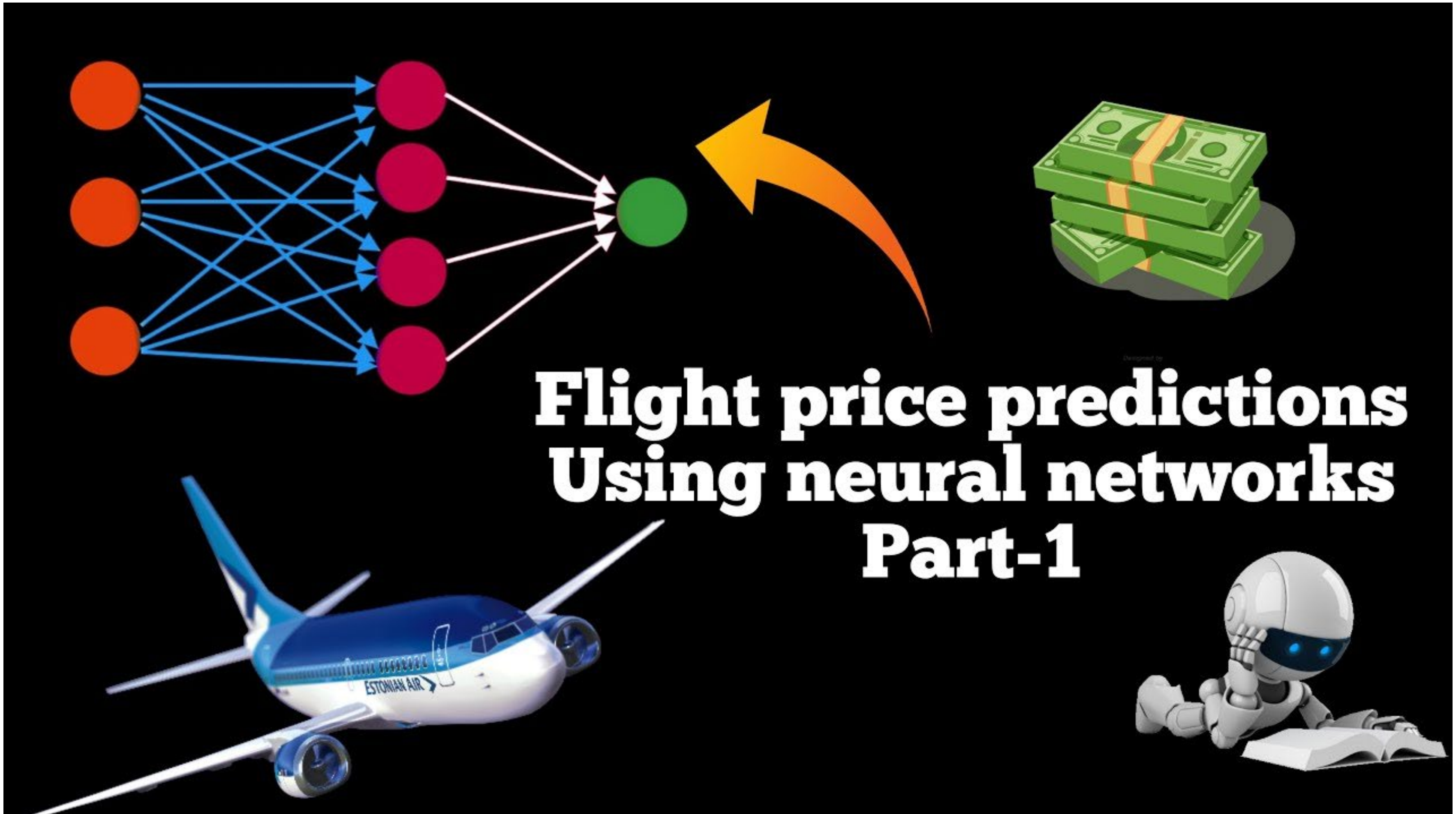
Flight Price Prediction

SUBMITTED BY:-

Naincy Joshi

The Project is based on the “Flight Price Prediction”.We have to predict the prices of the test data.we have two sets of data i.e train_data and the test_data.Based upon the study of train_data we will predict the prices of the flight of test_data.The price of the flight gets fluctuate depend upon the various variable such as duration,stoppage,class(economy class or the bussiness class).



Dataset Details

Train_data
- Airline
- Date_of_Journey
- Source
- Destination
- Route
- Dep_Time
- Arrival_Time
- Duration
- Total_Stops
- Additional_Info
- Prices

Test_data
- Airline
- Date_of_Journey
- Source
- Destination
- Route
- Dep_Time
- Arrival_Time
- Duration
- Total_Stops
- Additional_Info

Data Analysis

- Train_data.info() will give you the datatype in the train_dataset
- 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

- test_data.info() will give the datatype of the test data set.
- 0 Airline 2671 non-null object
- 1 Date_of_Journey 2671 non-null object
- 2 Source 2671 non-null object 3
- Destination 2671 non-null object
- 4 Route 2671 non-null object
- 5 Dep_Time 2671 non-null object
- 6 Arrival_Time 2671 non-null object
- 7 Duration 2671 non-null object
- 8 Total_Stops 2671 non-null object
- 9 Additional_Info 2671 non-null object

- In train_data ,all are “object” datatype except the price.
- In test_data,all are “object” datatype.we have to predict the price of the test_data.

Train_data and the test_data.isna().sum() will give us the missing value in the dataset.

- 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

- Airline 0
- Date_of_Journey 0
- Source 0
- Destination 0
- Route 0
- Dep_Time 0
- Arrival_Time 0
- Duration 0
- Total_Stops 0
- Additional_Info 0

In train_data ,we have to convert the “Object” datatype into the int or floats or the other category which computer understand.

In test_data ,every variable is an “object” type we need to convert the datatype into the machine language.

In the train_data we got 1 missing value in the “Route” and the “Total_stops”.In that column there is a ‘NAN’ value so we will drop that column.

In [49]:
```
train_data[train_data['Total_Stops'].isnull()]
```

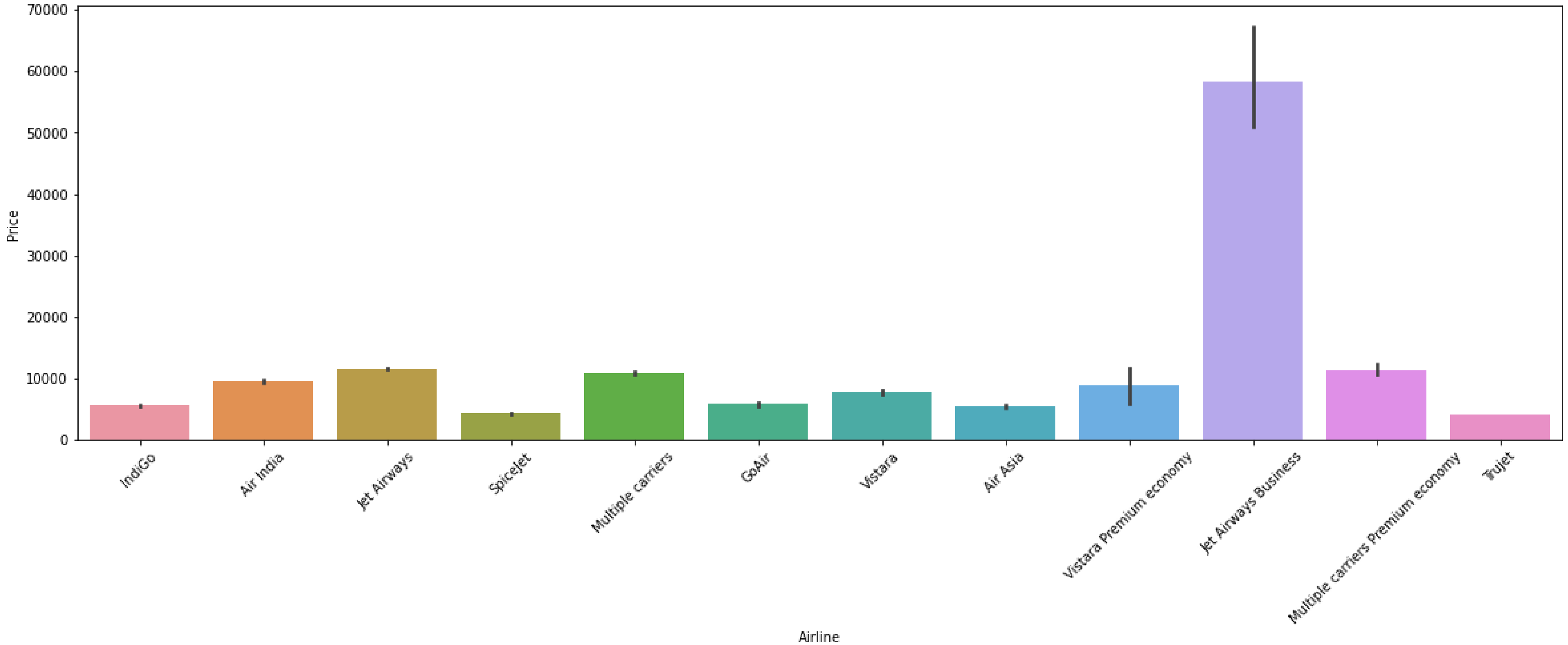
Out[49]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
9039	1	6/05/2019	2	1	NaN	09:45	09:25 07 May	23h 40m	NaN	No info	7480

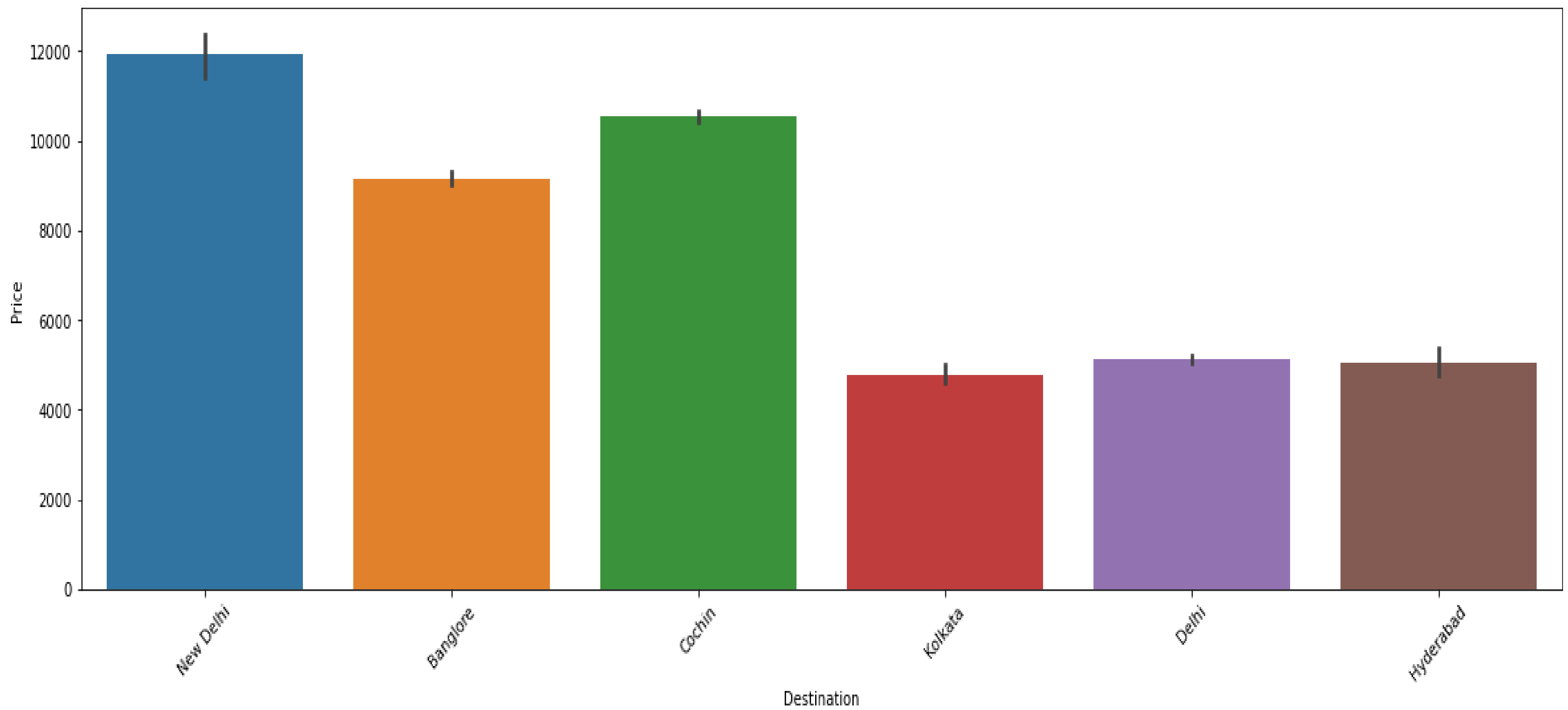
We Will Drop one "NAN" of the "Total_Stops"

In [50]:
```
train_data=train_data[train_data['Total_Stops'].notnull()]
```

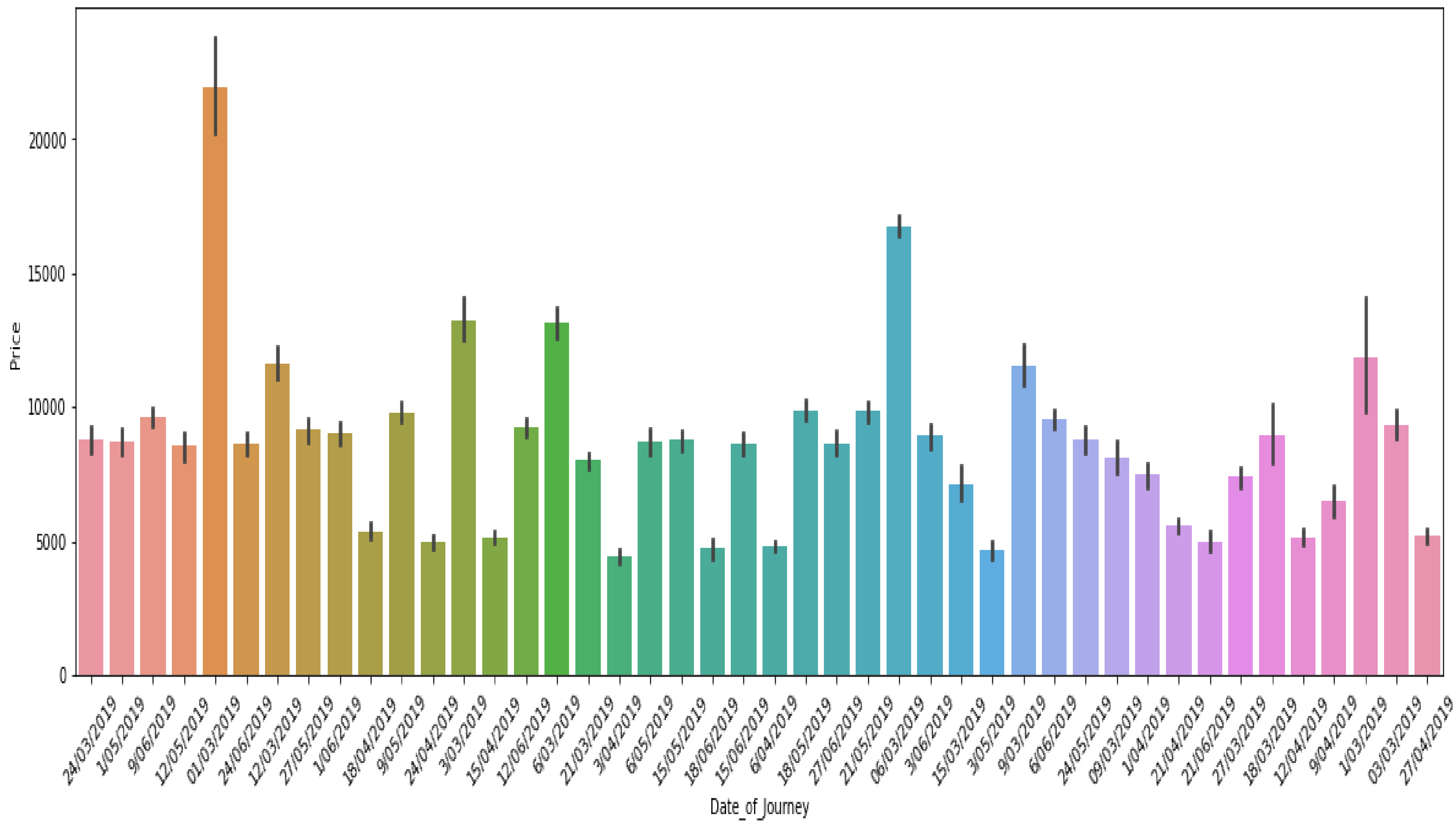
We will plot feature variable such as “Airlines” with the target variable “prices”



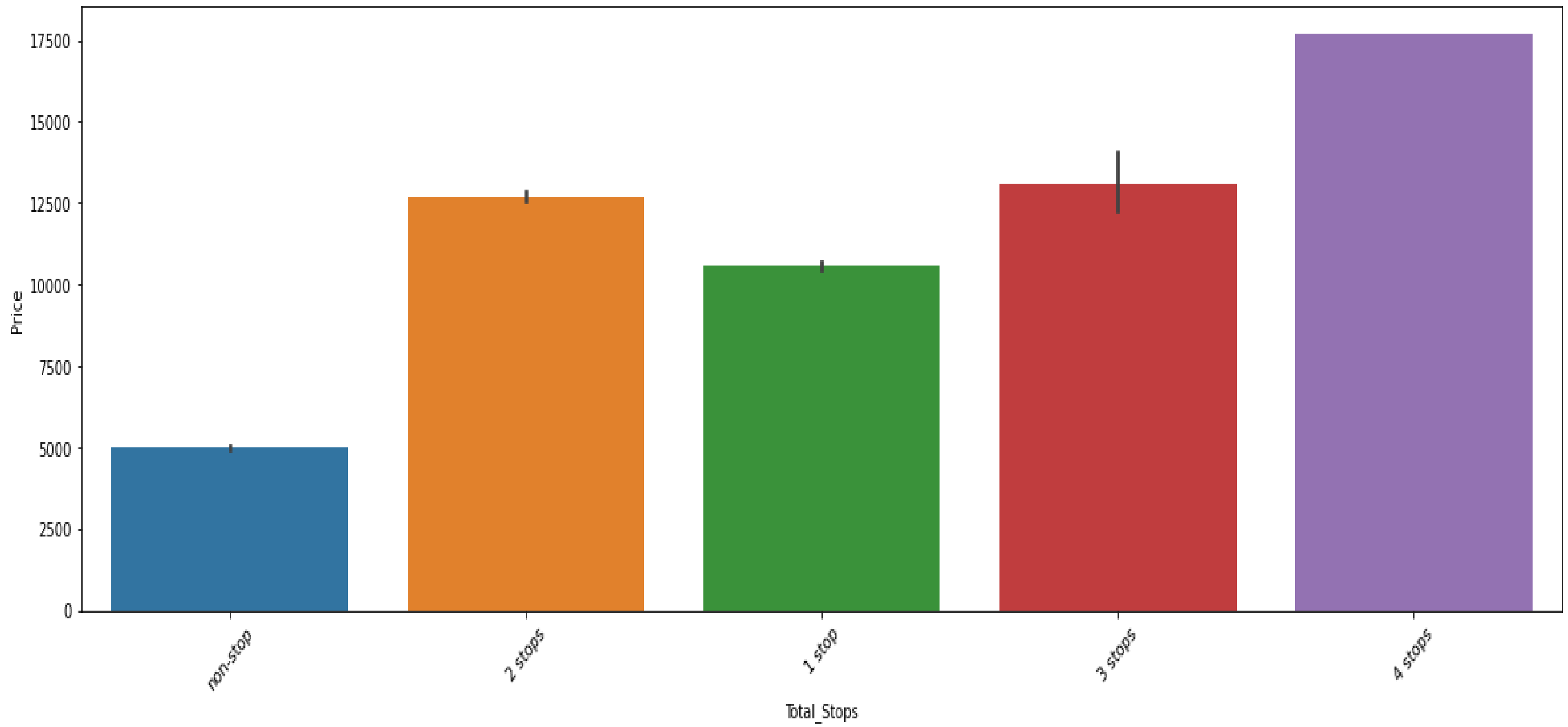
Plotting the barplot of “destination”with the target variable “prices”



Barplot of “date_of_journey” with the target variable “prices”



Plotting the barplot of “Total_Stops” with the target variable “prices”



EDA Conclusion Remark

- Jet airways business class has the highest ticket prices of the Airlines and the least prices of the airlines are Trujet and the spicejet.
- The delhi,cochin,bangalore flight is more expensive than the other flights.
- People traveled more on 1/03/19 ,so the sales prices are also more.
- Prices of the flight is directly proportional to the number of stoppage i.e if the number of stoppage is more ,the ticket fare is also more.

Pre-Processing Pipeline

- Now we will encode the “object” datatype
- We have two dataset,so first we have to see whether they are set in terms of True
- set(train_data['Airline'])**==**set(test_data['Airline'])
- Out[17]:False
- train_data=train_data[train_data['Airline']!='Trujet']
- set(train_data['Airline'])==set(test_data['Airline'])
- Out [18]:True

Now we will encode the “Airlines” variable using LabelEncoder()

- le=preprocessing.LabelEncoder()
- train_data['Airline']=le.fit_transform(train_data['Airline'])
- test_data['Airline']=le.fit_transform(test_data['Airline'])
- In the same way we will perform in each column feature and convert the object datatype using LabelEncoder()

The data in the "Additional_info" in both the datset will make use of the common feature.

Now WE will see the "Additional_Info" Feature of both the dataset

```
In [33]: train_data['Additional_Info'].value_counts()

Out[33]: No info                         8343
         In-flight meal not included     1982
         No check-in baggage included     320
         1 Long layover                    19
         Change airports                    7
         Business class                     4
         No Info                            3
         Red-eye flight                     1
         2 Long layover                     1
         1 Short layover                    1
         Name: Additional_Info, dtype: int64
```

```
In [34]: test_data['Additional_Info'].value_counts()

Out[34]: No info                         2148
         In-flight meal not included      444
         No check-in baggage included      76
         Business class                     1
         Change airports                    1
         1 Long layover                     1
         Name: Additional_Info, dtype: int64
```

```
In [35]: set(train_data['Additional_Info'])==set(test_data['Additional_Info'])

Out[35]: False
```

```
In [35]: set(train_data['Additional_Info'])==set(test_data['Additional_Info'])

Out[35]: False

In [36]: train_data['Additional_Info']=train_data['Additional_Info'].replace('No Info','No info')

In [37]: train_data['Additional_Info']=train_data['Additional_Info'].replace(['2 Long layover',
                                                                              '1 Short layover',
                                                                              'Red-eye flight'],'Rare')

In [38]: train_data=train_data[train_data['Additional_Info'] !='Rare']

In [39]: set(train_data['Additional_Info'])==set(test_data['Additional_Info'])

Out[39]: True

In [40]: train_data['Additional_Info']=le.fit_transform(train_data['Additional_Info'])
         test_data['Additional_Info']=le.fit_transform(test_data['Additional_Info'])
```

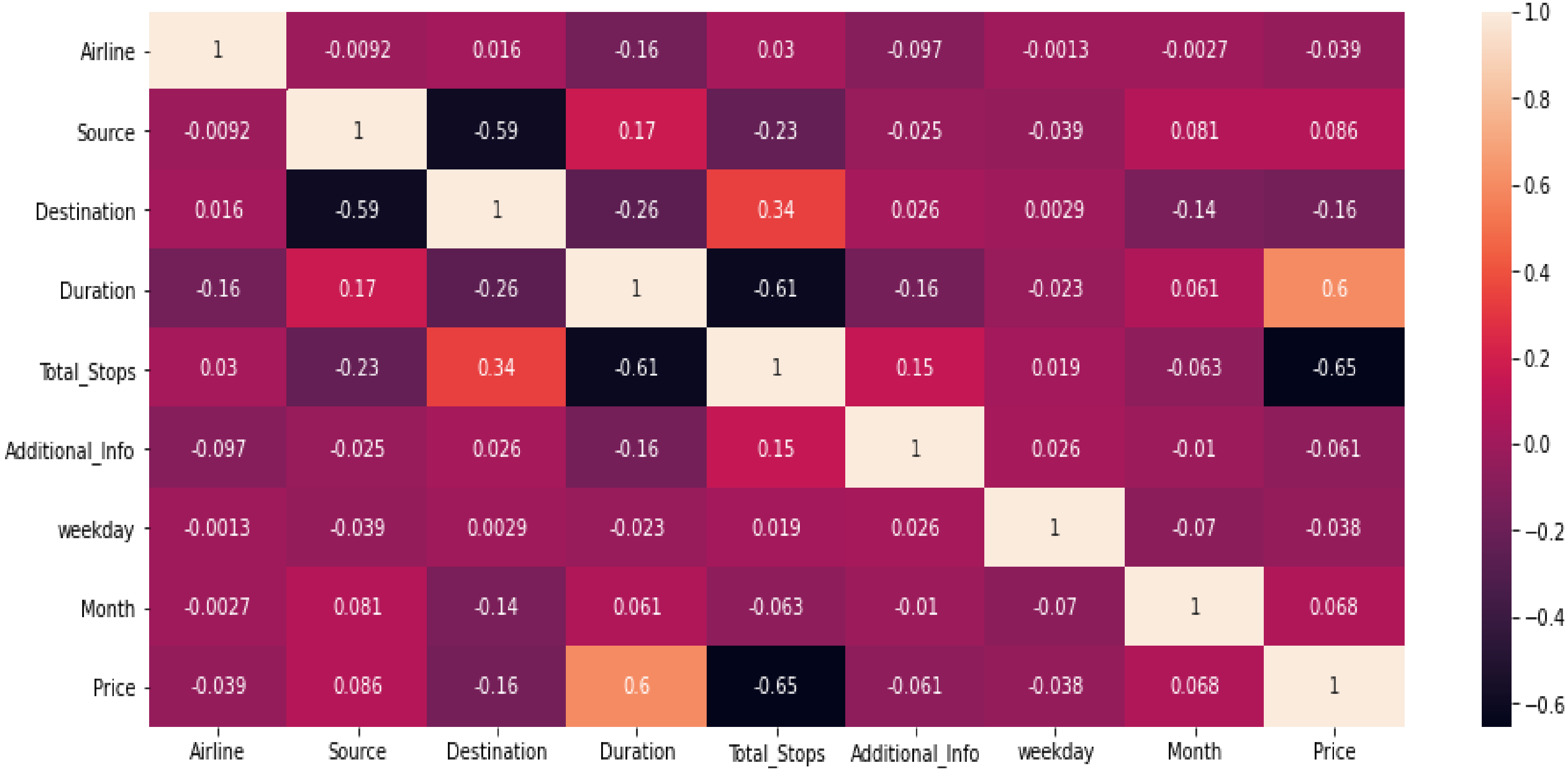
We have used .replace method to make the dataset equal so that we can convert both the datatype into int or float which a machine language can understand and perform the task.

In [44]:

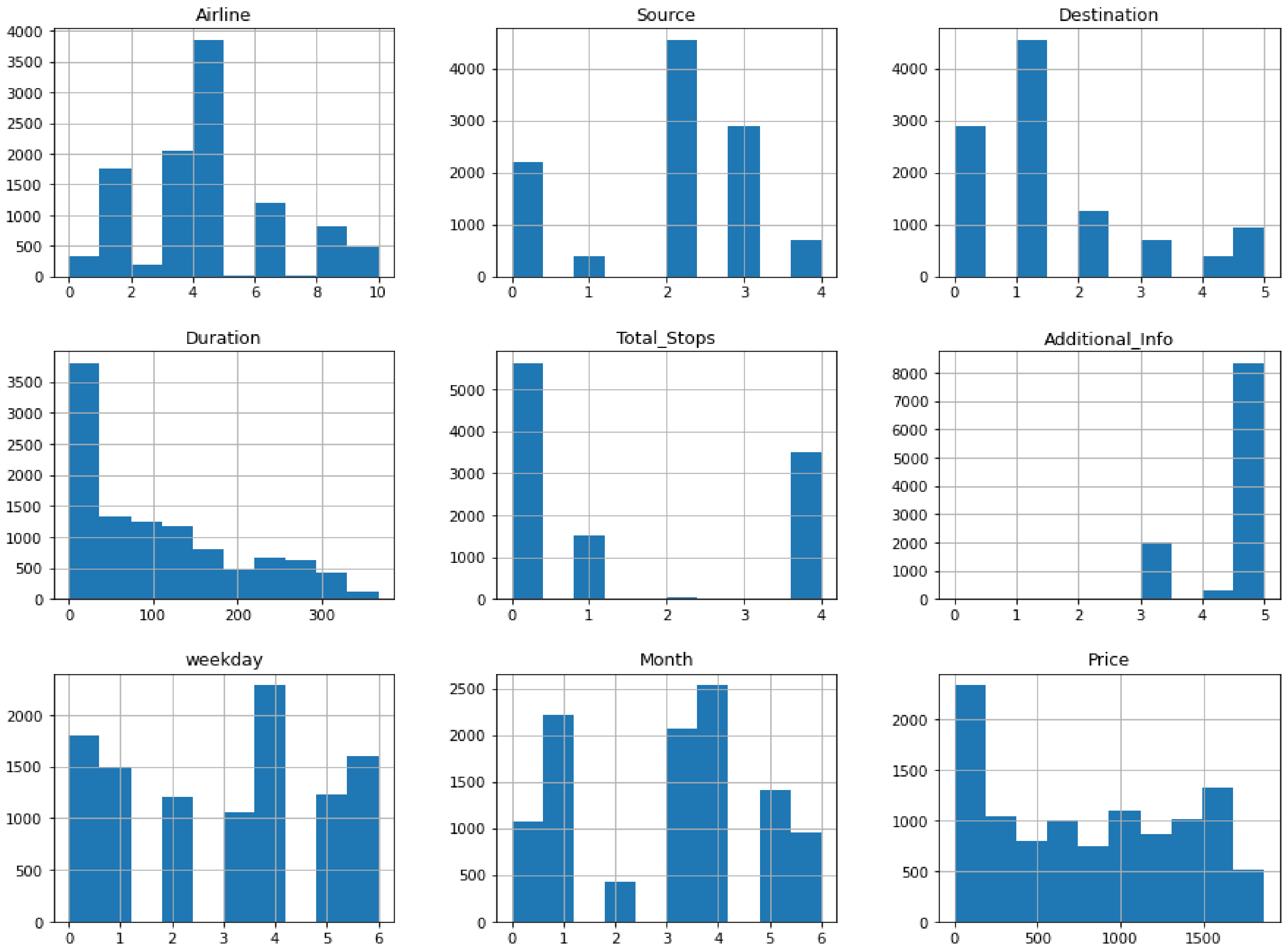
```
train_data['weekday'] = train_data['Date_of_Journey'].dt.day_name()
test_data['weekday'] = test_data['Date_of_Journey'].dt.day_name()

plt.figure(figsize=(12, 6))
sns.barplot(data=train_data, x='weekday', y='Price')
plt.show()

train_data['weekday'] = le.fit_transform(train_data['weekday'])
test_data['weekday'] = le.transform(test_data['weekday'])
```

train_data['Month'] = train_data['Date_of_Journey'].dt.month
test_data['Month'] = test_data['Date_of_Journey'].dt.month

We will make two columns such as “weekday” and the “month” and convert column the of Date_of_Journey .

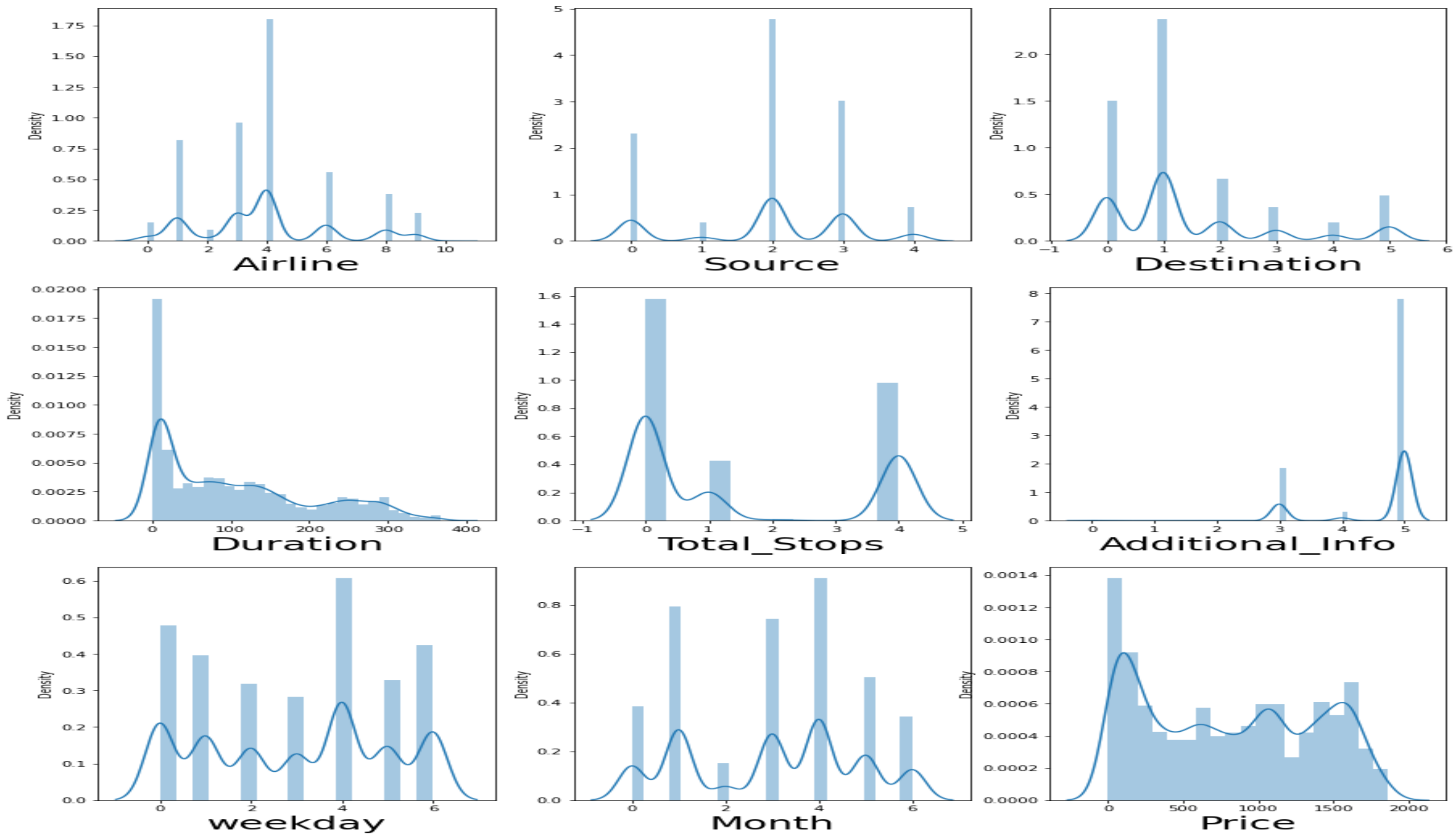


Data.corr() will check correlation between the features with the target column “Price”.The highest correlation is with the “duration”,”Destination” and the “total_stops”.



Airline
Source
Destination
Duration
Total_Stops
Additional_Info
weekday
Month
Price

Figure will show us the outlier present in the “price” column



Removing the oulier using Z-score:-

In [63]:
```
from scipy.stats import zscore
import numpy as np
```

In [64]:
```
z=np.abs(zscore(train_data))
threshold=3
train_data_new=train_data[(z<3).all(axis=1)]
print(train_data.shape)
print(train_data_new.shape)
```

```
(10678, 9)
(10648, 9)
```

In [65]:
```
percent_loss=(10678-10648)/10678*100
print(percent_loss)
```

```
0.2809514890428919
```

Splitting the data into “X” and “Y”

- The loss of data is 0.2% which is acceptable.
- X = train_data_new.drop(['Price'],axis=1)
- Y = train_data_new['Price']
- Splitting the data into “X” as a feature “Y” as a target column or vectors.
- Then we will split the data into training and testing data using train_test_split.
- We will train the data and test the data so we take the train and test data ratio(80:20)

Building the Model:-

Out[118]:

	Models	Accuracy Score	cross-val-score
0	RandomForestRegressor	0.845662	0.842677
1	LogisticRegression	0.845662	0.672407
2	<class 'sklearn.linear_model._stochastic_gradi...	0.845662	0.564790
3	DecisionTreeRegressor	0.845662	0.773681

Hyper-parameter Tunning

- There is a least difference between accuarcy score and the cross-validation score in "RandomForestRegressor".so we will do hyper-parameter tunning in this model to increase the accuracy score,
- For hyper-parameter tunning ,we will use K-fold and we got accuracy score is 84.35%

Predicting the fare price of the test_data

rows × 8 columns

In [88]:
```
test_dummy=pd.get_dummies(test_data.iloc[:,0:8])
```

In [89]:
```
regressor.fit(x_train,y_train)
predict=regressor.predict(test_dummy)
```

In [90]:
```
predict.shape
```

Out[90]:
```
(2671,)
```

In [91]:
```
print(predict)
```

```
[1661.66       1212.65       1400.46       ... 1658.83       1306.65
 1256.70666667]
```

In [92]:
```
test_dummy['Price']=predict
test_dummy
```

Out[92]:

	Airline	Source	Destination	Duration	Total_Stops	Additional_Info	weekday	Month	Price
0	4	2	1	655	0	5	4	6	1661.660000
1	3	3	0	240	0	5	4	12	1212.650000
2	4	2	1	1425	0	3	5	5	1400.460000
3	6	2	1	780	0	5	5	5	1218.582214
4	0	0	2	170	4	5	1	6	903.010000
...	...	...	...	...	...	...	...	...	...

Conclusion Remark

- The accuracy score of the model is 84.35% and there is a scope of increase in the accuracy.
- If we got more information such as seat allocations,when ticket get booked and the class.we can predict the model with good accuracy.