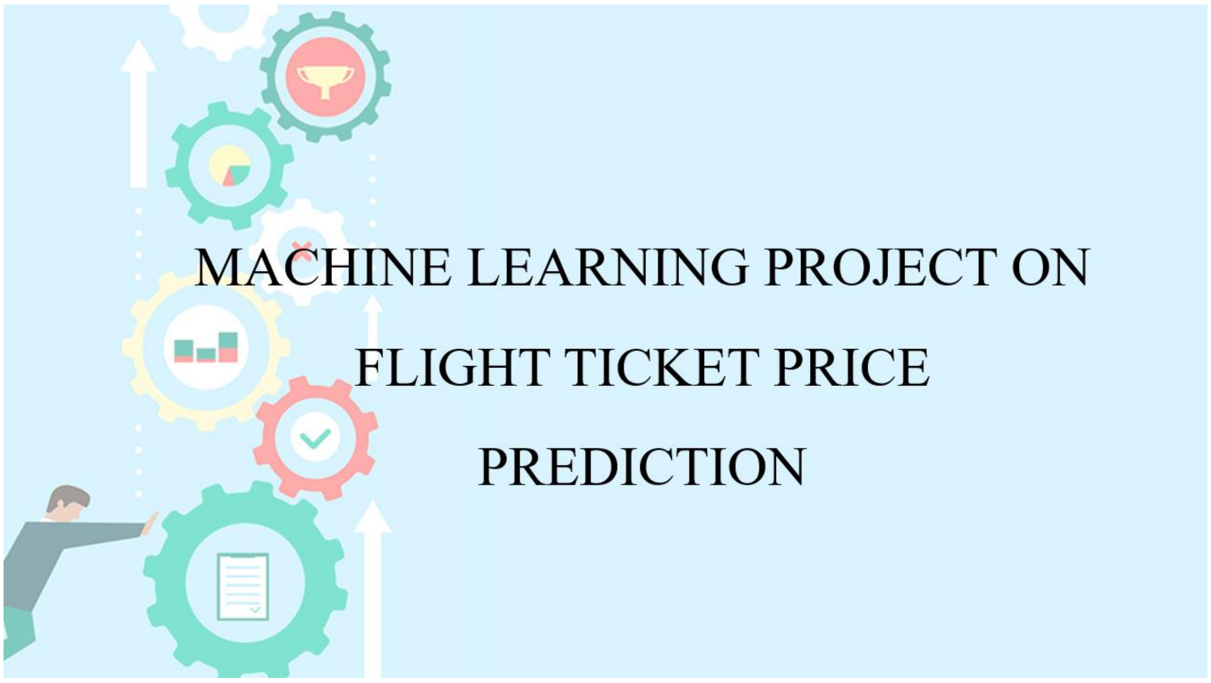
Flight Ticket Price Prediction



In this blog we need to use our skills and build a machine learning model to predict the price of the flight ticket. We can get dataset by using the link below:

https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects

Let's start our project by knowing what is machine learning.

Machine Learning:

Machine learning is a capacity of the machines to analyse a set of data and build generic algorithms. There is no need to write codes, just feeding of data is enough it builds its logic based on it.

There are majorly two types of machine learning

- Supervised learning
- Un supervised learning

Supervised Learning

From the above context itself, it is so much evidence that there must be some supervisor as a teacher to carry on the process. Usually in this methodology, the training or the teaching is provided.

For Example, if it is to train to identify various kinds of fruits it must be like the shape of the fruit is round and a cavity is found at the top centre and the colour must be red. Which signifies Apple? If the shape is curved and long enough with the colour of green or yellow then it must be Banana. Now after this training it is given with another set of examples to identify.

Un Supervised Learning

This second type of learning in which there is no supervision or guidance required is called unsupervised learning. Here it can act without any guidance required. For example, if a picture of dogs and cats is given together to analyse, it has no information on Cats or Dogs. But still, it can categorize based on the similarities between them by analysing the patterns, Size, shape, figures, and differences.

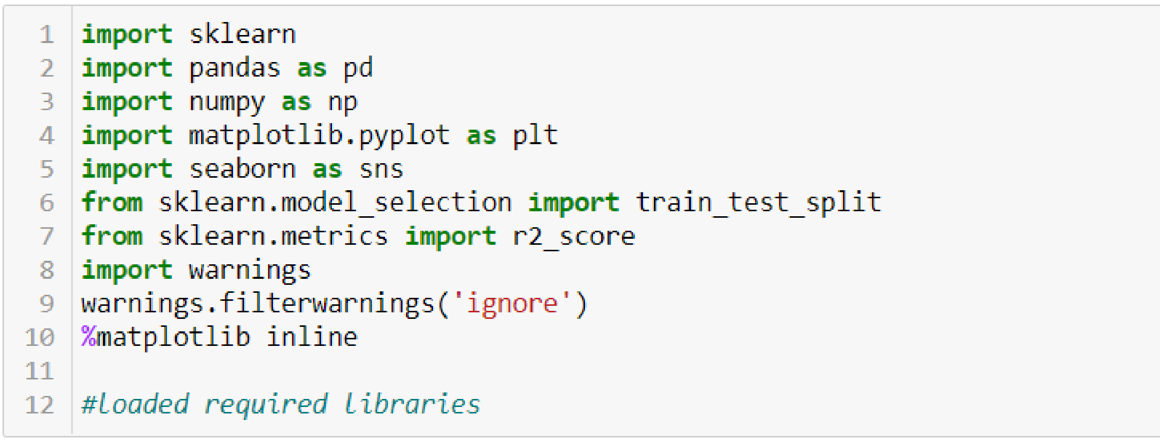
Now let’s see the steps that we follow while building our model

- Problem Definition.
- Data Analysis.
- Building Machine Learning Models.
- Model Deployment
- Concluding Remarks.

Problem Definition:

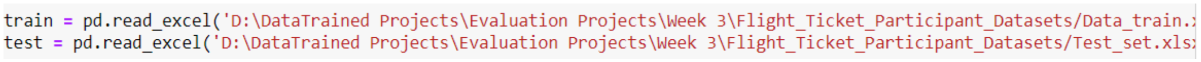
Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travellers saying that flight ticket prices are so unpredictable. Here you will be provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

Now, let’s begin our model building by importing all the required libraries



```python
import sklearn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

#loaded required libraries
```

In the next step we need to extract the downloaded dataset by using pandas.



```python
train = pd.read_excel('D:\DataTrained Projects\Evaluation Projects\Week 3\Flight_Ticket_Participant_Datasets/Data_train.
test = pd.read_excel('D:\DataTrained Projects\Evaluation Projects\Week 3\Flight_Ticket_Participant_Datasets/Test_set.xls
```

Our dataset consists of both test and training data separately, so we whatever the preprocessing steps we perform on our training data, we need to perform same on test data also.

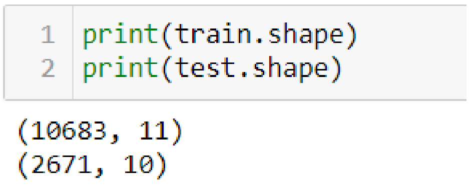
<u>Data Analysis</u>:

Now, we will start our analysis on data.

The size of our dataset :

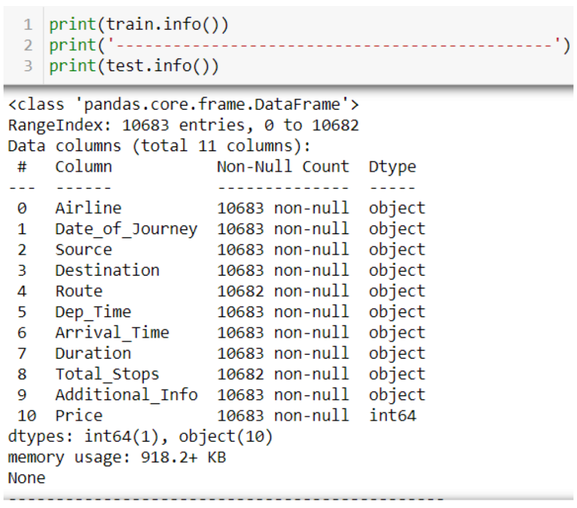
Size of training set: **10683** records

Size of test set: **2671** records



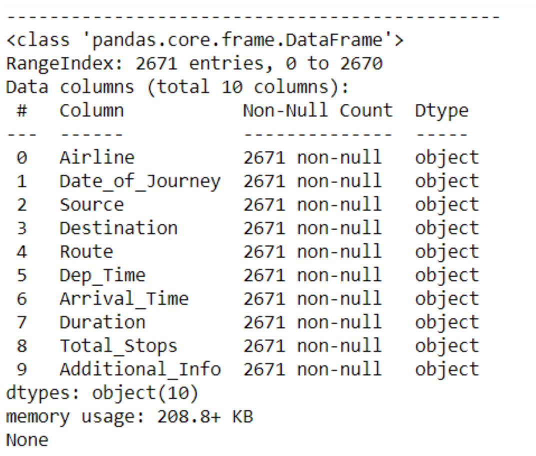
```python
print(train.shape)
print(test.shape)
```

```
(10683, 11)
(2671, 10)
```



```
print(train.info())
print('---------------------------------------------')
print(test.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
None
---------------------------------------------
```



```
---------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 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
dtypes: object(10)
memory usage: 208.8+ KB
None
```

We can see that most of the columns are in object datatype, so we need to convert all categorical columns to numerical columns.

Features:

Airline: The name of the airline.

Date_of_Journey: The date of the journey

Source: The source from which the service begins.

Destination: The destination where the service ends.

Route: The route taken by the flight to reach the destination.

Dep_Time: The time when the journey starts from the source.

Arrival_Time: Time of arrival at the destination.

Duration: Total duration of the flight.

Total_Stops: Total stops between the source and destination.

Additional_Info: Additional information about the flight

Price: The price of the ticket.

```
print(train.isnull().values.any())
train.isnull().sum()
```

```
True

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
```

There are two null values in entire dataset(one in Route and other in Additional_Info), so we can drop those na values.

```
train.dropna(inplace=True)
```

In the next step of preprocessing, we will seperate date and time column s, later we will drop the actual columns.

```
train['day'] = train['Date_of_Journey'].str.split('/').str[0].astype(int)
train['month'] = train['Date_of_Journey'].str.split('/').str[1].astype(int)
train.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

```
train['dep_hour'] = pd.to_datetime(train['Dep_Time']).dt.hour
train['dep_minutes'] = pd.to_datetime(train['Dep_Time']).dt.minute
train.drop(['Dep_Time'], axis = 1, inplace = True)
```

```
train['Arrival_hour'] = pd.to_datetime(train.Arrival_Time).dt.hour
train['Arrival_min'] = pd.to_datetime(train.Arrival_Time).dt.minute
train.drop(['Arrival_Time'], axis = 1, inplace = True)
```

```
test['day'] = test['Date_of_Journey'].str.split('/').str[0].astype(int)
test['month'] = test['Date_of_Journey'].str.split('/').str[1].astype(int)
test.drop(['Date_of_Journey'], axis = 1, inplace = True)
```

```
test['dep_hour'] = pd.to_datetime(test['Dep_Time']).dt.hour
test['dep_min'] = pd.to_datetime(test['Dep_Time']).dt.minute
test.drop(['Dep_Time'], axis = 1, inplace = True)
```

```
test['Arrival_hour'] = pd.to_datetime(test.Arrival_Time).dt.hour
test['Arrival_min'] = pd.to_datetime(test.Arrival_Time).dt.minute
test.drop(['Arrival_Time'], axis = 1, inplace = True)
```

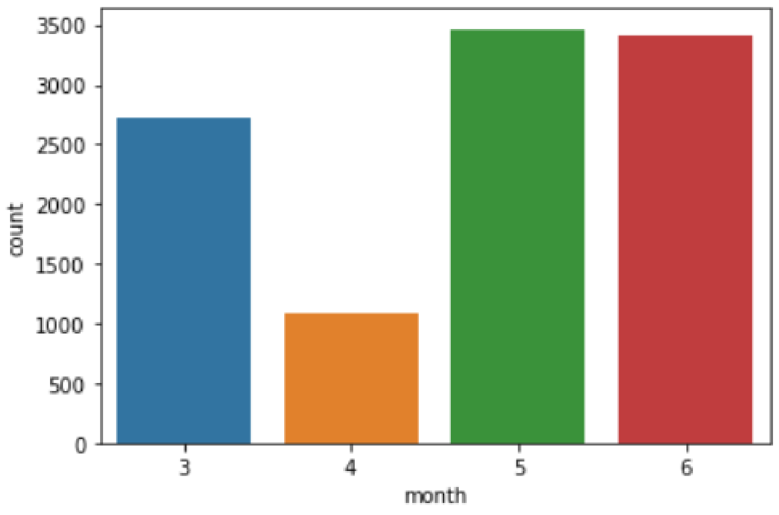
We can see that capital_gain and capital_gain are having difference between 50% quartile and mean values, so there is a chance to have outliers in those columns.

Data Visualization :

In data visualization we mostly used countplot from seaborn.

```
sns.countplot(x='month',data=train)
```

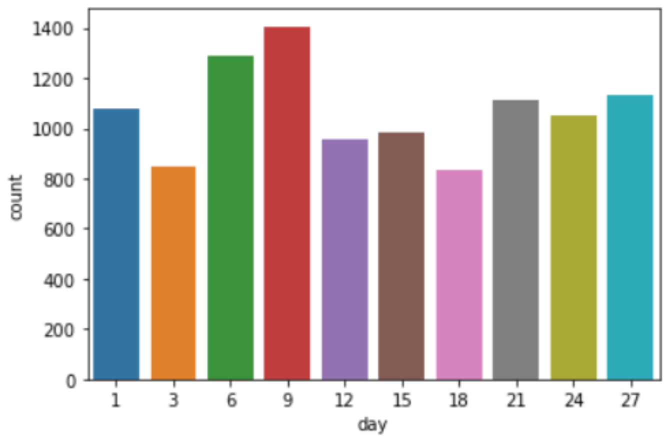
```
<AxesSubplot:xlabel='month', ylabel='count'>
```



We could see that more number of flights are present during 5,6 months

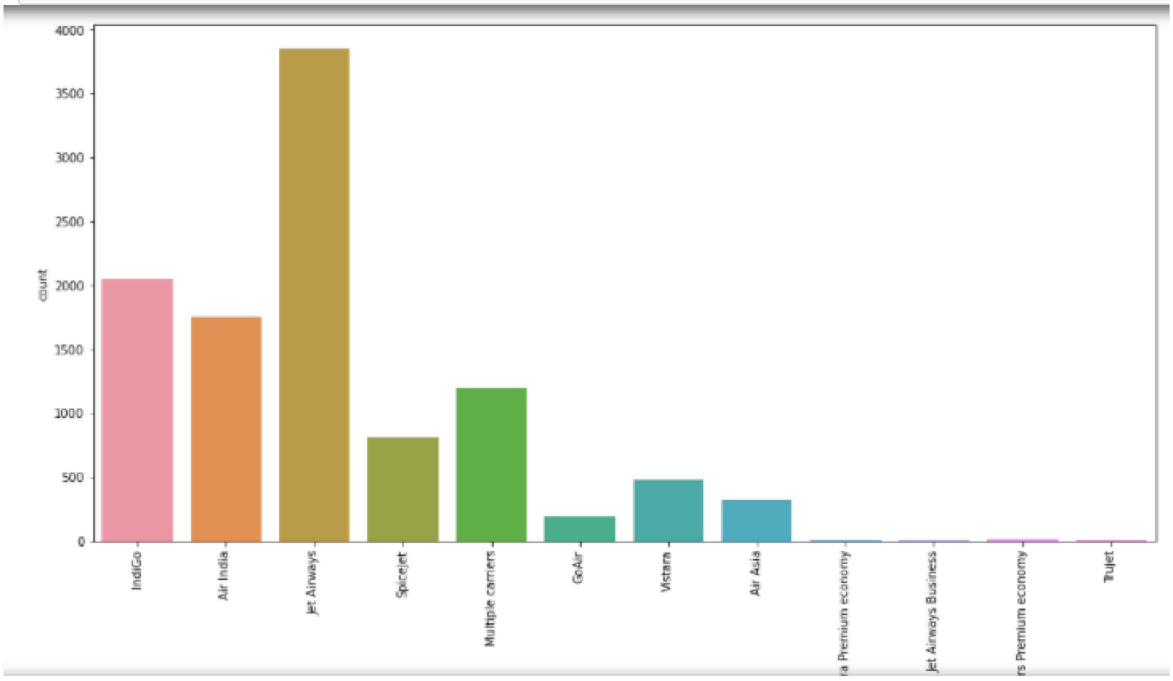
```
sns.countplot(x='day',data=train)
```

```
<AxesSubplot:xlabel='day', ylabel='count'>
```



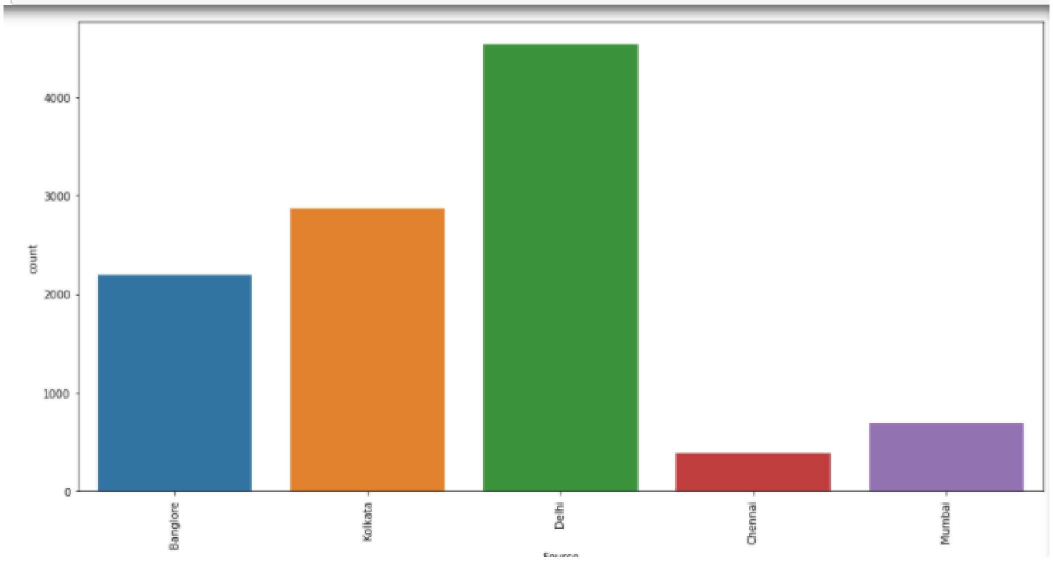
We can observe that more number of flights are traveling in most days of second week

```
plt.figure(figsize=[16,8])
sns.countplot(x='Airline',data=train)
plt.xticks(rotation = 90)
plt.show()
```



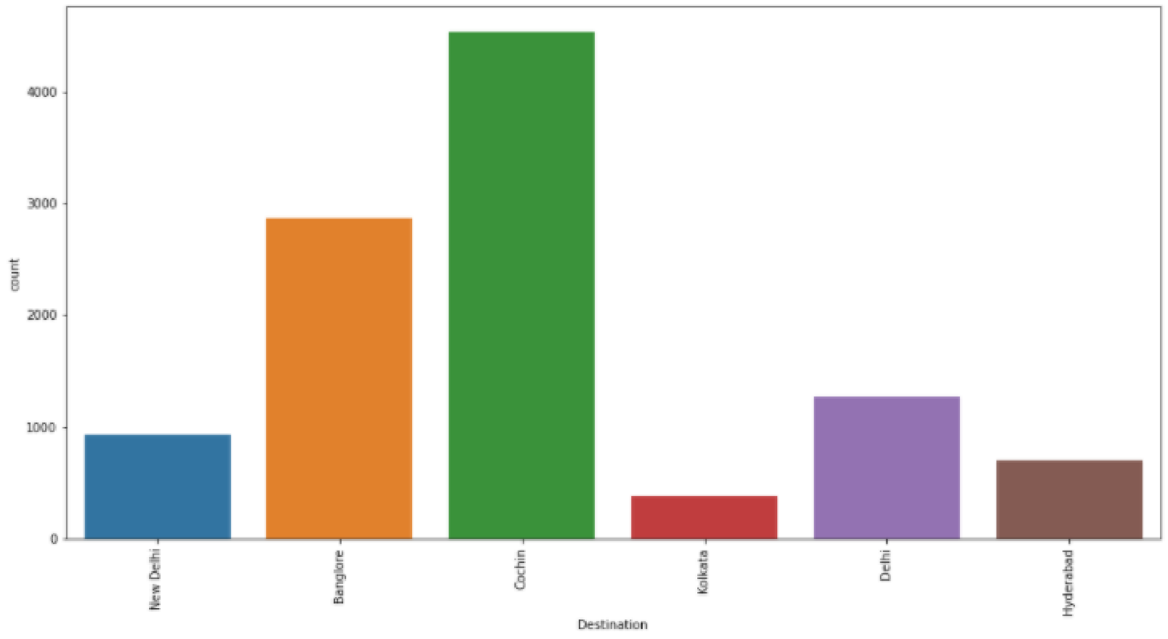
We can see that jet airways is having more number of flights when compared with other companies.

```
plt.figure(figsize=[16,8])
sns.countplot(x='Source',data=train)
plt.xticks(rotation = 90)
plt.show()
```



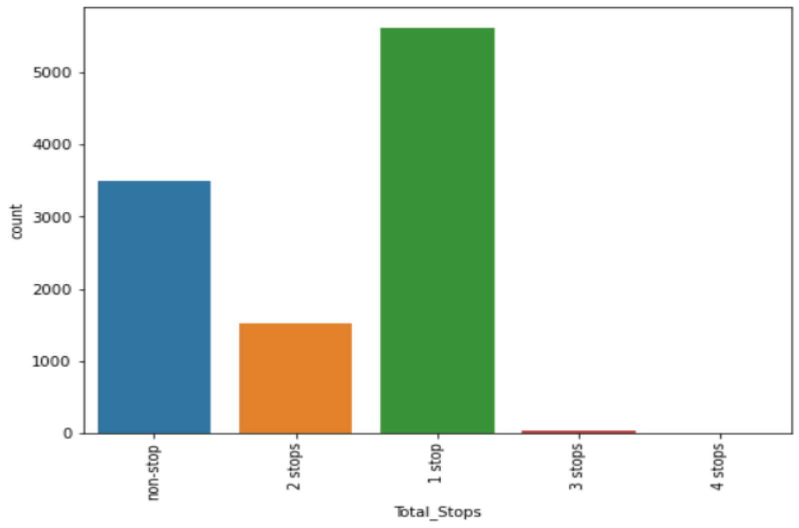
More number of flights are originating from Delhi

```
plt.figure(figsize=[16,8])
sns.countplot(x='Destination',data=train)
plt.xticks(rotation = 90)
plt.show()
```



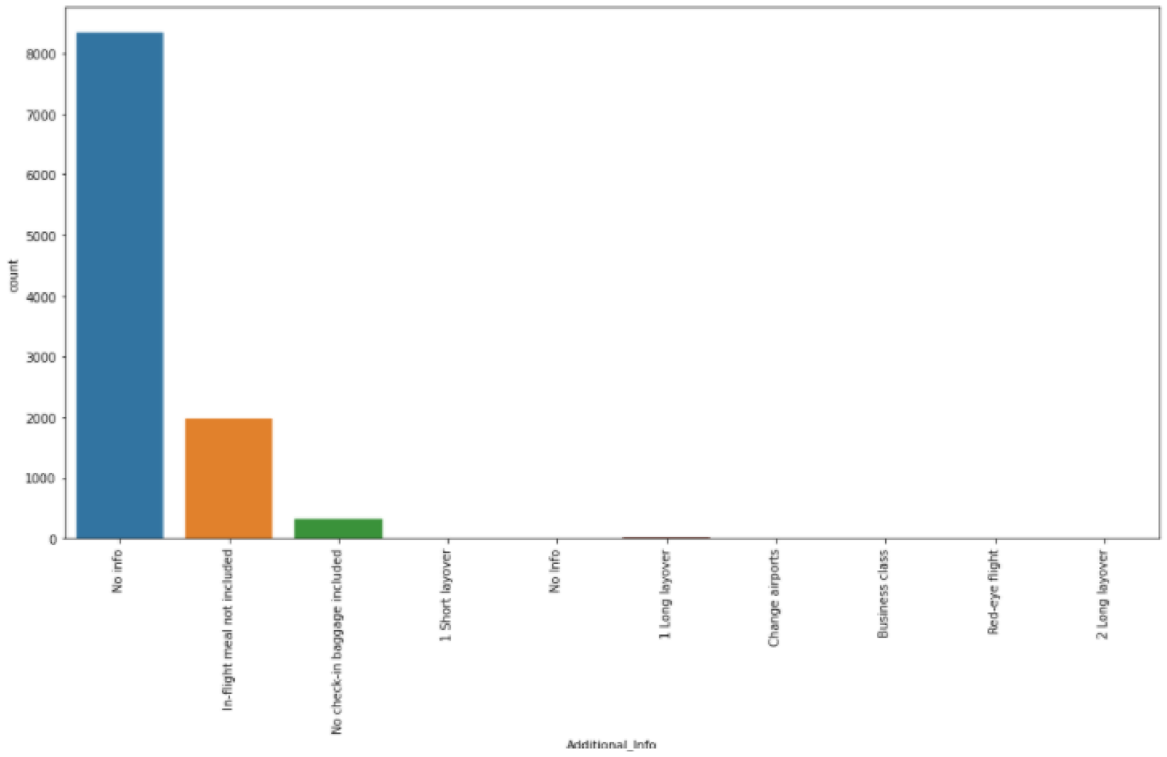
More number of flights has their destination as Cochin

```
plt.figure(figsize=[8,6])
sns.countplot(x='Total_Stops',data=train)
plt.xticks(rotation = 90)
plt.show()
```



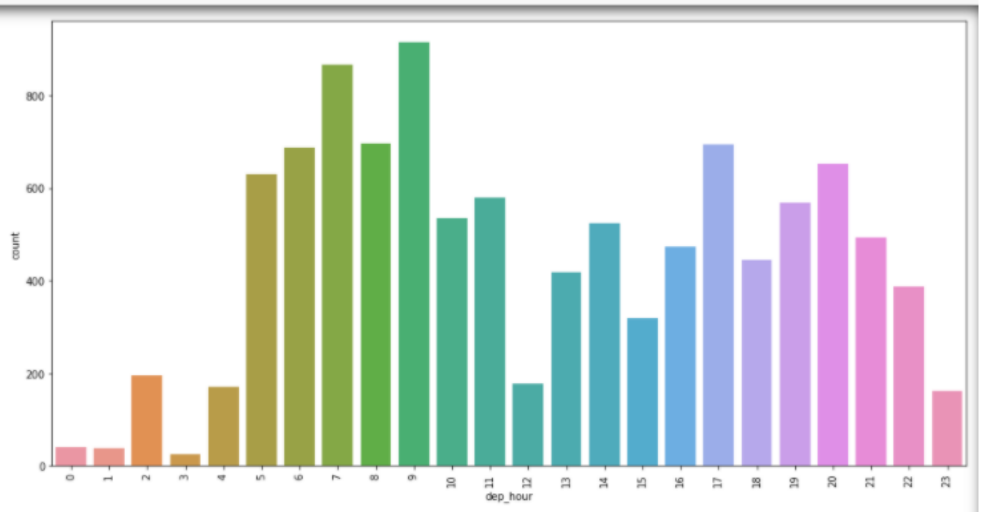
More number of flights are of 1stop

```
plt.figure(figsize=[16,8])
sns.countplot(x='Additional_Info',data=train)
plt.xticks(rotation = 90)
plt.show()
```



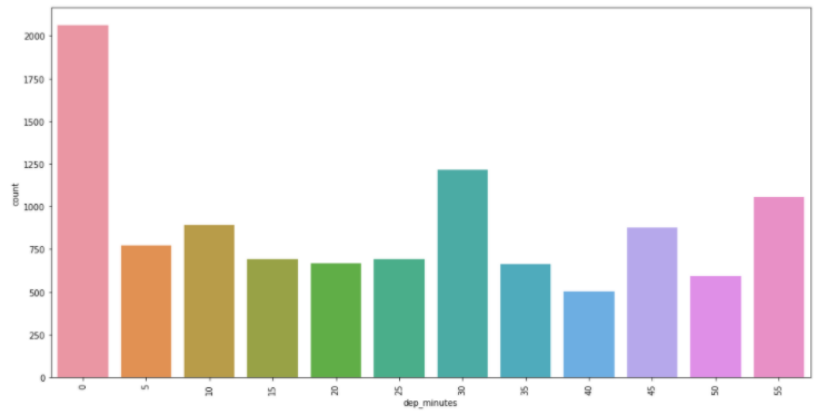
We can see that most of the companies are not providing extra information other than the required information.

```
plt.figure(figsize=[16,8])
sns.countplot(x='dep_hour',data=train)
plt.xticks(rotation = 90)
plt.show()
```



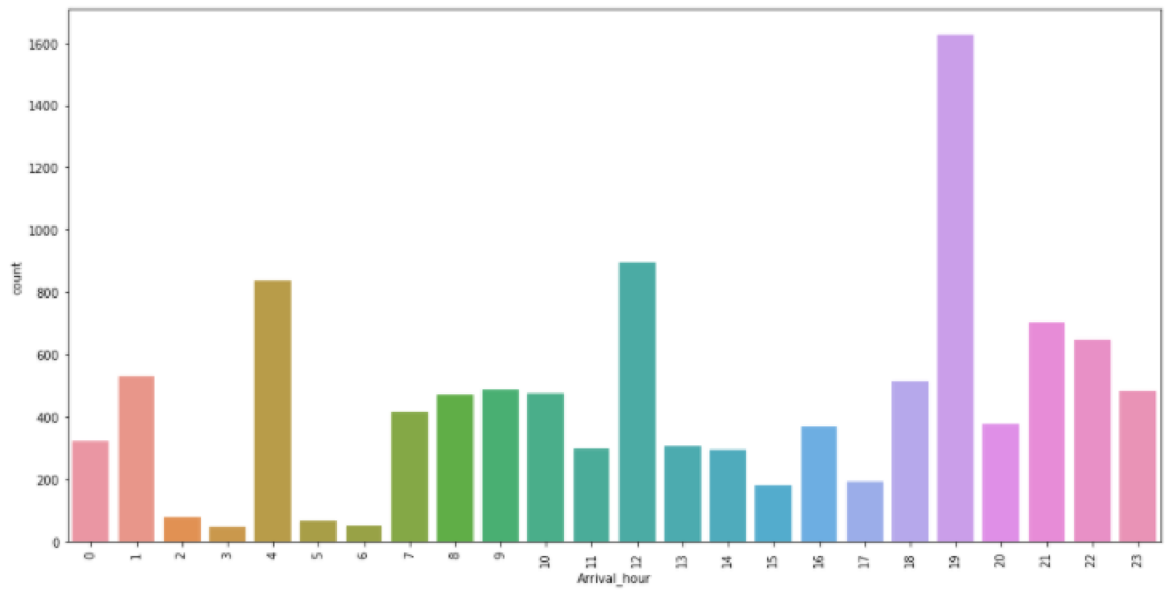
We can observe that more number of fligts are leaving between 5am-10am and 5pm-9pm.

```
plt.figure(figsize=[16,8])
sns.countplot(x='dep_minutes',data=train)
plt.xticks(rotation = 90)
plt.show()
```



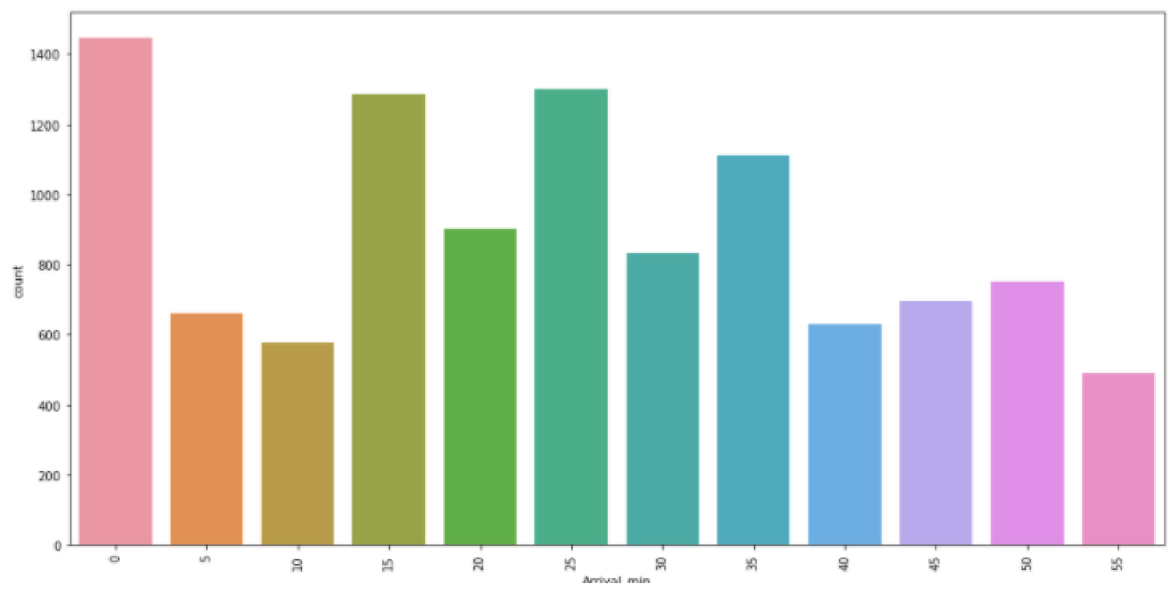
We can observe that more flights are leaving before starting 5 minutes of every hour.

```
plt.figure(figsize=[16,8])
sns.countplot(x='Arrival_hour',data=train)
plt.xticks(rotation = 90)
plt.show()
```



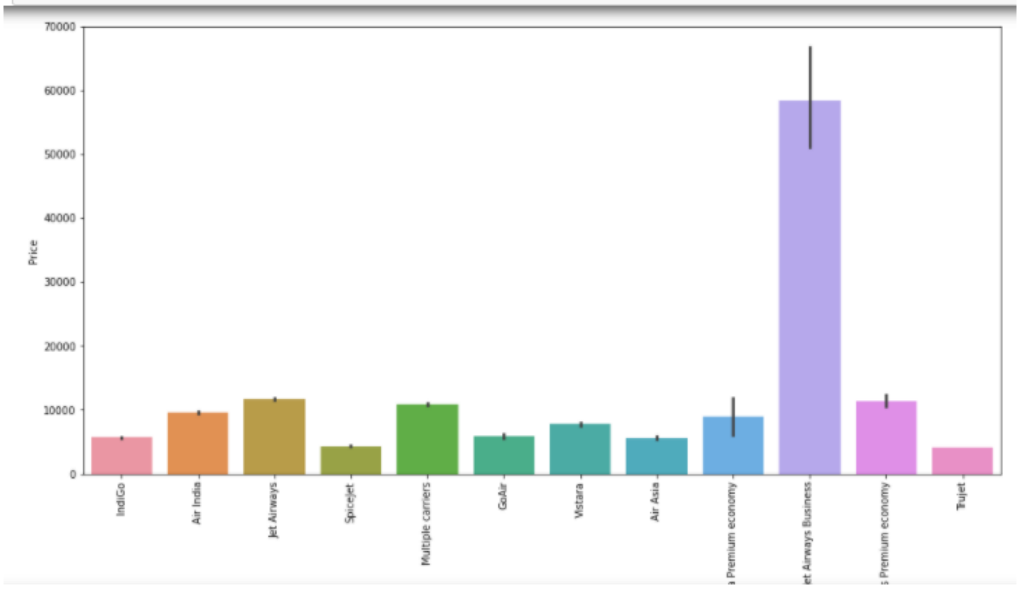
More number of flights are arriving at 7pm,12pm,4am,9pm,10pm.

```
plt.figure(figsize=[16,8])
sns.countplot(x='Arrival_min',data=train)
plt.xticks(rotation = 90)
plt.show()
```



More number of flights are arriving in the first 5 min and also during the mid hour we can see more number number of flights arriving.

```
plt.figure(figsize=[16,8])
sns.barplot(x='Airline',y='Price',data=train)
plt.xticks(rotation = 90)
plt.show()
```

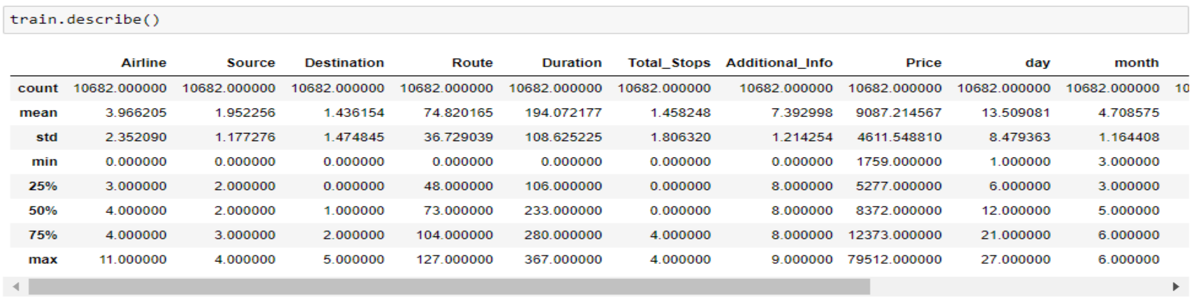


Jet airways business ticket is having highest price ticket when compared with all other companies.

In the next step of preprocessing, we will convert all categorical columns to numerical columns using label encoder.

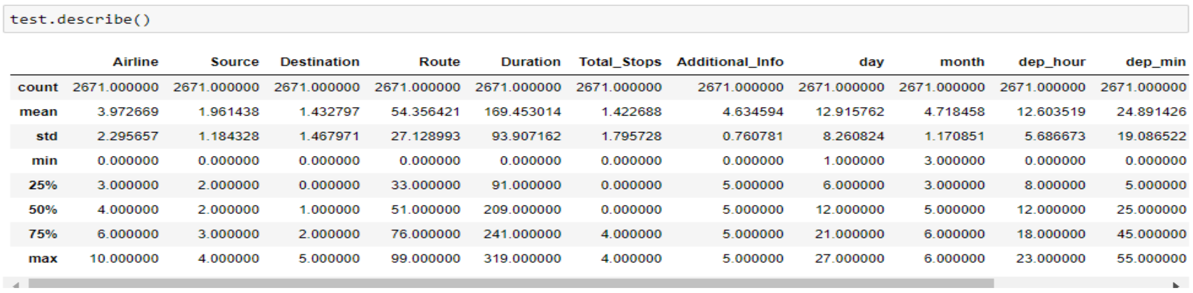
```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train['Airline'] = le.fit_transform(train['Airline'])
train['Source'] = le.fit_transform(train['Source'])
train['Destination'] = le.fit_transform(train['Destination'])
train['Route'] = le.fit_transform(train['Route'])
train['Duration'] = le.fit_transform(train['Duration'])
train['Total_Stops'] = le.fit_transform(train['Total_Stops'])
train['Additional_Info'] = le.fit_transform(train['Additional_Info'])

test['Airline'] = le.fit_transform(test['Airline'])
test['Source'] = le.fit_transform(test['Source'])
test['Destination'] = le.fit_transform(test['Destination'])
test['Route'] = le.fit_transform(test['Route'])
test['Duration'] = le.fit_transform(test['Duration'])
test['Total_Stops'] = le.fit_transform(test['Total_Stops'])
test['Additional_Info'] = le.fit_transform(test['Additional_Info'])
```

We applied label encoder process to both test and training data converted all the categorical columns to numerical columns.



```python
train.describe()
```

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	day	month	
count	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10
mean	3.966205	1.952256	1.436154	74.820165	194.072177	1.458248	7.392998	9087.214567	13.509081	4.708575	
std	2.352090	1.177276	1.474845	36.729039	108.625225	1.806320	1.214254	4611.548810	8.479363	1.164408	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1759.000000	1.000000	3.000000	
25%	3.000000	2.000000	0.000000	48.000000	106.000000	0.000000	8.000000	5277.000000	6.000000	3.000000	
50%	4.000000	2.000000	1.000000	73.000000	233.000000	0.000000	8.000000	8372.000000	12.000000	5.000000	
75%	4.000000	3.000000	2.000000	104.000000	280.000000	4.000000	8.000000	12373.000000	21.000000	6.000000	
max	11.000000	4.000000	5.000000	127.000000	367.000000	4.000000	9.000000	79512.000000	27.000000	6.000000	

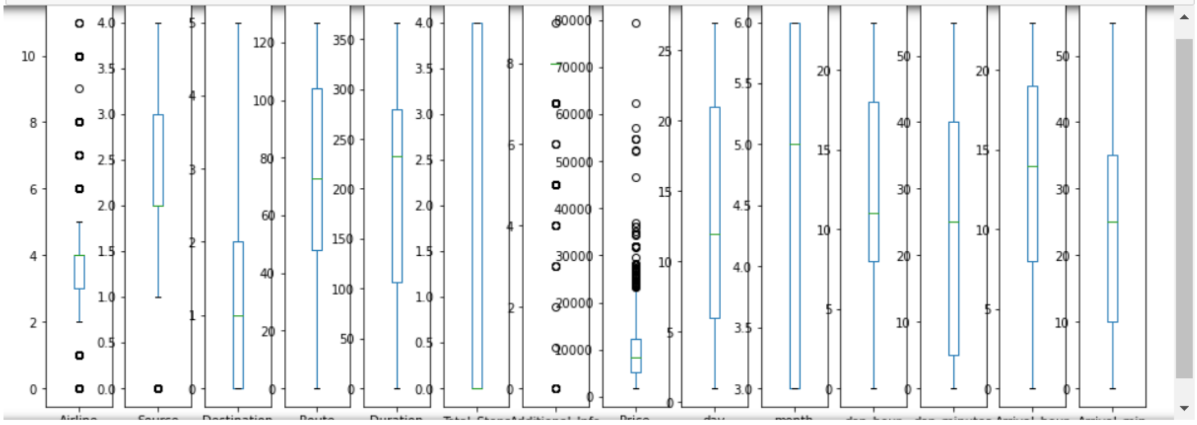


```python
test.describe()
```

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	day	month	dep_hour	dep_min
count	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000	2671.000000
mean	3.972669	1.961438	1.432797	54.356421	169.453014	1.422688	4.634594	12.915762	4.718458	12.603519	24.891426
std	2.295657	1.184328	1.467971	27.128993	93.907162	1.795728	0.760781	8.260824	1.170851	5.686673	19.086522
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000
25%	3.000000	2.000000	0.000000	33.000000	91.000000	0.000000	5.000000	6.000000	3.000000	8.000000	5.000000
50%	4.000000	2.000000	1.000000	51.000000	209.000000	0.000000	5.000000	12.000000	5.000000	12.000000	25.000000
75%	6.000000	3.000000	2.000000	76.000000	241.000000	4.000000	5.000000	21.000000	6.000000	18.000000	45.000000
max	10.000000	4.000000	5.000000	99.000000	319.000000	4.000000	5.000000	27.000000	6.000000	23.000000	55.000000

We can observe that the information provided above, that the values of 50% quartile and mean are almost same, so there might me less number of outliers if any.

```
train.plot(kind='box',subplots=True,figsize=[16,6])
plt.show()
```



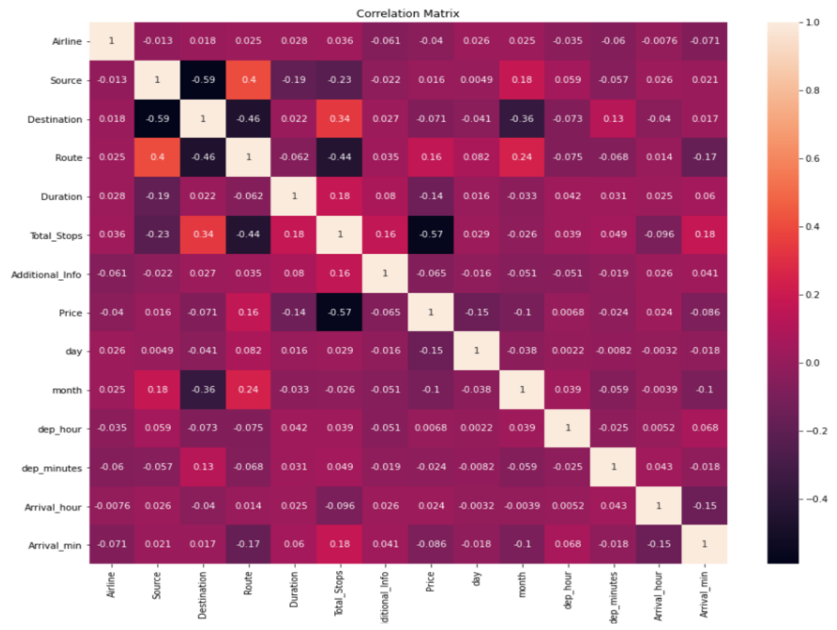
We have few outliers in dataset that can be considered.

```
train.plot(kind='density',subplots=True, sharex=False, legend=False, fontsize=1, figsize=[4,20])
plt.show()
```

```
sns.pairplot(train)
plt.show()
```

From density plot and pairplot we see that there is no much skewness in data.

```
plt.figure(figsize=[15,12])
sns.heatmap(corr_mat, annot=True)
plt.title('Correlation Matrix')
plt.show()
```



```
corr_target = train.corrwith(train['Price'], axis=0)
corr_target
```

```
Airline           -0.039565
Source             0.015999
Destination       -0.071122
Route              0.164149
Duration          -0.144280
Total_Stops       -0.571221
Additional_Info   -0.065463
Price              1.000000
day               -0.153774
month             -0.103643
dep_hour           0.006799
dep_minutes       -0.024458
Arrival_hour       0.024244
Arrival_min       -0.086155
dtype: float64
```

Every column is having some relation with target and Total_Stops is having strong relation with the price of the ticket.

Till now we have completed all the data preprocessing steps that are required to filter our data.

We started with checking for null values, proceeded with data visualization and concluded by checking for skewness.

Model Building:

Now it's time to start our model building process.

We will start by separating target column from our dataset.

```
x = train.drop('Price', axis=1)
y = train['Price']
```

Now we will split our data into training and test data.

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25, random_state=15)
```

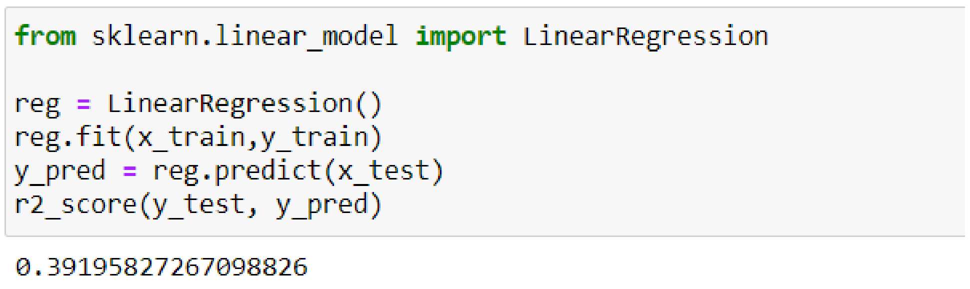
We are now ready to start our model building.

First we will use Linear Regression Model,

Linear Regression Model:

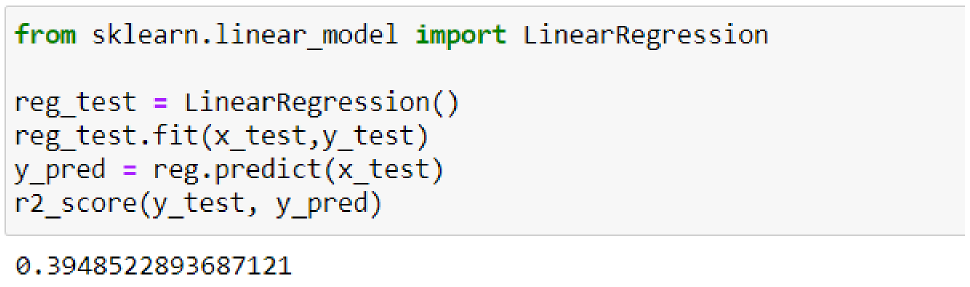
Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

We will use r2 score to check the accuracy of our model.



```
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
reg.fit(x_train,y_train)
y_pred = reg.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.391958272670988 26
```



```
from sklearn.linear_model import LinearRegression

reg_test = LinearRegression()
reg_test.fit(x_test,y_test)
y_pred = reg.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.3948522893687121
```

k-Nearest Neighbors (KNN)

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

```
from sklearn.neighbors import KNeighborsRegressor

KNR = KNeighborsRegressor()
KNR.fit(x_train, y_train)
y_pred=KNR.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.6209851416621182
```

```
from sklearn.neighbors import KNeighborsRegressor

KNR_test = KNeighborsRegressor()
KNR_test.fit(x_test, y_test)
y_pred=KNR.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.6209851416621182
```

Decision Tree Regressor:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

```
from sklearn.tree import DecisionTreeRegressor
RTD = DecisionTreeRegressor(random_state = 0)
RTD.fit(x_train, y_train)
y_pred = RTD.predict(x_test)
r2_score(y_test, y_pred)
```

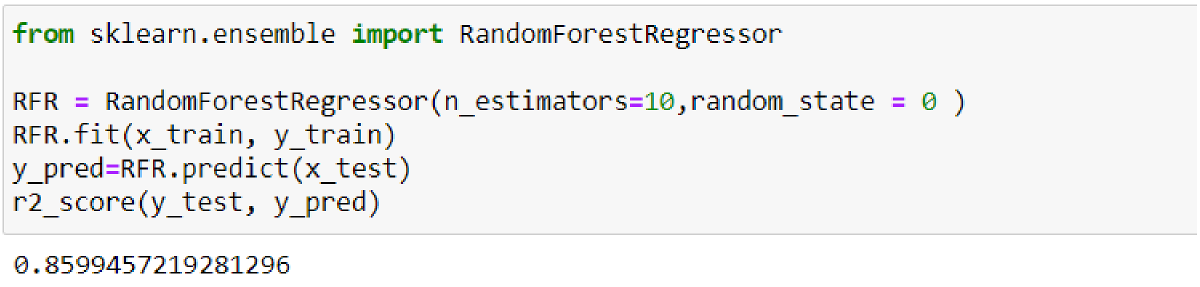
```
0.7823646116297907
```

```
from sklearn.tree import DecisionTreeRegressor
RTD_test = DecisionTreeRegressor(random_state = 0)
RTD_test.fit(x_test, y_test)
y_pred = RTD.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.9991699698441794
```

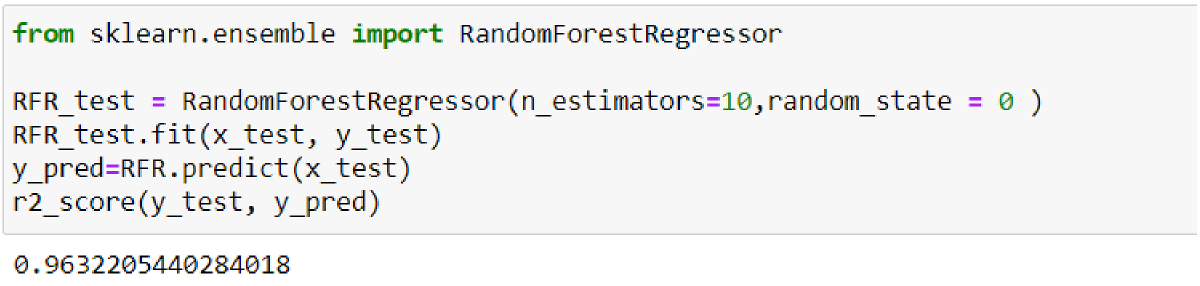
Random Forest Regression:

Random Forest Regression is a supervised learning algorithm that uses **ensemble learning** method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.



```
from sklearn.ensemble import RandomForestRegressor

RFR = RandomForestRegressor(n_estimators=10,random_state = 0 )
RFR.fit(x_train, y_train)
y_pred=RFR.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.8599457219281296
```



```
from sklearn.ensemble import RandomForestRegressor

RFR_test = RandomForestRegressor(n_estimators=10,random_state = 0 )
RFR_test.fit(x_test, y_test)
y_pred=RFR.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.9632205440284018
```

Support Vector Regression:

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

```
from sklearn.svm import SVR

SVR_Reg = SVR(kernel = "rbf",degree=3, C=40)
SVR_Reg.fit(x_train, y_train)
y_pred = SVR_Reg.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.12692991630232786
```

```
from sklearn.svm import SVR

SVR_Reg_test = SVR(kernel = "rbf",degree=3, C=40)
SVR_Reg_test.fit(x_test, y_test)
y_pred = SVR_Reg.predict(x_test)
r2_score(y_test, y_pred)
```

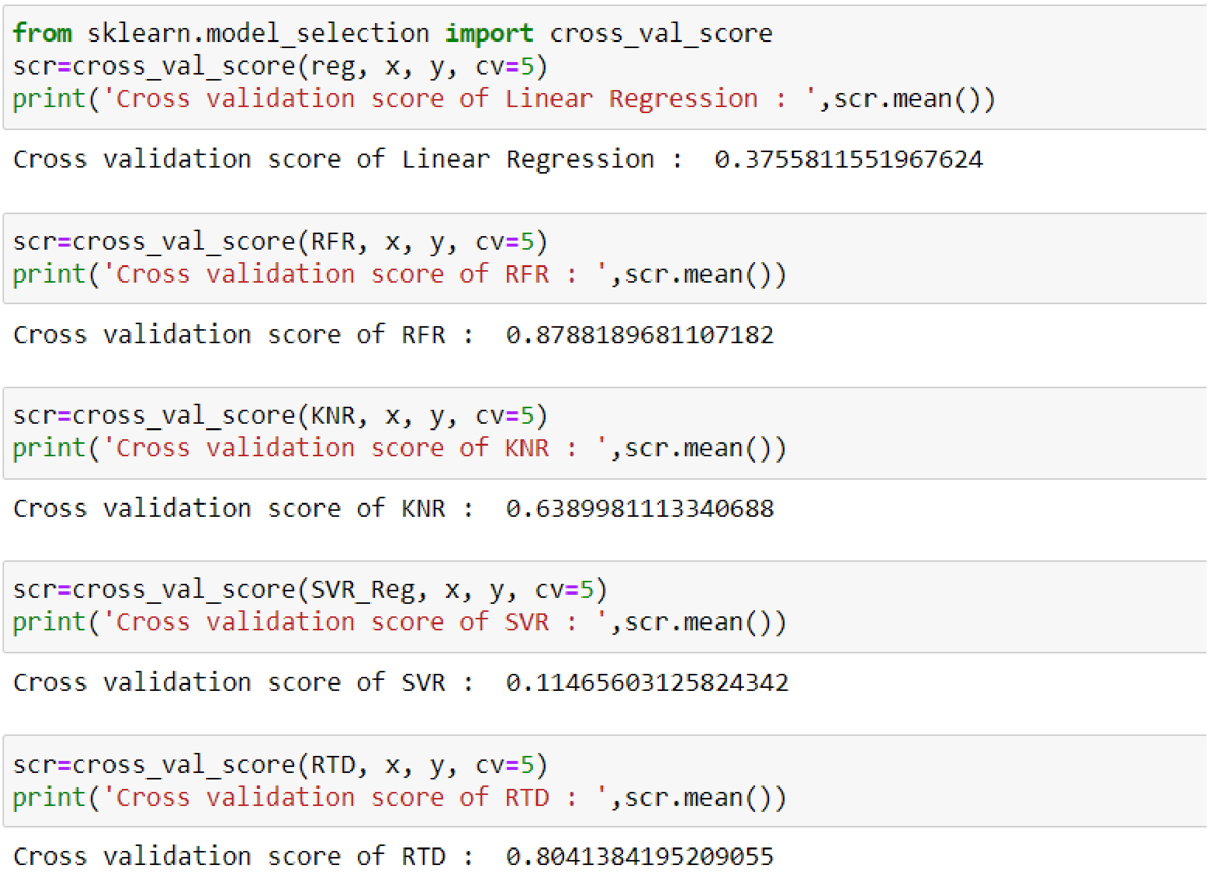
```
0.08700424526017103
```

By building our model using various algorithms we can select the best model which will give predictions with good accuracy, but by looking at the accuracy score we can’t conclude that which is the best model.

We need to perform cross validation to check the best model among all the models built.

Cross Validation:

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).



```python
from sklearn.model_selection import cross_val_score
scr=cross_val_score(reg, x, y, cv=5)
print('Cross validation score of Linear Regression : ',scr.mean())
```

```
Cross validation score of Linear Regression :  0.3755811551967624
```

```python
scr=cross_val_score(RFR, x, y, cv=5)
print('Cross validation score of RFR : ',scr.mean())
```

```
Cross validation score of RFR :  0.8788189681107182
```

```python
scr=cross_val_score(KNR, x, y, cv=5)
print('Cross validation score of KNR : ',scr.mean())
```

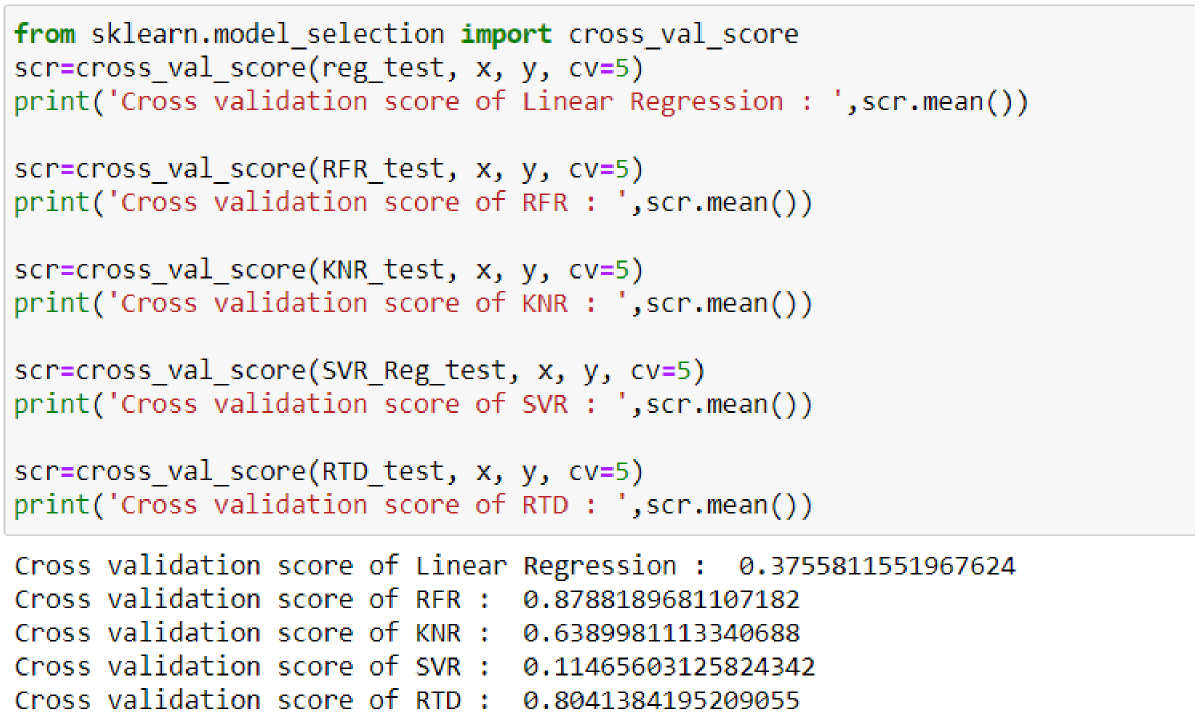
```
Cross validation score of KNR :  0.6389981113340688
```

```python
scr=cross_val_score(SVR_Reg, x, y, cv=5)
print('Cross validation score of SVR : ',scr.mean())
```

```
Cross validation score of SVR :  0.11465603125824342
```

```python
scr=cross_val_score(RTD, x, y, cv=5)
print('Cross validation score of RTD : ',scr.mean())
```

```
Cross validation score of RTD :  0.8041384195209055
```



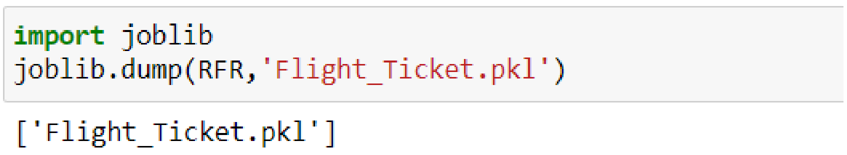
```python
from sklearn.model_selection import cross_val_score
scr=cross_val_score(reg_test, x, y, cv=5)
print('Cross validation score of Linear Regression : ',scr.mean())

scr=cross_val_score(RFR_test, x, y, cv=5)
print('Cross validation score of RFR : ',scr.mean())

scr=cross_val_score(KNR_test, x, y, cv=5)
print('Cross validation score of KNR : ',scr.mean())

scr=cross_val_score(SVR_Reg_test, x, y, cv=5)
print('Cross validation score of SVR : ',scr.mean())

scr=cross_val_score(RTD_test, x, y, cv=5)
print('Cross validation score of RTD : ',scr.mean())
```

```
Cross validation score of Linear Regression :  0.3755811551967624
Cross validation score of RFR :  0.8788189681107182
Cross validation score of KNR :  0.6389981113340688
Cross validation score of SVR :  0.11465603125824342
Cross validation score of RTD :  0.8041384195209055
```

Random Forest Regressor is having least difference between cross validation score and accuracy score so we can choose Random Forest Regressor as best model.

Deployment:

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data.



```
import joblib
joblib.dump(RFR,'Flight_Ticket.pkl')
```

```
['Flight_Ticket.pkl']
```

Conclusion:

As part of our study, we understood how to build a proper machine learning model and what are the points that are to be considered and systematic workflow.

I will be happy to take any kind of suggestions for the betterment of project.

Hope, you liked my work and thanks for reading.