FLIP ROBO

Flight Price Prediction Project

Submitted by:

Raghavulu Patnala

ACKNOWLEDGMENT

Thanks for giving me the opportunity to work in Flip Robo Technologies as Intern and would like to express my gratitude to Data Trained Institute as well for trained me in Data Science Domain. This helps me to do my projects well and understand the concepts.

Resources Referred – Google, GitHub, Blogs for conceptual referring

INTRODUCTION

- ## Business Problem Framing

We need to predict the flight price here as we know already that flight prices will vary due to many factors like festive time and booking ticket at a last moment and based on airlines also prices will vary too.

Keeping the flight full as they want it because last minute purchases are expensive.

Depends on the route and the duration between the places. This is also one of the factors that they can raise the price of the ticket at any time.

- ## Motivation for the Problem Undertaken

Due to the high price strategy, all cannot afford to travel in flight, and it is the fastest mode of travel now a days.

Here, Predicting the prices will help us to know the cheapest and best route and it will help us to find the price of the flight.

This will help the all kinds of people to conclude when the prices will be high and when it will be less.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  Target variable is **Price** for our problem. Hence It is *Regression Problem as the data is continuous variable*.

- Data Sources and their formats

  Data has been collected by me from one of the official websites of flight and it has 2305 rows and 9 columns.

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

	Unnamed: 0	AirlineName	Stops	Duration	Source	Destination	DepartureTime	ArrivalTime	Price
0	0	Air India	direct	2h 30m	BLR Bengaluru Intl	DEL Indira Gandhi Intl	13:20	15:50	7889
1	1	Air India	direct	2h 35m	BLR Bengaluru Intl	DEL Indira Gandhi Intl	10:25	13:00	6936
2	2	Vistara	direct	2h 35m	BLR Bengaluru Intl	DEL Indira Gandhi Intl	21:10	23:45	7484
3	3	SpiceJet	direct	2h 40m	BLR Bengaluru Intl	DEL Indira Gandhi Intl	21:55	00:35	7472
4	4	Vistara	direct	2h 40m	BLR Bengaluru Intl	DEL Indira Gandhi Intl	17:50	20:30	7484
...	...	...	...	...	...	...	...	...	...
2300	748	United Airlines	2 stops	41h 30m	LHR Heathrow	DEL Indira Gandhi Intl	10:25	09:25	1154916
2301	749	Air Canada	1 stop	36h 55m	LHR Heathrow	DEL Indira Gandhi Intl	15:00	09:25	1276588
2302	750	Air Canada	1 stop	40h 45m	LHR Heathrow	DEL Indira Gandhi Intl	11:10	09:25	1276588
2303	751	Austrian Airlines	3 stops	41h 40m	USM Ko Samui	LHR Heathrow	19:40	06:20	1595550
2304	752	Austrian Airlines	3 stops	46h 10m	USM Ko Samui	LHR Heathrow	19:40	10:50	1714796

2305 rows × 9 columns

Data doesn't have any null values or missing data. So, we are good to pre-process the data.

```python
# Summary of each column and its datatype,

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2305 entries, 0 to 2304
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   AirlineName    2305 non-null   object
 1   Stops          2305 non-null   object
 2   Duration       2305 non-null   object
 3   Source         2305 non-null   object
 4   Destination    2305 non-null   object
 5   DepartureTime  2305 non-null   object
 6   ArrivalTime    2305 non-null   object
 7   Price          2305 non-null   int64 
dtypes: int64(1), object(7)
memory usage: 144.2+ KB
```

information of the dataset is data type is 8 object type and no null values present in a dataset and total 2305 rows and 8 columns

```python
# Checking if any null values in a dataset,

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

```
AirlineName      0
Stops            0
Duration         0
Source           0
Destination      0
DepartureTime    0
ArrivalTime      0
Price            0
dtype: int64
```

- Data Pre-processing Done

We can see that we have object datatypes for most of the columns in dataset.

So, we need to convert those categorical columns into numerical columns as pre- processing step for better model.

I am applying Datetime index to split the hour and minute from departure / arrival time and duration columns.

```python
# splitting duration column which has string and integer,

df['Duration'] = df['Duration'].str.split(' ')
df['dur_hr'] = df['Duration'].str[0]
df['dur_hr'] = df['dur_hr'].str.split('h')
df['dur_hr'] = df['dur_hr'].str[0]

df['dur_min'] = df['Duration'].str[1]
df['dur_min'] = df['dur_min'].str.split('m')
df['dur_min'] = df['dur_min'].str[0]


# changing datatype into Int for duration hour and min column,

df['dur_hr'] = df['dur_hr'].astype(int)
df['dur_min'] = df['dur_min'].astype(int)

# dropping duration columns,

df = df.drop(columns = ['Duration'], axis = 1)
df.head()
```

	AirlineName	Stops	Source	Destination	Price	dep_hr	dep_min	arr_hr	arr_min	dur_hr	dur_min
0	Air India	direct	BLR Bengaluru Intl	DEL Indira Gandhi Intl	7889	13	20	15	50	2	30
1	Air India	direct	BLR Bengaluru Intl	DEL Indira Gandhi Intl	6936	10	25	13	0	2	35
2	Vistara	direct	BLR Bengaluru Intl	DEL Indira Gandhi Intl	7484	21	10	23	45	2	35
3	SpiceJet	direct	BLR Bengaluru Intl	DEL Indira Gandhi Intl	7472	21	55	0	35	2	40
4	Vistara	direct	BLR Bengaluru Intl	DEL Indira Gandhi Intl	7484	17	50	20	30	2	40

As you can see from the above snap, we still have stops, departure and arrival place as categorical column. Applying **Label Encoder ()** Technique to rest of the column,

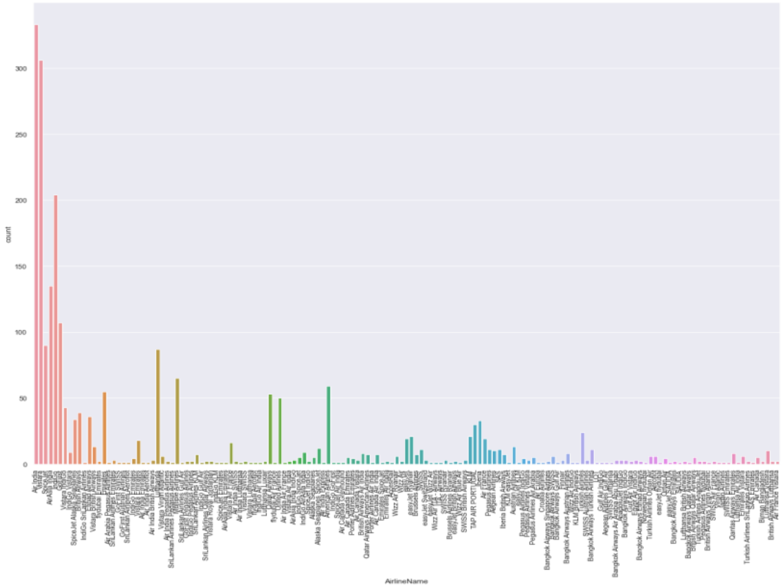
```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
col = ['AirlineName','Stops','Source','Destination']
for i in col:
    df[col] = df[col].apply(le.fit_transform)

df
```

	AirlineName	Stops	Source	Destination	Price	dep_hr	dep_min	arr_hr	arr_min	dur_hr	dur_min
0	13	4	0	2	7889	13	20	15	50	2	30
1	13	4	0	2	6936	10	25	13	0	2	35
2	127	4	0	2	7484	21	10	23	45	2	35
3	109	4	0	2	7472	21	55	0	35	2	40
4	127	4	0	2	7484	17	50	20	30	2	40
...	...	...	...	...	...	...	...	...	...	...	...
2300	125	2	7	2	1154916	10	25	9	25	41	30
2301	5	1	7	2	1276588	15	0	9	25	36	55
2302	5	1	7	2	1276588	11	10	9	25	40	45
2303	29	3	10	9	1595550	19	40	6	20	41	40
2304	29	3	10	9	1714796	19	40	10	50	46	10

2305 rows × 11 columns

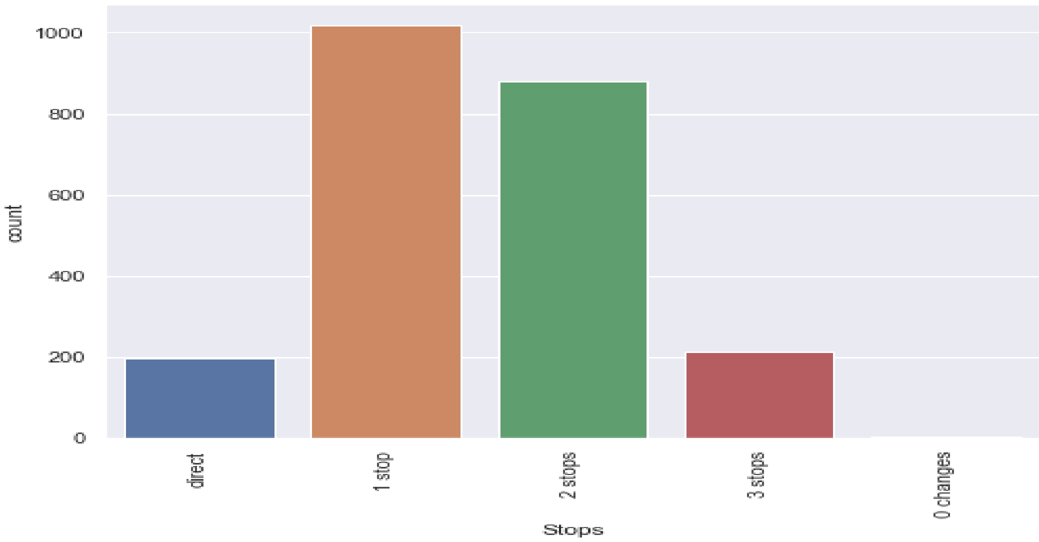
- Data Inputs- Logic- Output Relationships

  Aa we can see that there are different types airline names as per mentioned below chart,we can see most of the flights names are AirIndia,Vistara,Indigo

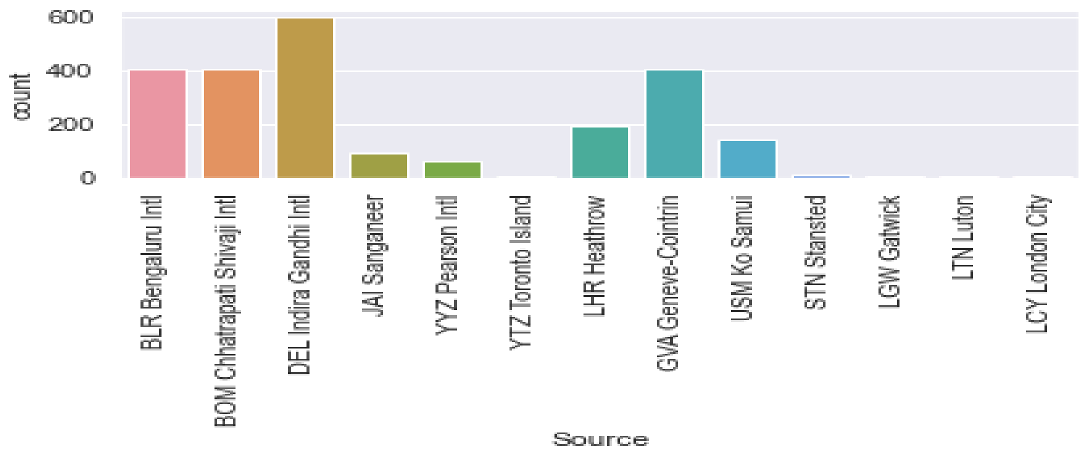


As we can see that there are stops such as – Direct, 1 /2/3 stops and either we can change or not change.

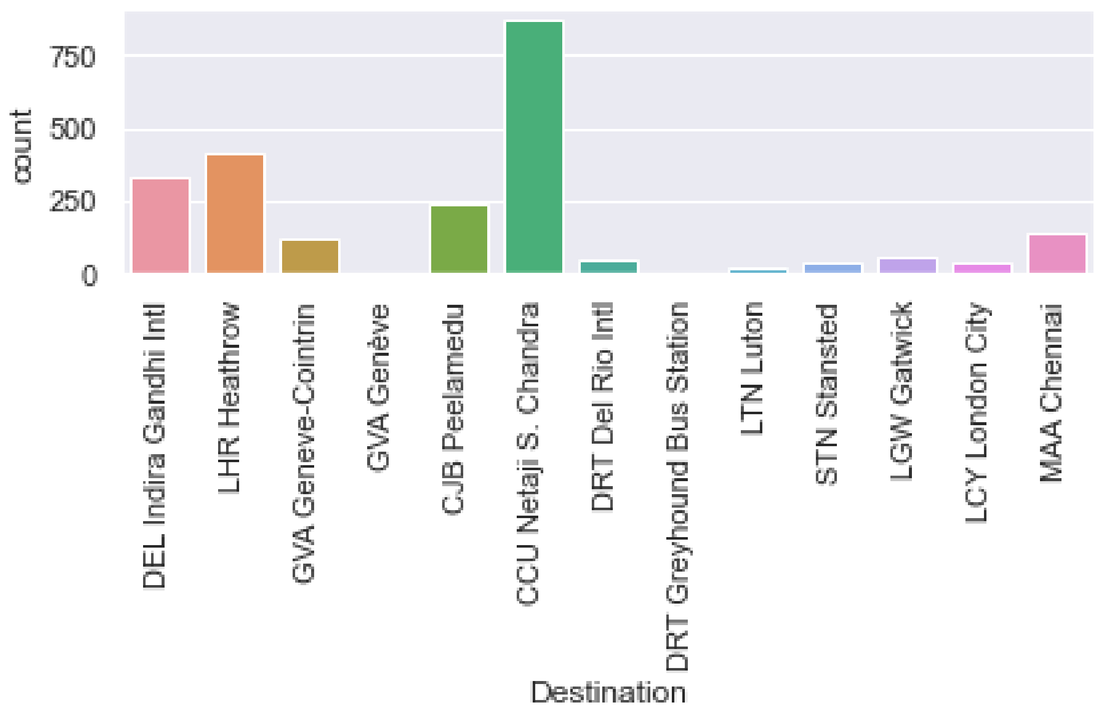
So as per the below chart, we can see that most of the flights has minimum 1 or 2 stops to proceed further.



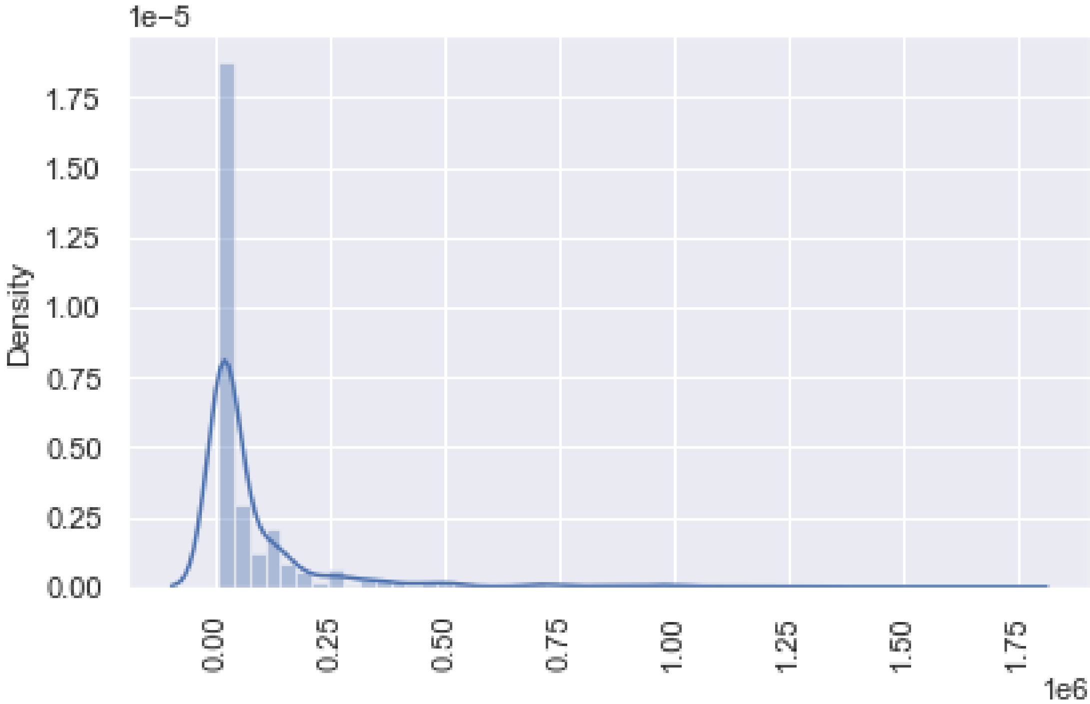
Passengers most booked departure place is DEL (Indhira Gandhi Intl)



Most of the arrival place is CCU Netaji S Chandra



Target variable price has some skewness on right side and its because depends on place, price will be varying



- Hardware and Software Requirements and Tools Used

  Libraries – Scikit Learn, Pandas, NumPy

  Label Encoder to encode the categorical values and convert into Numerical values.

  Metric – MSE, RMSE, R2 Score

  Model Selection – Train_Test _split for splitting the data into trainand test dataset.

  CV Score to check the model is over fit or under fit.

  Grid Search CV for hyper parameter tuning the model

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

1. Random Forest Regressor
2. Gradient Boost Regressor
3. Ada Boost Regressor
4. K Neighbors Regressor

- Run and evaluate selected models

```
# SPlitting X and Y

x = df_new.drop(columns = ['Price'])
y = df_new['Price']
```

```
#Scaling the data for normalize the range of values to 0-1.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

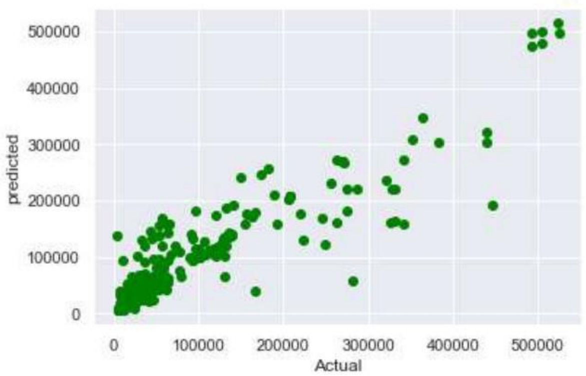
```
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
```

```
# Train test split

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.20, random_state = 555
```

```
# Random Forest regressor
rfr = RandomForestRegressor()
rfr.fit(x_train,y_train)
y_pred = rfr.predict(x_test)
scr_rfr = cross_val_score(rfr,x,y,cv=5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_rfr.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", rfr.score(x_train,y_train))
print("Test Score", rfr.score(x_test,y_test))
```

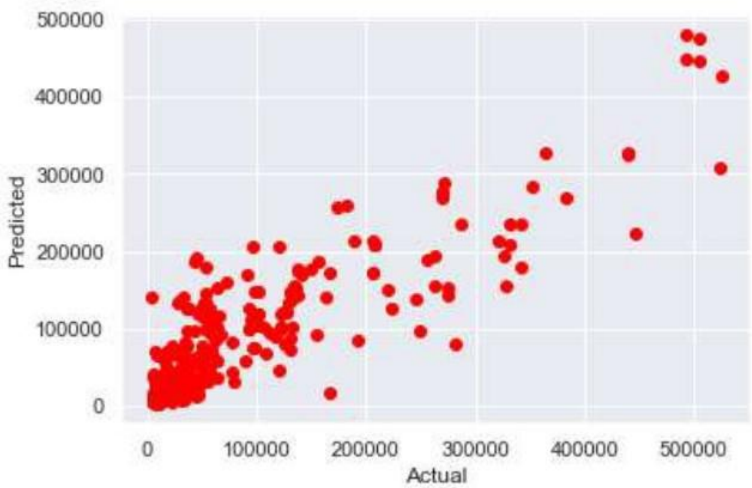
```
r2_Score 0.8604489217259426
CV Score 0.8311985772356756
MSE 1270851750.2627993
RMSE 35649.007703760835
Train Score 0.9766959077607412
Test Score 0.8604489217259426
```

```
plt.scatter(y_test,y_pred, color = 'green')       #Scatter Matrix for Actual VS predicted f
plt.xlabel("Actual")
plt.ylabel("predicted")
plt.show()
```



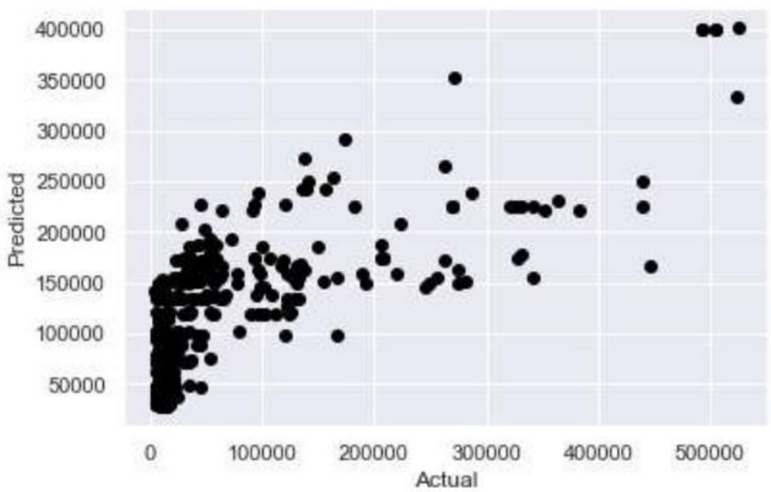
```python
# Gradient Boost Regression

from sklearn.ensemble import GradientBoostingRegressor

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

scr_gb = cross_val_score(gb,x,y,cv = 5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_gb.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", gb.score(x_train,y_train))
print("Test Score", gb.score(x_test,y_test))

plt.scatter(y_test,y_pred, color = 'red')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
r2_Score 0.7957871144618289
CV Score 0.750327822503668
MSE 1859708332.0469546
RMSE 43124.33572876172
Train Score 0.8364313549319184
Test Score 0.7957871144618289
```



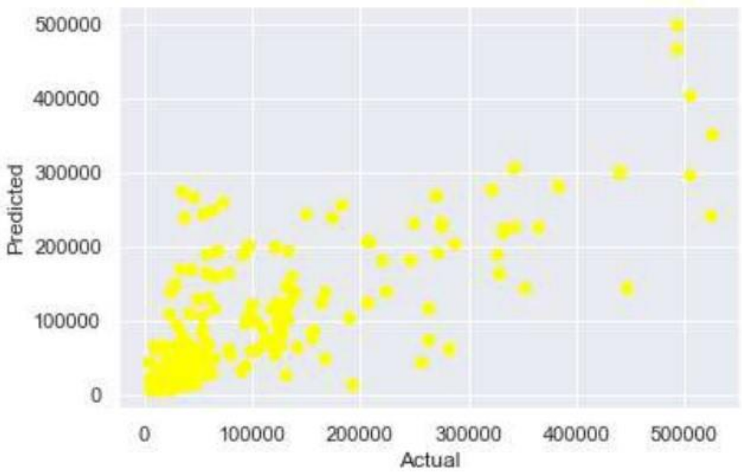
```python
#AdaBoostRegressor

ab = AdaBoostRegressor()
ab.fit(x_train,y_train)
y_pred = ab.predict(x_test)

scr_ab = cross_val_score(ab,x,y,cv = 5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_ab.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", ab.score(x_train,y_train))
print("Test Score", ab.score(x_test,y_test))

plt.scatter(y_test,y_pred, color = 'black')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
r2_Score 0.21178847713732152
CV Score 0.23983500085952233
MSE 7178016865.1950655
RMSE 84723.17785113507
Train Score 0.25967663959613807
Test Score 0.21178847713732152
```



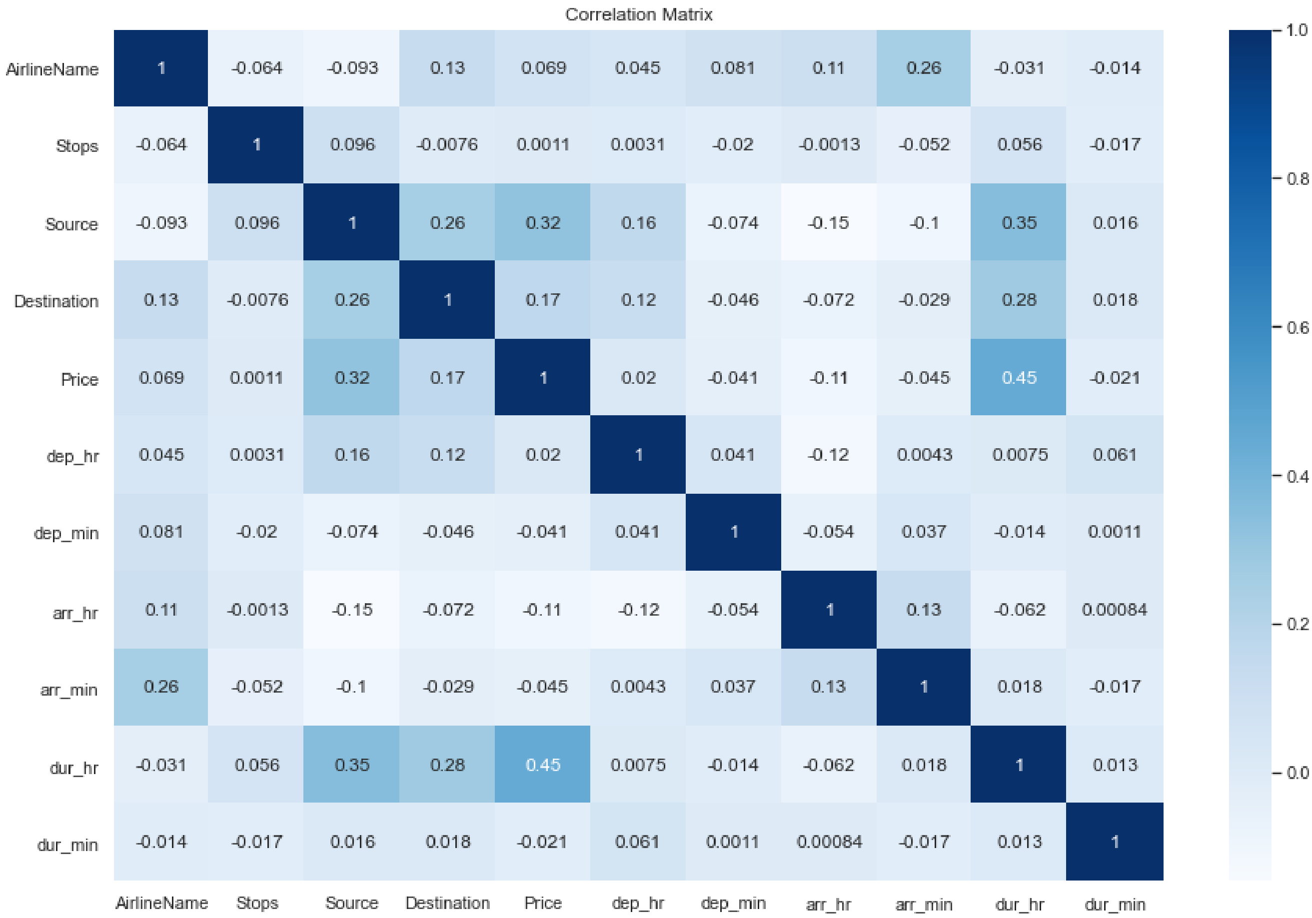
```
# K Neighbors regression

knr = KNeighborsRegressor(n_neighbors = 5)
knr.fit(x_train,y_train)
y_pred = knr.predict(x_test)

scr_knr = cross_val_score(knr,x,y,cv=5)

print("r2_Score", r2_score(y_test,y_pred))
print("CV Score", scr_knr.mean())
print("MSE",mean_squared_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("Train Score", knr.score(x_train,y_train))
print("Test Score", knr.score(x_test,y_test))

plt.scatter(y_test,y_pred, color = 'yellow')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.show()
```

```
r2_Score 0.6906919227930592
CV Score 0.6288493939944896
MSE 2816780179.3469625
RMSE 53073.34716547433
Train Score 0.7627770313893869
Test Score 0.6906919227930592
```



- Visualizations

To visualize the graphs, we have used matplotlib library and seaborn library.



- Interpretation of the Results

```
from sklearn.model_selection import GridSearchCV
#creating parameter list of random forest regressor to pass in GridSearchCV
parameters = {'bootstrap': [True, False],
              'n_estimators': [1,100],
             'criterion': ['squared_error','mse','absolute_error','mae'],
             'min_samples_leaf': [1, 2, 4],
             'min_samples_split' : [1,10],
             'max_depth' : [1,10],
             'max_features' : ['auto', 'sqrt', 'log2']}
```

```
grid = GridSearchCV(RandomForestRegressor(),parameters,cv = 5)
grid.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'bootstrap': [True, False],
                         'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'max_depth': [1, 10],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [1, 10],
                         'n_estimators': [1, 100]})
```

```
grid.best_params_
```

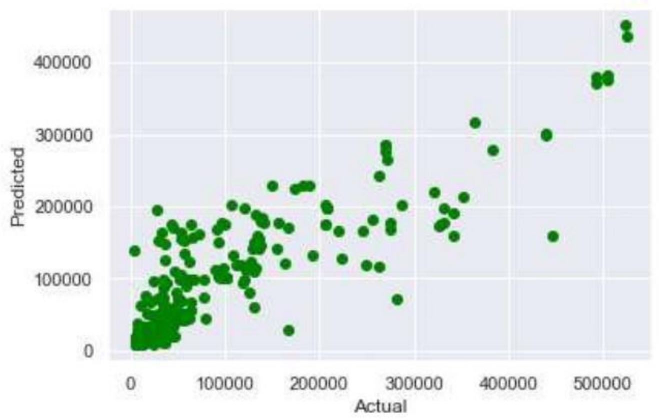
```
{'bootstrap': False,
 'criterion': 'mse',
 'max_depth': 10,
 'max_features': 'log2',
 'min_samples_leaf': 1,
 'min_samples_split': 10,
 'n_estimators': 100}
```

```
Fnal_model = RandomForestRegressor(bootstrap= 'True',criterion = 'mse',max_depth = 20,max_
                                          min_samples_split = 10 ,n_estimators = 100,min_s
Fnal_model.fit(x_train,y_train)
pred = Fnal_model.predict(x_test)
print("accuracy score of the final model is :",r2_score(y_test,pred))
```

```
accuracy score of the final model is : 0.8210548199147409
```

```
plt.scatter(y_test,y_pred, color = 'green')
plt.xlabel("Actual")
plt.ylabel("Predicted")
```

```
Text(0, 0.5, 'Predicted')
```



The final model has an accuracy of the 82.10 %

CONCLUSION

- Key Findings and Conclusions of the Study

  As this project is about predicting the prices of flight, it is a regression problem as the target variables are continuous range.

  Used r2 score, MSE as a metrics to calculate the model accuracy.

  Data is Collected by me from kayak.co for predicting the price of flight.

  The dataset doesn’t have any null or missing values.

- Learning Outcomes of the Study in respect of Data Science

  Random forest and Gradient Boost Algorithm have high accuracy score and I have used GridSearch CV for Hyper parametertuning to check accuracy score finally I got after tuning 83%.

  This is kind of different as the data is not present and we need to collect it to build a model but helps me to learn more and most important is that I am getting hands-on experience more on Data Science Concepts.

  Thanks, Flip Robo and Data Trained for this wonderful Opportunity!!