FLIGHT PRICE PREDICTION

Submitted by:

NISHANT POKHRIYAL

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to FlipRobo who gave me this opportunity to do this wonderful project on Flight Price Prediction, which also helped me in doing a lot of research about the flights and how its prices fluctuate and I came to know about a lot of other new things I am really thankful to them.

I have searched many websites to scrape the data and finally found the best two sites i.e. Yatra.com & Easemytrip.com. I was able to find almost all the required data from here such as Airline name, duration of flight, departure time, arrival time, prices, etc.

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

INTRODUCTION

- ## Business Problem Framing

  Flight fares are not fixed and they fluctuate too often. The cheapest available ticket on a given flight gets more and less expensive over time. So we have to predict the prices of flights.

- ## Conceptual Background of the Domain Problem

  Prices of the flights depend on many factors such as:

  1. Airlines, Big players charge big because their Brand value is good.
  2. Destination, longer the destination, higher the prices.
  3. Stops, if the fight is non stop then fares will be less, if there are few stops then fares will be high.

  The ticket prices fluctuate quite often, This usually happens as an attempt to maximize revenue based on time of purchase, to make sure last minute purchases are expensive, keeping the flight as full as they want it by fluctuating the prices. So we have to scrape the data of flight fares with other features and to build a model to predict prices of flights.

- ## Review of Literature

  It is very difficult for the customer to purchase a flight ticket at the minimum price. There are many factors on which the pricing of the ticket depends. For this project I have scraped the data from sites like Yatra.com and Easemytrip.com. i.e. name, date, source, destination, departure time, arrival time, duration, stops and prices.

  Scraped all the data and saved it into dataframe and then saved it as a csv file.

  We have done the required exploratory data analysis such as separating hour and minutes from departure, arrival time and

duration, converting string data into integer and created dummies for the different variables.

We have created several machine learning models to predict the prices of the flights, The best performing model is Random Forest Regressor with approximately 64.8% of accuracy and 63.6% of cross validation score. Other models accuracy are (Linear Regression model : 58.9%), (Lasso Regression : 58.4%), (Gradient Boosting Regressor : 65.5%).

- Motivation for the Problem Undertaken

It’s very hard to purchase the flight tickets at a minimum price so I wanted to collect the data for different types of flights and their prices so I could create a machine learning model that can predict the price of the flights depending on the factors affecting it.

Analytical Problem Framing

- Data Sources and their formats

We have scraped the data from two of the industry leading websites that are Yatra.com and Easemytrip.com. We have got all the required information about the flights, these are:

1. Name : Name of the Airline,
2. Date : Date of booking the tickets,
3. Source : Website from where the data is collected,
4. Destination : Destination of the Flight,
5. Departure time : Time of Departure,
6. Arrival time : Time of Arrival,
7. Duration : Total duration of the journey,
8. Stops : Total stops during the journey
9. Price : Price of the flight tickets.

```
]: flights
```

]:

	Name	Date	Source	Destination	Departure Time	Arrival Time	Duration	Stops	Price	Month	Day
0	Go First\nG8-2501	1900-09-23	Yatra	Mumbai	02:40	04:50	2h 10m	Non Stop	5,900	9	23
1	SpiceJet\nSG-8709	1900-09-23	Yatra	Mumbai	19:00	21:10	2h 10m	Non Stop	5,950	9	23
2	SpiceJet\nSG-8701	1900-09-23	Yatra	Mumbai	07:20	09:35	2h 15m	Non Stop	5,950	9	23
3	SpiceJet\nSG-8169	1900-09-23	Yatra	Mumbai	19:45	22:00	2h 15m	Non Stop	5,950	9	23
4	IndiGo\n6E-5306	1900-09-23	Yatra	Mumbai	15:25	17:20	1h 55m	Non Stop	5,954	9	23
...	...	...	...	...	...	...	...	...	...	...	...
1499	Vistara\nUK-940	1900-09-16	Ease My Trip	Chennai	19:45	09:50	14h 05m	1-stop	24,162	9	16
1500	Air India\nAI-806	1900-09-16	Ease My Trip	Chennai	08:00	23:35	15h 35m	1-stop	24,736	9	16
1501	Air India\nAI-617	1900-09-16	Ease My Trip	Chennai	11:05	23:35	12h 30m	2+-stop	24,736	9	16
1502	Vistara\nUK-647	1900-09-16	Ease My Trip	Chennai	10:25	20:15	09h 50m	2+-stop	24,799	9	16
1503	Vistara\nUK-970	1900-09-16	Ease My Trip	Chennai	08:45	22:45	14h 00m	1-stop	25,797	9	16

1504 rows × 11 columns

- Data Preprocessing Done

Created different columns for month and day

```
from datetime import datetime
fl['Date']=pd.to_datetime(fl['Date'], format='%b %d')
```

```
fl['Month'] = pd.DatetimeIndex(fl['Date']).month
fl['Day'] = pd.DatetimeIndex(fl['Date']).day
```

```
flights['Date']=pd.to_datetime(flights['Date'], format=' %d %b')
```

```
flights['Month'] = pd.DatetimeIndex(flights['Date']).month
flights['Day'] = pd.DatetimeIndex(flights['Date']).day
```

After saving the dataframe into a csv file, we got an extra column named unnamed: 0 so we dropped it and we also dropped column date because we already have the month and day column.

```
f.drop('Unnamed: 0', axis=1, inplace=True)
```

```
f.drop('Date',axis=1,inplace=True)
```

No null values are present in the data

```
f.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1504 entries, 0 to 1503
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Name            1504 non-null   object
 1   Source          1504 non-null   object
 2   Destination     1504 non-null   object
 3   Departure Time  1504 non-null   object
 4   Arrival Time    1504 non-null   object
 5   Duration        1504 non-null   object
 6   Stops           1504 non-null   object
 7   Price           1322 non-null   object
 8   Month           1504 non-null   int64 
 9   Day             1504 non-null   int64 
dtypes: int64(2), object(8)
memory usage: 117.6+ KB
```

Extracted hour and minutes from the departure, arrival time and duration.

```
f['Departure Time']=pd.to_datetime(f['Departure Time'])
# Extracting Departure Hours
f['Departure Hour']=f['Departure Time'].dt.hour

# Extracting Departure Minutes
f['Departure Min']=f['Departure Time'].dt.minute

# Dropping Departure Time column because it is of no use now
f.drop(['Departure Time'],axis=1,inplace=True)
```

```
f['Arrival Time']=pd.to_datetime(f['Arrival Time'])
# Extracting Arrival Hours
f['Arrival Hour']=f['Arrival Time'].dt.hour

# Extracting Arrival Minutes
f['Arrival Min']=f['Arrival Time'].dt.minute

# Dropping Arrival Time column because it is of no use now
f.drop(['Arrival Time'],axis=1,inplace=True)
```

```
# Extracting hour and minutes from Duartion

# Assigning and converting Duartion column into list
duration = list(f["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) !=2:                    # Checking if Duration contains only hour or minute
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"    # Adds 0 minute
        else:
            duration[i]= "0h " + duration[i]           # Adds 0 hour


duration_hour = []
duration_min = []

for i in range(len(duration)):
    duration_hour.append(int(duration[i].split(sep = "h")[0]))
    duration_min.append(int(duration[i].split(sep = "m")[0].split()[-1]))
```

- Data Inputs- Logic- Output Relationships

  These columns are the factors on which the price of the flight depends, like name of the airline is the surety of quality, if the Airline is reputed then they'll charge big amount for the service.

  If the destination is long then the fare will be high and if the destination is short then fare will be less.

  The price of the flight also depends on the departure timings. Price will be different for different times.

Duration of the Journey also play an important role in deciding the price, shorter duration has lower prices than longer durations.

Price is also dependent on the total number of stops in the journey.

- Assumptions related to the problem under consideration

  We have only collected the data for domestic flights.

- Hardware and Software Requirements and Tools Used
  1. Jupyter Notebook
  2. Module : Datetime
  3. Packages : Selenium, Web driver, Pandas, Numpy, sklearn, matplotlip, seaborn
  4. Dataset : https://github.com/nishantpokhriyal/Internship/blob/main/Flight%20Price%20Prediction/scraped_flights.csv

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  Listing down all the algorithms used for the training and testing.

```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
```

- Run and Evaluate selected models

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import r2_score
```

```
def results(actual,pred):
    mae=mean_absolute_error(actual,pred)
    mse=mean_squared_error(actual,pred)
    r2=r2_score(actual,pred)

    print('Mean Absolute Error :',mae,'\n')
    print('Mean Squared Error :',mse,'\n')
    print('r2 Score :',r2,'\n')
```

Linear Regression : Linear regression analysis is used to predict the price of the flight based on the value of another variables.

```
ln=LinearRegression()
ln.fit(x_train,y_train)
pred=ln.predict(x_test)

print('\033[1m' + 'LINEAR REGRESSION MODEL SUMMERY\n' + '\033[0m')
print('Predicted Price',pred)
print('\nActual Price\n',y_test)
print('\n')
results(y_test,pred)
accuracy(ln)
```

```
Mean Absolute Error : 2401.8502178994772

Mean Squared Error : 9193096.926691832

r2 Score : 0.5896123233046361

Cross_val_Accuracy: -41203550173596473288829173 76.00 %

Cross_val_Standard Deviation: 123610650520789408871371243 52.00 %
```

Lasso Regression : Lasso regression is used to obtain the subset of predictors that minimizes prediction error for a quantitative response variable.

```
ls=Lasso(alpha=0.0001)
ls.fit(x_train,y_train)
pred_ls=ls.predict(x_test)

print('\033[1m' + 'LASSO REGRESSION MODEL SUMMERY\n' + '\033[0m')
print('Predicted Price',pred_ls)
print('\nActual Price\n',y_test)
print('\n')
results(y_test,pred_ls)
accuracy(ls)
```

```
Mean Absolute Error : 2399.252882756697

Mean Squared Error : 9300309.71711723

r2 Score : 0.5848262530254296

Cross_val_Accuracy: 47.77 %

Cross_val_Standard Deviation: 7.79 %
```

Gradient Boosting Regressor : GBR is used because it prevents overfitting by making predictions for all individual decision trees and averaging the regression results.

```
gb=GradientBoostingRegressor()
gb.fit(x_train,y_train)
pred_gb=gb.predict(x_test)

print('\033[1m' + 'GRADIENT BOOSTING REGRESSOR MODEL SUMMERY\n' + '\033[0m')
print('Predicted Price',pred_gb)
print('\nActual Price\n',y_test)
print('\n')
results(y_test,pred_gb)
accuracy(gb)
```

```
Mean Absolute Error : 2080.868937761348

Mean Squared Error : 7723555.283901217

r2 Score : 0.6552139138677577

Cross_val_Accuracy: 56.33 %

Cross_val_Standard Deviation: 6.74 %
```

Random Forest Regressor : Random forest is used because it fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
rr=RandomForestRegressor()
rr.fit(x_train, y_train)
pred_rr=rr.predict(x_test)

print('\033[1m' + 'RANDOM FOREST REGRESSOR MODEL SUMMERY\n' + '\033[0m')
print('Predicted Price',pred_rr)
print('\nActual Price\n',y_test)
print('\n')
results(y_test,pred_rr)
accuracy(rr)
```

```
Mean Absolute Error : 1906.0725852650494

Mean Squared Error : 7864074.519210141

r2 Score : 0.6489410155213904

Cross_val_Accuracy: 63.61 %

Cross_val_Standard Deviation: 5.32 %
```

- Key Metrics for success in solving problem under consideration

  Mean Squared Error :

  This metric is used as It essentially finds the average of the squared difference between the target value and the value predicted by the regression model.

  Mean Absolute Error :

  Mean Absolute Error is the average of the difference between the ground truth and the predicted values.

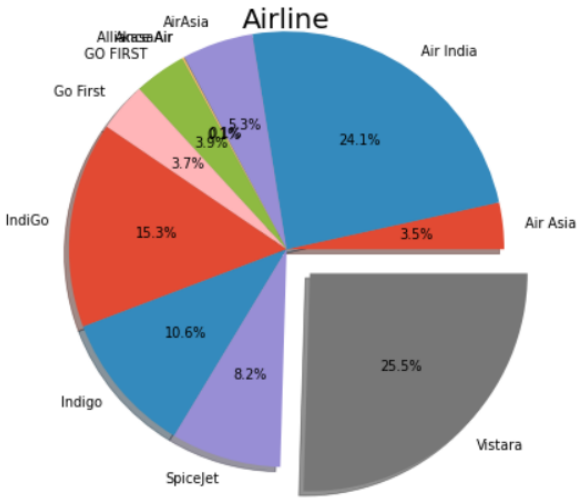
  R2 Score :

  It is Coefficient of determination, is the proportion of the variance in the dependent variable(price) that is predictable from the independent variables.

- Visualizations

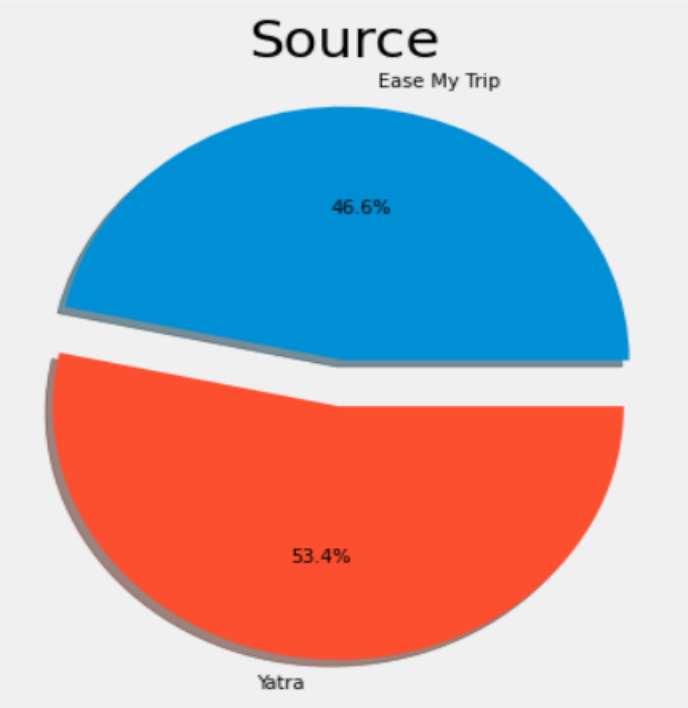
  Pie Chart : It is used to see the percentage of all variables.

```
Unique Values : ['Air Asia' 'Air India' 'AirAsia' 'AkasaAir' 'Alliance Air' 'GO FIRST'
 'Go First' 'IndiGo' 'Indigo' 'SpiceJet' 'Vistara']
Frequency Values : [52, 362, 79, 1, 2, 58, 55, 230, 159, 123, 383]
```



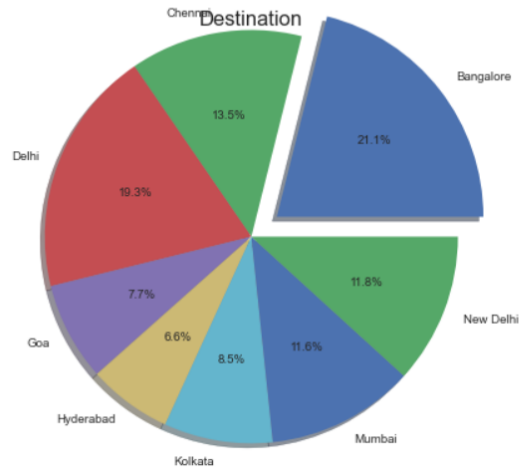
- Maximum number of flights are from Vistara and minimum number are from Akasa Air.

```
Unique Values : ['Ease My Trip' 'Yatra']
Frequency Values : [701, 803]
```



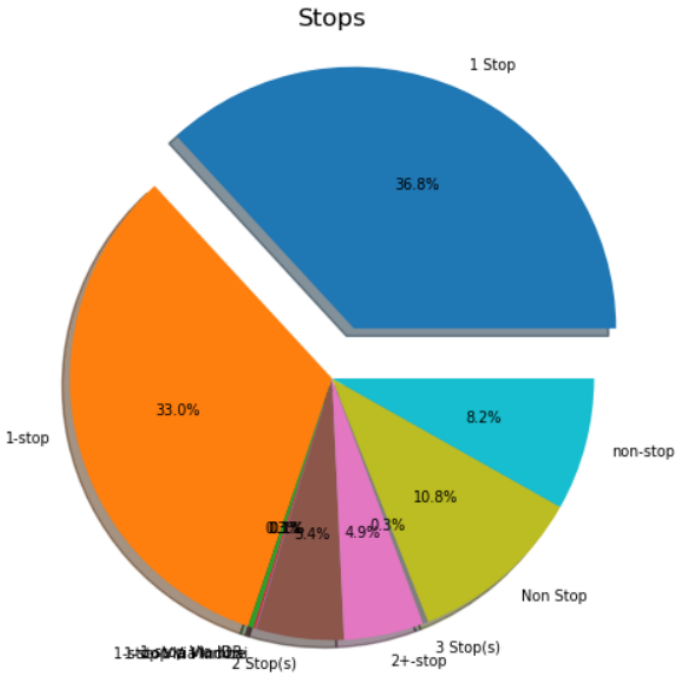
- Maximum number of data is from Yatra.com.

```
Unique Values : ['Bangalore' 'Chennai' 'Delhi' 'Goa' 'Hyderabad' 'Kolkata' 'Mumbai'
 'New Delhi']
Frequency Values : [317, 203, 290, 116, 99, 128, 174, 177]
```



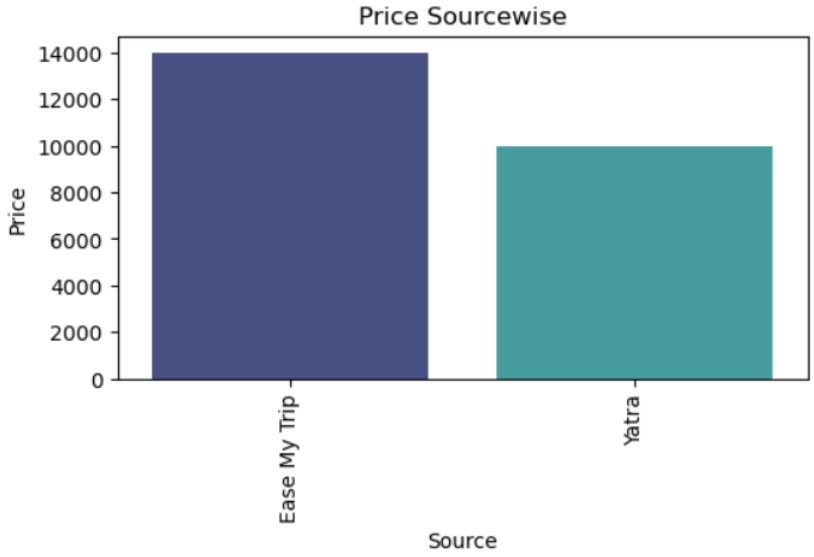
- Most of the flights destination is Bangalore and least number of flights are going to Hyderabad.

```
Unique Values : ['1 Stop' '1-stop' '1-stop Via IDR' '1-stop Via Indore'
 '1-stop Via Mumbai' '2 Stop(s)' '2+-stop' '3 Stop(s)' 'Non Stop'
 'non-stop']
Frequency Values : [554, 496, 5, 2, 1, 81, 74, 5, 163, 123]
```

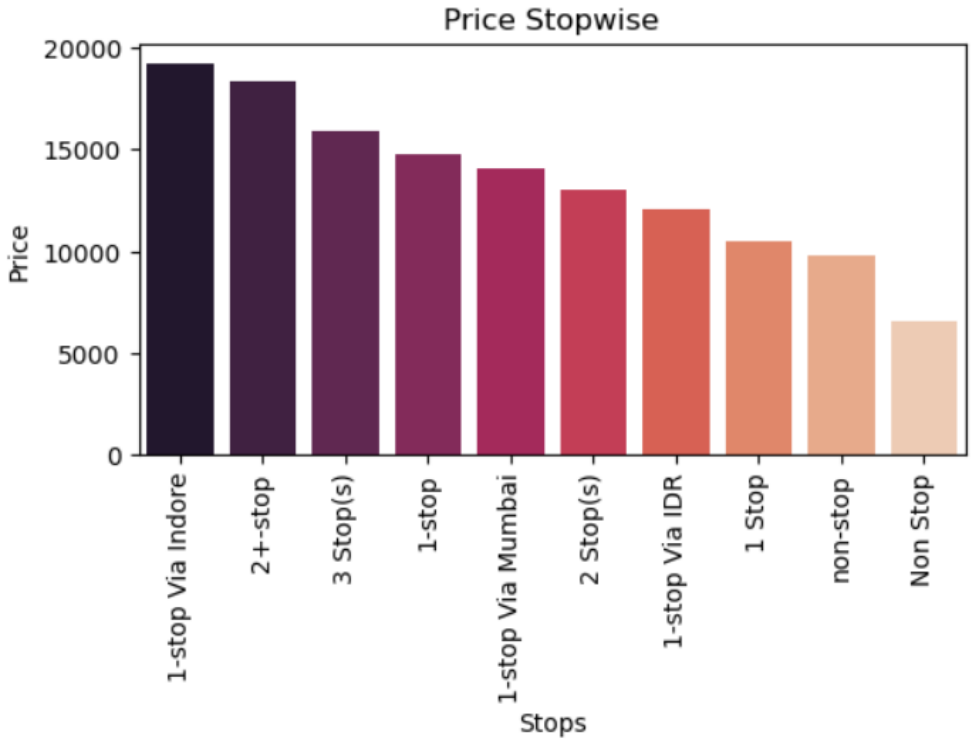


- Flights having 1 stop are most in numbers and flights having 1 stop via IDR, 1 stop via Indore and 1 stop via Mumbai are the least.

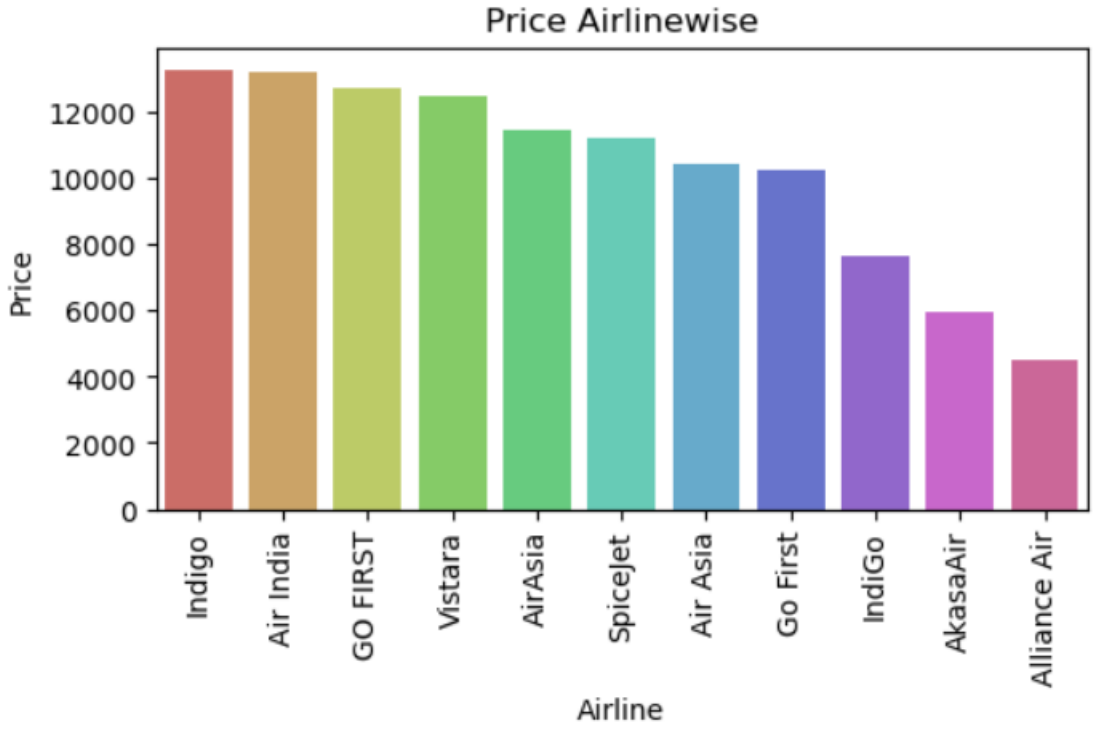
Bar Plot : It is used to see the plot showing prices with different variables.



- Easemytrip has higher price flights as compared to Yatra.

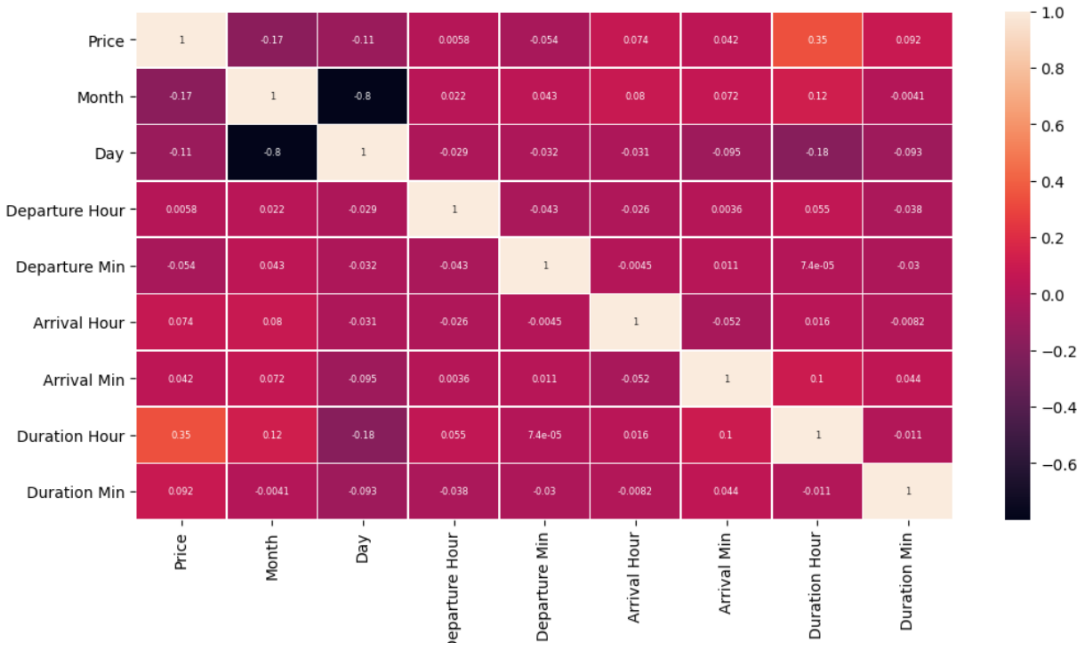


- Prices are on a higher side where stops are more, 1 stop via Indore has the maximum price and Non stop flight has minimum price.



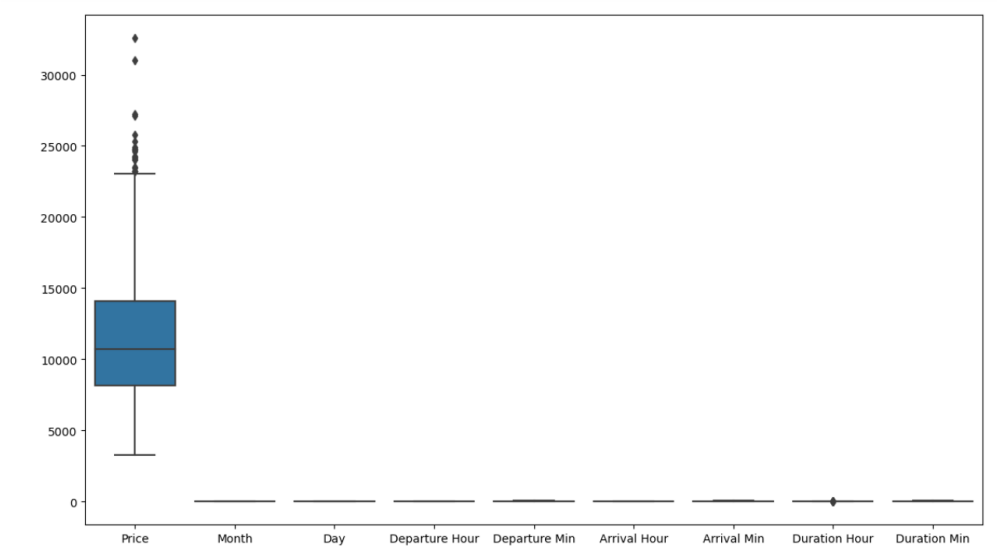
- Indigo and Air India has the maximum prices as compared to the other Airlines and Alliance Air has the minimum price.

Heatmap : It is used to see the correlation between different variables.



- Variables are not much correlated with price and some are negatively correlated also, only duration_hour is little correlated with price.

Box plot : It is used to see the plot showing outliers in varibales.



- Price, and Duration_Hour has outliers but But we haven't dropped any of them because that data is important for us.

- Interpretation of the Results

  On the basis of my analysis I found that only the prices of Indigo , Air India , Go first and Vistara went above 12000. Prices increases as the stops increases.

```
: print('Predicted Result','\n', pred_rr)
  print('Actual Result','\n', y_test)
```

```
Predicted Result
 [12626.04       14966.77        8927.21       18356.68166667
  8439.17       15534.92        9195.49       11714.06
  9398.21        8299.82        8589.12       20521.76

Actual Result
       Price
829    15353
1442   13955
539    11127
1482   21546
282     9037
...      ...
```

As we can see model is performing well.

CONCLUSION

- Key Findings and Conclusions of the Study

  In my observation I have found that prices are majorly depending upon the Airlines, number of stops and destination.

- Learning Outcomes of the Study in respect of Data Science

  I've got to know so much new things during this project like how the prices of Airlines fluctuate so much depending on the factors affecting it such as destination, number of stops and Airlines. I was able to understand this by the help of visualization libraries like matplotlib and seaborn by creating plots of different variables. I have created different types of regression models like Linear Regression, Lasso regression, Gradient Boosting Regression and Random forest regressor to predict the prices by training the data and the best performing model was Random Forest regressor with 64.8 % of accuracy score.