Project report on:

Flight Price Prediction

Submitted by:

Suraj Kumar Soni

Internship Batch = 29

ACKNOWLEDGMENT

It is my sensual gratification to present this report on the FLIGHT PRICE PREDICTION project. Working on this project was a good experience that has given me very informative knowledge.

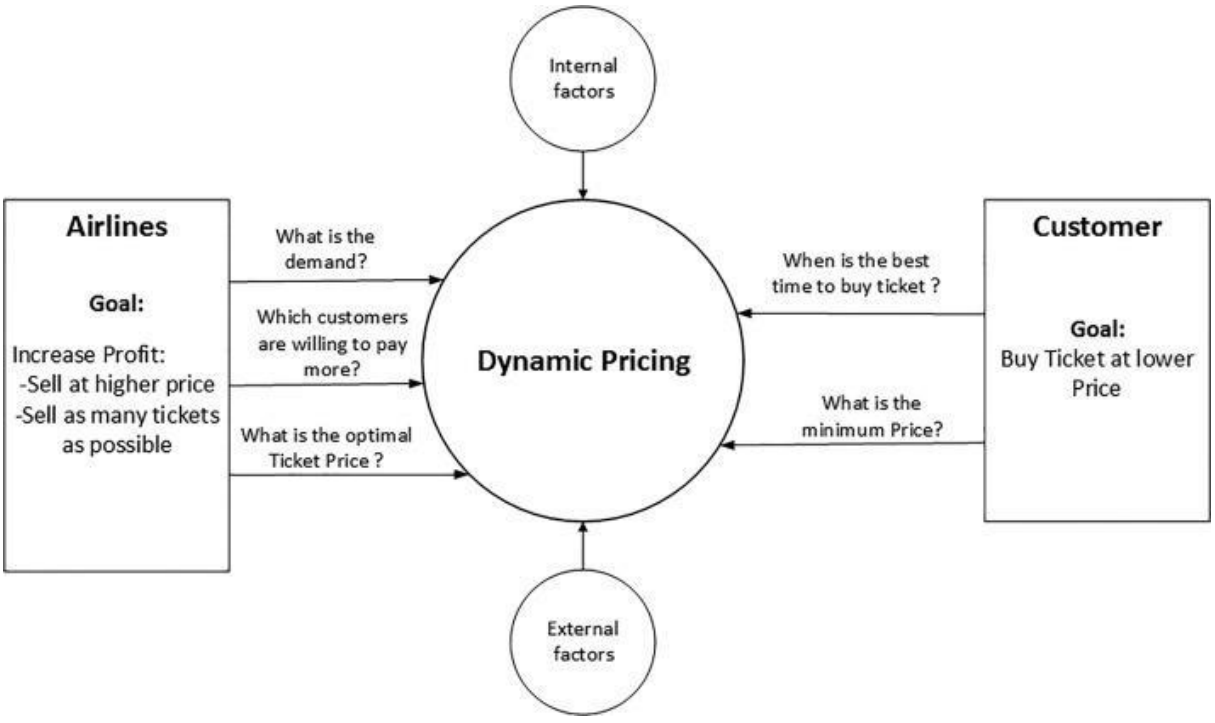
I would like to express my sincere thanks to Mr. Shwetank Mishra for a regular follow-up and valuable guidance provided throughout.

And I am also thankful to FlipRobo Technologies Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

INTRODUCTION

Business Problem Framing

The airline industry is considered one of the most sophisticated industries 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 lead 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.



The conceptual background of the domain problem

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 that is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

So, we have to work on a project where we will collect data on flight fares with other features and work to make a model to predict the fares of flights.

Analytical Problem Framing

Mathematical/Analytical modeling of the problem

For the given flight price prediction project I have scraped the flight prices along with some other features from a well-known website that is 'yatra.com'. And this data is framed into a data frame and saved into a .csv file. After that, I have fetched this data set and performed some data processing and some EDA. The data set is having around 5808 rows and 10 columns.

Loading the data

```
#lets import the dataset
df = pd.read_csv("Flight_prices.csv")
df
```

	Unnamed: 0	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price
0	0	Air Asia	12:40	20:15	7h 35m	New Delhi	Mumbai	No Meal Fare	1 Stop	5,953
1	1	Air Asia	11:55	20:15	8h 20m	New Delhi	Mumbai	No Meal Fare	1 Stop	5,953
2	2	Air Asia	16:15	06:20	14h 05m	New Delhi	Mumbai	No Meal Fare	1 Stop	5,953
3	3	Go First	18:50	20:45	1h 55m	New Delhi	Mumbai	No Meal Fare	Non Stop	5,954
4	4	Go First	09:05	11:05	2h 00m	New Delhi	Mumbai	No Meal Fare	Non Stop	5,954
...	...	...	...	...	...	...	...	...	...	...
5800	5800	Air India	08:55	08:20	23h 25m	Lucknow	Jaipur	No Meal Fare	1 Stop	9,302
5801	5801	Air India	08:55	09:20	24h 25m	Lucknow	Jaipur	No Meal Fare	2 Stop(s)	16,287
5802	5802	Air India	14:45	09:20	18h 35m	Lucknow	Jaipur	No Meal Fare	2 Stop(s)	16,865
5803	5803	Air India	08:55	09:20	24h 25m	Lucknow	Jaipur	No Meal Fare	2 Stop(s)	16,865
5804	5804	Air India	15:30	09:20	17h 50m	Lucknow	Jaipur	Free Meal	3 Stop(s)	19,749

5805 rows x 10 columns

```
#checking info about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5805 entries, 0 to 5804
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Unnamed: 0         5805 non-null   int64 
 1   Airline            5805 non-null   object
 2   Departure_time     5805 non-null   object
 3   Time_of_arrival    5805 non-null   object
 4   Duration           5805 non-null   object
 5   Source             5805 non-null   object
 6   Destination        5805 non-null   object
 7   Meal_availability  5805 non-null   object
 8   Number_of_stops    5805 non-null   object
 9   Price              5805 non-null   object
dtypes: int64(1), object(9)
memory usage: 453.6+ KB
```

Great, looking at the data information we can see we have one unwanted column named as 'Unnamed: 0' I will drop this column as it is not contributing to our predictions. And rest all columns are given as object data type which are required to convert into respective data types. And luckily we are not having any null values in our data set.

Data Processing

As we need to convert the data types of some feature for that I followed dome of the data processing steps.

Duration column:

```
#Extracting numerical data using 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
```

This column is having string entries in hours and minutes; I have separated hours and minutes into two different columns by splitting and then by using both of these columns filled the entries into Duration column as given in above figure.

Departure_time & Time_of_arrival:

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

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

As similar to the case of duration column these two columns are also having the time but in a string format. And by using above code steps I have fetched numerical values for the time and filled to respective columns. And the extra columns from these three cases have deleted from the data set.

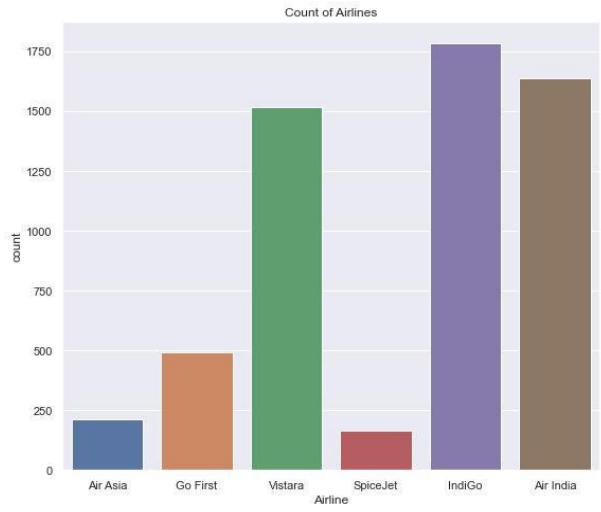
Number_of_stops:

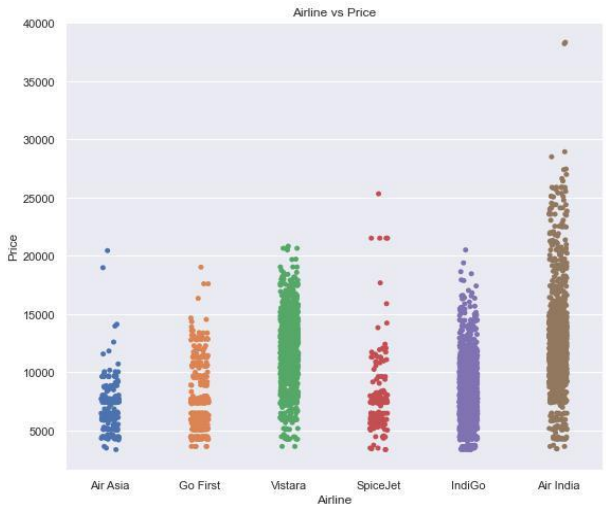
```
df.Number_of_stops.replace({"Non Stop": 0,
            "1 Stop": 1,
            "2 Stop(s)": 2,
            "3 Stop(s)": 3,
            "4 Stop(s)": 4},
           inplace = True)
```

This column 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.

Exploratory Data Analysis

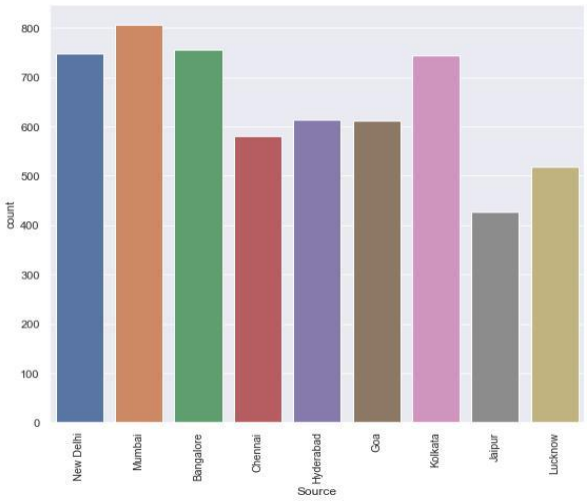
Airline:

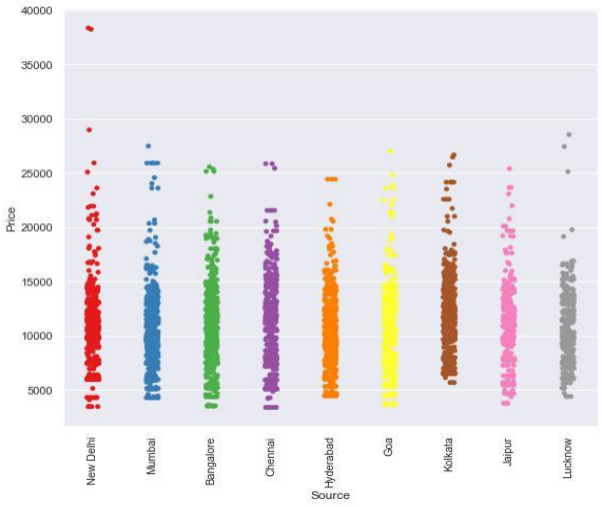




Above figure is showing two different plots one is count plot for the column Airline and another is a strip plot showing relation between Airline and Prices. The count plot will tell us that there are more numbers of flights of Vistara, IndiGo and Air India than others. Flights of Spice Jet are very less in numbers. Strip plot will tell us that Air India is having flights with higher prices.

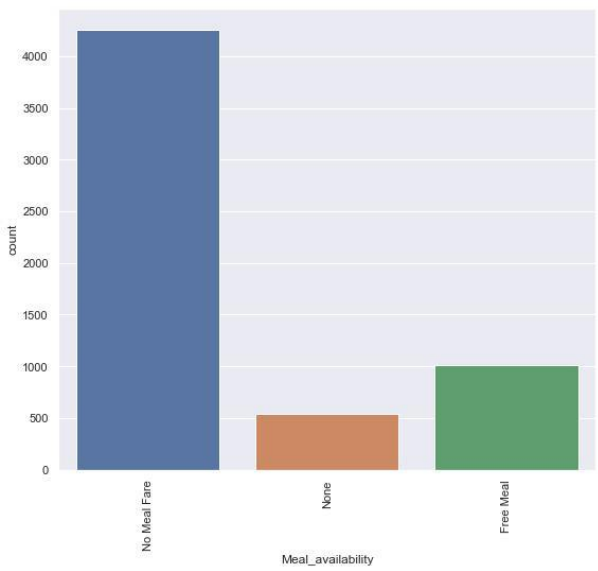
Source:

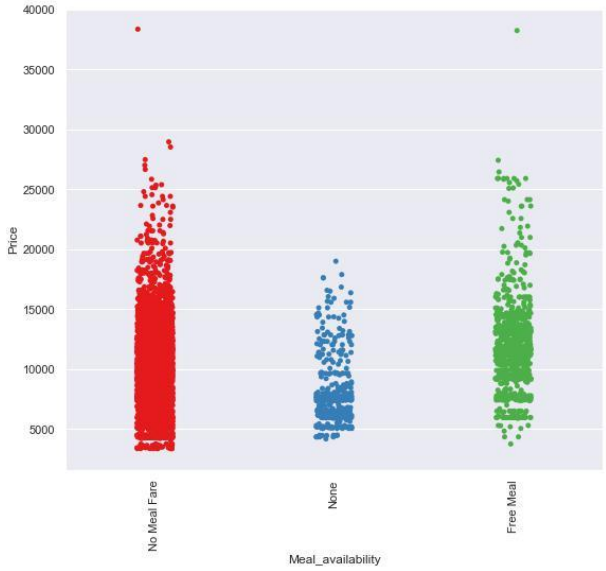




Here we are having more number of flights from New Delhi, Mumbai, Bangalore and Kolkata than other cities. Looking at the strip plot we can say flights from New Delhi are having somewhat higher prices than other cities.

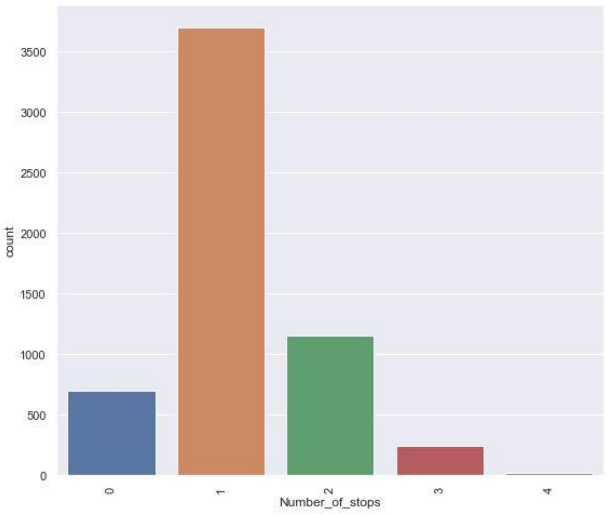
Meal_availability:

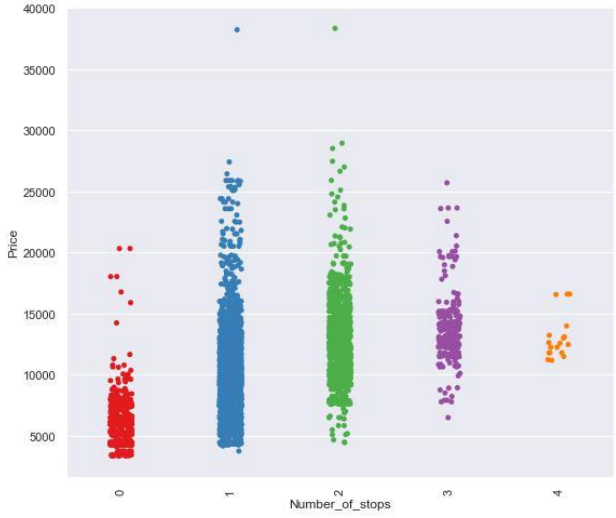




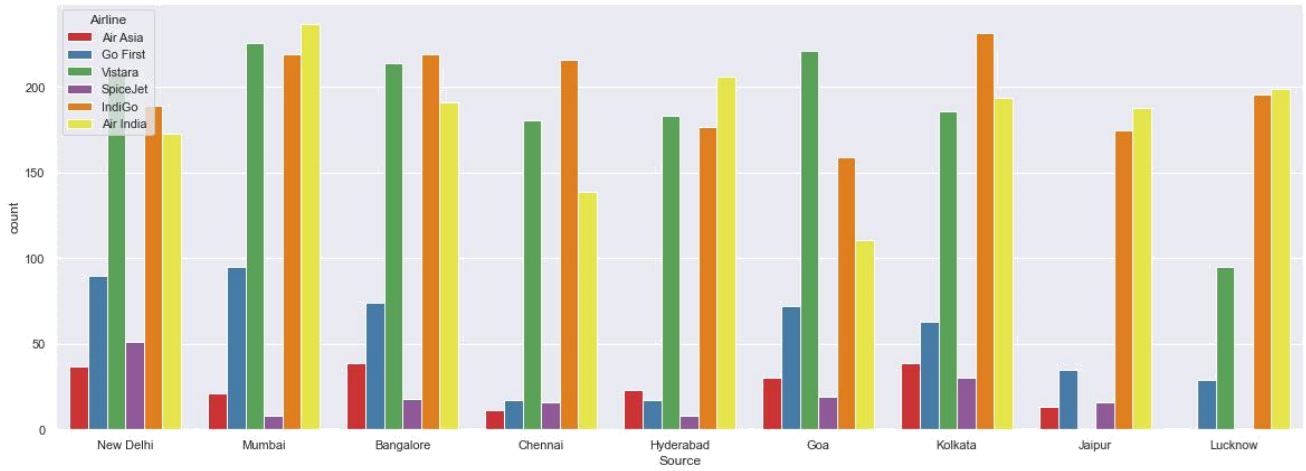
Looking at above plots we can conclude that more number of flights are not including Meal Fare, some of with providing free meal and rest few are with None category.

Number_of_stops:

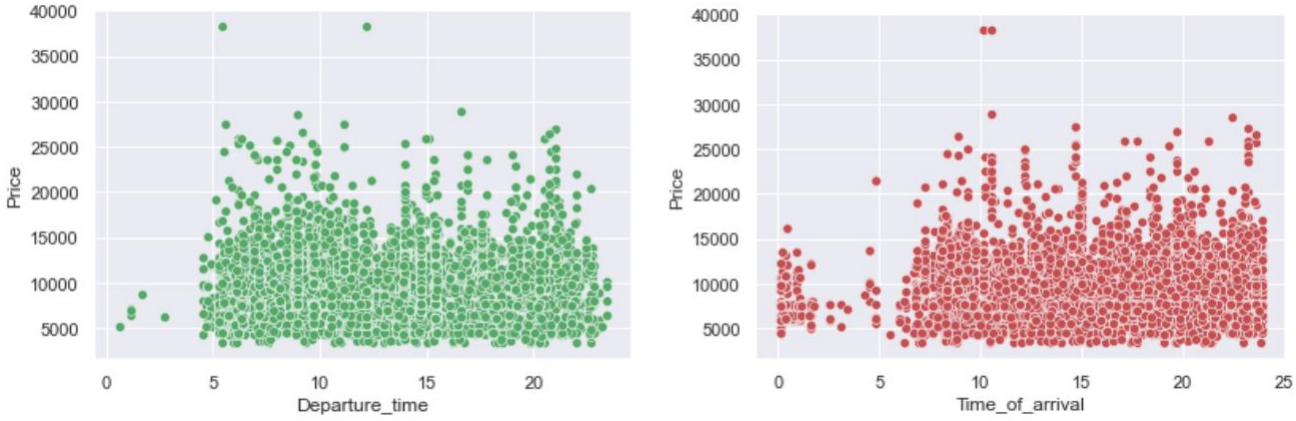




The above count plot will tell us that most of the flights are with 1 stop and very few are with 3 and 4 stops during the Journey. We can see that the prices are increasing with the number of stops.

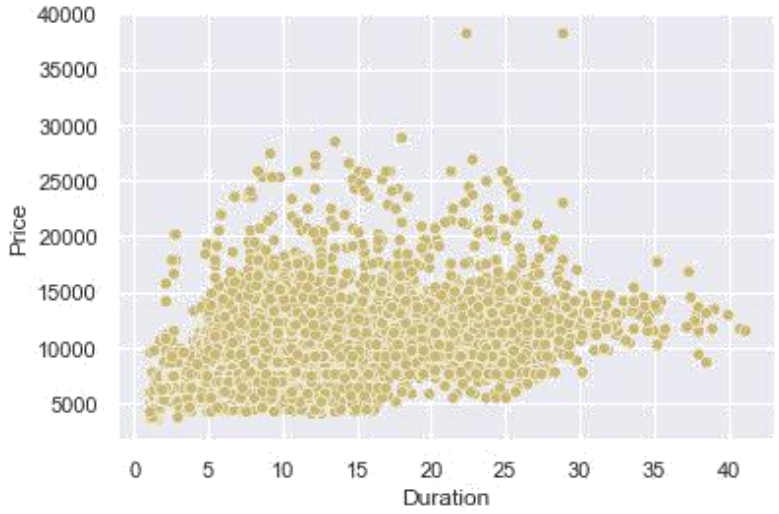


The above plot is showing the region wise count of airlines which will tell us that Jaipur is not having any flight of Vistara and the city Lucknow is not having any flights of Spice Jet. And all cities are having flights of IndiGo and Air India with higher count.



The first scatter plot is showing 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.

Second scatter plot is showing relation between Time of arrival and flight prices, which will tell us that very few numbers of flights are arriving in the early morning that is around 0 to 5 am. We can say the flight prices are not much dependent on the time of arrival.



The above figure is representing the scatter plot of Duration vs Price. Looking at this figure we can say that there is some linear relation between price and duration. The prices increase with duration.

Hardware and Software Requirements and Tools Used

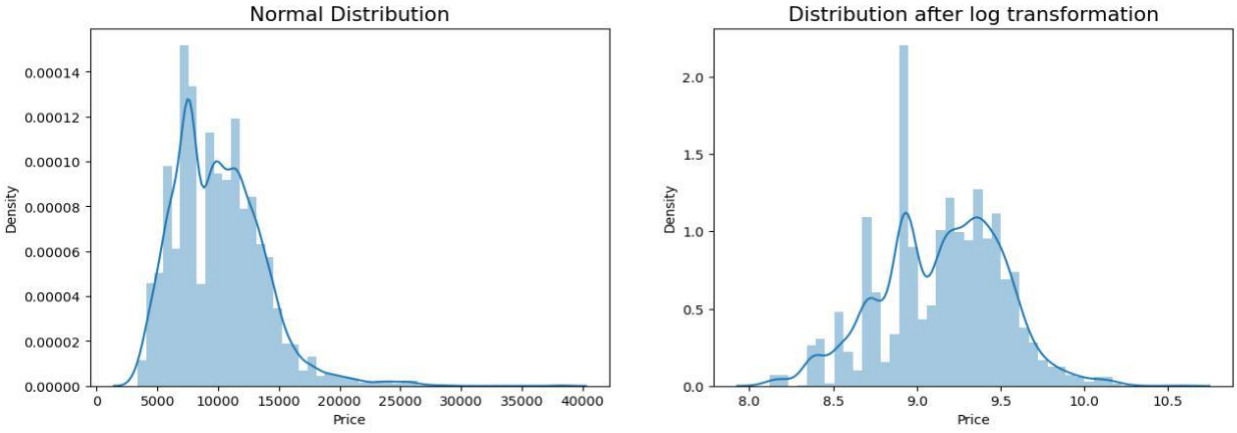
- For hardware I have used my laptop that have i5 processor and 8gb ram
- For software I have used Jupyter notebook
- For Tools I have use this following library-
  1. Numpy
  2. Pandas
  3. Seaborn
  4. Matplotlib
  5. Sklearn

Data Pre-processing

- Outlier removing(using z_score method)
- Skewness treatment
- Normalizing the data(applying StandardScaler to numerical features)
- Encoding categorical features(Using OrdinalEncoder)

Model Development and Evaluation

Applying log transformation to our target variable



As we can see our target variable is right skewed thats why I will apply log transfomation to it for better results.

For this project I have applied StandardScaler to numerical features for bringing it to a common scale and used ordinal encoder for categorical features. After doing such pre-processing steps I used this data for model building with the help of train_test_split. I have defined a function to train and evaluate our algorithms. For this task I have used many regression algorithms and selected LGBMRegressor as it is giving better performance than other algorithms. Linear models were giving least difference in r2-score and cv-score but in this case the r2-score was very less compared to tree based algorithms. Other than LGBMRegressor algorithms showing the problem of over-fitting so I selected LGBMRegressor which is not over-fitting much compared to others.

For this project I have used following algorithms:

- LinearRegressionC
- LassoCV
- RidgeCV
- DecisionTreeRegressor
- RandomForestRegressor
- XGBRegressor
- ExtraTreesRegressor
- LGBMRegressor

From all of these above models LGBMRegressor was giving me good performance.

Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

- I have used mean absolute error which gives magnitude of difference between the prediction of an observation and the true value of that observation.
- I have used root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions.
- I have used r2 score which tells us how accurate our model is.
- Also I have checked both training and testing r2-scores to check the over-fitting and under-fitting.
- Cross Validation score to check the model performance on overall data.

Hyperparameter Tuning

I did hyperparameter tuning for LGBMRegressor for the parameters like 'boosting_type', 'max_depth', 'learning_rate', 'n_estimators' using GridSearchCV

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

After running the code for above mentioned parameters I got the values which are indicated in the above figure as best parametric values for our final model.

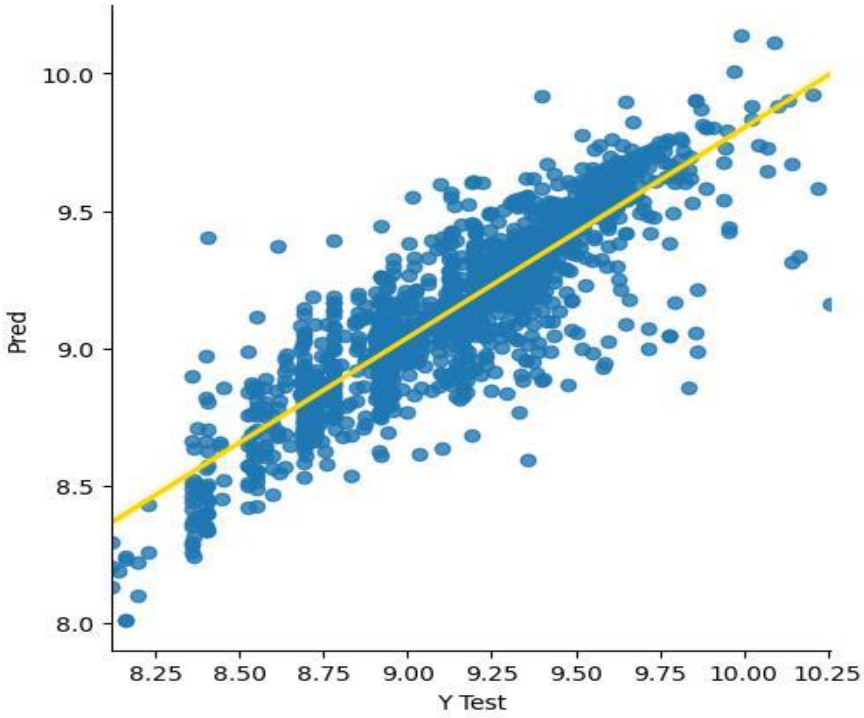
Using these parametric values I trained our final model and got good r2-score better than earlier.

Final Model

```
#lets train and test our final model with best parameters
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 : 0.12700025773946114
RMSE : 0.18493538527362255
------------------------------

r2 Score: 75.1935409688982 %
```

Let's see the graph for actual vs predicted values



Great we have achieved a better r2 score after doing hyperparameter tuning than earlier.

Conclusion

Key findings of the study

In this project we have scraped the flight data from yatra.com. Then the .csv file is loaded into a data frame.

Luckily we don't have any missing values in our data set.

Looking at the data set we understand that there are some features needs to be processed like converting the data types, and get the actual value from the string entries from the time related columns.

After the data is been processed I have done some EDA to understand the relation among features and the target variable.

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

Limitations of this work and scope for the future work

As looking at the features we came to know that the numbers of features are very less, due to which we are getting somewhat lower r2-scores.

Some algorithms are facing over-fitting problem which may be because of less number of features in our dataset.

We can get a better r2 score than now by fetching some more features from the web scraping by that we may also reduce the over fitting problem in our models.