:-

* Air India is the most preferred airline with the highest row count, followed by Vistara and indigo.

* Count for AirAsia, Gofirst, spicejet and flydubai is quite low.

source:-

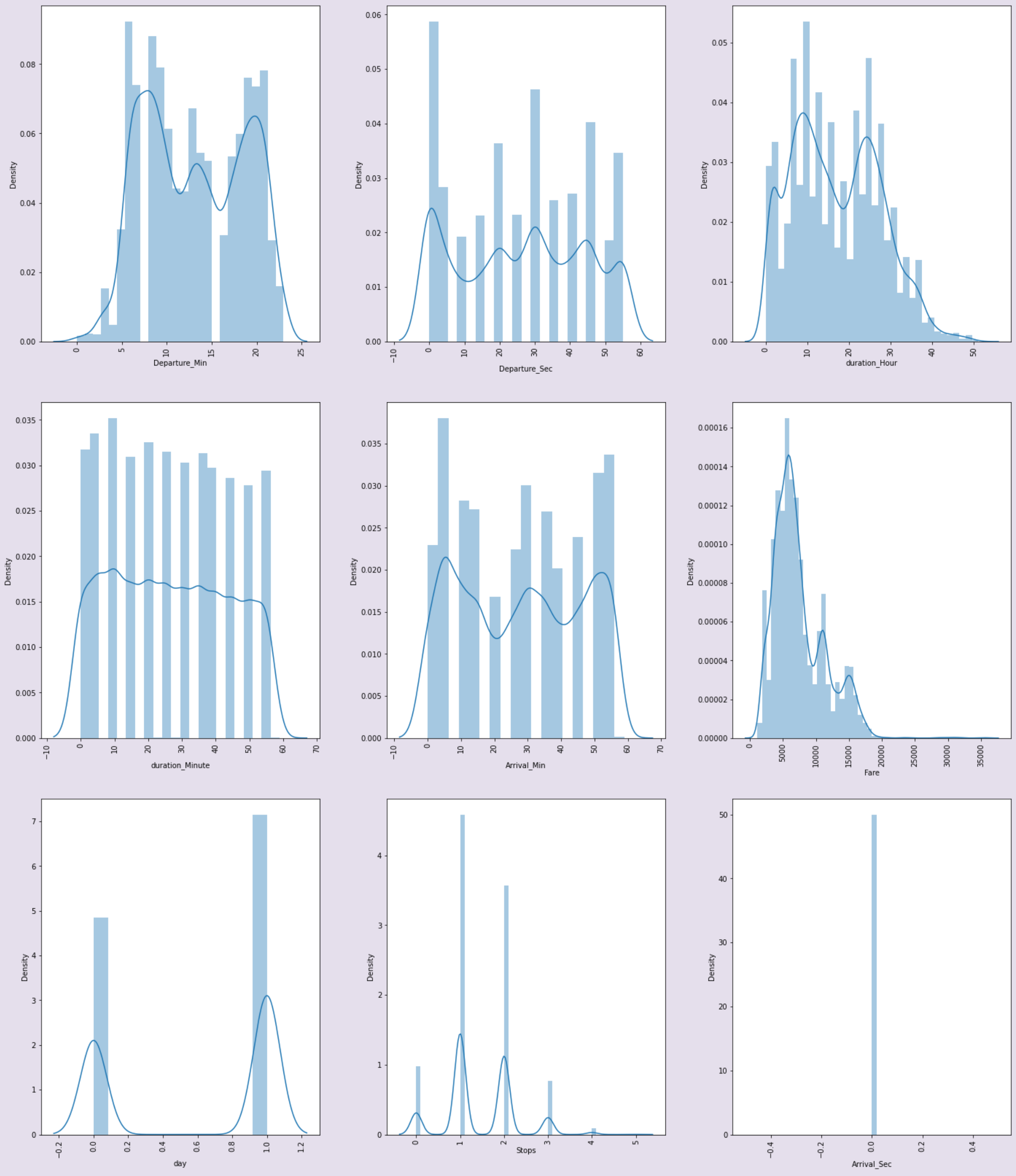
* Most of the flights take off from Bengaluru.

* Patna has the minimum number of flights

Destination:

* most of the flights destination is Bengaluru.

* Ras al Khayumah has least number of flight destination.



We make the below observations from the numerical data –

Total stops:-
* Majority of the flights have stops as 1, flights with 3 and 4 stops are quite low

Departure_hour:-

* Majority of the flights tend to fly in the early morning time
* Count of flights taking off during 5:00 - 10:00 is also high, Afternoon flights are less in number.

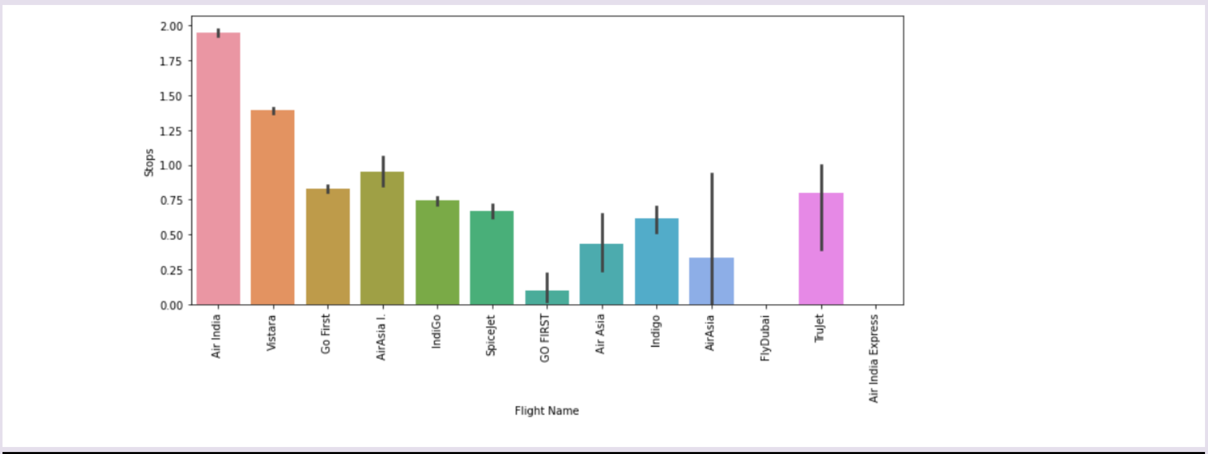
Departure_minute:-
* Most flights take off at whole hours (Mins as 00)

Arrival time hour:-
* Majority of the flights reach its destination in the evening time around 18:00-19:00
* This seems to be because majority of the flights have take-off times in the morning and hence land after in the evening

Arrival time minute:-
* This distribution is similar and does not give out any dedicated information

Travel hours:-
* Majority of the flights have travel time for around 10-15 hours, these are international flights
* Some flights have time around 30 hours too, this could be because of the number of stops in between

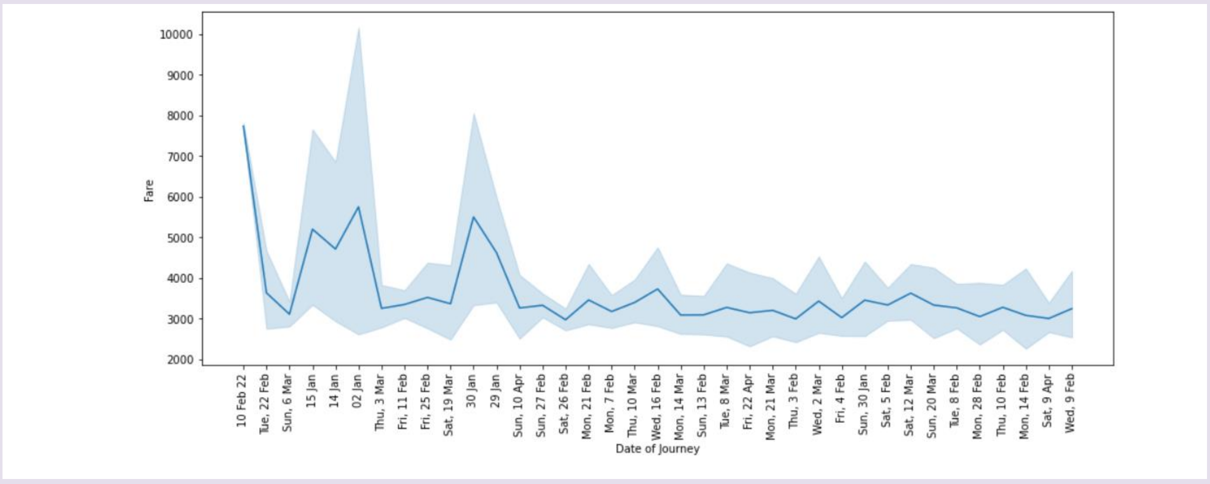
Travel mins:-
* The data is divided and is not pointing towards any specific points



from the above graph we can see that:-

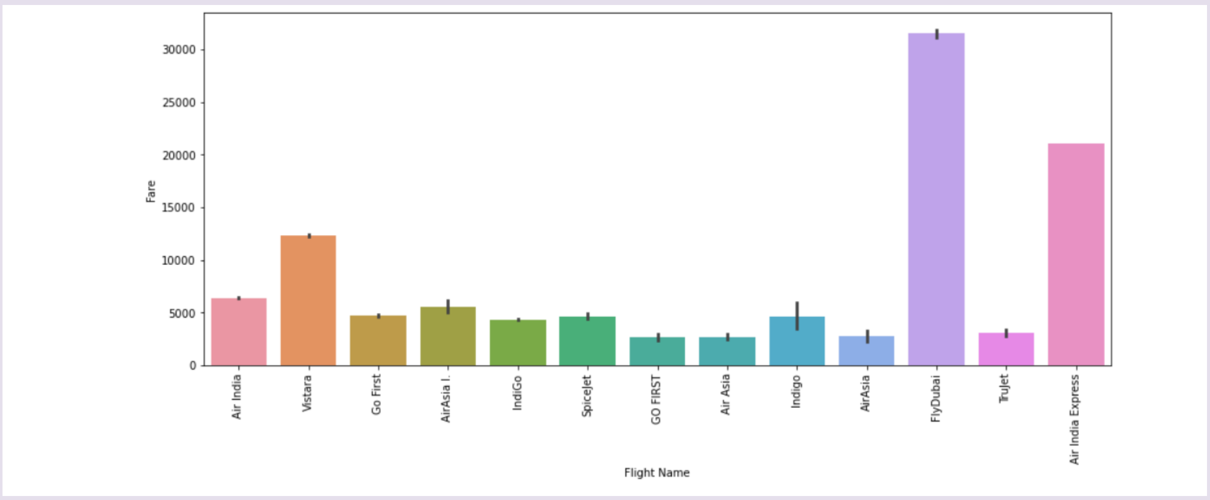
* Air india has the maximum number of stops between the source and destination.

* Go first has the least number of stops between the source and destination.



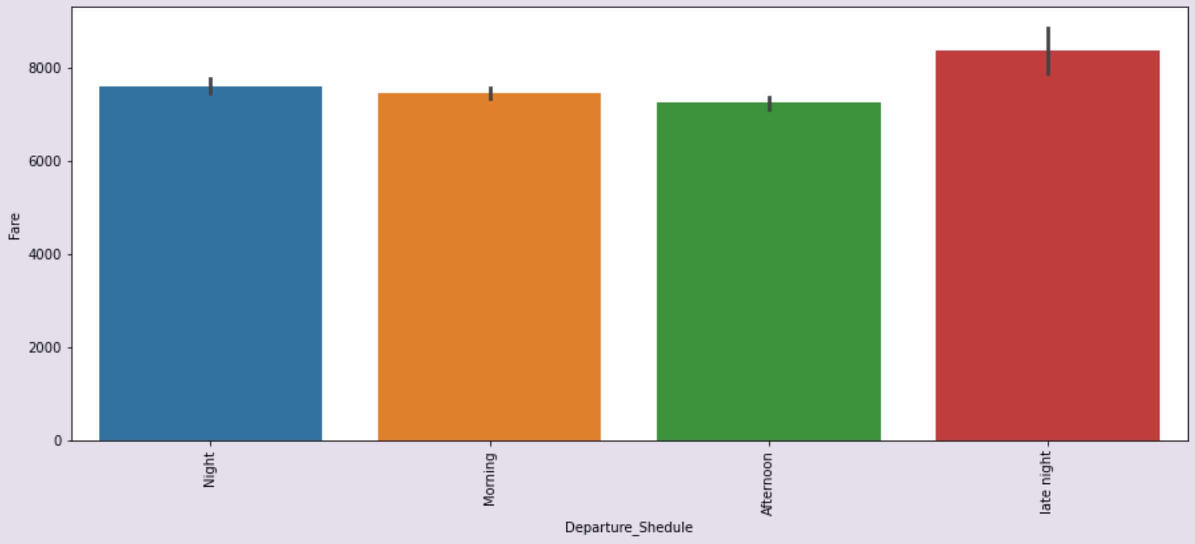
from the above graph we can observe that:-

* the price of flight decreases according to the date of journey. as the date crosses by the fare decreases.



from the above graph we could observe that:

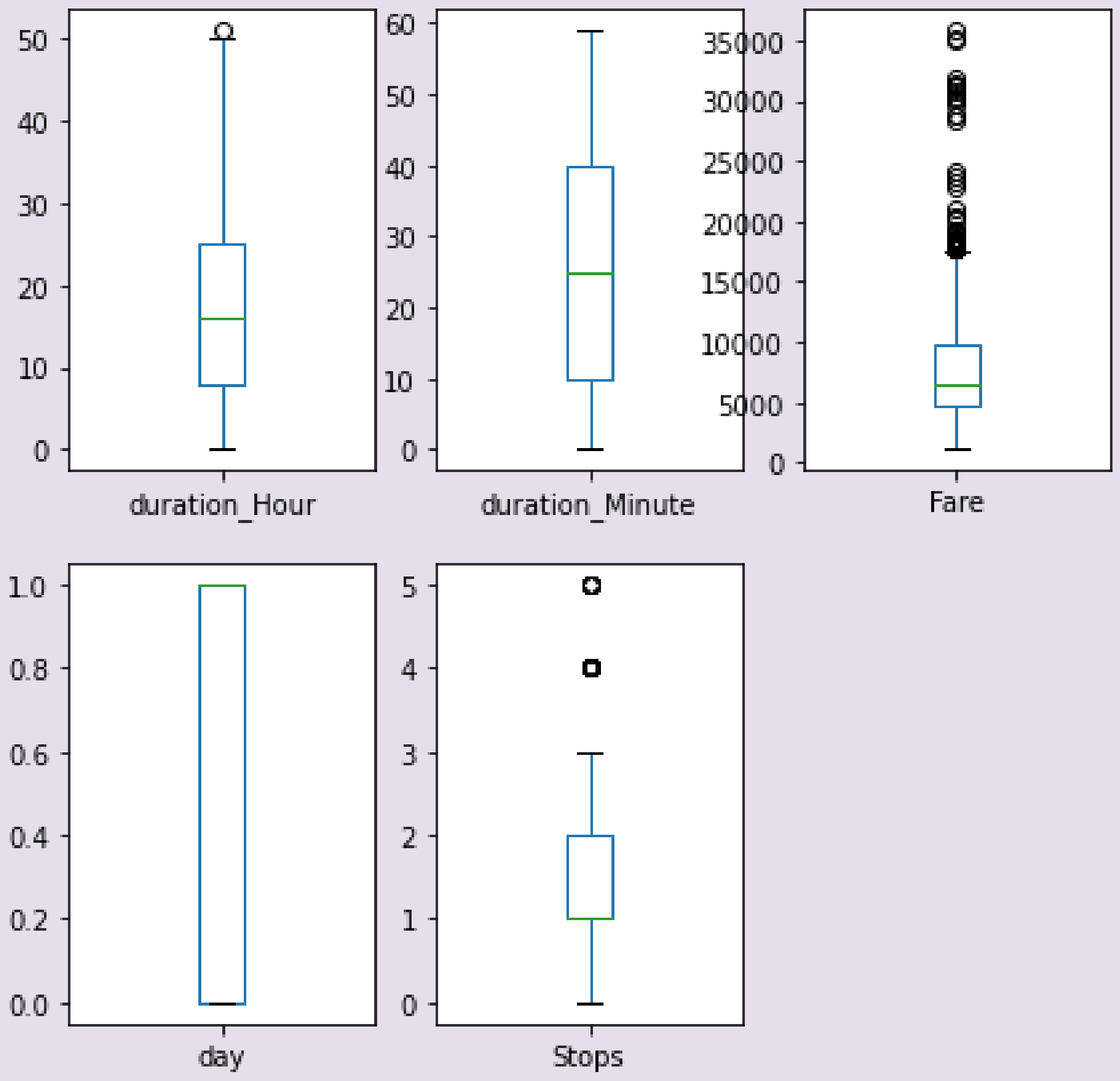
* Flydubai has the maximum number of Fare.
* Go first, AirAsia,Trujet has the minimum Fares
* indigo is expensive then jet airways.



From the above graph we could observe that:-

* night and late night flights are expensive as compared to morning afternoon flights

- **Outlier detection:-**



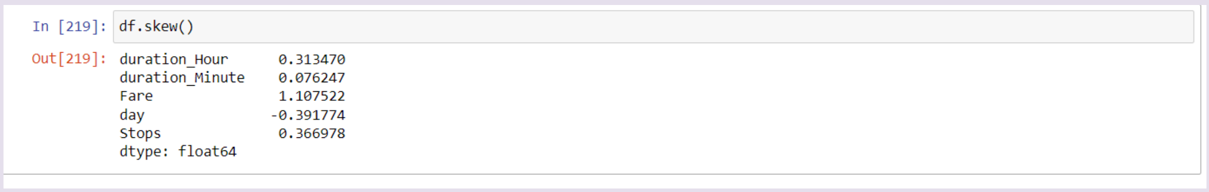
We make the below conclusions –

* Outliers are present in stops and price

*We will not remove outliers from total stops since price is impacted by number of stops

- **Skewness Treatment**

We now proceed with treating skewness in our data, which allows us to fit our data in a symmetric distribution, which further allows our model to learn better.

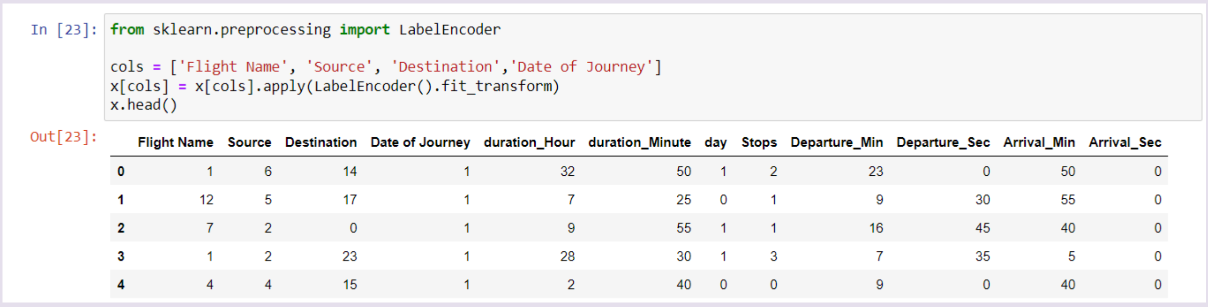


```
In [219]: df.skew()

Out[219]: duration_Hour      0.313470
          duration_Minute    0.076247
          Fare               1.107522
          day               -0.391774
          Stops              0.366978
          dtype: float64
```

We will not transform 'Price' column, since it is our target variable.

- **Encoding the categorical data:-**

  We encode the categorical data in this step, to convert it to integer type, since the model does not work on 'string' data. We use 'Label Encoder' to achieve the desired results –

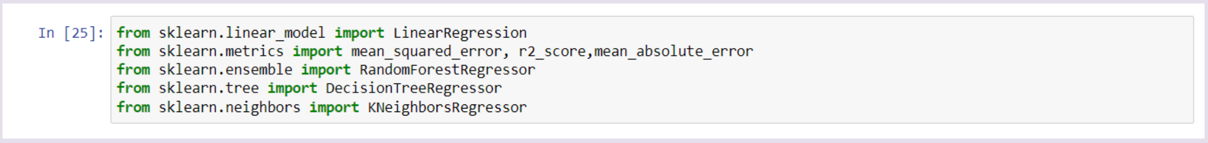


```
In [23]: from sklearn.preprocessing import LabelEncoder

         cols = ['Flight Name', 'Source', 'Destination','Date of Journey']
         x[cols] = x[cols].apply(LabelEncoder().fit_transform)
         x.head()

Out[23]:
```

	Flight Name	Source	Destination	Date of Journey	duration_Hour	duration_Minute	day	Stops	Departure_Min	Departure_Sec	Arrival_Min	Arrival_Sec
0	1	6	14	1	32	50	1	2	23	0	50	0
1	12	5	17	1	7	25	0	1	9	30	55	0
2	7	2	0	1	9	55	1	1	16	45	40	0
3	1	2	23	1	28	30	1	3	7	35	5	0
4	4	4	15	1	2	40	0	0	9	0	40	0

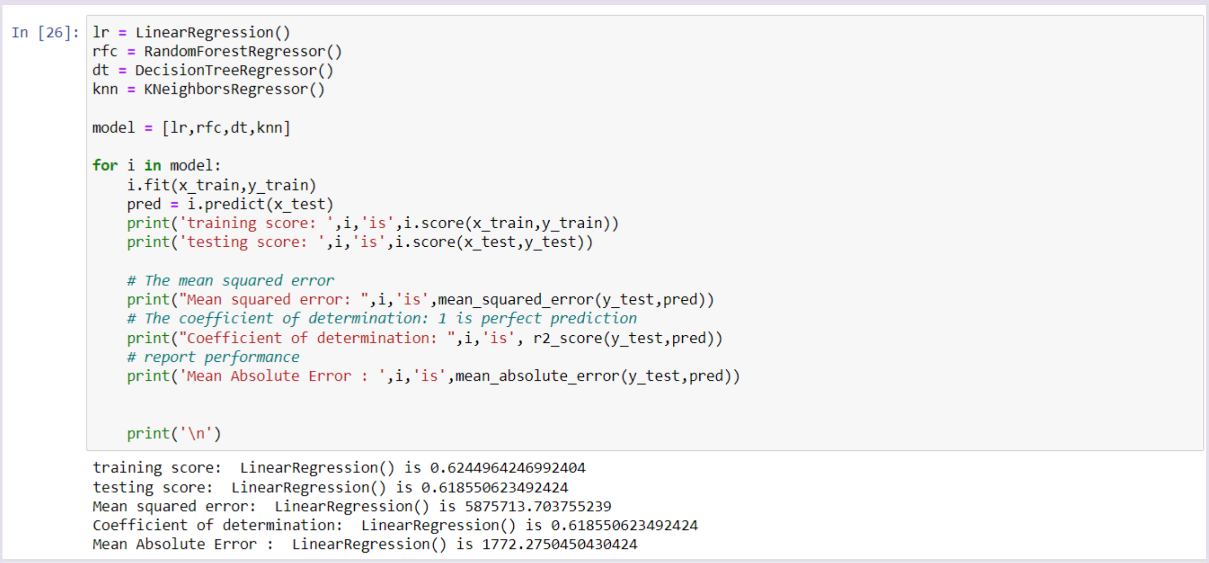
- **Fitting the Regression models:-**

  We now proceed to the main step of our machine learning, fitting the model and predicting the outputs. We fit the data into multiple regression models to compare the performance of all models and select the best model –



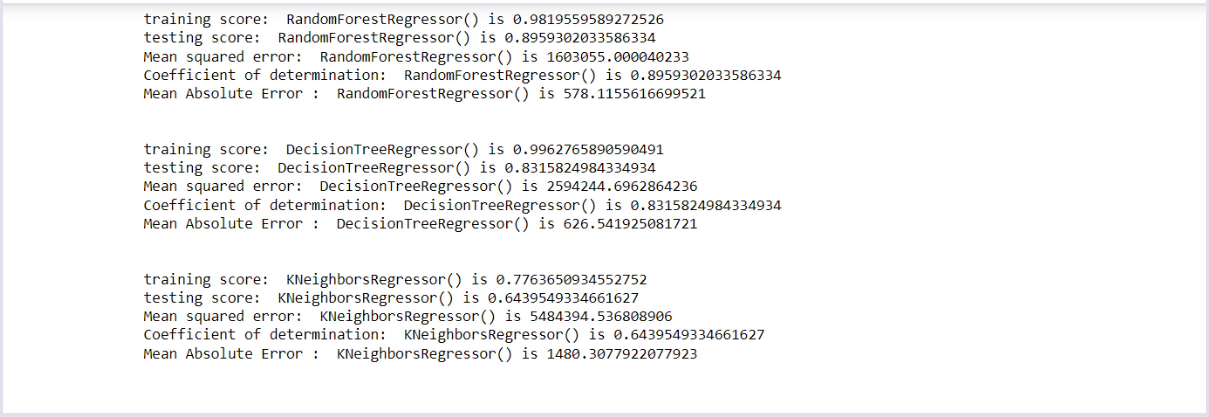
```
In [25]: from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, r2_score,mean_absolute_error
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.neighbors import KNeighborsRegressor
```

We use the below mentioned code snipped to fit the data into ML models and predict the output –



```
In [26]: lr = LinearRegression()
         rfc = RandomForestRegressor()
         dt = DecisionTreeRegressor()
         knn = KNeighborsRegressor()

         model = [lr,rfc,dt,knn]

         for i in model:
             i.fit(x_train,y_train)
             pred = i.predict(x_test)
             print('training score: ',i,'is',i.score(x_train,y_train))
             print('testing score: ',i,'is',i.score(x_test,y_test))

             # The mean squared error
             print("Mean squared error: ",i,'is',mean_squared_error(y_test,pred))
             # The coefficient of determination: 1 is perfect prediction
             print("Coefficient of determination: ",i,'is', r2_score(y_test,pred))
             # report performance
             print('Mean Absolute Error : ',i,'is',mean_absolute_error(y_test,pred))


             print('\n')
```

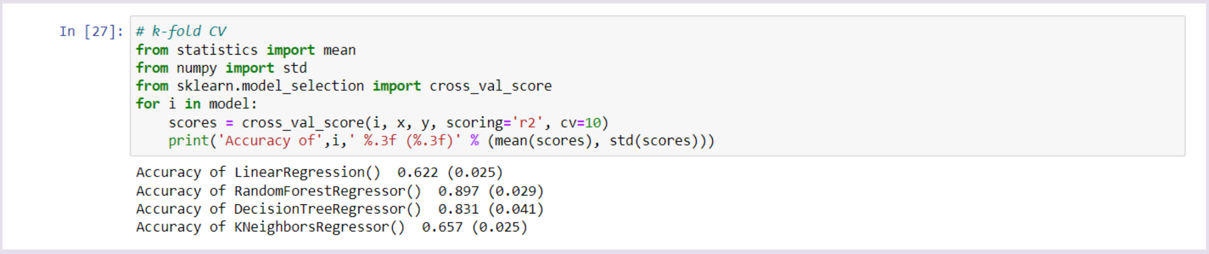
```
training score:  LinearRegression() is 0.6244964246992404
testing score:  LinearRegression() is 0.618550623492424
Mean squared error:  LinearRegression() is 5875713.703755239
Coefficient of determination:  LinearRegression() is 0.618550623492424
Mean Absolute Error :  LinearRegression() is 1772.2750450430424
```



```
training score:  RandomForestRegressor() is 0.9819559589272526
testing score:  RandomForestRegressor() is 0.8959302033586334
Mean squared error:  RandomForestRegressor() is 1603055.000040233
Coefficient of determination:  RandomForestRegressor() is 0.8959302033586334
Mean Absolute Error :  RandomForestRegressor() is 578.1155616699521


training score:  DecisionTreeRegressor() is 0.9962765890590491
testing score:  DecisionTreeRegressor() is 0.8315824984334934
Mean squared error:  DecisionTreeRegressor() is 2594244.6962864236
Coefficient of determination:  DecisionTreeRegressor() is 0.8315824984334934
Mean Absolute Error :  DecisionTreeRegressor() is 626.541925081721


training score:  KNeighborsRegressor() is 0.7763650934552752
testing score:  KNeighborsRegressor() is 0.6439549334661627
Mean squared error:  KNeighborsRegressor() is 5484394.536808906
Coefficient of determination:  KNeighborsRegressor() is 0.6439549334661627
Mean Absolute Error :  KNeighborsRegressor() is 1480.3077922077923
```

We achieve the best score using Random Forest regressor, with an r2_score of 89%. We also obtain the minimum values for mean_absolute_error, mean_squared_error and root_mean_squared_error (regression metrics) with this model.

- **<u>Cross Validation:-</u>**
  We perform the cross validation of our model to check if the model has any over fitting issue, by checking the ability of the model to make predictions on new data, using k-folds. We test the cross validation for every model.



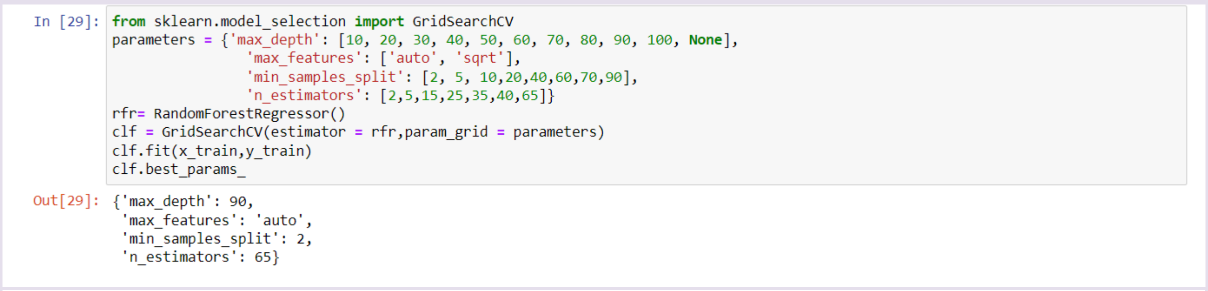
```
In [27]: # k-fold CV
         from statistics import mean
         from numpy import std
         from sklearn.model_selection import cross_val_score
         for i in model:
             scores = cross_val_score(i, x, y, scoring='r2', cv=10)
             print('Accuracy of',i,' %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy of LinearRegression()  0.622 (0.025)
Accuracy of RandomForestRegressor()  0.897 (0.029)
Accuracy of DecisionTreeRegressor()  0.831 (0.041)
Accuracy of KNeighborsRegressor()  0.657 (0.025)
```

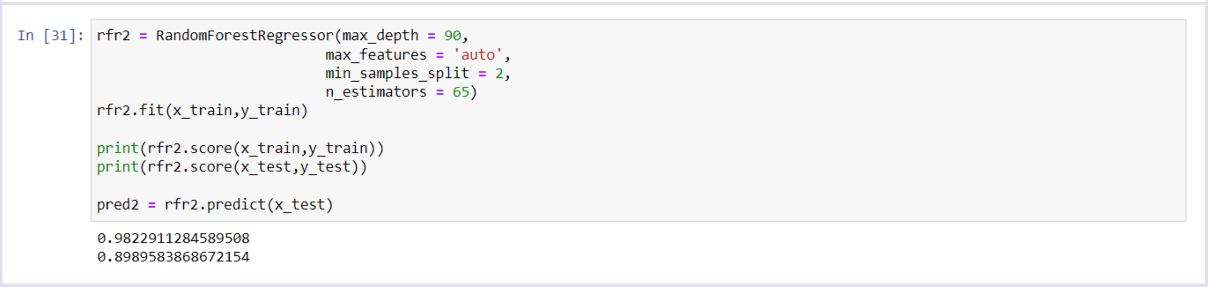
The Random Forest Regressor provides us a cross validation score of 89.7% .We will hyper tune the models to check if our accuracy improves.

- **<u>Hyper-parameter tuning the model:-</u>**

  Grid Search CV is a technique used to validate the model with different parameter combinations, by creating a grid of parameters and trying all the combinations to compare which combination gave the best results. We apply grid search on our model –



```
In [29]: from sklearn.model_selection import GridSearchCV
         parameters = {'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
                       'max_features': ['auto', 'sqrt'],
                       'min_samples_split': [2, 5, 10,20,40,60,70,90],
                       'n_estimators': [2,5,15,25,35,40,65]}
         rfr= RandomForestRegressor()
         clf = GridSearchCV(estimator = rfr,param_grid = parameters)
         clf.fit(x_train,y_train)
         clf.best_params_

Out[29]: {'max_depth': 90,
          'max_features': 'auto',
          'min_samples_split': 2,
          'n_estimators': 65}
```

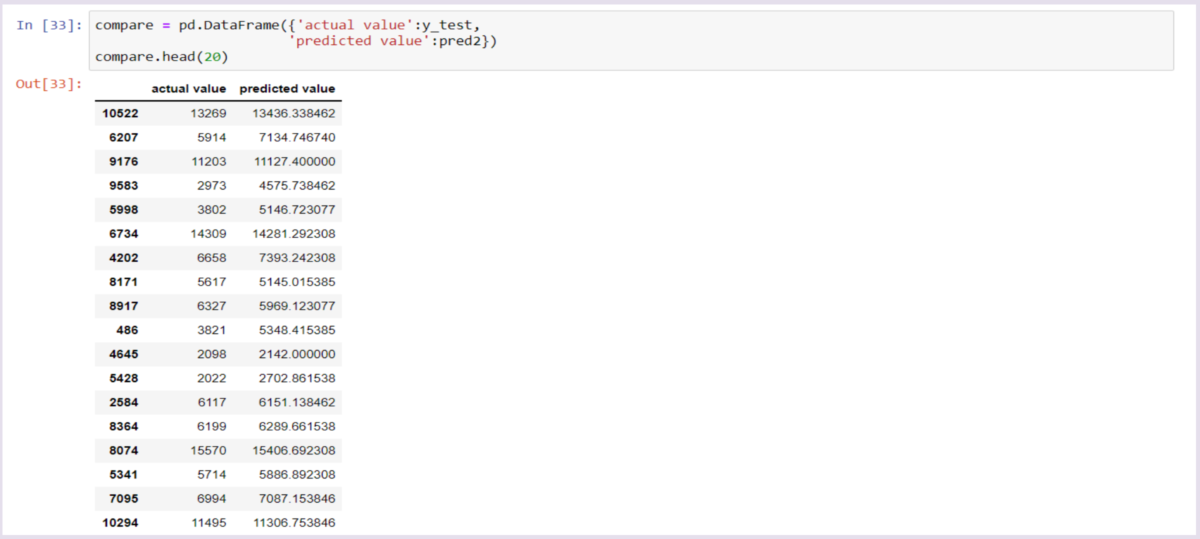


```
In [31]: rfr2 = RandomForestRegressor(max_depth = 90,
                                      max_features = 'auto',
                                      min_samples_split = 2,
                                      n_estimators = 65)
         rfr2.fit(x_train,y_train)

         print(rfr2.score(x_train,y_train))
         print(rfr2.score(x_test,y_test))

         pred2 = rfr2.predict(x_test)

         0.9822911284589508
         0.8989583868672154
```

  We have successfully boosted the accuracy of random forest regressor with 89.89% with grid search cv.
  Hence we select randomforest Regressor as our final model, save the model using best parameters, and create model object using pickle.

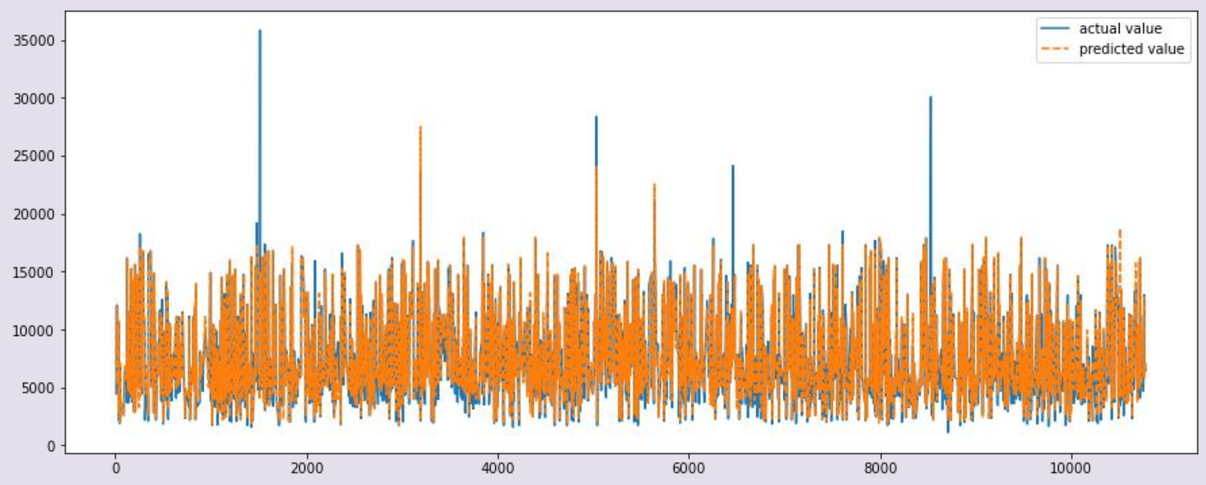
- **<u>Conclusion:-</u>**

  Further we have created a dataframe containing the Actual data and the predicted data. So that we can visualize how close the machine has predicted on testing data.



```
In [33]: compare = pd.DataFrame({'actual value':y_test,
                                 'predicted value':pred2})
         compare.head(20)

Out[33]:
```

	actual value	predicted value
10522	13269	13436.338462
6207	5914	7134.746740
9176	11203	11127.400000
9583	2973	4575.738462
5998	3802	5146.723077
6734	14309	14281.292308
4202	6658	7393.242308
8171	5617	5145.015385
8917	6327	5969.123077
486	3821	5348.415385
4645	2098	2142.000000
5428	2022	2702.861538
2584	6117	6151.138462
8364	6199	6289.661538
8074	15570	15406.692308
5341	5714	5886.892308
7095	6994	7087.153846
10294	11495	11306.753846



As we can observe in the above figure, model predictions and original prices are overlapping. This visual result confirms the high model score which we saw earlier.

- **Future Work:-**
  1. More routes can be added and the same analysis can be expanded to major airports and travel routes in India.
  2. The analysis can be done by increasing the data points and increasing the historical data used. That will train the model better accuracy and more savings.
  3. More rules can be added in the rule based learning based on our understand of the industry, also incorporating the offer periods given by the airlines.
  4. Developing a more user friendly interface for various routes giving more flexibility to the users.