FLIGHT PRICE PREDICTION PROJECT

Submitted by: Satyam Gupta

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

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

I want to thank my SME Miss. Khushboo Garg for providing the project and helping us to solve the problem and addressing out our Query in right time.

I would like to express my gratitude towards my parents & members of Flip Robo for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

INTRODUCTION

BUSINESS PROBLEM FRAMING

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

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ANALYTICAL MODELLING OF THE PROBLEM

In our scrapped dataset, our target variable "**Flight_price**" is a continuous variable. Therefore, we will be handling this modelling problem as regression.

This project is done in two parts:

- ➔ Data Collection phase
- ➔ Data Analysis phase
- ➔ Model building phase

Data Collection phase:

You have to scrape at least 2500 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.

Data Analysis Phase:

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?

Model building phase:

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. Follow the complete life cycle of data science. Include 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 data is collected from websites yatra.com. The data is scrapped using Web scraping technique and the framework used is Selenium.

➔We scrapped approx. 2500 records of the data and saved each data in a separate data frame.

➔In the end, we combined all the data frames into a single data frame and it looks like as follows:

```
import numpy as np
df=pd.DataFrame({})
df['Airline_Name']=Name
df['Source']=source
df['Destination']=destination
df['Departure_time']=dep_time
df['Arrival_time']=arr_time
df['Date of Journey']=Date_of_Journey
df['Number of stops']=Total_stops
df['Duration']=Duration
df['Price']=Price
df
```

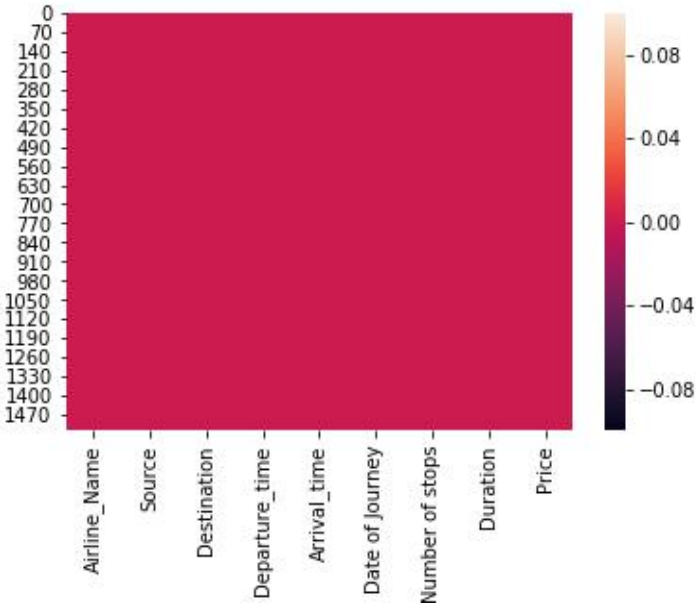
	Airline_Name	Source	Destination	Departure_time	Arrival_time	Date of Journey	Number of stops	Duration	Price
0	Go First	New Delhi	Mumbai	06:00	10:40	Thu, 25 Nov	1 Stop	4h 40m	4,941
1	Go First	New Delhi	Mumbai	05:25	14:05	Thu, 25 Nov	1 Stop	8h 40m	4,941
2	SpiceJet	New Delhi	Mumbai	06:35	10:15	Thu, 25 Nov	1 Stop	3h 40m	5,949
3	Air Asia	New Delhi	Mumbai	20:00	02:25\n+ 1 day	Thu, 25 Nov	1 Stop	6h 25m	5,953
4	Air Asia	New Delhi	Mumbai	09:35	20:35	Thu, 25 Nov	1 Stop	11h 00m	5,953
5	Air Asia	New Delhi	Mumbai	20:00	07:40\n+ 1 day	Thu, 25 Nov	1 Stop	11h 40m	5,953
6	Air Asia	New Delhi	Mumbai	18:35	06:45\n+ 1 day	Thu, 25 Nov	1 Stop	12h 10m	5,953
7	Air Asia	New Delhi	Mumbai	08:00	20:35	Thu, 25 Nov	1 Stop	12h 35m	5,953
8	Air Asia	New Delhi	Mumbai	18:35	07:15\n+ 1 day	Thu, 25 Nov	1 Stop	12h 40m	5,953
9	Air Asia	New Delhi	Mumbai	12:40	02:25\n+ 1 day	Thu, 25 Nov	1 Stop	13h 45m	5,953
10	Air Asia	New Delhi	Mumbai	12:10	02:25\n+ 1 day	Thu, 25 Nov	1 Stop	14h 15m	5,953
11	Air Asia	New Delhi	Mumbai	22:10	12:25\n+ 1 day	Thu, 25 Nov	1 Stop	14h 15m	5,953

DATA PRE-PROCESSING

Checking for null-values

```
sn.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e70f7772e8>
```



```
df.isnull().sum()
```

```
Airline_Name       0
Source             0
Destination        0
Departure_time     0
Arrival_time       0
Date of Journey    0
Number of stops    0
Duration           0
Price              0
dtype: int64
```

The above both graph is here to show that if there are any null values in the above dataset. In order to get clearity we have taken out the sum of the total null values down which is giving us the same output that is 0.

Data Columns

```
#Shows all the columns
df.columns
```

```
Index(['Airline_Name', 'Source', 'Destination', 'Departure_time',
       'Arrival_time', 'Date of Journey', 'Number of stops', 'Duration',
       'Price'],
      dtype='object')
```

```
#Checking for unique values present in each column
df.nunique()
```

```
Airline_Name         6
Source               1
Destination          1
Departure_time     134
Arrival_time       160
Date of Journey     15
Number of stops      4
Duration           164
Price              349
dtype: int64
```

Feature Engineering

- In this we have three columns Date of Journey, Departure time, and Arrival time to be feature engineered.
- Date of Journey is converted into date, day and month.
- Departure time is converted into hours and minutes and similarly Arrival time is converted into minutes and hours.
- Introduced new features Duration in seconds.
- Based upon the arrival time and departure time found out the whether flight arrived or departed in morning, afternoon and night.

```
df.head()
```

Departure_time	Arrival_time	Date of Journey	Number of stops	Duration	Price	Day	Month	Date	Departure_Hour	Departure_Minute	Arrival_Hour	Arrival_Minute	Duration(sec)
06:00	10:40	Thu, 25 Nov	1 Stop	4h 40m	4941	Thu	Nov	25	6	0	10	40	16800
05:25	14:05	Thu, 25 Nov	1 Stop	8h 40m	4941	Thu	Nov	25	5	25	14	5	31200
06:35	10:15	Thu, 25 Nov	1 Stop	3h 40m	5949	Thu	Nov	25	6	35	10	15	13200
20:00	02:25	Thu, 25 Nov	1 Stop	6h 25m	5953	Thu	Nov	25	20	0	2	25	23100
09:35	20:35	Thu, 25 Nov	1 Stop	11h 00m	5953	Thu	Nov	25	9	35	20	35	39600

Statistical summary of the dataset

```
#Gives statistical summary of data
df.describe()
```

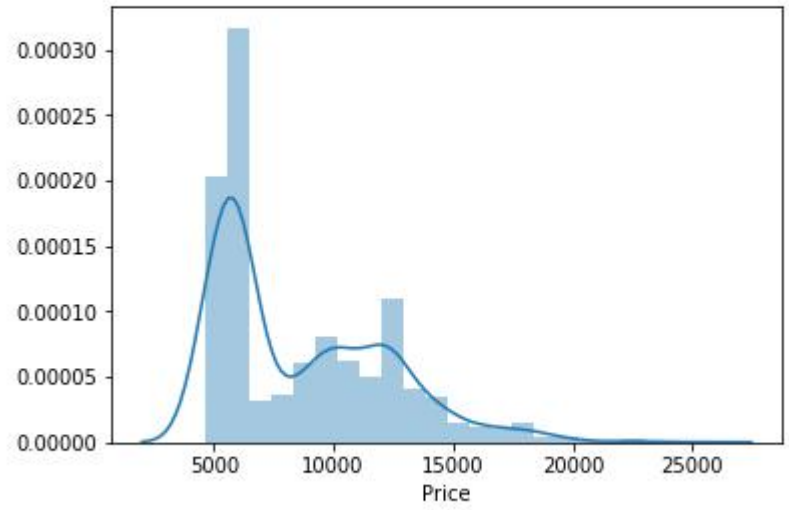
	Price
count	1529.000000
mean	8664.035317
std	3611.744765
min	4650.000000
25%	5953.000000
50%	7110.000000
75%	11468.000000
max	24815.000000

The above plot shows that the count of all column is equal to 1529 ,average price of ticket is 8664.03 and the minimum flight price is 4650 and maximum is 24815.

Distribution Plot and Outliers

```
#Plotting the distribution plot for price
sn.distplot(df['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e70fdba978>
```

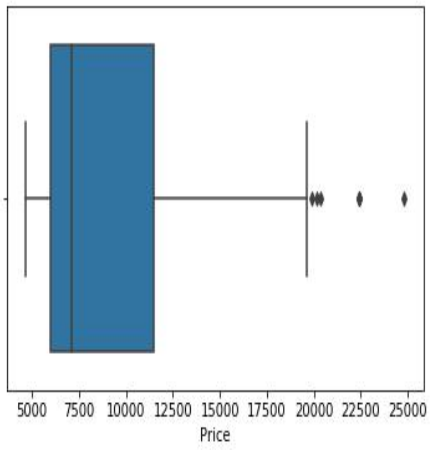


We can see that distribution plot is right skewed.

Checking Outliers

```
sn.boxplot(df['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e71118cb00>
```

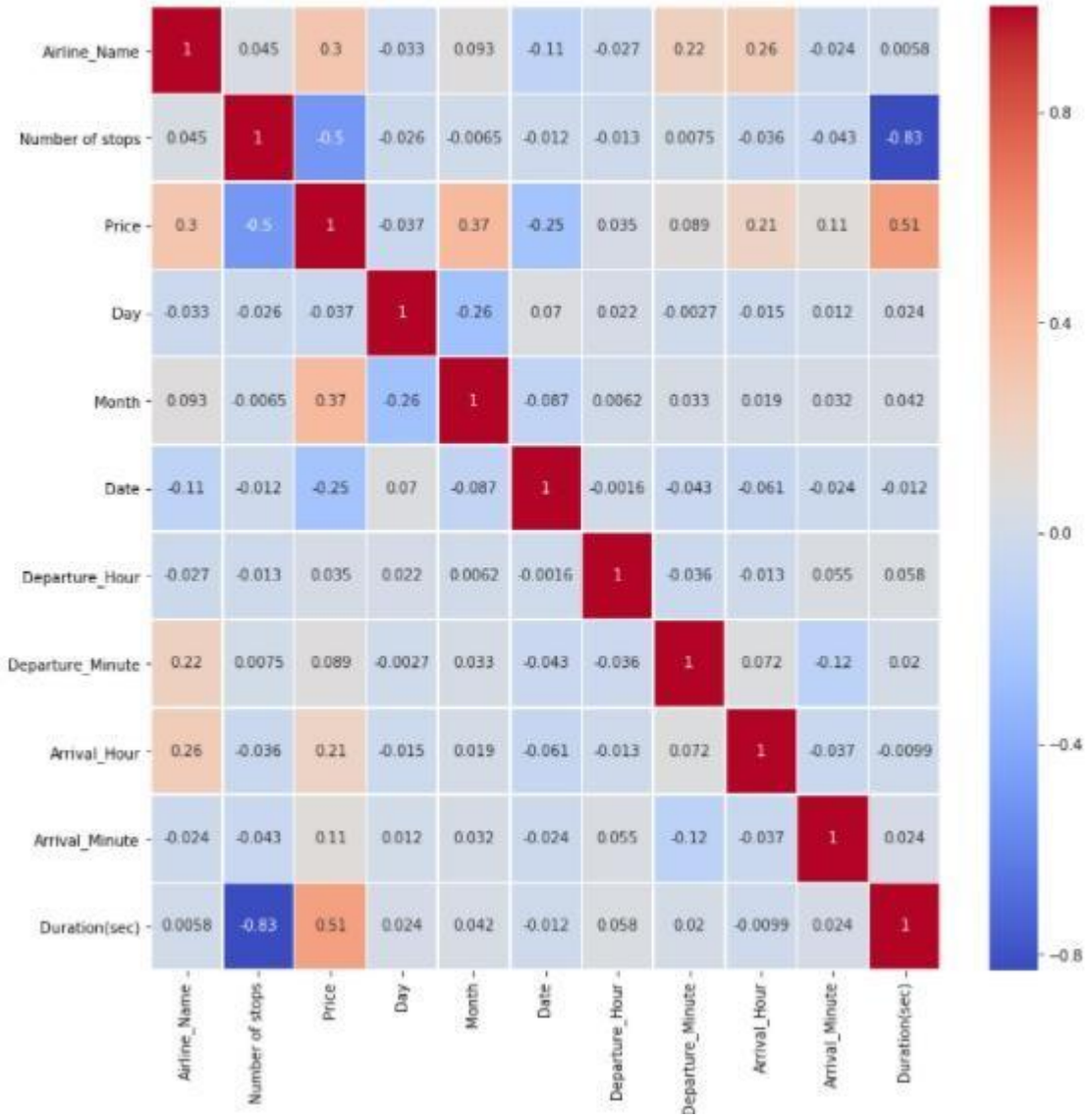


Outliers are present in the dataset, but as we know that it is the price of the flights which changes on the realtime, thus we will not remove the outliers here.

Plotting Heat Map

```
sn.heatmap(corr_matrix,annot=True,linewidths=.5,cmap='coolwarm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e70fd5cbe0>
```



HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used **Jupyter notebook** to do my python programming and analysis.

For using an CSV file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

Libraries Used:

Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn

from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
```

MODEL/S DEVELOPMENT AND EVALUATION

Preparing dataset for model training

Seperating data into independent and dependent variables.

```
x = df.drop('Price', axis = 1) #Set of input features
y = df['Price']  #Target variable
```

```
print(x.shape)
x.head()
```

```
(1529, 10)
```

	Airline_Name	Number of stops	Day	Month	Date	Departure_Hour	Departure_Minute	Arrival_Hour	Arrival_Minute	Duration(sec)
0	2	0.0	4	1	3	6	0	10	40	9.729194
1	2	0.0	4	1	3	5	25	14	5	10.348205
2	4	0.0	4	1	3	6	35	10	15	9.488048
3	0	0.0	4	1	3	20	0	2	25	10.047631
4	0	0.0	4	1	3	9	35	20	35	10.586610

```
print(y.shape)
y.head()
```

```
(1529,)
```

```
0    8.505525
1    8.505525
2    8.691146
3    8.691819
4    8.691819
Name: Price, dtype: float64
```

Using Label encoding and scaling the data

```
# Using Lable encoding
from sklearn.preprocessing import LabelEncoder
for col in df.columns:
    if df[col].dtypes == 'object':
        encoder = LabelEncoder()
        df[col] = encoder.fit_transform(df[col])
```

Using Standard Scaler

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
```

Model Building

Finding best random state and r2_score

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
MaxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=i)
    mod=LinearRegression()
    mod.fit(x_train,y_train)
    pred=mod.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>MaxAccu:
        MaxAccu=acc
        maxRS=i
print("Best R2_score is",MaxAccu,'on random state',maxRS)
```

```
Best R2_score is 0.6189760666735094 on random state 42
```

Creating train test split

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=maxRS)
```

Applying Different Models on the dataset

```
#Linear Regression
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)
y_pred=regressor.predict(x_test)
lr_score=r2_score(y_test,y_pred)
print("R2 score from Linear regression is",lr_score)
lr_cv=cross_val_score(regressor,x,y,scoring='r2',cv=5).mean()
lr_mae=mean_absolute_error(y_test,y_pred)
print('Mean absolute error : ',lr_mae)
lr_mse=mean_squared_error(y_test,y_pred)
print('Mean Squared error :',lr_mse)
print("Cross validation score is ",lr_cv)
```

```
R2 score from Linear regression is 0.6189760666735094
Mean absolute error :  0.18344110696470256
Mean Squared error : 0.057769222755463694
Cross validation score is  0.3630030516140986
```

The final outputs of the models we used are stored in a dataframe and it is shown below:

```
model=['Linear Regression','Support Vector Regressor','K Neighbors Regressor','Random Forest Regressor']

acc=[svr_score*100,lr_score*100,knn_score*100,rfr_score*100]
mae=[svr_mae,lr_mae,knn_mae,rfr_mae]
mse=[svr_mse,lr_mse,knn_mse,rfr_mse]
cv_score=[svr_cv*100,lr_cv*100,knn_cv*100,rfr_cv*100]

result=pd.DataFrame({'Model':model,'R2_score':acc,'Mean Absolute Error':mae,'Mean Squared Error':mse,'Cross validation_score':cv
_score})
result
# Creting dataframe to store R2_score,Mae and cv score of all the models.
```

	Model	R2_score	Mean Absolute Error	Mean Squared Error	Cross validation_score
0	Linear Regression	82.972598	0.107794	0.025816	33.998443
1	Support Vector Regressor	61.897607	0.183441	0.057769	36.300305
2	K Neighbors Regressor	83.170131	0.100205	0.025517	23.121090
3	Random Forest Regressor	83.919257	0.090071	0.024381	28.490928

We can see that Random Forest Regressor are performing well compared to other algorithms. Now we will try Hyperparameter Tuning to find out the best parameters and try to increase the scores.

Hyperparameter Tuning

Random Forest Regressor

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {
              "n_estimators"      : [10,50,100,200],
              "max_features"      : ["auto", "sqrt", "log2"],
              "min_samples_split" : [2,4,8],
              "bootstrap": [True, False],
              }
```

```
GCV=GridSearchCV(RandomForestRegressor(),parameters,cv=5)
```

```
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'bootstrap': [True, False],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [2, 4, 8],
                         'n_estimators': [10, 50, 100, 200]})
```

```
GCV.best_params_
```

```
{'bootstrap': False,
 'max_features': 'log2',
 'min_samples_split': 4,
 'n_estimators': 100}
```

```
Final_Model=RandomForestRegressor(bootstrap=False,max_features='log2',min_samples_split=4,n_estimators=100)
Final_Model.fit(x_train,y_train)
y_pred=Final_Model.predict(x_test)
acc=r2_score(y_test,y_pred)
print("R2 Score of the model is ",acc*100)
knn_mae=mean_absolute_error(y_test,y_pred)
print('Mean absolute error : ',knn_mae)
```

```
R2 Score of the model is  84.22781932886927
Mean absolute error :  0.08871526642077589
```

Here after trying with various model for the above dataset, I am going to choose Random forest regressor as the best model for predicting the price of the flights. Random Forest Regressor performs best with hyperparameter tuning, where accuracy increased from 83.91 to 84.22%.

FINAL THE MODEL

Saving the model

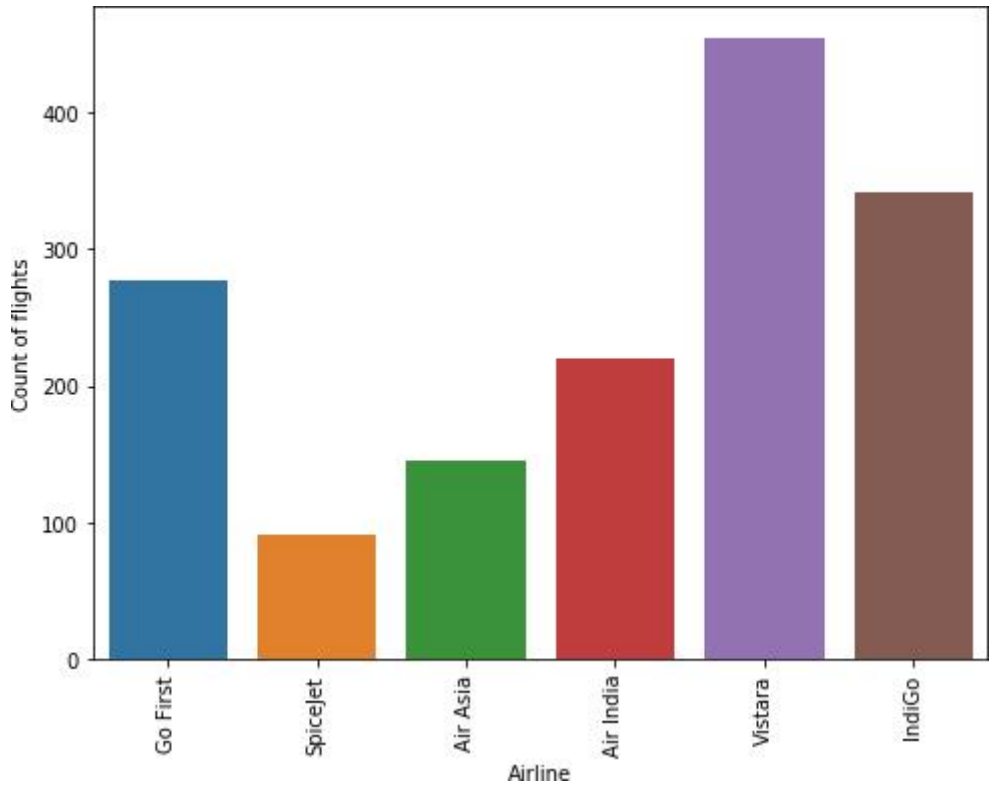
```
import joblib
import sys
sys.modules['sklearn.externals.joblib']=joblib
from sklearn.externals import joblib
```

```
joblib.dump(Final_Model,'Final_Model.pkl')
```

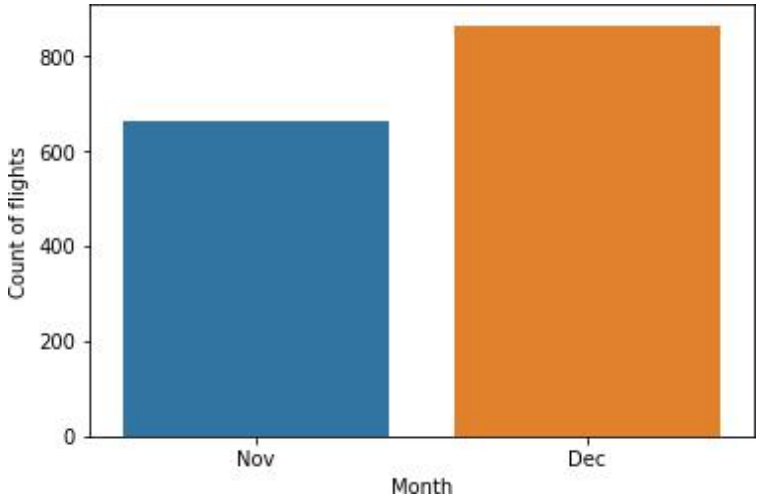
```
['Final_Model.pkl']
```

DATA VISUALIZATION

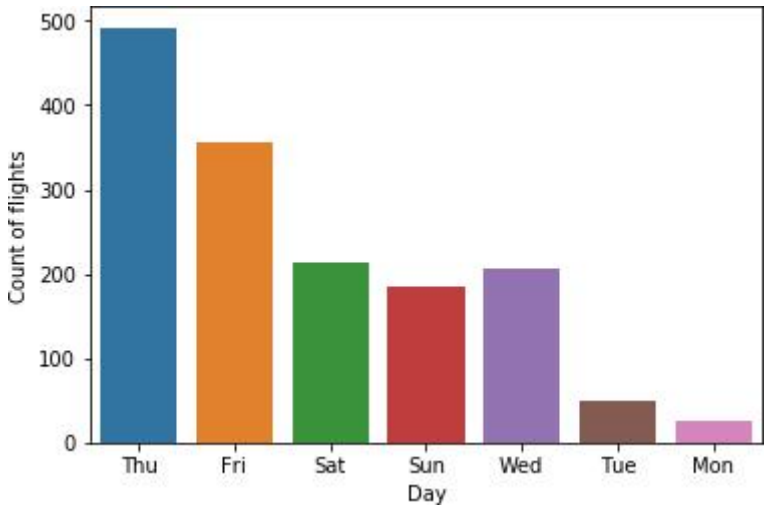
Airlines

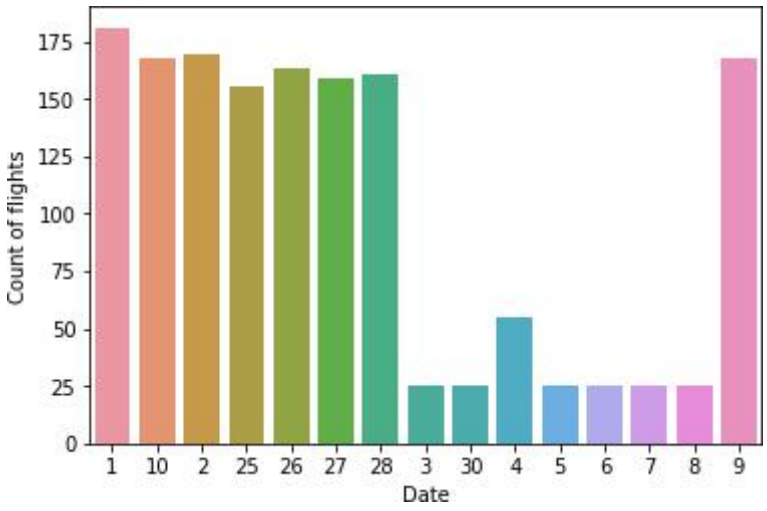


Month of flight

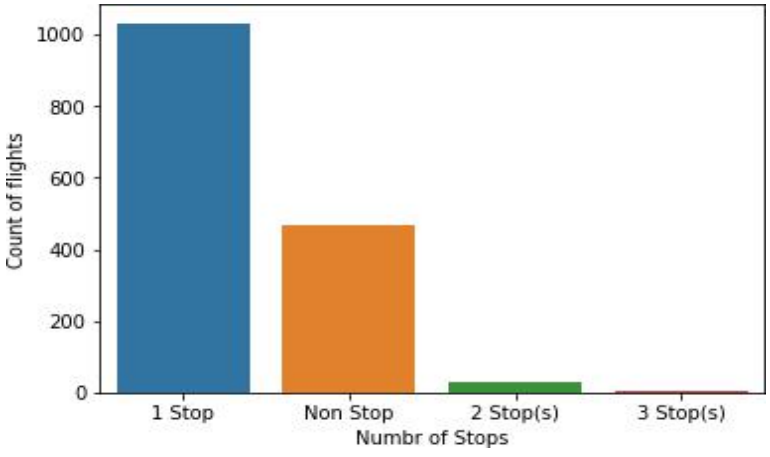


Flight by Day

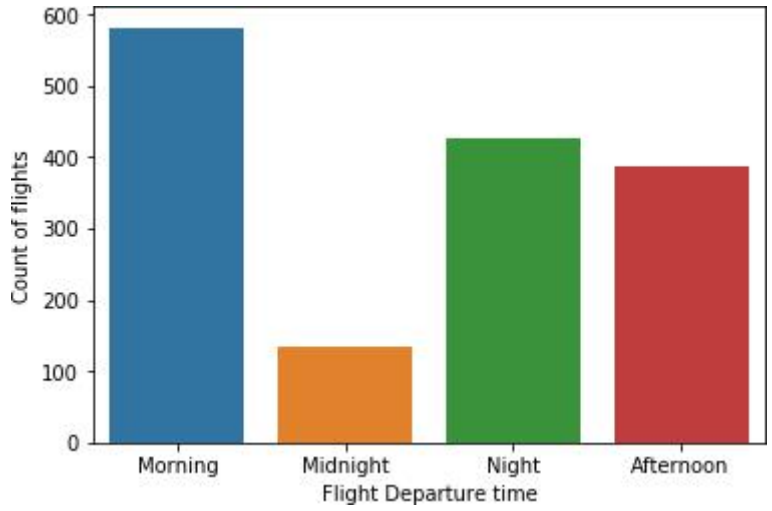




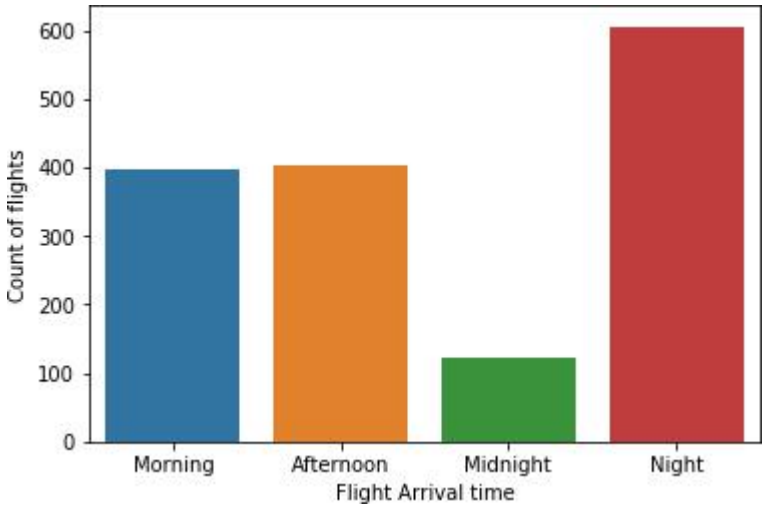
Number of Stops



Flight Departure Time



We can see that the least number of flight are in Midnight and Highest in Mornings.

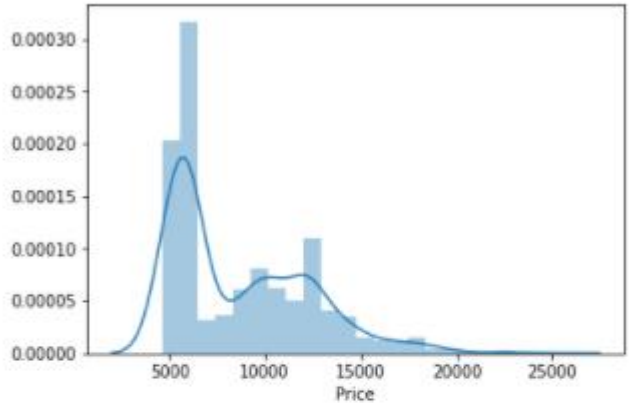


Most number of flights are arrived at Night time.

Distribution plot and Box Plot for Price

```
: #Plotting the distribution plot for price
  sn.distplot(df['Price'])
```

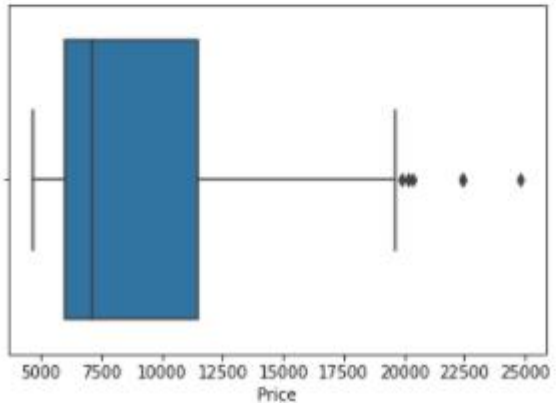
```
: <matplotlib.axes._subplots.AxesSubplot at 0x1e70fdba978>
```



We can see that distribution plot is right skewed.

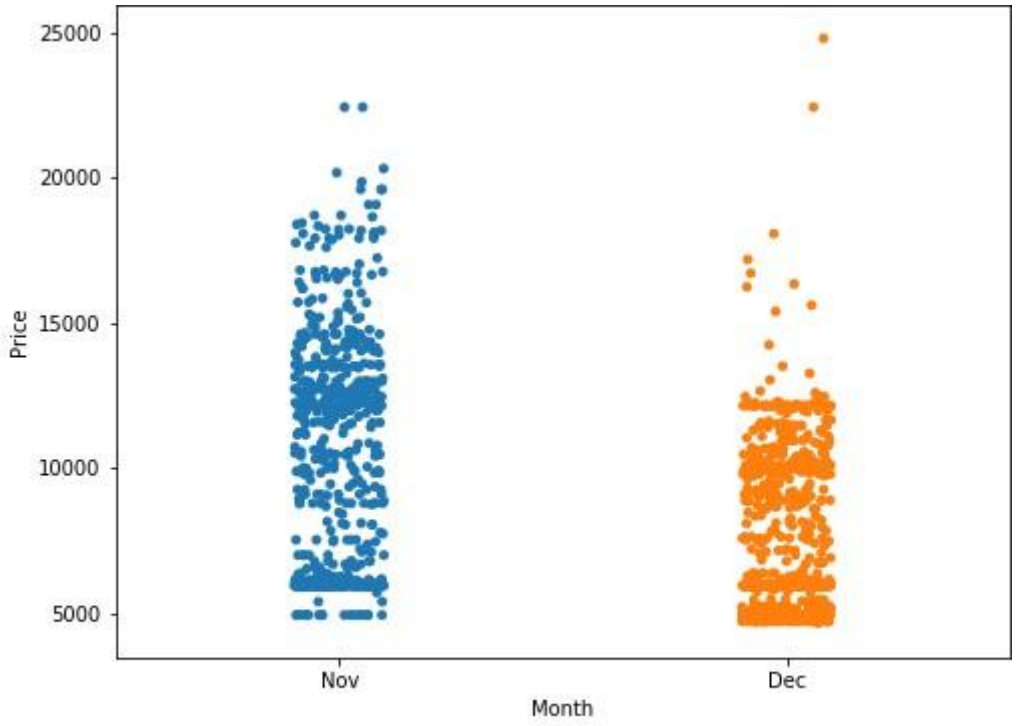
```
: sn.boxplot(df['Price'])
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x1e71118cb00>
```



Outliers are present in the dataset, but as we know that it is the price of the flights which changes on the real time, thus we will not remove the outliers here.

Month vs Price



December month has highest flight price.

Airlines vs Price

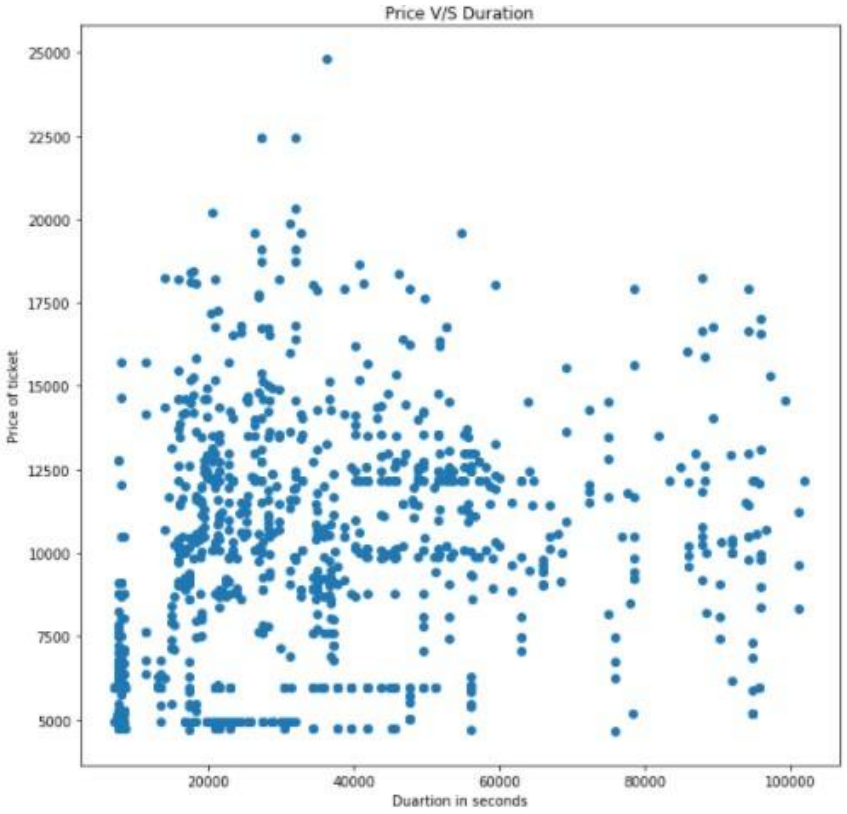
Air India and Go first shows the highest price of flight.

Date vs Price

Day vs Price

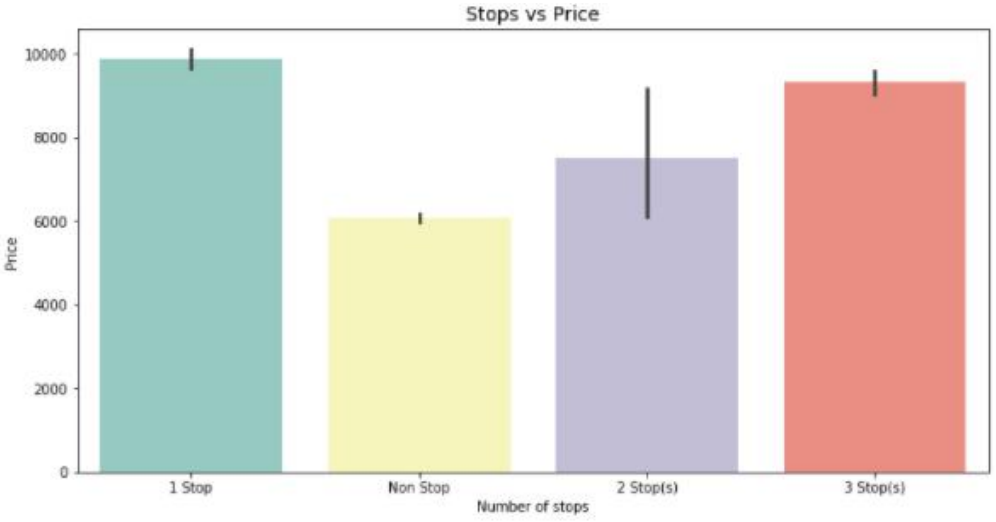
we can see that on weekend's prices are expensive than weekday's.

Price vs Duration(sec)



The above graph shows that relationship between price of the flight tiket and duration of the flight. we could see that how the price changes as there is any change in the duration of the flight.

Number of stop vs Price



One thing can be notice here is that, whichever flight has number of stops equal to one the price of flight is maximum for that and non stop flight has lowest price.

CONCLUSION

Key Findings and Conclusions of the Study

-> After the completion of this project, we got an insight of how to collect data, pre-processing the data, analyzing the data and building a model.

-> First, we collected the data flight prices from different websites like yatra and makemytrip it was done by using Web scraping. The framework used for web scraping was Selenium, which has an advantage of automating our process of collecting data.

-> We collected almost 1500 records of data which contained the flight price and other related features.

-> Then, the scrapped data was combined in a single data frame and saved in a csv file so that we can open it and analyze the data.

-> We did data cleaning, data-preprocessing steps like finding and handling null values, removing words from numbers, converting object to int type, data visualization, handling skewness, etc.

-> After separating our train and test data, we started running different machine learning classification algorithms to find out the best performing model.

-> We found that RandomForest Algorithms was performing well, according to their r2_score and cross val scores.

-> Then, we performed Hyperparameter Tuning techniques using GridSearchCV for getting the best parameters and improving the scores. In that, RandomForestRegressor performed well and we finalised that model.

> We saved or finalize the model using joblib library for further testing of the model.

The problems we faced during this project were:

- ➔Website was poorly designed because the scrapping took a lot of time and there were many issues in accessing to next page.
- ➔More negative correlated data were present.
- ➔No information for handling these fast-paced websites were informed so that we were consuming more time in web scraping itself.