FLIP ROBO

Project Name

FLIGHT PRICE PREDICTION

PROJECT REPORT

Submitted by:

Priyanka Saikia

ACKNOWLEDGMENT

I would like to express my deep sense of gratitude to my SME (Subject Matter Expert) **Mr. Shubham Yadav** as well as **Flip Robo Technologies** who gave me the golden opportunity to do this data analysis project on **Flight Price Prediction Project** , which also helped me in doing lots of research and I came to know about so many new things.
I have put in my all efforts while doing this project.

All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://github.com/
4) https://www.kaggle.com/
5) https://towardsdatascience.com/
6) https://www.analyticsvidhya.com/

Priyanka Saikia

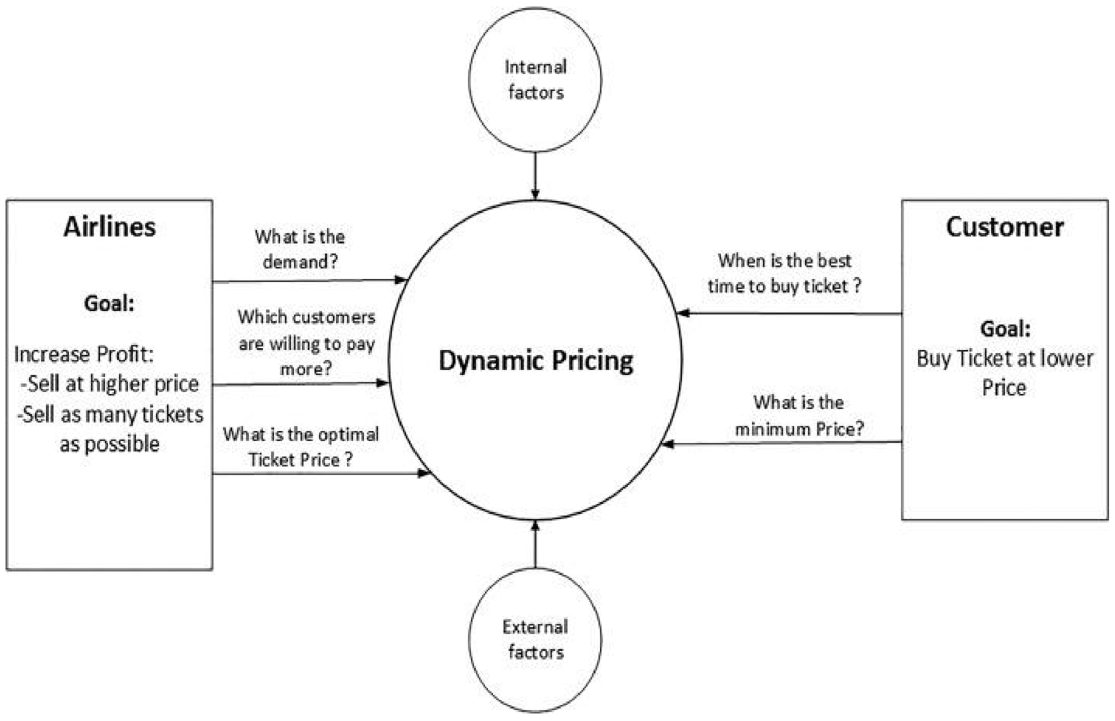
INTRODUCTION

- ## Business Problem Framing:

The airline industry is considered as one of the most sophisticated industry in using complex pricing strategies. Nowadays, ticket prices can vary dynamically and significantly for the same flight, even for nearby seats. The ticket price of a specific flight can change up to 7 times a day. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit. However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company losing revenue. Airlines companies are generally equipped with advanced tools and capabilities that enable them to control the pricing process. However, customers are also becoming more strategic with the development of various online tools to compare prices across various airline companies. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone.

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)


Internal factors
Airlines
Goal:
Increase Profit:
-Sell at higher price
-Sell as many tickets as possible
What is the demand?
Which customers are willing to pay more?
What is the optimal Ticket Price ?
Dynamic Pricing
When is the best time to buy ticket ?
What is the minimum Price?
Customer
Goal:
Buy Ticket at lower Price
External factors

- ## Conceptual Background of the Domain Problem:

  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.

- ## Review of Literature

  As per the requirement of client, we have scraped data from online sites like yatra.com and based on that data we have done analysis like based on which feature of the data, prices are changing and checked the relationship of flight price with all the feature like which flight he should choose.

- ## Motivation for the Problem Undertaken

  I have worked on this on the bases of client requirements and followed all the steps till model deployment.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  As per the client’s requirement for this flight price prediction project I have scraped data from yatra.com website. This is then saved into Excel format file. I have shared the script for web scraping into the GitHub repository.

  In our scraped dataset our target variable column "Price" is a continuous variable. Therefore, we will be handling this modelling problem as a Regression problem

  Then loaded this data into a dataframe. For checking datatypes and null values pandas.DataFrame.info() method has been used. I have gone through several EDA steps to analyse the data. After all the necessary steps I have built an ML model to predict the flight prices.

- ## Data Sources and their formats

  This project is done in three parts:
  - Data Collection
  - Data Analysis
  - Model Building

1. Data Collection

You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, more the data better the model. In this section you have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

2. Data Analysis

After cleaning the data, you have to do some analysis on the data.
Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? What is the best time to buy so that the consumer can save the most by taking the least risk? Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

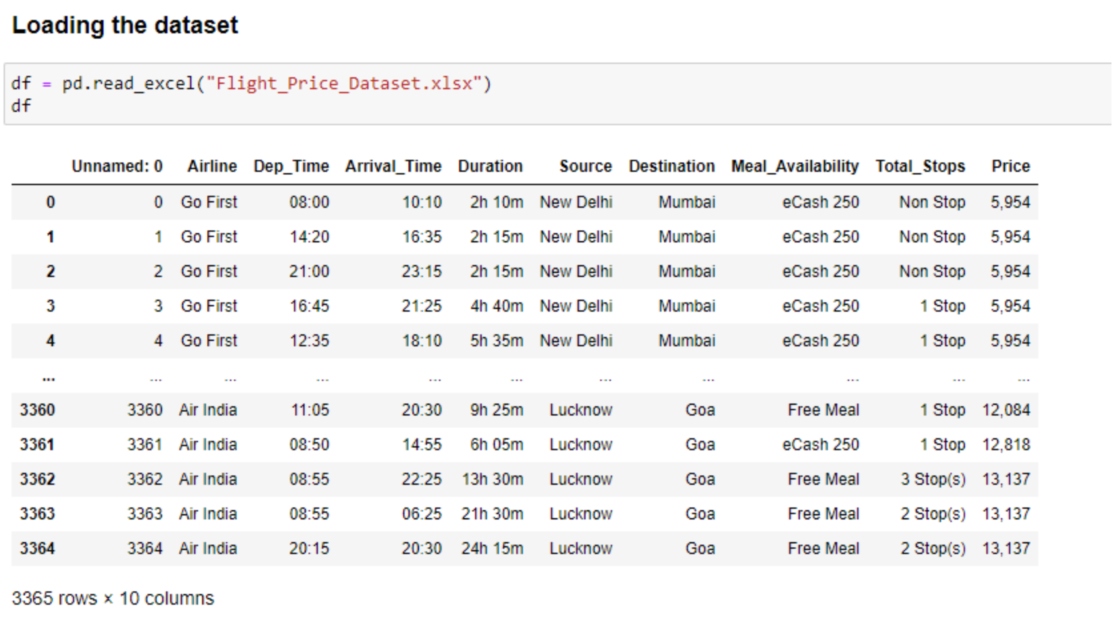
3. Model Building

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.
Following the complete life cycle of data science. Including all the steps like-
1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

- ## Data Sources and their formats

  The dataset is in the form of Excel file and consists of 9 columns (8 features and 1 label) with 3365 rows as explained below:

1) Airline: Name of the airline
2) Dep_time: Time of departure of flight from the source location
3) Arrival_Time: Time of arrival at destination
4) Duration: Total time of the journey from source to destination
5) Source: City name from where the flight is departing
6) Destination: Name of the city where flight is arriving
7) Meal_availability : Information about meal fare
8) Total_stops: Number of stops during the journey
9) Price : Flight fare



Loading the dataset

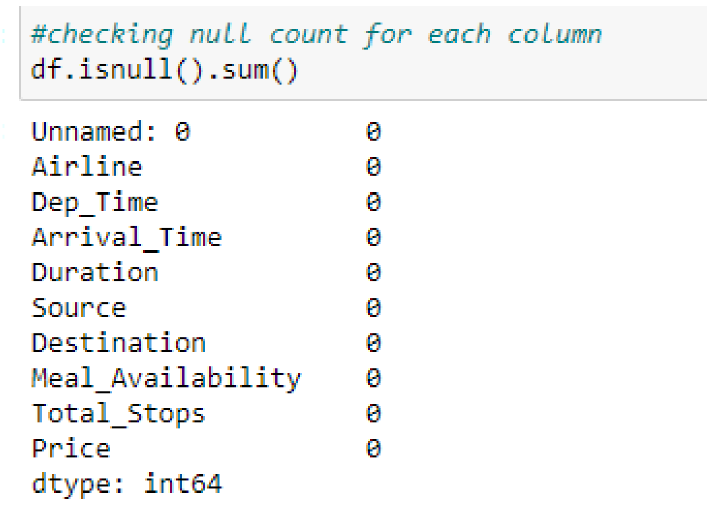
```
df = pd.read_excel("Flight_Price_Dataset.xlsx")
df
```

	Unnamed: 0	Airline	Dep_Time	Arrival_Time	Duration	Source	Destination	Meal_Availability	Total_Stops	Price
0	0	Go First	08:00	10:10	2h 10m	New Delhi	Mumbai	eCash 250	Non Stop	5,954
1	1	Go First	14:20	16:35	2h 15m	New Delhi	Mumbai	eCash 250	Non Stop	5,954
2	2	Go First	21:00	23:15	2h 15m	New Delhi	Mumbai	eCash 250	Non Stop	5,954
3	3	Go First	16:45	21:25	4h 40m	New Delhi	Mumbai	eCash 250	1 Stop	5,954
4	4	Go First	12:35	18:10	5h 35m	New Delhi	Mumbai	eCash 250	1 Stop	5,954
...	...	...	...	...	...	...	...	...	...	...
3360	3360	Air India	11:05	20:30	9h 25m	Lucknow	Goa	Free Meal	1 Stop	12,084
3361	3361	Air India	08:50	14:55	6h 05m	Lucknow	Goa	eCash 250	1 Stop	12,818
3362	3362	Air India	08:55	22:25	13h 30m	Lucknow	Goa	Free Meal	3 Stop(s)	13,137
3363	3363	Air India	08:55	06:25	21h 30m	Lucknow	Goa	Free Meal	2 Stop(s)	13,137
3364	3364	Air India	20:15	20:30	24h 15m	Lucknow	Goa	Free Meal	2 Stop(s)	13,137

3365 rows × 10 columns

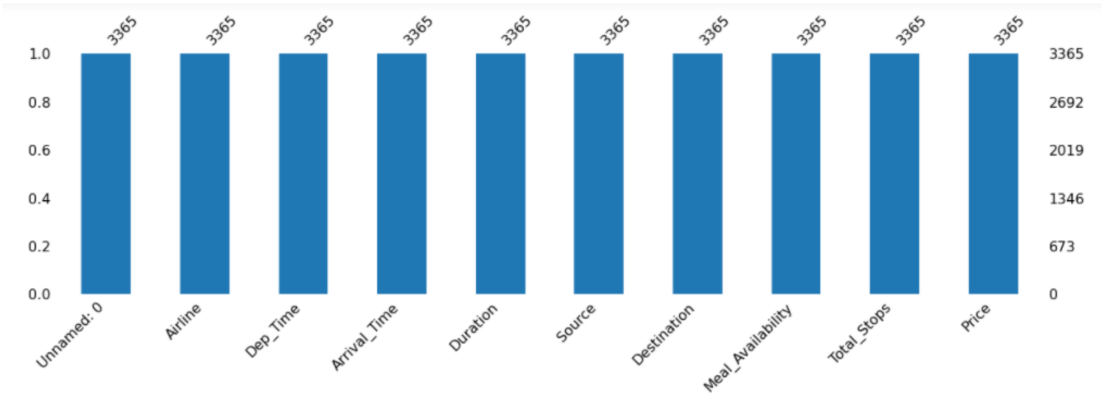
- ## Data Pre-processing Done:

For the data pre-processing step, I checked through the dataframe for missing values and renamed values that needed a better meaningful name.

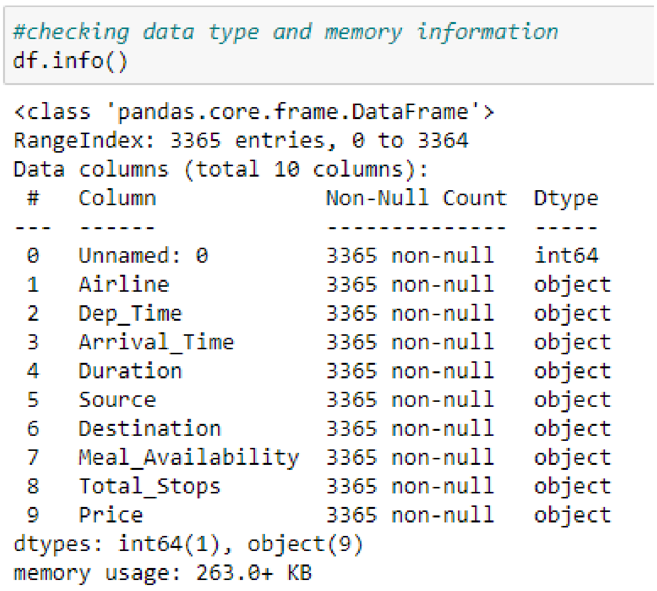


```
#checking null count for each column
df.isnull().sum()
```

```
Unnamed: 0          0
Airline             0
Dep_Time            0
Arrival_Time        0
Duration            0
Source              0
Destination         0
Meal_Availability   0
Total_Stops         0
Price               0
dtype: int64
```



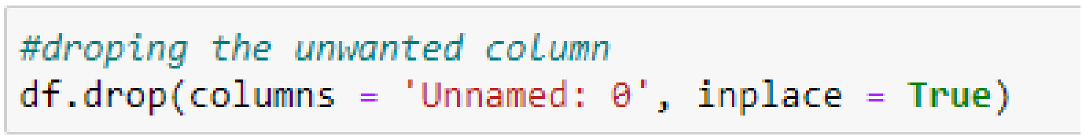
Checked the datatype details for each column to understand the numeric ones and its further conversion process.



```
#checking data type and memory information
df.info()
```

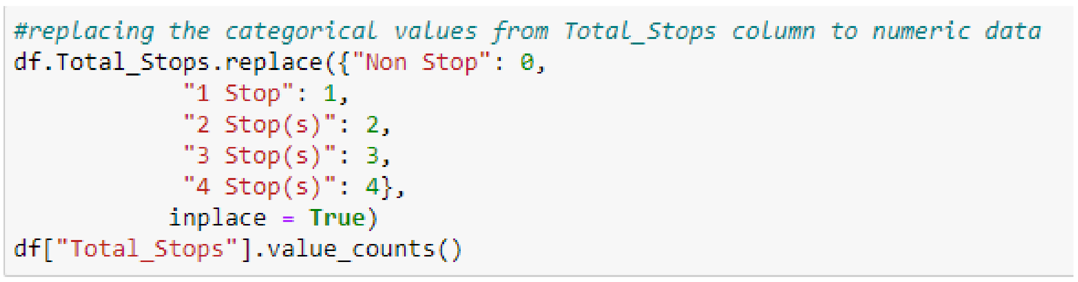
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3365 entries, 0 to 3364
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        3365 non-null   int64
 1   Airline           3365 non-null   object
 2   Dep_Time          3365 non-null   object
 3   Arrival_Time      3365 non-null   object
 4   Duration          3365 non-null   object
 5   Source            3365 non-null   object
 6   Destination       3365 non-null   object
 7   Meal_Availability 3365 non-null   object
 8   Total_Stops       3365 non-null   object
 9   Price             3365 non-null   object
dtypes: int64(1), object(9)
memory usage: 263.0+ KB
```

Dropping some unwanted columns.



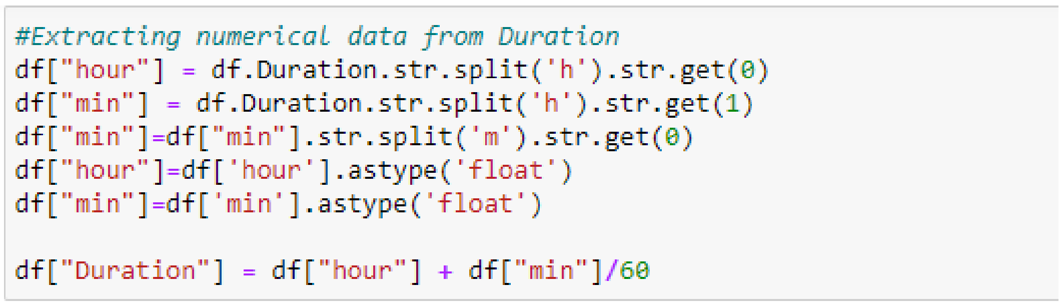
```
#droping the unwanted column
df.drop(columns = 'Unnamed: 0', inplace = True)
```

The various data processing performed on our data set are shown below with the code.



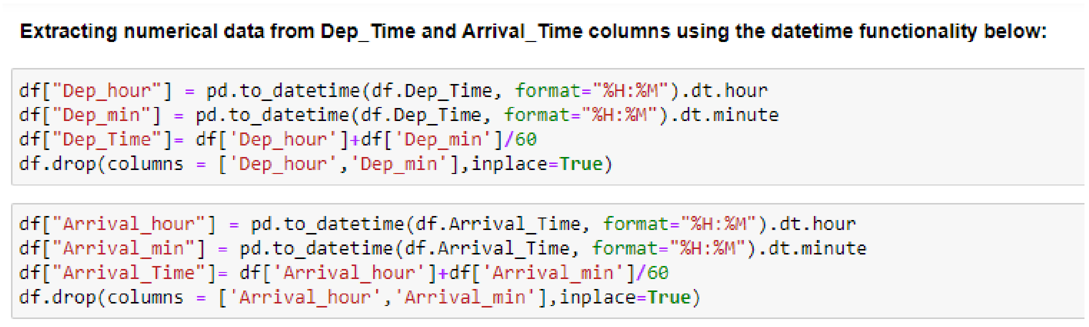
```
#replacing the categorical values from Total_Stops column to numeric data
df.Total_Stops.replace({"Non Stop": 0,
             "1 Stop": 1,
             "2 Stop(s)": 2,
             "3 Stop(s)": 3,
             "4 Stop(s)": 4},
            inplace = True)
df["Total_Stops"].value_counts()
```

The column Total_Stops is a categorical column and filled with string values, but we need to fill the values in ordinal manner so I have replaced the entries with corresponding numeric values.



```
#Extracting numerical data from Duration
df["hour"] = df.Duration.str.split('h').str.get(0)
df["min"] = df.Duration.str.split('h').str.get(1)
df["min"]=df["min"].str.split('m').str.get(0)
df["hour"]=df['hour'].astype('float')
df["min"]=df['min'].astype('float')

df["Duration"] = df["hour"] + df["min"]/60
```

The above Duration column was having string entries in hours and minutes; I have separated hours and minutes into two different columns by splitting.



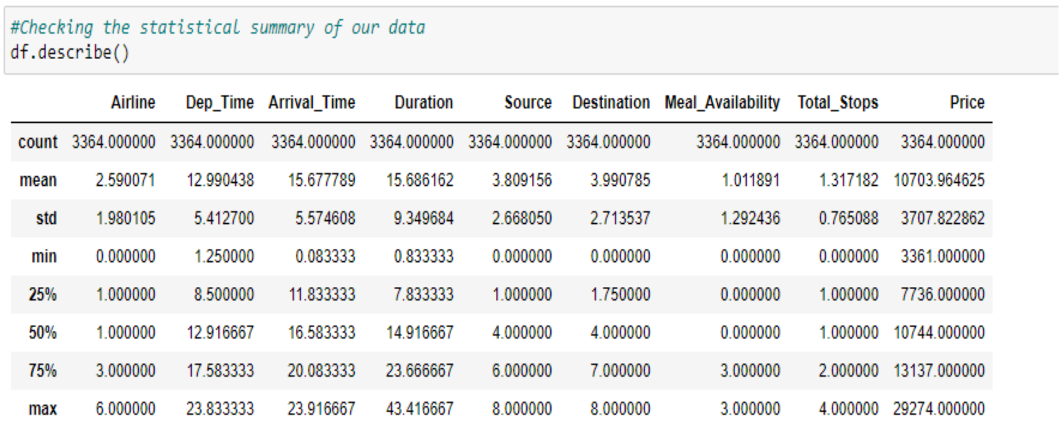
Extracting numerical data from Dep_Time and Arrival_Time columns using the datetime functionality below:

```
df["Dep_hour"] = pd.to_datetime(df.Dep_Time, format="%H:%M").dt.hour
df["Dep_min"] = pd.to_datetime(df.Dep_Time, format="%H:%M").dt.minute
df["Dep_Time"]= df['Dep_hour']+df['Dep_min']/60
df.drop(columns = ['Dep_hour','Dep_min'],inplace=True)
```

```
df["Arrival_hour"] = pd.to_datetime(df.Arrival_Time, format="%H:%M").dt.hour
df["Arrival_min"] = pd.to_datetime(df.Arrival_Time, format="%H:%M").dt.minute
df["Arrival_Time"]= df['Arrival_hour']+df['Arrival_min']/60
df.drop(columns = ['Arrival_hour','Arrival_min'],inplace=True)
```

As similar to the case of duration column, these two columns was also having the time but in a string format. And by using datetime functionality, I have fetched numerical values for the time and filled the respective columns.

Next, I used the "describe" method to check the count, mean, standard deviation, minimum, maximum, 25%, 50% and 75% quartile data.

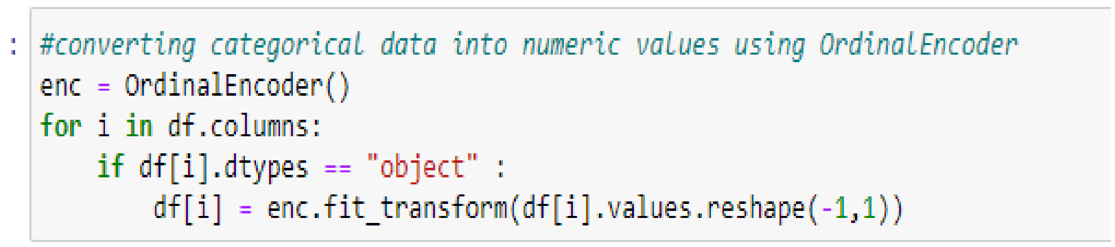


```
#Checking the statistical summary of our data
df.describe()
```

	Airline	Dep_Time	Arrival_Time	Duration	Source	Destination	Meal_Availability	Total_Stops	Price
count	3364.000000	3364.000000	3364.000000	3364.000000	3364.000000	3364.000000	3364.000000	3364.000000	3364.000000
mean	2.590071	12.990438	15.677789	15.686162	3.809156	3.990785	1.011891	1.317182	10703.964625
std	1.980105	5.412700	5.574608	9.349684	2.668050	2.713537	1.292436	0.765088	3707.822862
min	0.000000	1.250000	0.083333	0.833333	0.000000	0.000000	0.000000	0.000000	3361.000000
25%	1.000000	8.500000	11.833333	7.833333	1.000000	1.750000	0.000000	1.000000	7736.000000
50%	1.000000	12.916667	16.583333	14.916667	4.000000	4.000000	0.000000	1.000000	10744.000000
75%	3.000000	17.583333	20.083333	23.666667	6.000000	7.000000	3.000000	2.000000	13137.000000
max	6.000000	23.833333	23.916667	43.416667	8.000000	8.000000	3.000000	4.000000	29274.000000

- ## Data Inputs- Logic- Output Relationships

The input data were initially all object datatype so had to clean the data by removing unwanted information like "h" and "m" from Flight_Duration column and ensuring the numeric data are converted accordingly. I then used Ordinal Encoding method to convert all the categorical feature columns to numeric format.



```
: #converting categorical data into numeric values using OrdinalEncoder
  enc = OrdinalEncoder()
  for i in df.columns:
      if df[i].dtypes == "object" :
          df[i] = enc.fit_transform(df[i].values.reshape(-1,1))
```

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

- Hardware and Software Requirements and Tools Used

➢ Hardware Used:
RAM: 8 GB
CPU: AMD A8 Quad Core 2.2 Ghz
GPU: AMD Redon R5 Graphics

➢ Software Tools used:
Programming language: Python 3.0
Distribution: Anaconda Navigator
Browser-based language shell: Jupyter Notebook

➢ Libraries/Packages Used:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Scipy.stats
- Sklearn

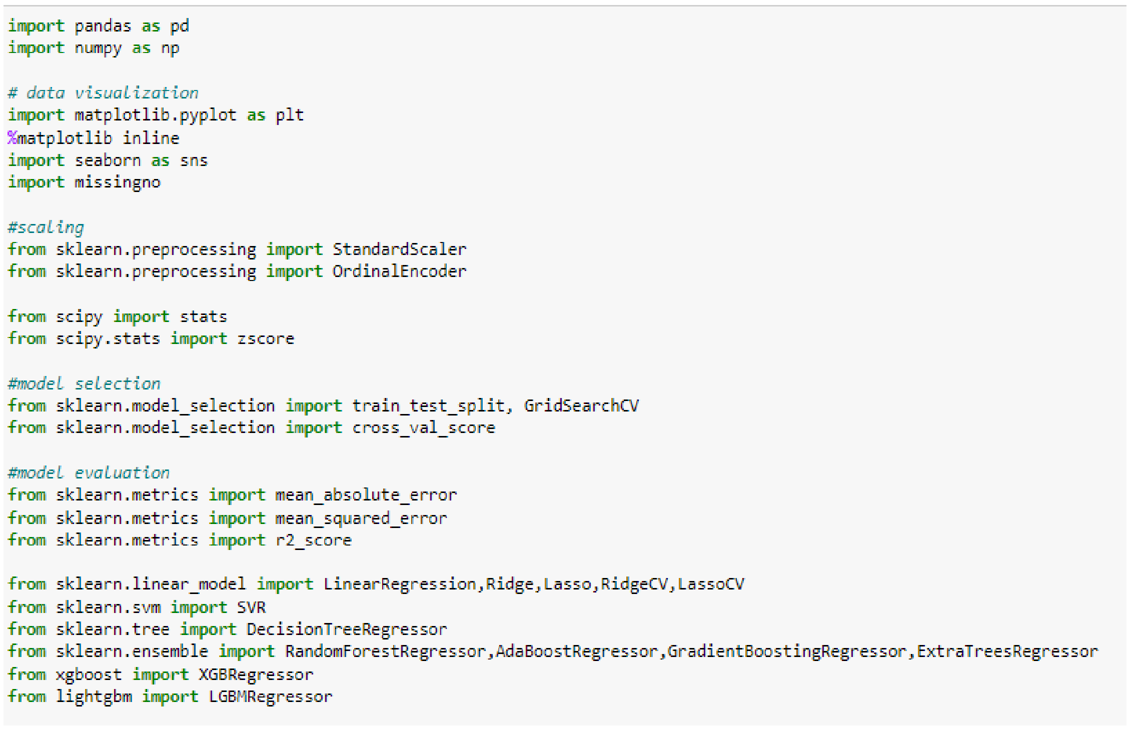
Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

1. Cleaning the dataset from unwanted scraped details.
2. Renaming values with meaningful information.
3. Encoding the categorical data to get numerical input data.
4. Comparing different models and identify the suitable model.
5. R2 score is used as the primary evaluation metric.
6. MSE and RMSE are used as secondary metrics.
7. Cross Validation Score was used to ensure there is no over-fitting or under-fitting models.

- Testing of Identified Approaches (Algorithms)

Libraries and Machine Learning Regression models that were used in this project are shown below.



```python
import pandas as pd
import numpy as np

# data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import missingno

#scaling
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder

from scipy import stats
from scipy.stats import zscore

#model selection
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score

#model evaluation
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

from sklearn.linear_model import LinearRegression,Ridge,Lasso,RidgeCV,LassoCV
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor,ExtraTreesRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
```

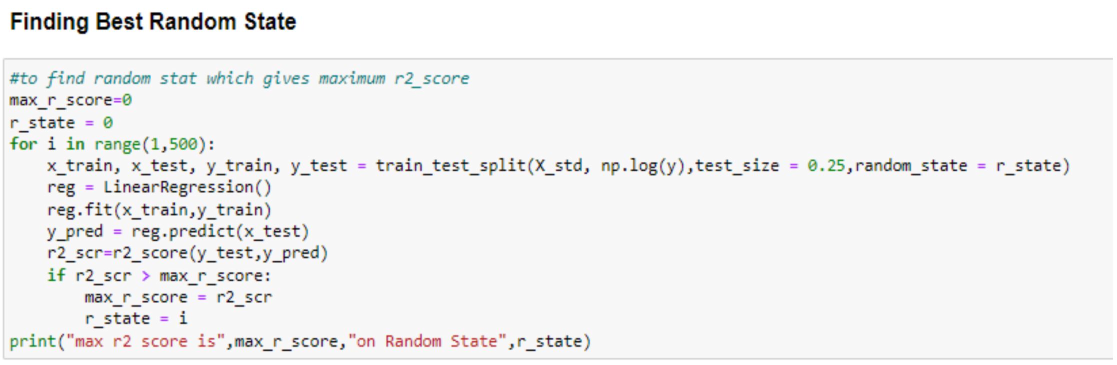
The list of all the algorithms used for the training and testing Regression model are listed below:

- Linear Regression Model
- Ridge Regularization Model
- Lasso Regularization Model
- Decision Tree Regression Model
- Random Forest Regression Model
- Gradient Boosting Regression Model
- Ada Boost Regression Model
- Extra Trees Regression Model
- XGB Regressor
- LGBM Regressor

From all of these above models, LGBMRegressor gave me good performance.

• Run and Evaluate selected models

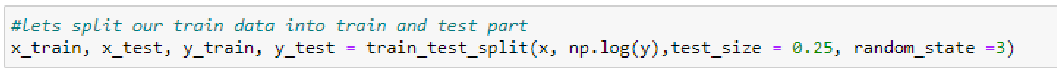
To perform training and testing operation(s) following functions has been defined for which codes are as follows:



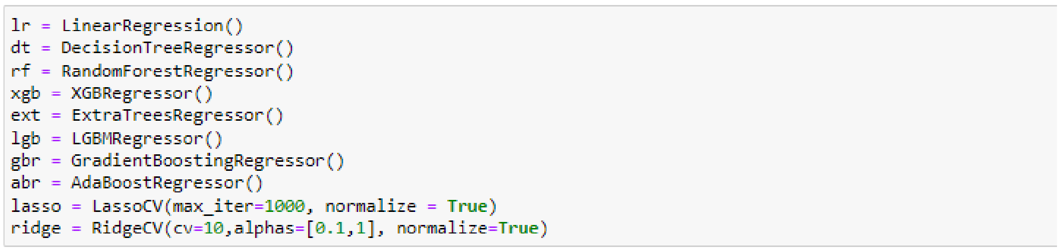
Finding Best Random State

```python
#to find random stat which gives maximum r2_score
max_r_score=0
r_state = 0
for i in range(1,500):
    x_train, x_test, y_train, y_test = train_test_split(X_std, np.log(y),test_size = 0.25,random_state = r_state)
    reg = LinearRegression()
    reg.fit(x_train,y_train)
    y_pred = reg.predict(x_test)
    r2_scr=r2_score(y_test,y_pred)
    if r2_scr > max_r_score:
        max_r_score = r2_scr
        r_state = i
print("max r2 score is",max_r_score,"on Random State",r_state)
```

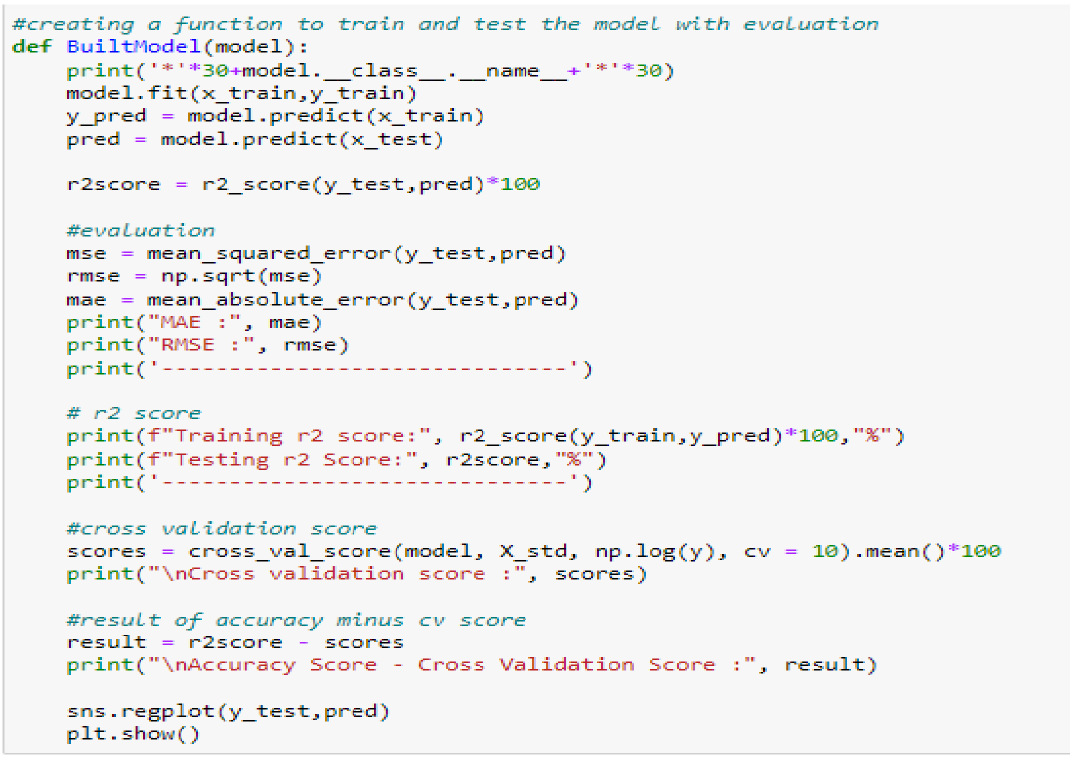
Building Machine Learning Model for Regression with Evaluation Metrics:



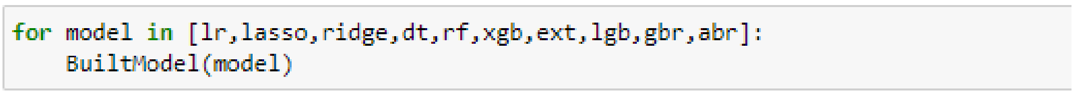
```python
#lets split our train data into train and test part
x_train, x_test, y_train, y_test = train_test_split(x, np.log(y),test_size = 0.25, random_state =3)
```



```python
lr = LinearRegression()
dt = DecisionTreeRegressor()
rf = RandomForestRegressor()
xgb = XGBRegressor()
ext = ExtraTreesRegressor()
lgb = LGBMRegressor()
gbr = GradientBoostingRegressor()
abr = AdaBoostRegressor()
lasso = LassoCV(max_iter=1000, normalize = True)
ridge = RidgeCV(cv=10,alphas=[0.1,1], normalize=True)
```

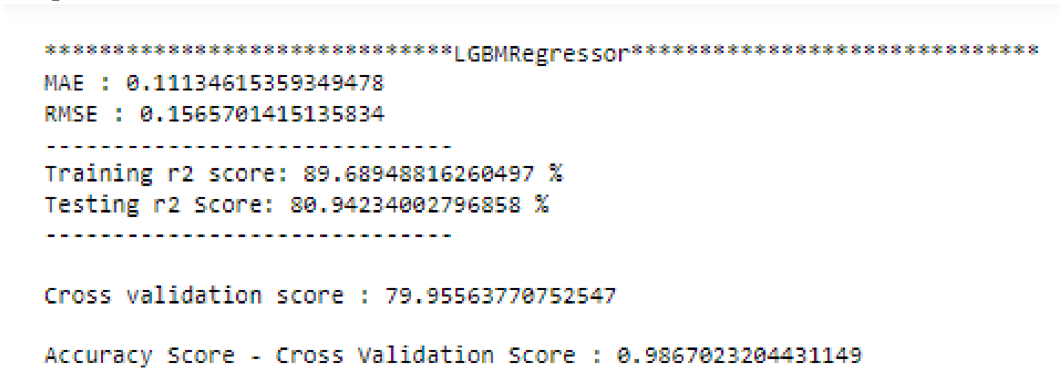


```python
#creating a function to train and test the model with evaluation
def BuiltModel(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    model.fit(x_train,y_train)
    y_pred = model.predict(x_train)
    pred = model.predict(x_test)

    r2score = r2_score(y_test,pred)*100

    #evaluation
    mse = mean_squared_error(y_test,pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test,pred)
    print("MAE :", mae)
    print("RMSE :", rmse)
    print('------------------------------')

    # r2 score
    print(f"Training r2 score:", r2_score(y_train,y_pred)*100,"%")
    print(f"Testing r2 Score:", r2score,"%")
    print('------------------------------')

    #cross validation score
    scores = cross_val_score(model, X_std, np.log(y), cv = 10).mean()*100
    print("\nCross validation score :", scores)

    #result of accuracy minus cv score
    result = r2score - scores
    print("\nAccuracy Score - Cross Validation Score :", result)

    sns.regplot(y_test,pred)
    plt.show()
```



```python
for model in [lr,lasso,ridge,dt,rf,xgb,ext,lgb,gbr,abr]:
    BuiltModel(model)
```

Output:

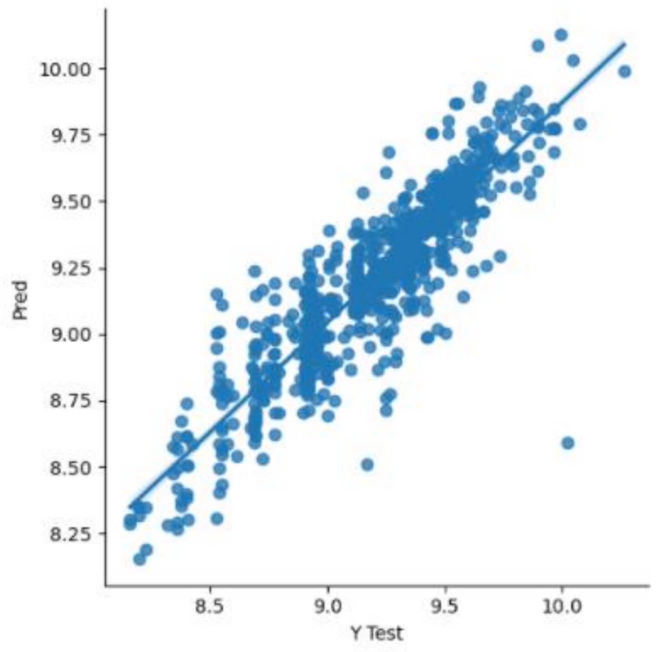


```
******************************LGBMRegressor******************************
MAE : 0.11134615359349478
RMSE : 0.1565701415135834
------------------------------
Training r2 score: 89.68948816260497 %
Testing r2 Score: 80.94234002796858 %
------------------------------

Cross validation score : 79.95563770752547

Accuracy Score - Cross Validation Score : 0.9867023204431149
```

From the model performance comparison, I found LGBMRegressor is showing good performance and less difference in r2-score and cv-score so I selected LGBMRegressor as our best suitable algorithm for our final model.

Plot of actual vs predicted values

```
data = pd.DataFrame({'Y Test':y_test , 'Pred':pred},columns=['Y Test','Pred'])
sns.lmplot(x='Y Test',y='Pred',data=data,palette='rainbow')
plt.show()
```



Saving the best model:

```
import joblib
joblib.dump(model,"Flight_Price_Prediction.pkl")
```

```
['Flight_Price_Prediction.pkl']
```

- Key Metrics for success in solving problem under consideration:

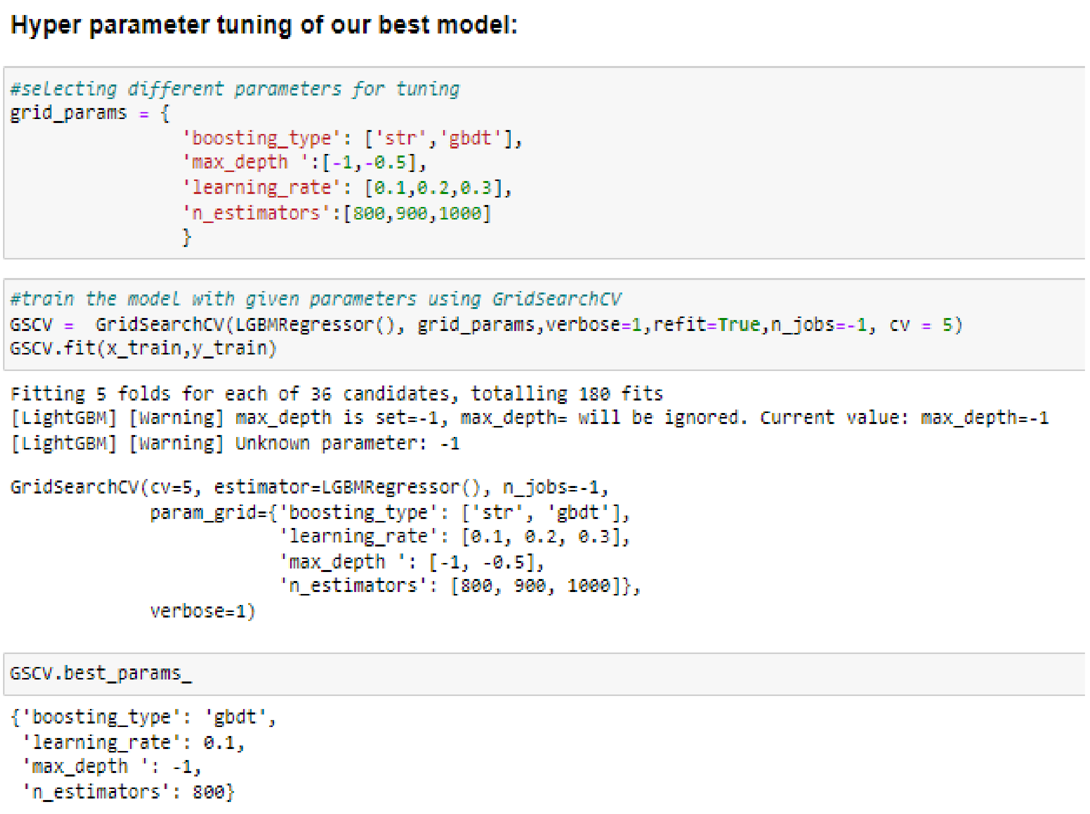
To find out best performing model, the following metrices are used:

- RMSE Score: Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.
- R2 Score: The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R

squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

- Cross Validation Score: Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are common tactics that you can use to select the value of k for your dataset (I have used 5-fold validation in this project). There are commonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn.
- Hyper Parameter Tuning: In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

Code:



Hyper parameter tuning of our best model:

```
#selecting different parameters for tuning
grid_params = {
                'boosting_type': ['str','gbdt'],
                'max_depth ':[-1,-0.5],
                'learning_rate': [0.1,0.2,0.3],
                'n_estimators':[800,900,1000]
                }
```

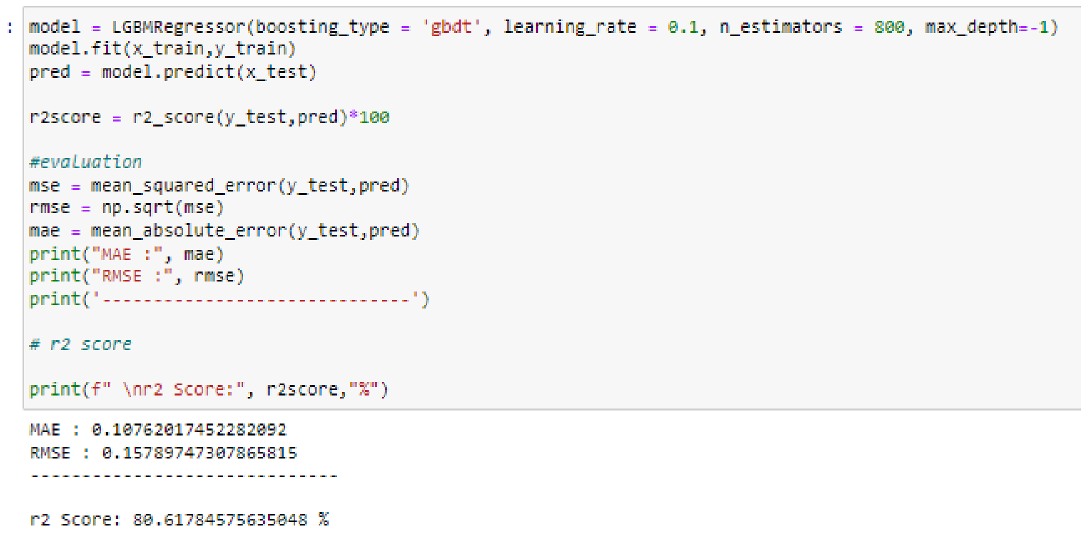
```
#train the model with given parameters using GridSearchCV
GSCV =  GridSearchCV(LGBMRegressor(), grid_params,verbose=1,refit=True,n_jobs=-1, cv = 5)
GSCV.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
[LightGBM] [Warning] max_depth is set=-1, max_depth= will be ignored. Current value: max_depth=-1
[LightGBM] [Warning] Unknown parameter: -1

GridSearchCV(cv=5, estimator=LGBMRegressor(), n_jobs=-1,
             param_grid={'boosting_type': ['str', 'gbdt'],
                         'learning_rate': [0.1, 0.2, 0.3],
                         'max_depth ': [-1, -0.5],
                         'n_estimators': [800, 900, 1000]},
             verbose=1)
```

```
GSCV.best_params_
```

```
{'boosting_type': 'gbdt',
 'learning_rate': 0.1,
 'max_depth ': -1,
 'n_estimators': 800}
```

Final model score after plugging in the best parameter values:



```
model = LGBMRegressor(boosting_type = 'gbdt', learning_rate = 0.1, n_estimators = 800, max_depth=-1)
model.fit(x_train,y_train)
pred = model.predict(x_test)

r2score = r2_score(y_test,pred)*100

#evaluation
mse = mean_squared_error(y_test,pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test,pred)
print("MAE :", mae)
print("RMSE :", rmse)
print('-----------------------------')

# r2 score

print(f" \nr2 Score:", r2score,"%")
```

```
MAE : 0.10762017452282092
RMSE : 0.15789747307865815
-----------------------------

r2 Score: 80.61784575635048 %
```

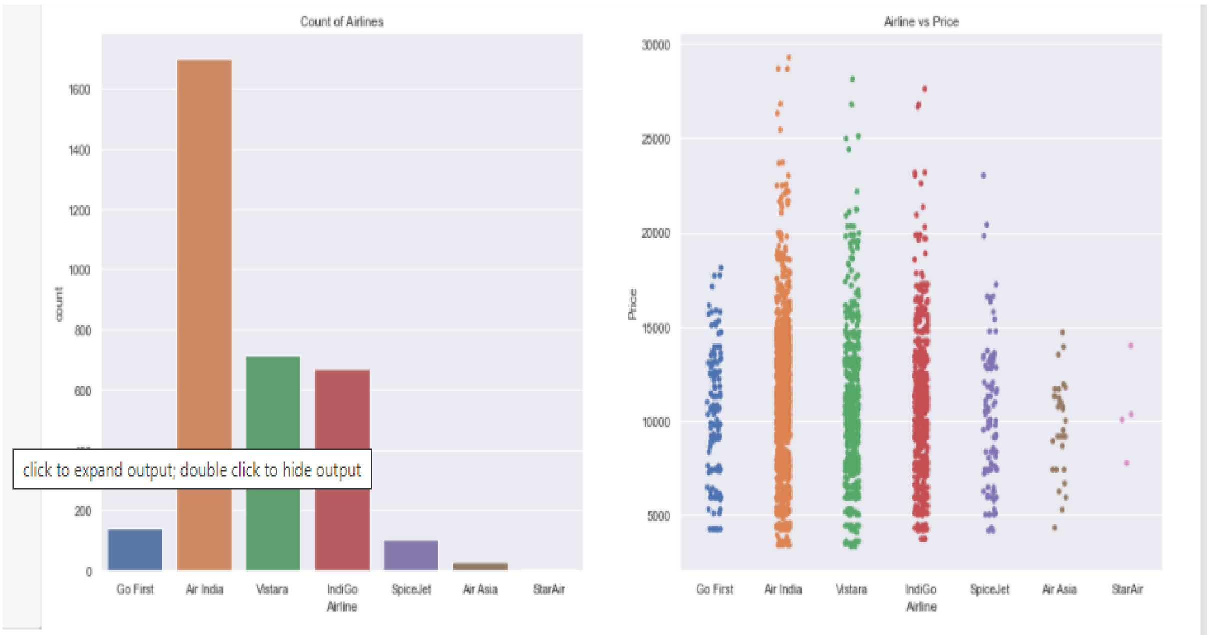
- Visualizations:

To better understand the data, the following types of visualizations have been used.

➢ Univariate analysis is the simplest form of data analysis where the data being analyzed contains only one variable. In this project, distribution plot, count plot, and bar plot has been used.

Count Plot:

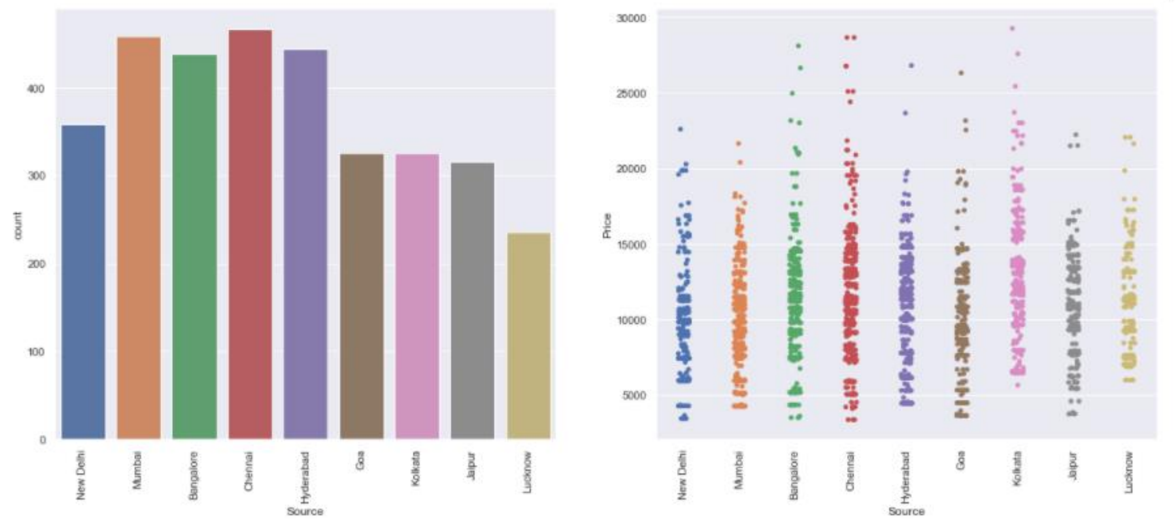
1.



Observations:

- The count of Air India airline is highest, followed by Vistara and then Indigo.
- The count of airline is lowest for StarAir followed by Air Asia.
- From the Airline Vs Price plot, we see that the flight price for Air India is highest, which is followed by Vistara, Indigo and then SpiceJet.
- The flight price is least for StarAir airline and then Air Asia.

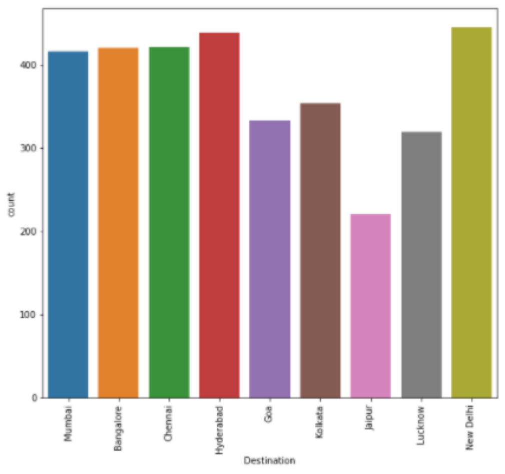
2.

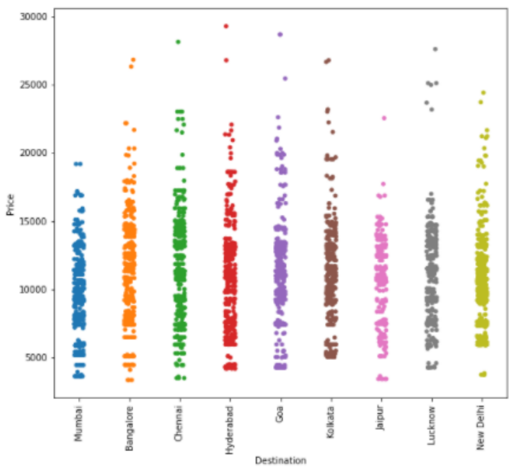


Observations:

- Majority of flights travel from source Chennai which is followed by Mumbai, Hyderabad and then Bangalore.
- Other locations from which people like to fly are New Delhi, Goa, Kolkata and Jaipur.
- From the count plot, it seems that less people fly from Lucknow.
- From the source vs price plot, we can see that flight price is the highest from Kolkata, which is followed by Chennai, Bangalore, and then Hyderabad.
- The flight prices from New Delhi, Lucknow and Jaipur is almost similar.
- The least flight price is from Chennai, Bangalore, Delhi, Goa and Jaipur.

3.

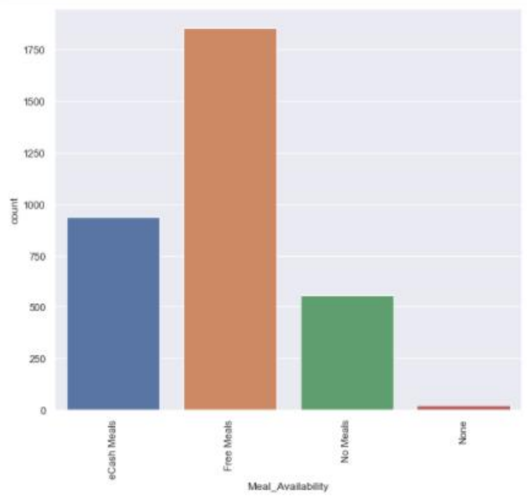


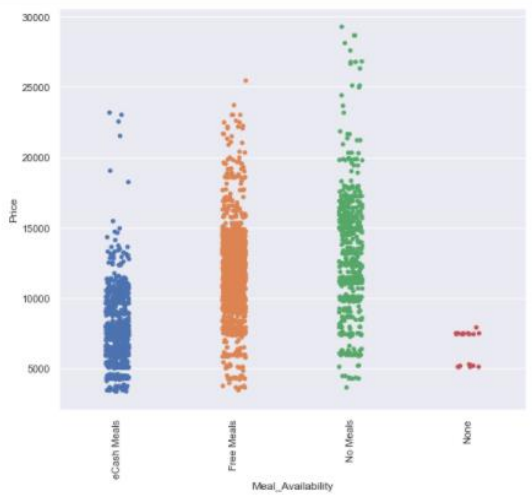


Observations:

- More number of flights fly to New Delhi and then to Hyderabad.
- The number of flights to Bangalore, Chennai and Mumbai is almost similar.
- The number of flights flying to Jaipur is lowest as compared to the other locations.
- Flight price is highest to travel to Hyderabad, then to Goa, Chennai and Lucknow.
- To travel to Bangalore and Kolkata, the flight price is similar.
- Prices are low for flights flying to Mumbai, and Jaipur.

4.



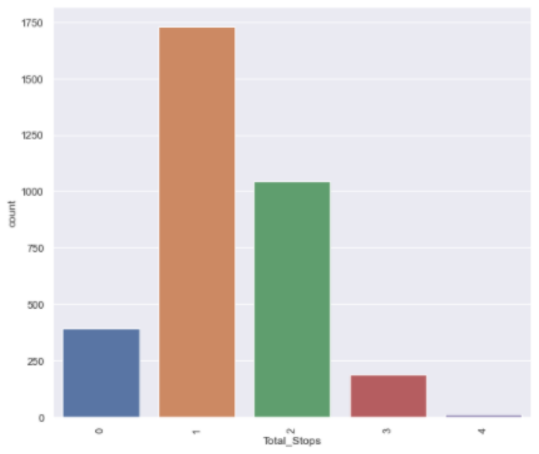


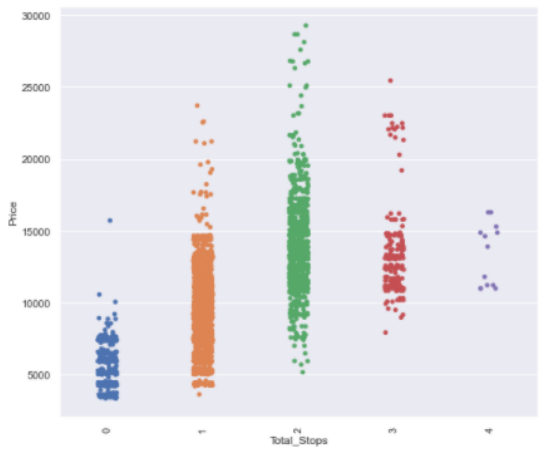
Observations:

- From the above plot, we can see that more number of flights offering free meals which are probably for tickets that include those prices and meal services.
- Next, we can see that flights are offering the eCash meals option that can be redeemed to purchase food during long journey flights, mostly with multiple stops.
- Lastly, there are flights that are not offering any meals which may be because they are flying short distances and duration.

- From the Meal_Availability vs Price plot, we can conclude that Flight prices is higher for flights offering no meals, which is followed by Flights offering free meals, and lastly flights offering eCash meals.

5.

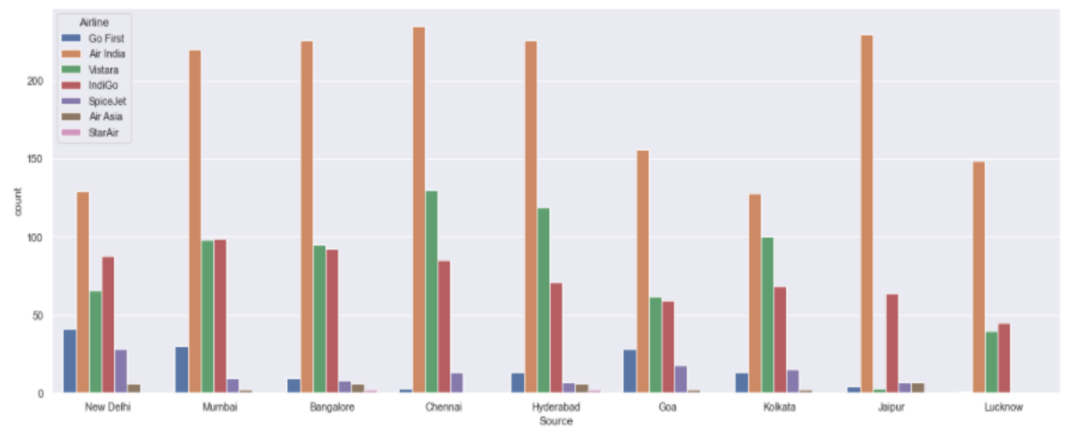




Observations:

- Higher number of People are buying flight tickets that have 1 stop layover.
- Next, we see that people buys flight tickets having 2 stops, which is followed by people getting non-stop flight tickets.
- In domestic flights we rarely see 3 or 4 stops, hence the number of stops is very less in this case.
- The flight price is the highest for flights having 2 stops, followed by flights having 3 and 1 stops.
- The Non-stop flights tickets price is the least.

6.

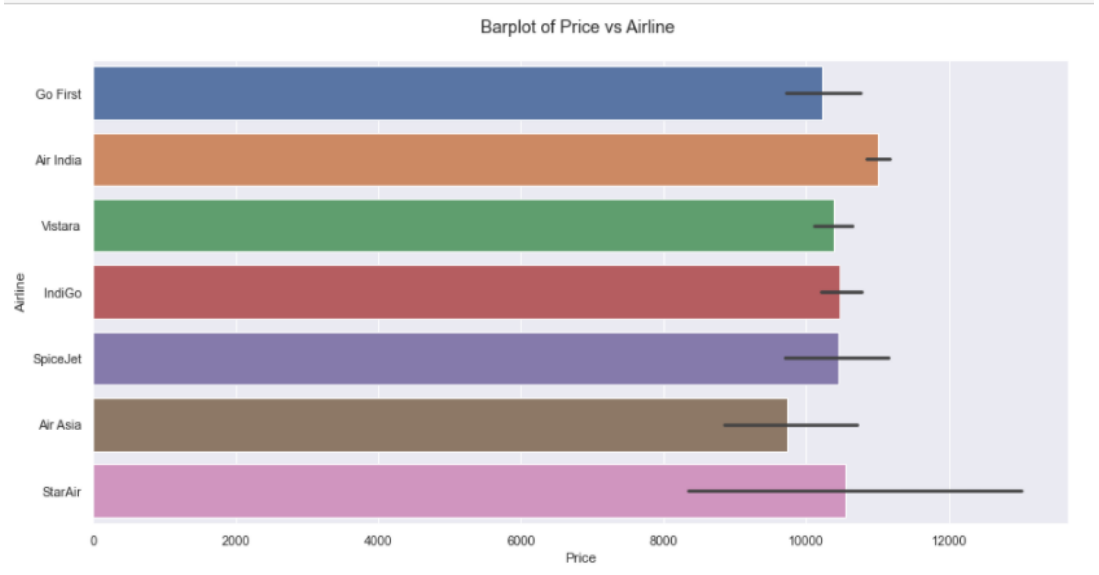


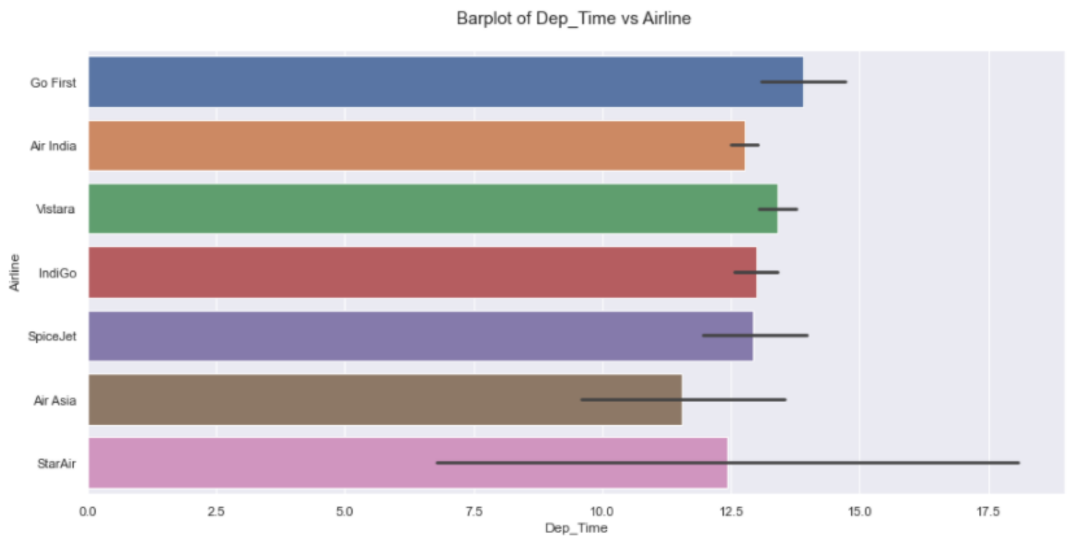
Observations:

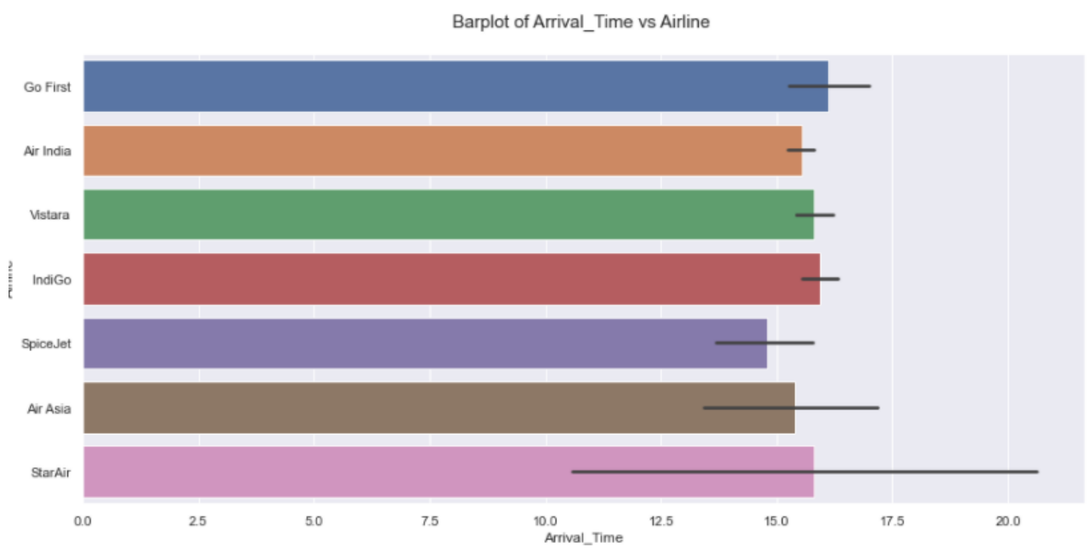
- The above plot is showing the region wise count of Airlines that shows that Jaipur is not having any flight of Vistara and the city Lucknow is not having any flight of SpiceJet,Air Asia, StarAir, and Go First.
- All cities are having higher count of flight of Air India compared to other flights

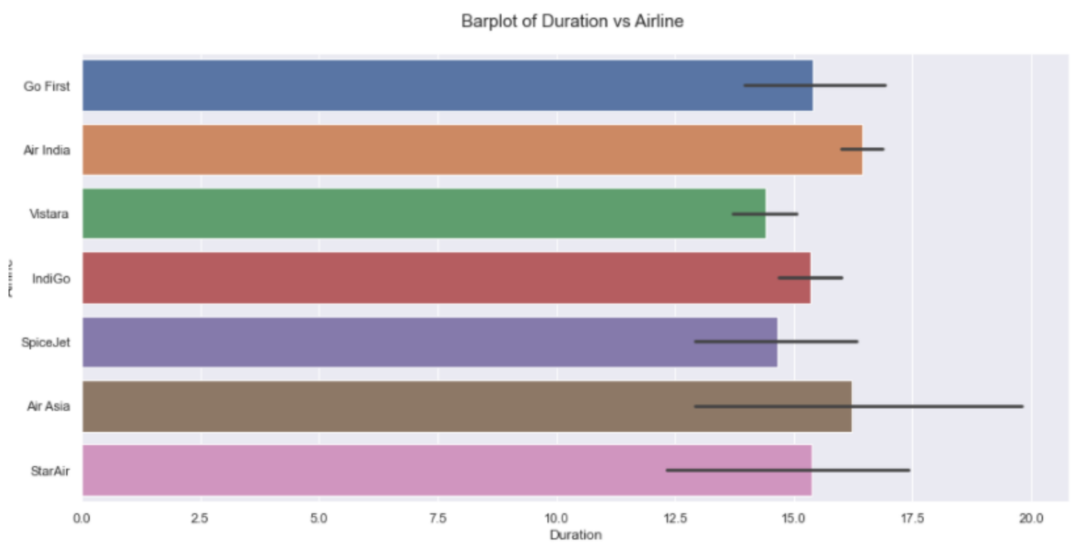
Bar Plots:

1.





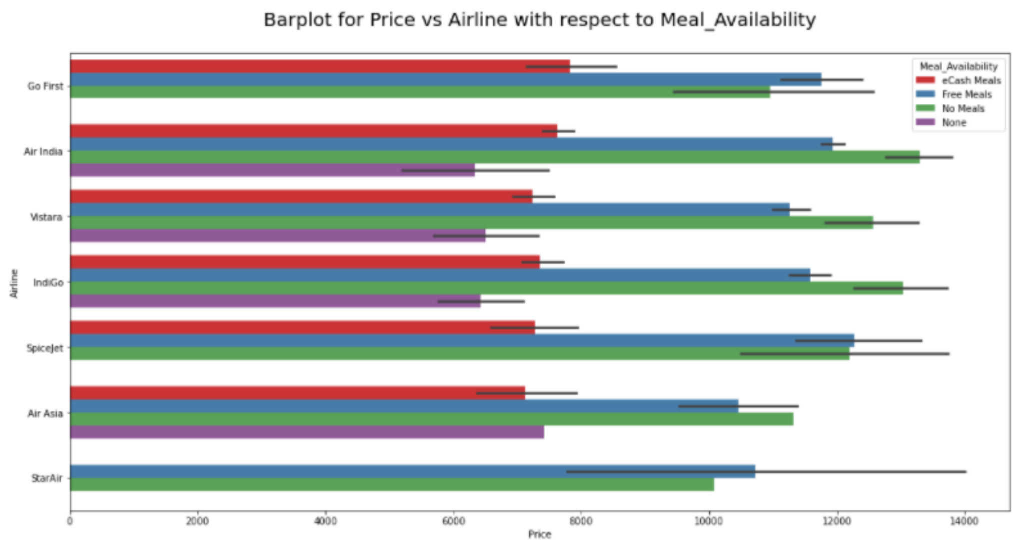


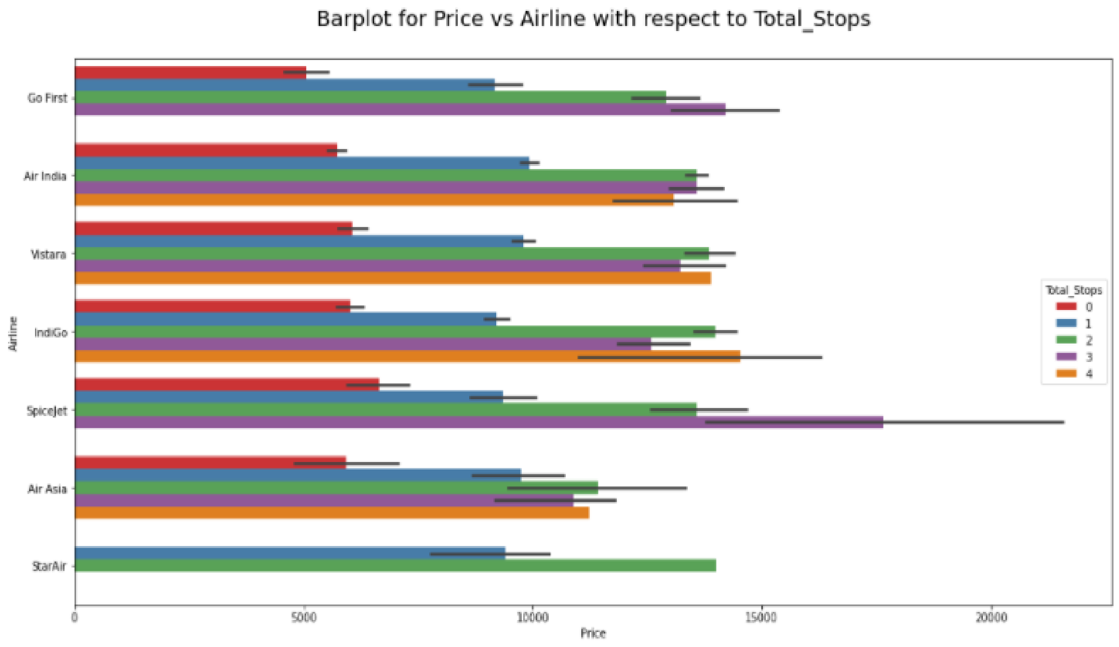


Observations:

- Comparing the barplots for Flight Prices vs Airline. we can clearly see that Air India has the highest flight prices while the other airlines like IndiGo, Vistara, SpiceJet, Go First, and StarAir lies in the similar price range, whereas Air Asia has the lowest fare among all.
- When we observe the barplot for Departure time vs Airline we can see that Go First takes the highest departure time while Air Asia takes the least departure time.
- Considering the barplot for Arrival time vs Airline we can see that Go First and StarAir collectively takes the highest arrival time while SpiceJet takes the least time to arrive at the destination.
- Looking at the barplot for Duration vs Airline, we observe that Air India and Air Asia takes the highest flight duration while Indigo and Vistara takes the lowest flight duration.

2.

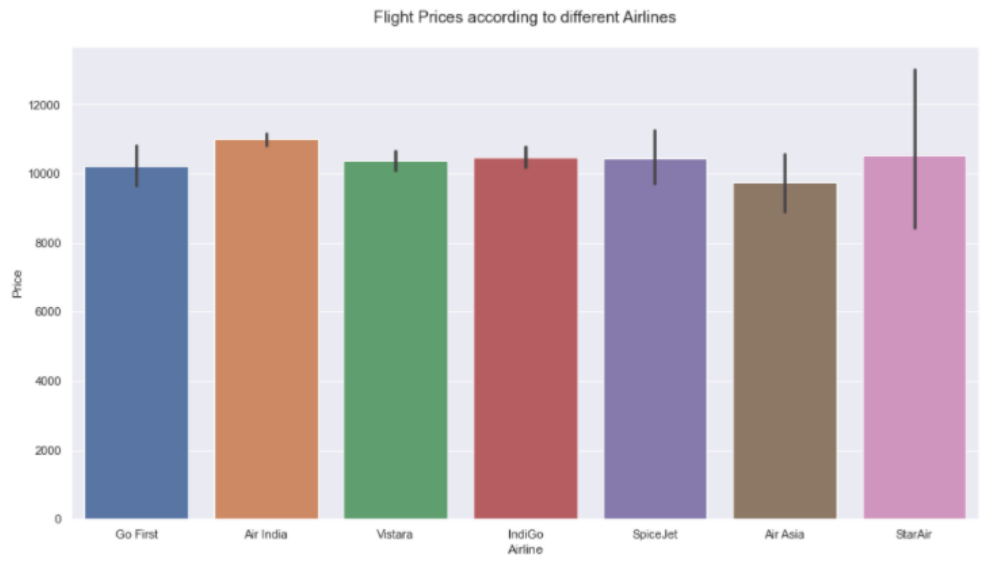


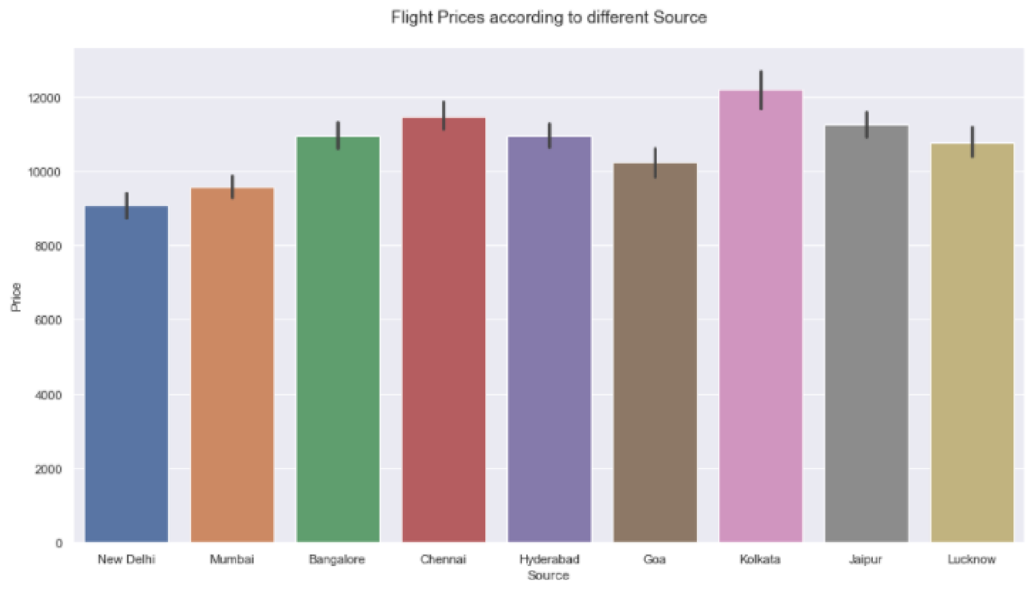


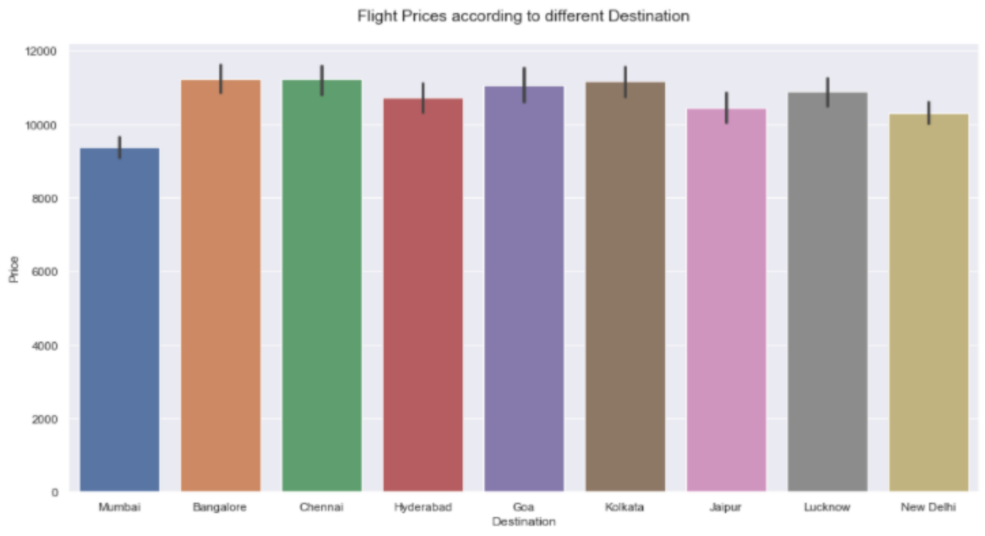
Observations:

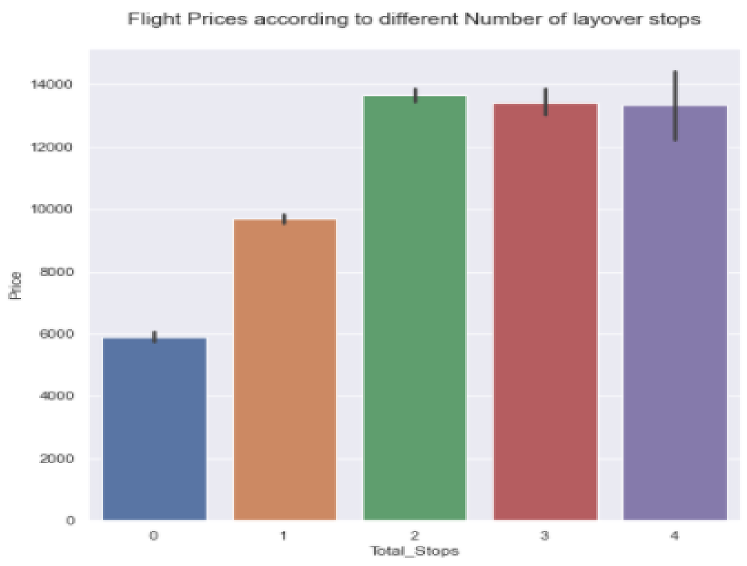
- SpiceJet is having the highest free meal service on their flight as compared to all other flights. While Air India and Indigo is having the highest no meal option on their flights, but these flights are having high ticket price.
- The eCash meal service is not available for StarAir flights.
- As it can see from the second plot that the flights with 0 stops or rather direct flights for every airline is cheaper as compared to 1 or more stops.
- Next, flights with 2 and 3 stops have a considerably high price and number of flights available in those records are high too.
- Except for SpiceJet, StarAir and Go First Airline, all other airlines are having 4 layover stops even in a domestic environment.
- StarAir has only flights available with 1 and 2 stops.

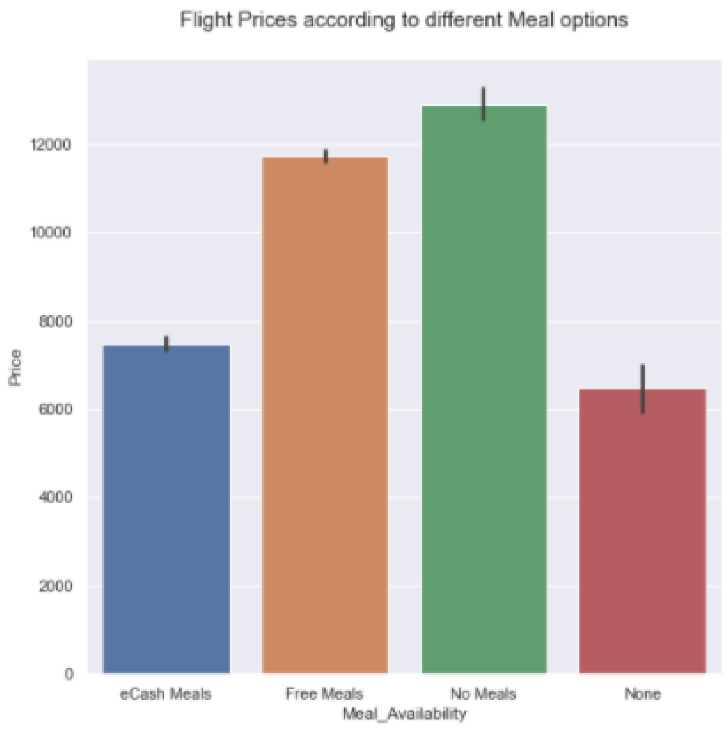
3.


Flight Prices according to different Airlines
Price
0
2000
4000
6000
8000
10000
12000
Go First
Air India
Vistara
IndiGo
SpiceJet
Air Asia
StarAir
Airline


Flight Prices according to different Source
Price
0
2000
4000
6000
8000
10000
12000
New Delhi
Mumbai
Bangalore
Chennai
Hyderabad
Goa
Kolkata
Jaipur
Lucknow
Source


Flight Prices according to different Destination
Price
0
2000
4000
6000
8000
10000
12000
Mumbai
Bangalore
Chennai
Hyderabad
Goa
Kolkata
Jaipur
Lucknow
New Delhi
Destination





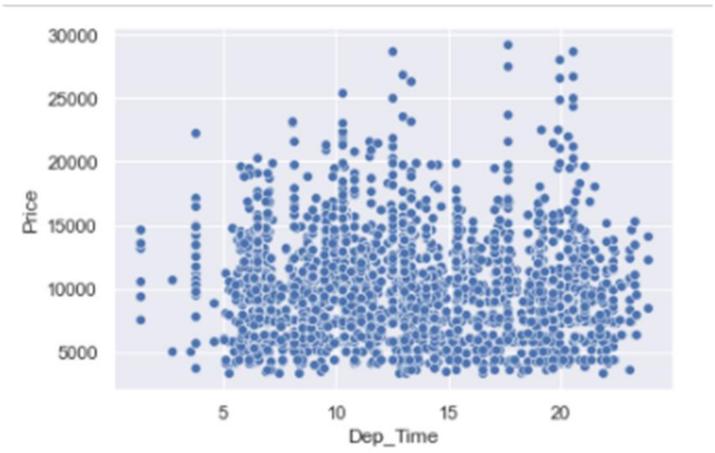
Observations:

- Airfare in Air India is quite high as compared to other airlines as they provide free meal service which probably is just meal cost included in the tickets.
- Airfare is almost similar for other remaining Airline while Air Asia is having the least fare.
- Flight prices when departing from cities like Kolkata, Jaipur and Chennai is higher, while the others flights are having somewhat similar flight prices except for New Delhi which is having the least fare.
- Similarly prices when arriving from cities like Kolkata, Bangalore, and Chennai is higher, however the lowest price is for Mumbai city, and the rest falls in similar price range.

- From the barplot of Total_Stops, we can see that the flight prices are higher for flights having 2,3 and 4 Stops, Whereas prices are cheaper for direct flights and flights having a single stop.
- As per the data collected, we see that no meals service flight tickets cost the highest as compared to flights with free meals service; and the lowest price observed are for eCash meal service wherein the frequent flyers can use their flight points as discounts on the ticket prices.

Scatter Plots:

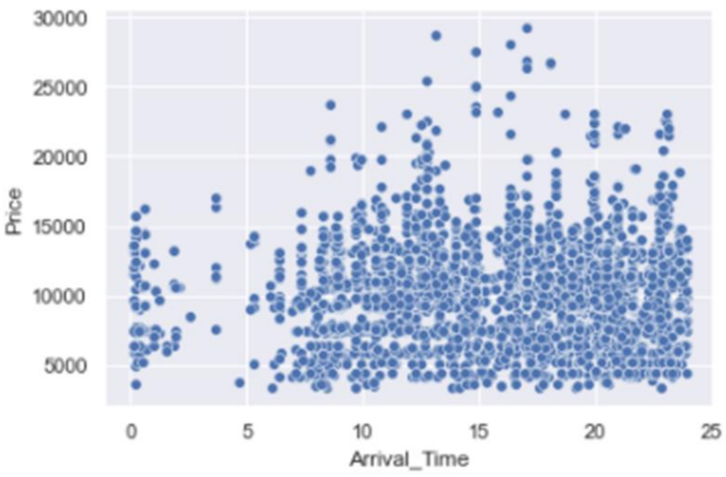
1.



Observation:

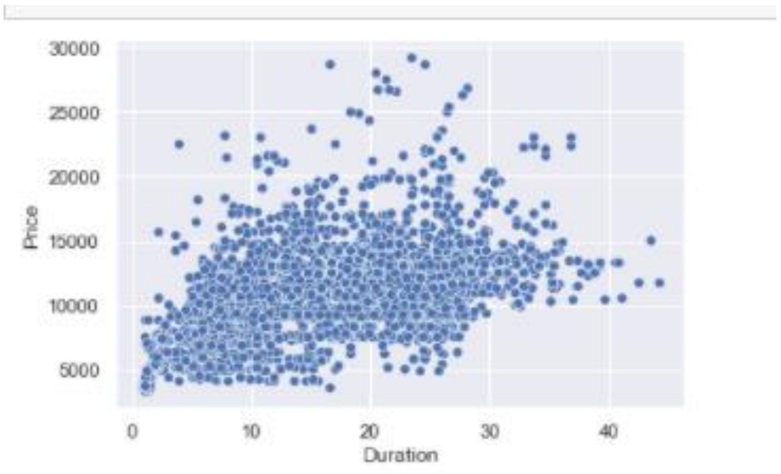
- Above scatter plot is showing the relationship between Departure time and flight prices. We can observe that there are very few flights departing in the early morning which are having lower price as well.

2.



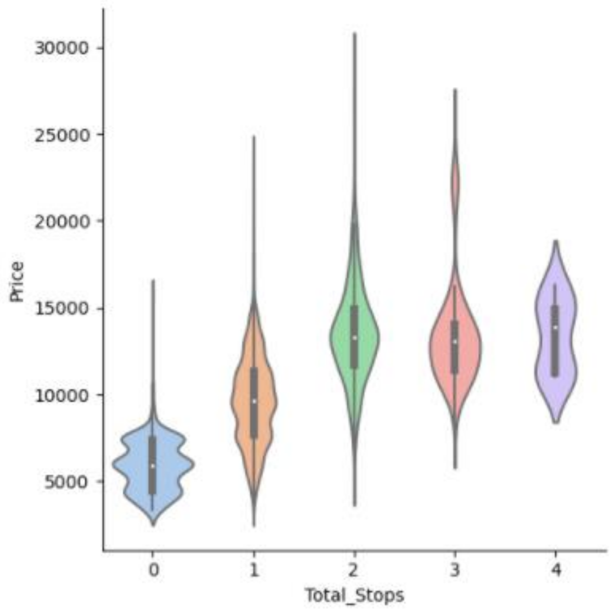
Observation: The above scatter plot is showing relation between Time of arrival and flight prices, which shows that a few number of flights are arriving in the early morning around 0 to 5 am. As we can see in the plot, the flight prices are not much dependent on the time of arrival.

3.



Observation: From the above plot, we see that as the overall flight duration increases, the flight prices increases too and that makes direct flight depart and arrive in a relatively shorter period of time.

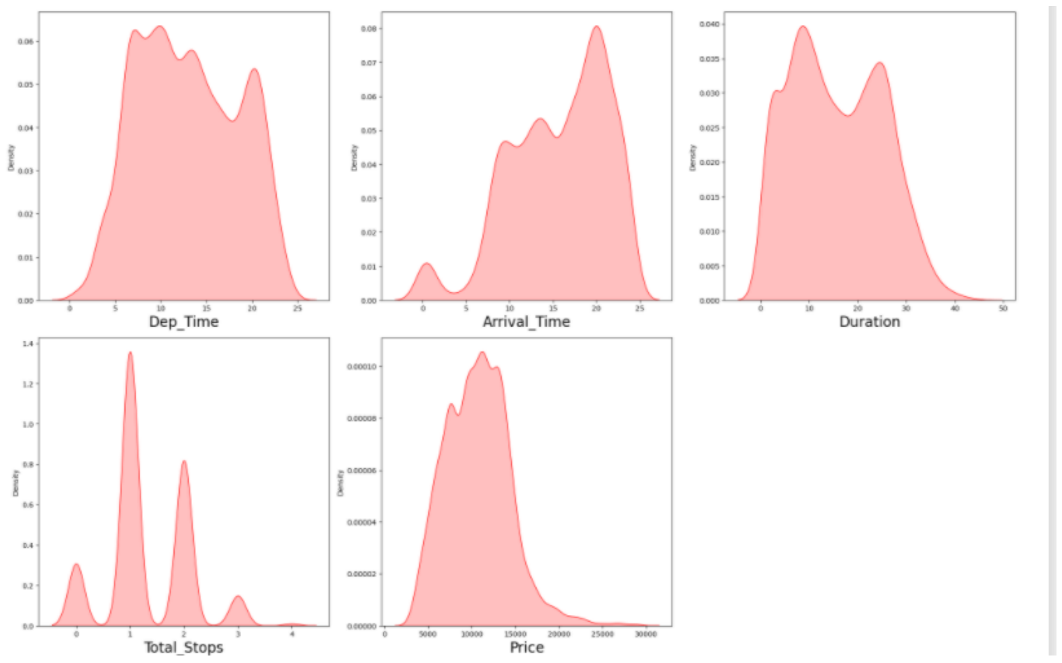
<u>Violin Plot:</u>



Observation: From the above violin plot, we can conclude that the Price for flights having stops more than 1 costs higher than the flights with stops 1 or none.

Distribution Plot:

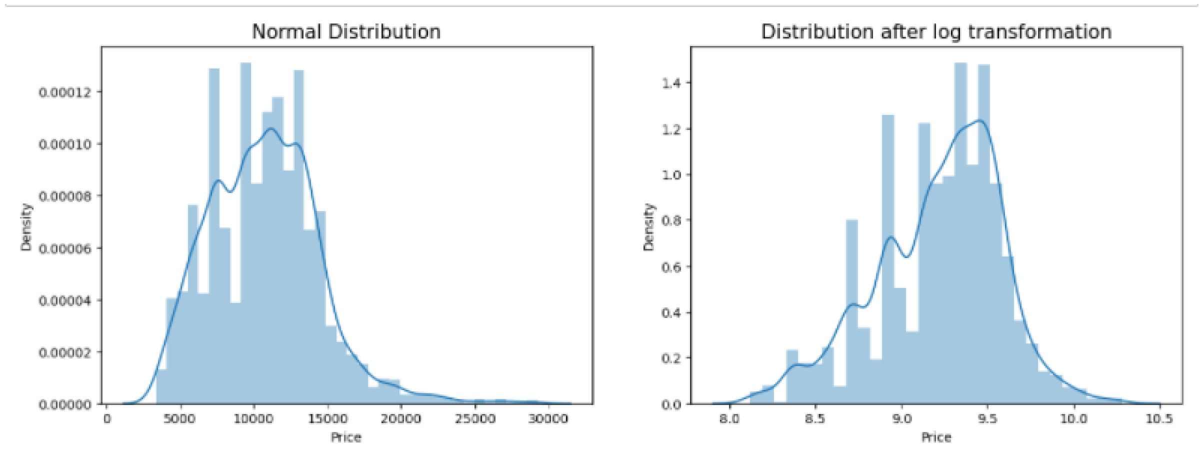
1.



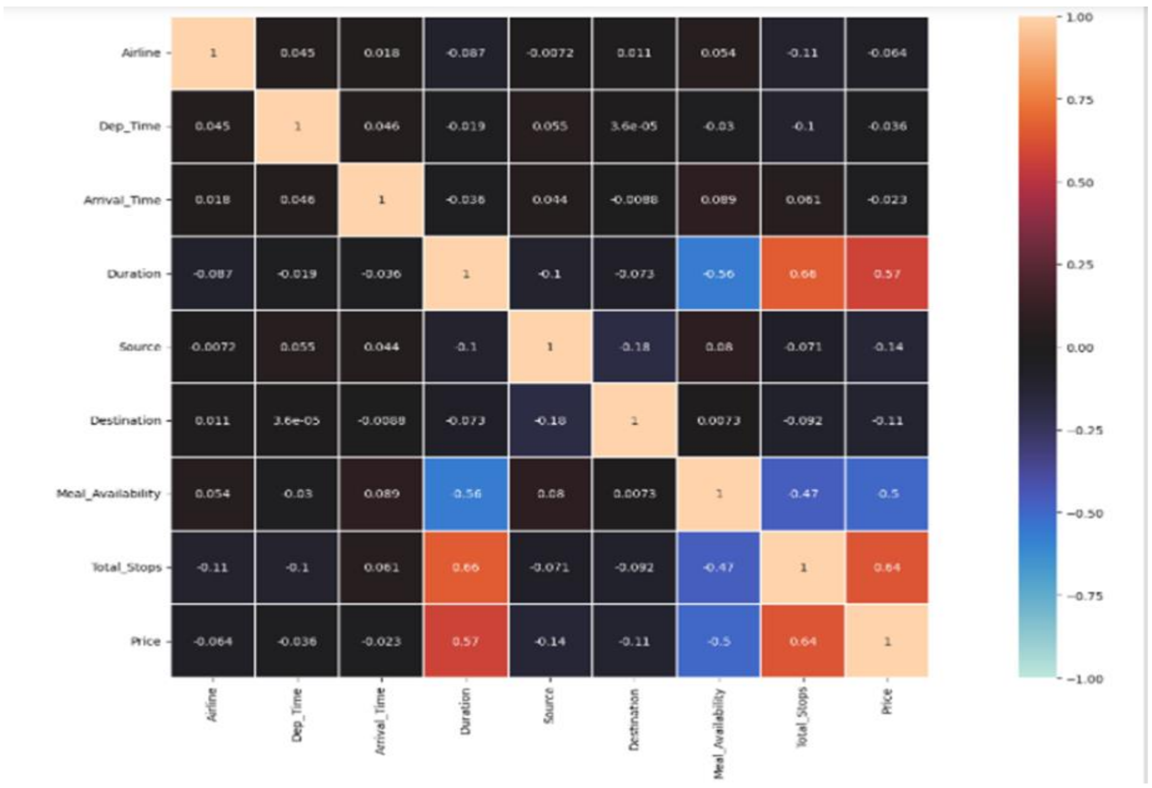
Observation: Looking at the distribution of the features, we can see that there are some outliers present in our data and data is skewed too. Hence we have to remove both skewness and outliers from our data.

2.

Applying log transformation to our target variable:



Heat-Map:



Observation: From the above correlation heatmap, we can see that except for Total_Stops and Duration, all other features is having negative correlation with our target 'Price'.

- ## Interpretation of the Results

  From the above EDA we can easily understand the relationship between features and we can even see which things are affecting the price of flights. 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 have tried 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.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  - From the model performance comparison, it is clear that LGBMRegressor performs well with R2 score of 80.61% and lowest difference between accuracy_score and cross_val_score, hence I selected LGBMRegressor as our final model.
  - In this project we have scraped the flight data from airline webpage "yatra.com".

- Features like flight duration, number of stops during the journey and the availability of meals are playing major role in predicting the prices of the flights.
- It would help customers to predict future flight prices and plan the journey accordingly because it is difficult for airlines to maintain prices since it changes dynamically due to different conditions. Hence by using Machine Learning techniques we can solve this problem.
- The above research will help our client to study the latest flight price market and with the help of the model built he can easily predict the price ranges of the flight, and also will helps him to understand based on what factors the fight price is decided.

Learning Outcomes of the Study in respect of Data Science

- Visualization part helped to understand the data as it provides graphical representation of huge data.
- It assisted to understand the feature importance, outliers or skewness detection and to compare the independent-dependent features.
- Data cleaning is the most important part of model building and therefore before model building, I made sure the data is cleaned.
- I have generated multiple regression machine learning models to get the best model wherein I found LGBMRegressor Model being the best based on the metrics I have used.
- I ensured that at least I get a decent prediction confidence percentage.

Limitations of this work and Scope for Future Work

- Some algorithms are facing over-fitting problem which may be because of lesser number of features in our dataset.
- We can get a better r2 score by fetching some more features by web scraping which may help to reduce the over fitting problem in our models.
- Limitation of the study is that in the volatile changing market we have taken the data, to be more precise we have taken the data at the time of pandemic and recent data, so when the pandemic ends the market correction might happen slowly.
- Therefore based on that again the deciding factors of it may change and we have shortlisted and taken these data from the important cities across India.
- If the customer is from the different country our model might fail to predict the accuracy price of that flight.