FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

SUBHAJIT DAS

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I would also like to thank my mentor in Fliprobo, Sapna Verma, for providing me with the problem statement for performing this wonderful task.

Some of the reference sources are as follows:

- Coding Ninjas
- Medium.com
- Analytics Vidhya
- StackOverflow
- cleartrip.com for data collection
- Kaggle for dataset

INTRODUCTION

BUSINESS PROBLEM FRAMING

The objective of the project was to build the model to predict the price of flighths with the available independent variables. This model can then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics.

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

MOTIVATION FOR THE PROBLEM UNDERTAKEN

To understand real world problems where Machine Learning and Data Analysis can be applied to help organizations in various domains to make better decisions with the help of which they can gain profit or can be escaped from any loss which otherwise could be possible without the study of data .

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ ANALYTICAL MODELING OF THE PROBLEM

This is a Regression problem, where our end goal is to predict the Prices of flights based on data. I have collected the data from cleartrip.com and Kaggle and will be making the model based on the data I have collected.

DATA SOURCES AND THEIR FORMATS

The data was collected from cleartrip.com in csv format.

Below is the description of the data

Data Description:

1. Airline

Name of the airline the flight was booked on. There is possibility that multiple airline exist for flights with 1 stop or more

2. Date_of_Journey

Date when the journey will be commencing

3. Source

Station from where the flight will begin

4. Destination

Station where journey will end

5. Route

All the en-route destination the flight will touch, before reaching the destination.

6. Dep_Time

Time the flight will depart from the origin station

7. Arrival_Time

Time the flight will arrive at the final detination.

8. Duration

Total time the fight take to reach its final destination from origin.

9. Total_Stops

The number of stops flight take (it does not include source and destination)

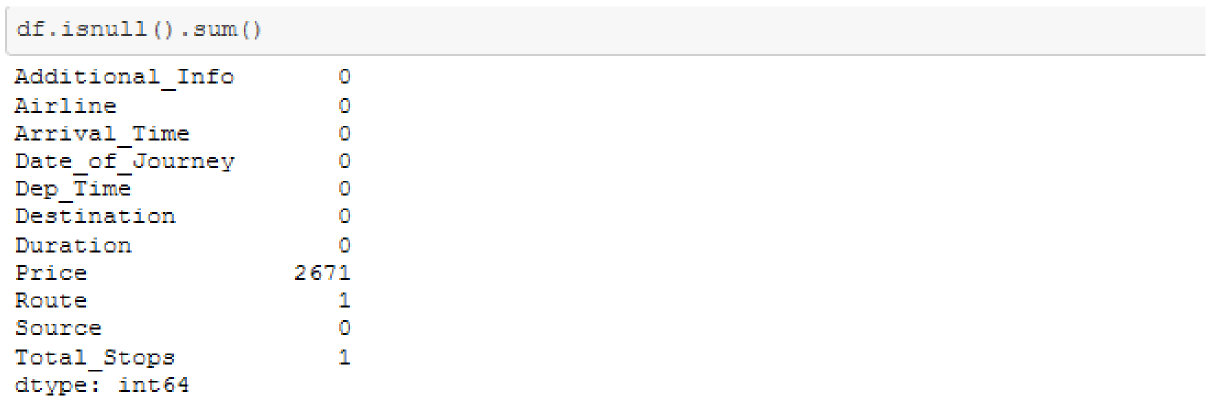
10. Price

This is our traget variable which we needs to predict, the price of the flight

DATA PREPROCESSING DONE

After loading all the required libraries we loaded the data into our jupyter notebook.

Feature Engineering has been used for cleaning of the data. Some unused columns have been deleted and even some columns have been bifurcated which was used in the prediction. We first done data cleaning. We first looked for the missing values and the same was filled.



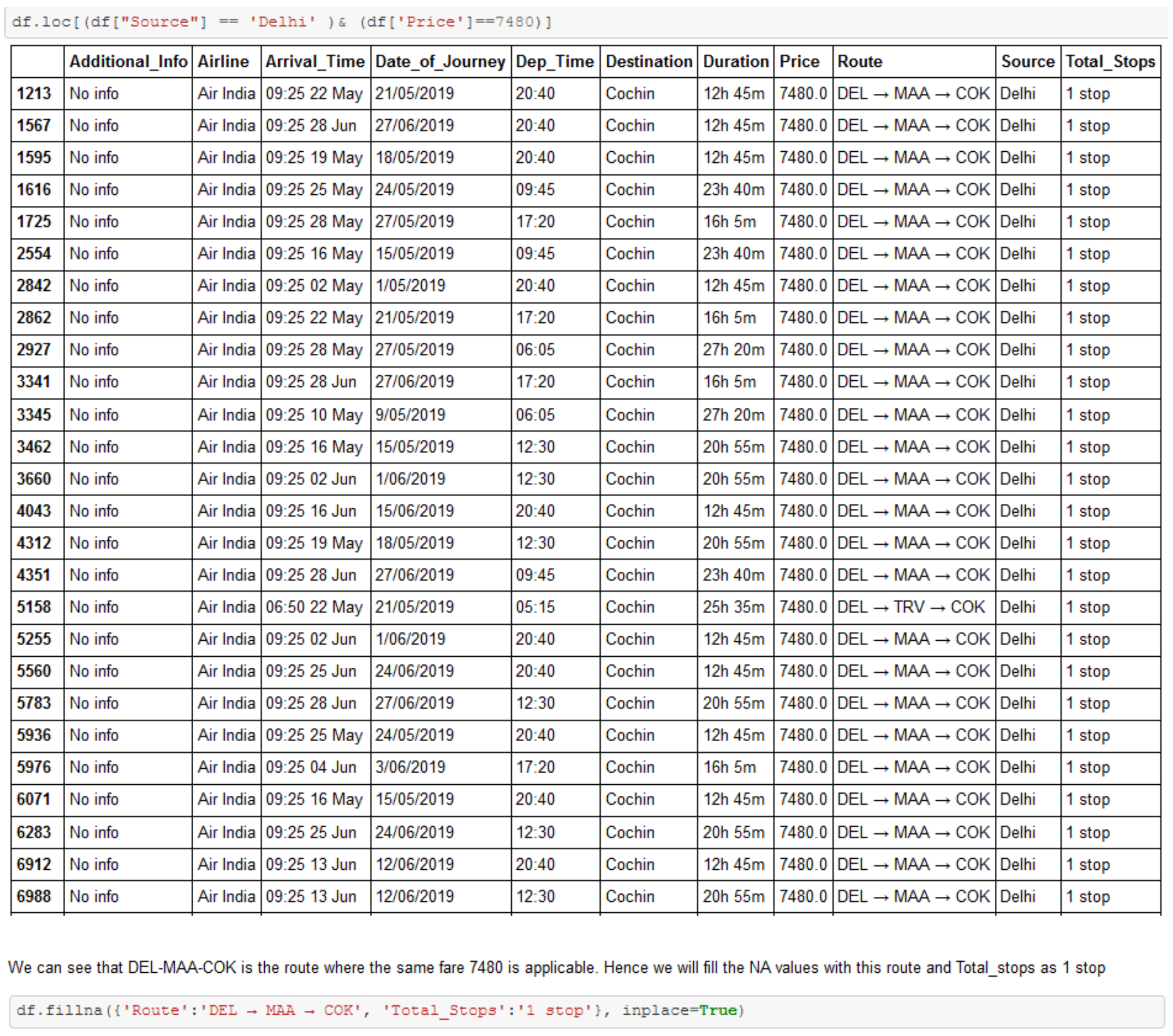
```
df.isnull().sum()
```

```
Additional_Info       0
Airline               0
Arrival_Time          0
Date_of_Journey       0
Dep_Time              0
Destination           0
Duration              0
Price              2671
Route                 1
Source                0
Total_Stops           1
dtype: int64
```

2671 null values in Price is of test data which we need to predict. There is one null value of Route and Total stops

Since we merged the train and test data set 2671 was the target variable which we need to predict and not actual Null values.

There was only one Null value in Route and Total stops. The same was filled by chceking the price for similar source and destination.



```
df.loc[(df["Source"] == 'Delhi' )& (df['Price']==7480)]
```

	Additional_Info	Airline	Arrival_Time	Date_of_Journey	Dep_Time	Destination	Duration	Price	Route	Source	Total_Stops
1213	No info	Air India	09:25 22 May	21/05/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
1567	No info	Air India	09:25 28 Jun	27/06/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
1595	No info	Air India	09:25 19 May	18/05/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
1616	No info	Air India	09:25 25 May	24/05/2019	09:45	Cochin	23h 40m	7480.0	DEL → MAA → COK	Delhi	1 stop
1725	No info	Air India	09:25 28 May	27/05/2019	17:20	Cochin	16h 5m	7480.0	DEL → MAA → COK	Delhi	1 stop
2554	No info	Air India	09:25 16 May	15/05/2019	09:45	Cochin	23h 40m	7480.0	DEL → MAA → COK	Delhi	1 stop
2842	No info	Air India	09:25 02 May	1/05/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
2862	No info	Air India	09:25 22 May	21/05/2019	17:20	Cochin	16h 5m	7480.0	DEL → MAA → COK	Delhi	1 stop
2927	No info	Air India	09:25 28 May	27/05/2019	06:05	Cochin	27h 20m	7480.0	DEL → MAA → COK	Delhi	1 stop
3341	No info	Air India	09:25 28 Jun	27/06/2019	17:20	Cochin	16h 5m	7480.0	DEL → MAA → COK	Delhi	1 stop
3345	No info	Air India	09:25 10 May	9/05/2019	06:05	Cochin	27h 20m	7480.0	DEL → MAA → COK	Delhi	1 stop
3462	No info	Air India	09:25 16 May	15/05/2019	12:30	Cochin	20h 55m	7480.0	DEL → MAA → COK	Delhi	1 stop
3660	No info	Air India	09:25 02 Jun	1/06/2019	12:30	Cochin	20h 55m	7480.0	DEL → MAA → COK	Delhi	1 stop
4043	No info	Air India	09:25 16 Jun	15/06/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
4312	No info	Air India	09:25 19 May	18/05/2019	12:30	Cochin	20h 55m	7480.0	DEL → MAA → COK	Delhi	1 stop
4351	No info	Air India	09:25 28 Jun	27/06/2019	09:45	Cochin	23h 40m	7480.0	DEL → MAA → COK	Delhi	1 stop
5158	No info	Air India	06:50 22 May	21/05/2019	05:15	Cochin	25h 35m	7480.0	DEL → TRV → COK	Delhi	1 stop
5255	No info	Air India	09:25 02 Jun	1/06/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
5560	No info	Air India	09:25 25 Jun	24/06/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
5783	No info	Air India	09:25 28 Jun	27/06/2019	12:30	Cochin	20h 55m	7480.0	DEL → MAA → COK	Delhi	1 stop
5936	No info	Air India	09:25 25 May	24/05/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
5976	No info	Air India	09:25 04 Jun	3/06/2019	17:20	Cochin	16h 5m	7480.0	DEL → MAA → COK	Delhi	1 stop
6071	No info	Air India	09:25 16 May	15/05/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
6283	No info	Air India	09:25 25 Jun	24/06/2019	12:30	Cochin	20h 55m	7480.0	DEL → MAA → COK	Delhi	1 stop
6912	No info	Air India	09:25 13 Jun	12/06/2019	20:40	Cochin	12h 45m	7480.0	DEL → MAA → COK	Delhi	1 stop
6988	No info	Air India	09:25 13 Jun	12/06/2019	12:30	Cochin	20h 55m	7480.0	DEL → MAA → COK	Delhi	1 stop

We can see that DEL-MAA-COK is the route where the same fare 7480 is applicable. Hence we will fill the NA values with this route and Total_stops as 1 stop

```
df.fillna({'Route':'DEL → MAA → COK', 'Total_Stops':'1 stop'}, inplace=True)
```

Following process was followed to clean the data

1. We can see that arrival time for some of the flights was not updated correctly. Some of the flights along with the arrival time date is also mentioned. We separated the date using the split command.

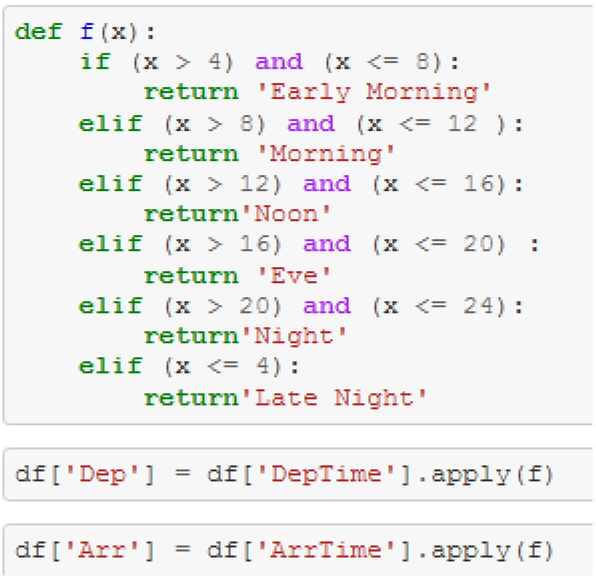
2. The destination/source value was changed from a city name to the city code. Since Route columns has city code updated. This will help machine to understand both the source and the route correctly.

3. The total duration was changed into minutes. For that first we had splitted the 'h' and 'm' from the columns and separated the hour and minutes columns.

4. We made column for each route the flights take. We named this as a Via points.

5. From the Date_of_Journey column we extracted the month.

6. The time was changed to early morning, morning, Noon, Eve and Night. For that we created the function and apply while creating the new column.



```
def f(x):
    if (x > 4) and (x <= 8):
        return 'Early Morning'
    elif (x > 8) and (x <= 12 ):
        return 'Morning'
    elif (x > 12) and (x <= 16):
        return'Noon'
    elif (x > 16) and (x <= 20) :
        return 'Eve'
    elif (x > 20) and (x <= 24):
        return'Night'
    elif (x <= 4):
        return'Late Night'
```

```
df['Dep'] = df['DepTime'].apply(f)
```

```
df['Arr'] = df['ArrTime'].apply(f)
```

7. Seasons plays an important role in determining the flight price. We all know that flight price will be high during summer and Winter vacations. Generally the fares are lower during the monsoon.

Hence we created a new column as Season and named it High, Low and Shoulder season depending on the month the flight was operating.

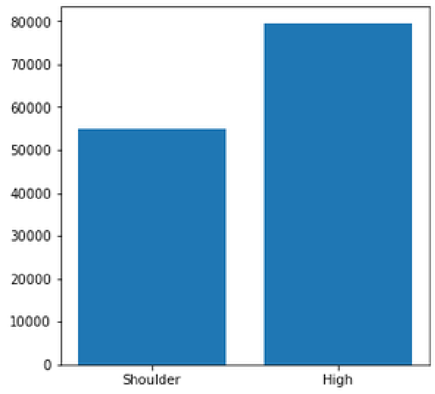
8. Each station was assigned with the same numeric value across each columns. e.g. AMD will have a numeric value 1 each column if AMD is there.

This will help machine to understand the correct source, destination and via points

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

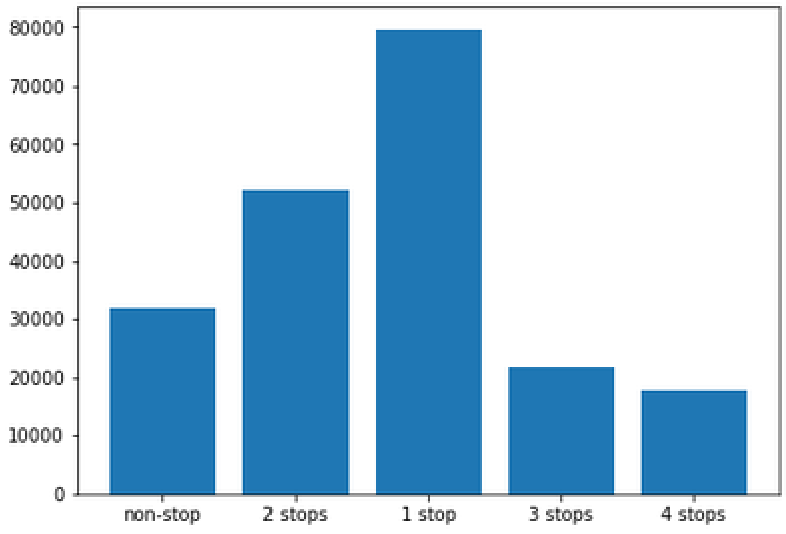
Here we check the correlation between all our feature variables with target variable label

```
fig = plt.figure(figsize =(5,5))
plt.bar(df['Season'], df['Price'])
plt.show()
```



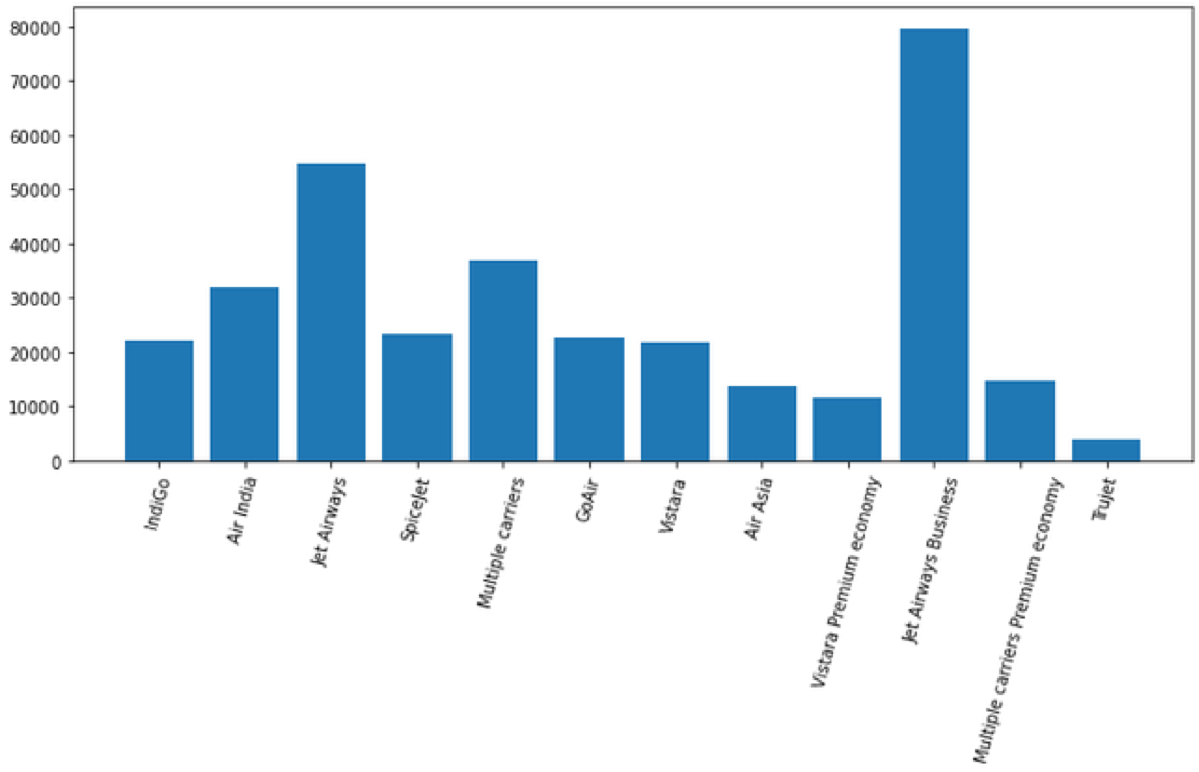
Since there is no data from the month that makes the low season, we can clearly see that price of the ticket is high during the High Season

```
fig = plt.figure(figsize =(7,5))
plt.bar(df['Total_Stops'], df['Price'])
plt.show()
```



Flight with 1 stop has the higher price. This could be because of the cabin customer is flying on.

```
fig = plt.figure(figsize =(12,5))
plt.bar(df['Airline'], df['Price'])
plt.xticks(rotation=75)
plt.show()
```



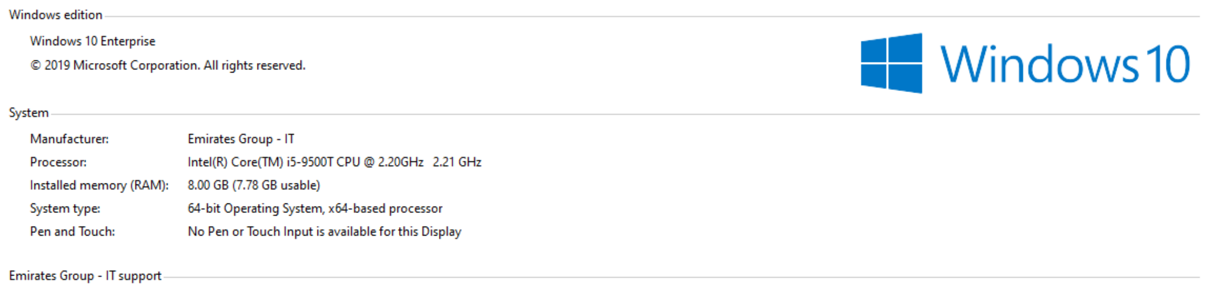
We can observe that fares are high for Jet airways business. True Jet has the the lowest price.

Set of assumptions related to the problem under consideration

By looking into the target variable label we assumed that it was a Regression type of problem.

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:



Windows edition

Windows 10 Enterprise

© 2019 Microsoft Corporation. All rights reserved.

System

Manufacturer:	Emirates Group - IT
Processor:	Intel(R) Core(TM) i5-9500T CPU @ 2.20GHz 2.21 GHz
Installed memory (RAM):	8.00 GB (7.78 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Emirates Group - IT support

SOFTWARE:

Jupyter Notebook (Anaconda 3) – `Python 3.8.5`

LIBRARIES:

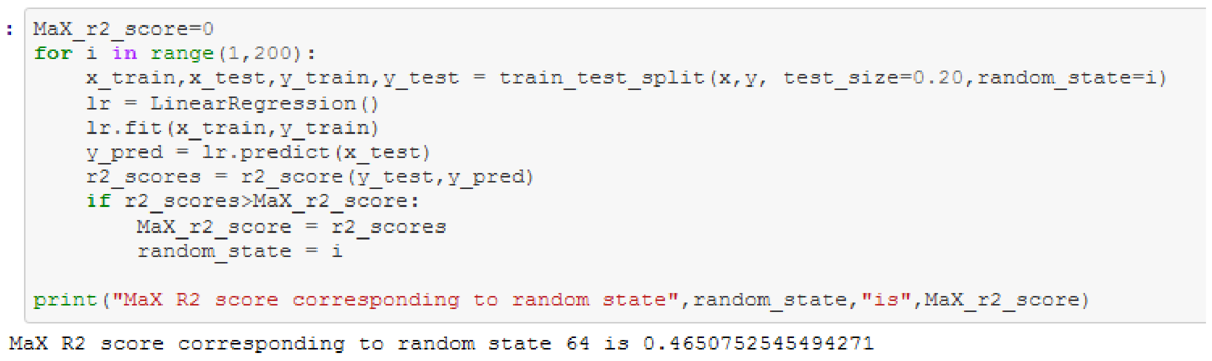
The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition sklearn standardscaler, GridSearchCV, joblib.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Lasso,Ridge,ElasticNet
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

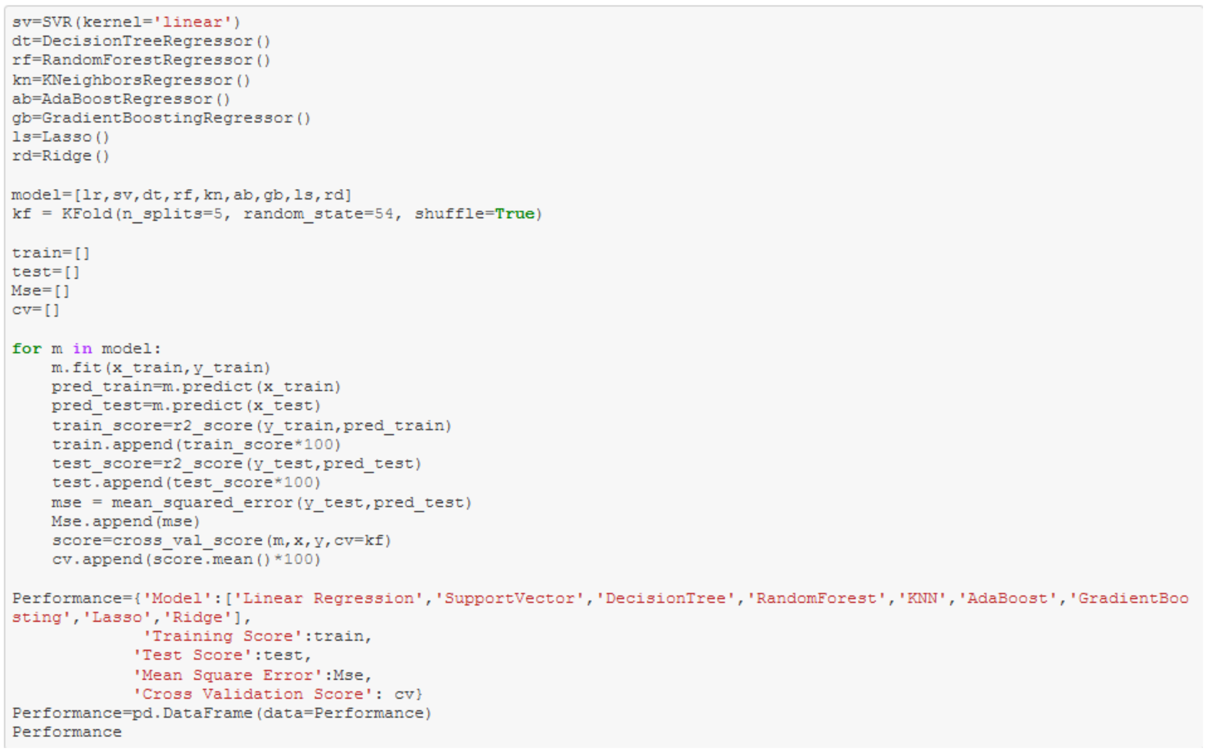
MODEL/S DEVELOPMENT AND EVALUATION



```
: MaX_r2_score=0
  for i in range(1,200):
      x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.20,random_state=i)
      lr = LinearRegression()
      lr.fit(x_train,y_train)
      y_pred = lr.predict(x_test)
      r2_scores = r2_score(y_test,y_pred)
      if r2_scores>MaX_r2_score:
          MaX_r2_score = r2_scores
          random_state = i

  print("MaX R2 score corresponding to random state",random_state,"is",MaX_r2_score)
```

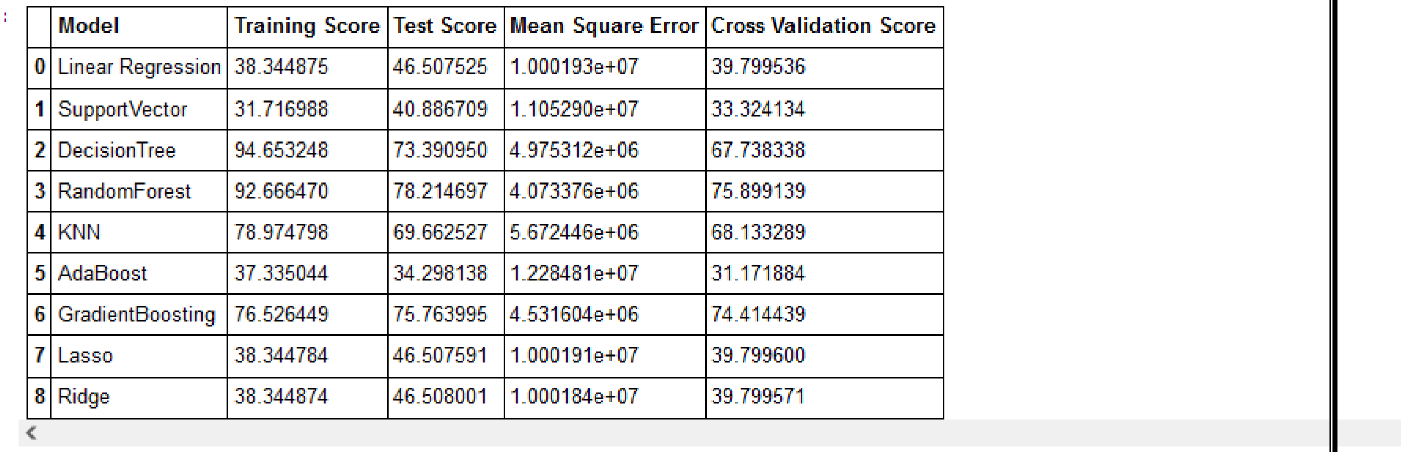
```
MaX R2 score corresponding to random state 64 is 0.4650752545494271
```

RUN AND EVALUATE SELECTED MODELS



```
sv=SVR(kernel='linear')
dt=DecisionTreeRegressor()
rf=RandomForestRegressor()
kn=KNeighborsRegressor()
ab=AdaBoostRegressor()
gb=GradientBoostingRegressor()
ls=Lasso()
rd=Ridge()

model=[lr,sv,dt,rf,kn,ab,gb,ls,rd]
kf = KFold(n_splits=5, random_state=54, shuffle=True)

train=[]
test=[]
Mse=[]
cv=[]

for m in model:
    m.fit(x_train,y_train)
    pred_train=m.predict(x_train)
    pred_test=m.predict(x_test)
    train_score=r2_score(y_train,pred_train)
    train.append(train_score*100)
    test_score=r2_score(y_test,pred_test)
    test.append(test_score*100)
    mse = mean_squared_error(y_test,pred_test)
    Mse.append(mse)
    score=cross_val_score(m,x,y,cv=kf)
    cv.append(score.mean()*100)

Performance={'Model':['Linear Regression','SupportVector','DecisionTree','RandomForest','KNN','AdaBoost','GradientBoo
sting','Lasso','Ridge'],
             'Training Score':train,
            'Test Score':test,
            'Mean Square Error':Mse,
            'Cross Validation Score': cv}
Performance=pd.DataFrame(data=Performance)
Performance
```

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION



	Model	Training Score	Test Score	Mean Square Error	Cross Validation Score
0	Linear Regression	38.344875	46.507525	1.000193e+07	39.799536
1	SupportVector	31.716988	40.886709	1.105290e+07	33.324134
2	DecisionTree	94.653248	73.390950	4.975312e+06	67.738338
3	RandomForest	92.666470	78.214697	4.073376e+06	75.899139
4	KNN	78.974798	69.662527	5.672446e+06	68.133289
5	AdaBoost	37.335044	34.298138	1.228481e+07	31.171884
6	GradientBoosting	76.526449	75.763995	4.531604e+06	74.414439
7	Lasso	38.344784	46.507591	1.000191e+07	39.799600
8	Ridge	38.344874	46.508001	1.000184e+07	39.799571

Observation

1. GradientBoosting model has performed well with least difference between test and CV score.

2. GradientBoosting test accuracy is 75% with CV score of 74%

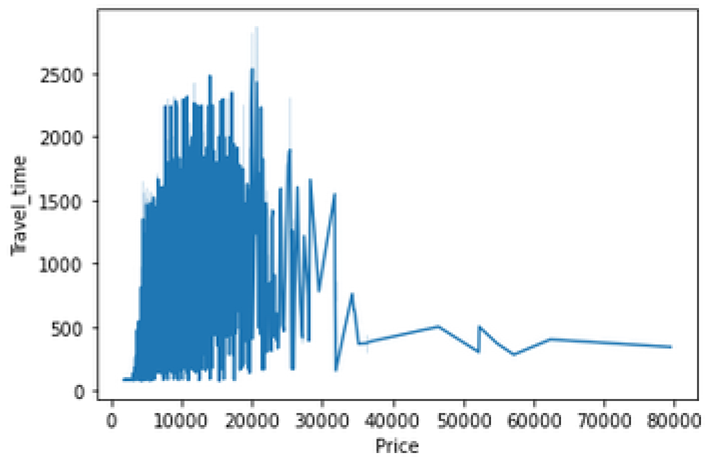
3. RandomForest has given us the best test accuracy of 78% with CV score of 75% ¶

We used the R2 score and difference between test and cross validation to select the model.

VISUALIZATION

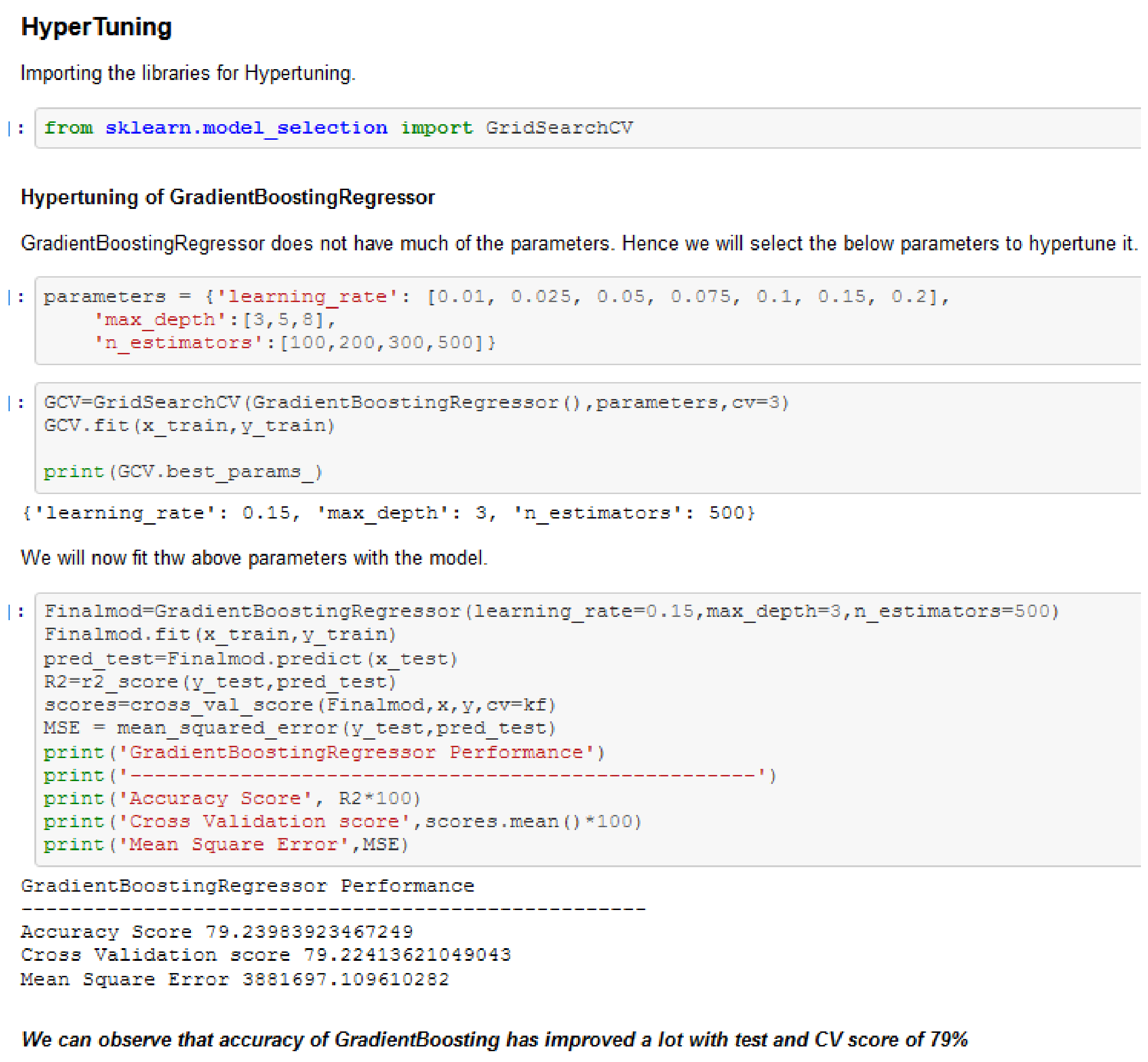
Below technique was used for data visualization

```
: sns.lineplot(x='Price',y='Travel_time',data=df)

: <AxesSubplot:xlabel='Price', ylabel='Travel_time'>
```



There is no corelation we can find with travel time and price.

INTERPRETATION OF THE RESULTS



HyperTuning

Importing the libraries for Hypertuning.

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

Hypertuning of GradientBoostingRegressor

GradientBoostingRegressor does not have much of the parameters. Hence we will select the below parameters to hypertune it.

```
parameters = {'learning_rate': [0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2],
     'max_depth':[3,5,8],
     'n_estimators':[100,200,300,500]}
```

```
GCV=GridSearchCV(GradientBoostingRegressor(),parameters,cv=3)
GCV.fit(x_train,y_train)

print(GCV.best_params_)
```

```
{'learning_rate': 0.15, 'max_depth': 3, 'n_estimators': 500}
```

We will now fit thw above parameters with the model.

```
Finalmod=GradientBoostingRegressor(learning_rate=0.15,max_depth=3,n_estimators=500)
Finalmod.fit(x_train,y_train)
pred_test=Finalmod.predict(x_test)
R2=r2_score(y_test,pred_test)
scores=cross_val_score(Finalmod,x,y,cv=kf)
MSE = mean_squared_error(y_test,pred_test)
print('GradientBoostingRegressor Performance')
print('---------------------------------------------------')
print('Accuracy Score', R2*100)
print('Cross Validation score',scores.mean()*100)
print('Mean Square Error',MSE)
```

```
GradientBoostingRegressor Performance
---------------------------------------------------
Accuracy Score 79.23983923467249
Cross Validation score 79.22413621049043
Mean Square Error 3881697.109610282
```

We can observe that accuracy of GradientBoosting has improved a lot with test and CV score of 79%

Our model performance increased slightly after Hypertuning.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

In this project we have tried to show how the flight prices vary and what are the factors related to the changing of flight prices. The Gradient boosting regressor model has performed well in predicting the prices..

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

This project has demonstrated the importance of modelling and predicting data.

Through different powerful tools of visualization we were able to analyse and interpret different hidden insights about the data.

Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or underfitting.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.