FLIP ROBO

Flight Price Prediction Project

Submitted by:

Atharva Ramesh Shelar

ACKNOWLEDGMENT

I would like to express my special gratitude to the FlipRobo team for giving me this opportunity to deal with a real-life dataset and for helping me improve my analytical and observation skills.

Thanks to DataTrained, who facilitated my internship at FlipRobo.

References used in this project

1. Documentation and GitHub repository for DataTraind

2. scikit-learn library documentation

3. Toward data science blogs

4. Airline: Wikipedia

5. Fox News: "9 surprising factors that influence the price of your airline ticket"

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

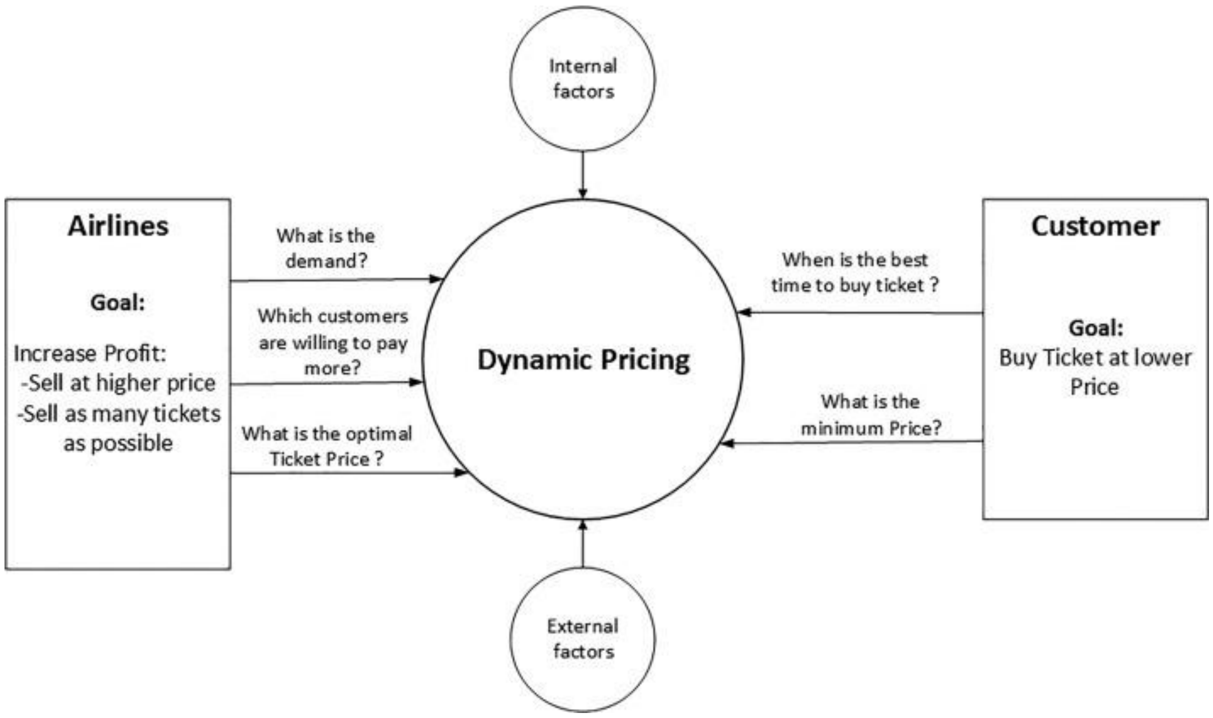
purchases)

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.

- Conceptual Background of the Domain Problem

On the airlines side, the main goal is increasing revenue and maximizing profit. According to Narangajavana et al., 2014, airlines utilize various kinds of pricing strategies to determine optimal ticket prices: long-term pricing policies, yield pricing which describes the impact of production conditions on ticket prices, and dynamic pricing which is mainly associated with dynamic adjustment of ticket prices in response to various influencing factors. Long term-pricing policies and yield pricing are associated with internal working of the specific airline and do not help that much in predicting dynamic fluctuations in price. On the other hand, dynamic pricing enables a more optimal forecasting of ticket prices based on vibrant factors such as changes in demand and price discrimination (Malighetti et al., 2009). However, dynamic pricing is challenging as it is highly influenced by various factors including internal factors, external factors, competition among airlines and strategic customers. Internal factors consist of features such as historical ticket price data, ticket purchase date and departure date, season, holidays, supply (number of available airlines and flights), fare class, availability of seats, recent market demand and flight distance. External factors include features such as occurrence of some event at the origin or destination city like terrorist attacks, natural disaster (hurricane, earthquake, tsunami, etc.), political instability (protest, strike, coup, resignation), concerts, festivals, conferences, political gatherings and sports events, competitors' promotions, weather conditions and economic activities.


Internal factors
Airlines
Goal:
Increase Profit:
-Sell at higher price
-Sell as many tickets as possible
What is the demand?
Which customers are willing to pay more?
What is the optimal Ticket Price ?
Dynamic Pricing
When is the best time to buy ticket ?
What is the minimum Price?
Customer
Goal:
Buy Ticket at lower Price
External factors

- ## Motivation for the Problem Undertaken

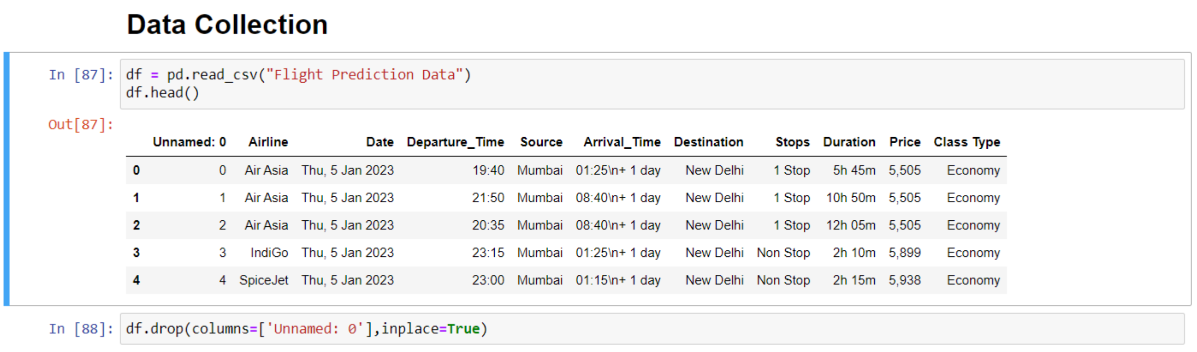
As part of the internship program, the project is collaborating with Fliprobo Technologies.

to early prediction of price to get wide idea about flight pricing Flight prices are affected by many factors, for example, class type, same-day flight ticket, early booking, distance, and many others.

Analytical Problem Framing

- ### Data Sources and their formats

Using Selenium, data from yatra.com is scraped from January 6, 2023, to January 12, 2023. Data is scraped for the Mumbai-Delhi flight in economy class, business class, and premium economy class. Around 2000 flight data are collected.



Data Collection

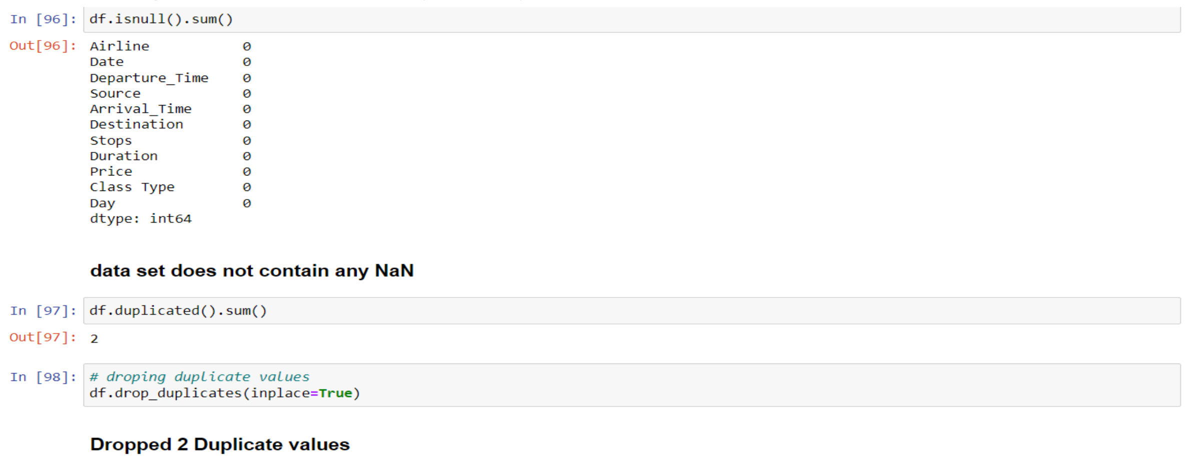
```
In [87]: df = pd.read_csv("Flight Prediction Data")
         df.head()
Out[87]:
```

	Unnamed: 0	Airline	Date	Departure_Time	Source	Arrival_Time	Destination	Stops	Duration	Price	Class Type
0	0	Air Asia	Thu, 5 Jan 2023	19:40	Mumbai	01:25\n+ 1 day	New Delhi	1 Stop	5h 45m	5,505	Economy
1	1	Air Asia	Thu, 5 Jan 2023	21:50	Mumbai	08:40\n+ 1 day	New Delhi	1 Stop	10h 50m	5,505	Economy
2	2	Air Asia	Thu, 5 Jan 2023	20:35	Mumbai	08:40\n+ 1 day	New Delhi	1 Stop	12h 05m	5,505	Economy
3	3	IndiGo	Thu, 5 Jan 2023	23:15	Mumbai	01:25\n+ 1 day	New Delhi	Non Stop	2h 10m	5,899	Economy
4	4	SpiceJet	Thu, 5 Jan 2023	23:00	Mumbai	01:15\n+ 1 day	New Delhi	Non Stop	2h 15m	5,938	Economy

```
In [88]: df.drop(columns=['Unnamed: 0'],inplace=True)
```

- ### Data Preprocessing Done

No missing values and only 2 duplicate entries in dataset



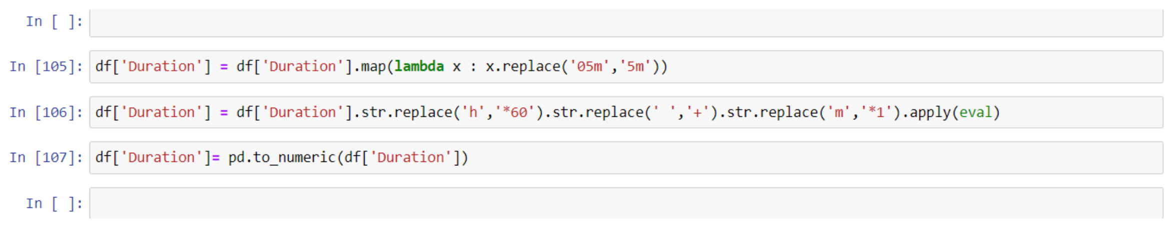
```
In [96]: df.isnull().sum()
Out[96]: Airline           0
         Date              0
         Departure_Time    0
         Source            0
         Arrival_Time      0
         Destination       0
         Stops             0
         Duration          0
         Price             0
         Class Type        0
         Day               0
         dtype: int64
```

data set does not contain any NaN

```
In [97]: df.duplicated().sum()
Out[97]: 2

In [98]: # droping duplicate values
         df.drop_duplicates(inplace=True)
```

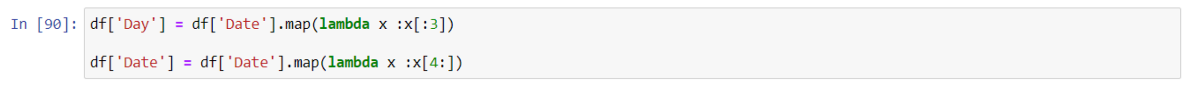
Dropped 2 Duplicate values

Conversion of from hour&min format into Minutes format



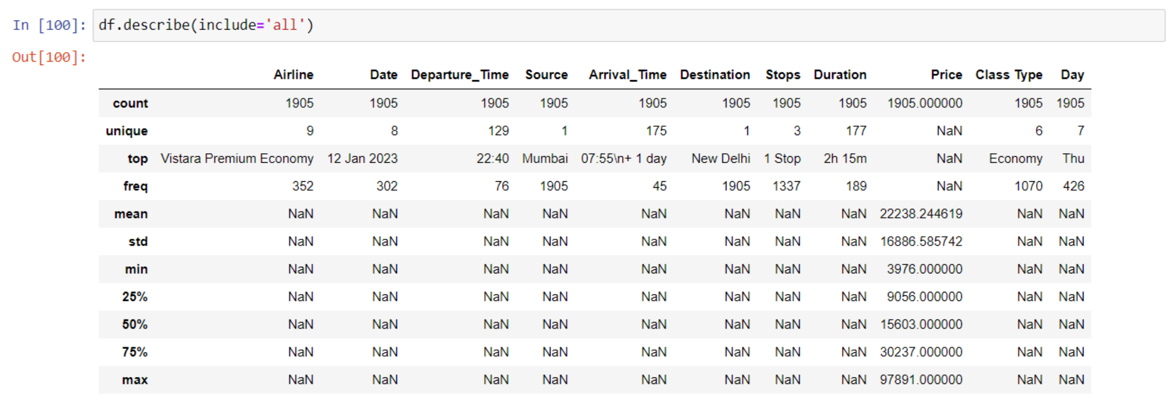
```
In [ ]:

In [105]: df['Duration'] = df['Duration'].map(lambda x : x.replace('05m','5m'))

In [106]: df['Duration'] = df['Duration'].str.replace('h','*60').str.replace(' ','+').str.replace('m','*1').apply(eval)

In [107]: df['Duration']= pd.to_numeric(df['Duration'])

In [ ]:
```

Create new column for Day and Date



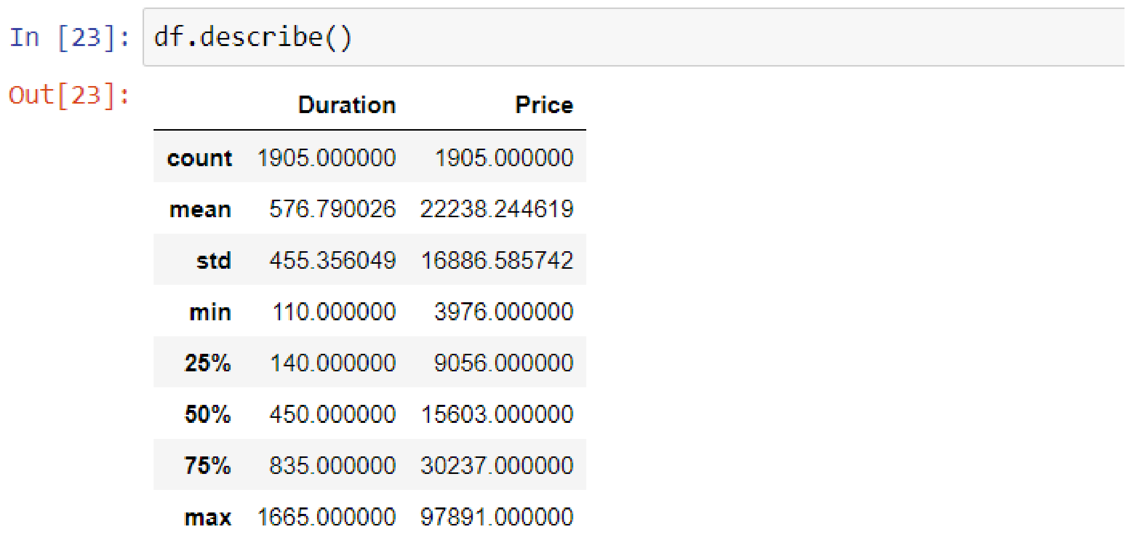
```
In [90]: df['Day'] = df['Date'].map(lambda x :x[:3])

         df['Date'] = df['Date'].map(lambda x :x[4:])
```

Using the describe function, you can get an overall view of the data.



```
In [100]: df.describe(include='all')
Out[100]:
```

	Airline	Date	Departure_Time	Source	Arrival_Time	Destination	Stops	Duration	Price	Class Type	Day
count	1905	1905	1905	1905	1905	1905	1905	1905	1905.000000	1905	1905
unique	9	8	129	1	175	1	3	177	NaN	6	7
top	Vistara Premium Economy	12 Jan 2023	22:40	Mumbai	07:55\n+ 1 day	New Delhi	1 Stop	2h 15m	NaN	Economy	Thu
freq	352	302	76	1905	45	1905	1337	189	NaN	1070	426
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	22238.244619	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	16886.585742	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3976.000000	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	9056.000000	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	15603.000000	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	30237.000000	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97891.000000	NaN	NaN



```
In [23]: df.describe()
Out[23]:
```

	Duration	Price
count	1905.000000	1905.000000
mean	576.790026	22238.244619
std	455.356049	16886.585742
min	110.000000	3976.000000
25%	140.000000	9056.000000
50%	450.000000	15603.000000
75%	835.000000	30237.000000
max	1665.000000	97891.000000

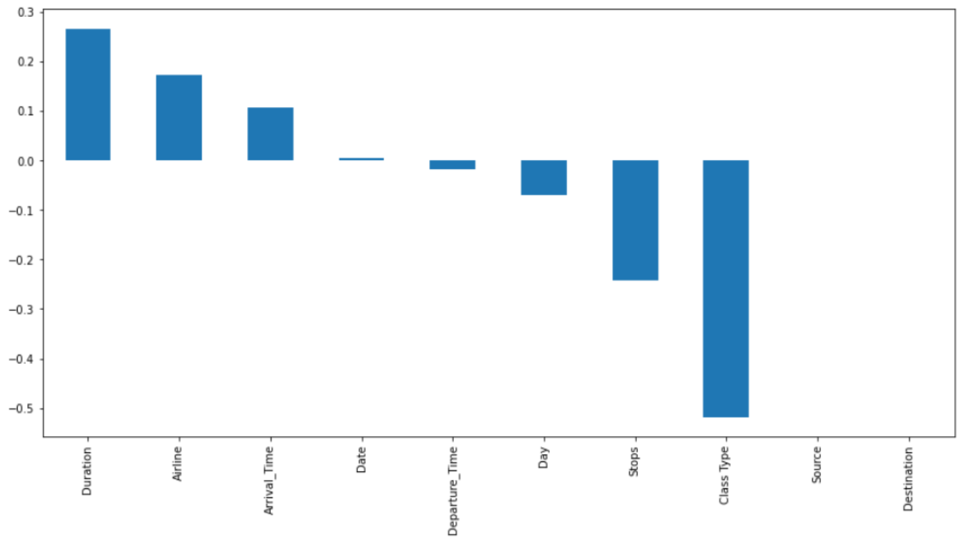
- Data Inputs- Logic- Output Relationships

Source and Destination has no relation with price beacuse there is only 1 source and destination

Class Type, Duration and Airline have strong relationship with Price respectively.

Date and Departure time have poor relationship with price

```
In [48]: plt.figure(figsize = (15,7))
         df.corr()['Price'].drop(['Price']).sort_values(ascending=False).plot(kind='bar')
         plt.show()
```



- Hardware and Software Requirements and Tools Used

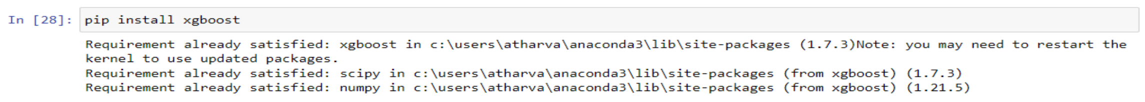
Hardware Used:

1. Processor : AMD Ryzen 5
2. Ram : 6 GB
3. GPU : Integrated Graphic card 2 GB

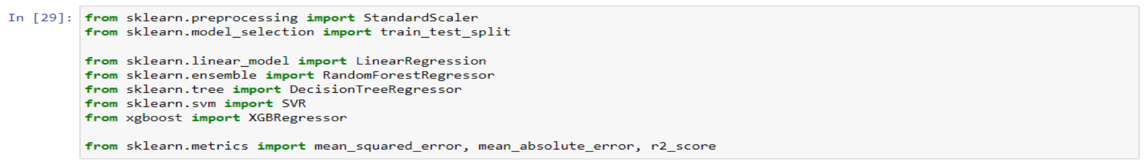
Software Utilised:

1. Anaconda
2. Selenium
3. Jupyter Notebook
4. Libraries Used

```
In [1]: import pandas as pd
        import numpy as np

        import matplotlib.pyplot as plt
        import seaborn as sns

        import warnings
        warnings.filterwarnings('ignore')
```



```
In [28]: pip install xgboost

Requirement already satisfied: xgboost in c:\users\atharva\anaconda3\lib\site-packages (1.7.3)Note: you may need to restart the
kernel to use updated packages.
Requirement already satisfied: scipy in c:\users\atharva\anaconda3\lib\site-packages (from xgboost) (1.7.3)
Requirement already satisfied: numpy in c:\users\atharva\anaconda3\lib\site-packages (from xgboost) (1.21.5)
```



```
In [29]: from sklearn.preprocessing import StandardScaler
         from sklearn.model_selection import train_test_split

         from sklearn.linear_model import LinearRegression
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.svm import SVR
         from xgboost import XGBRegressor

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

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

By offering competitive prices, flight price prediction model helps to increase consumer traffic. Price is a continuous data type, so we use regression models like liner regression to predict it.

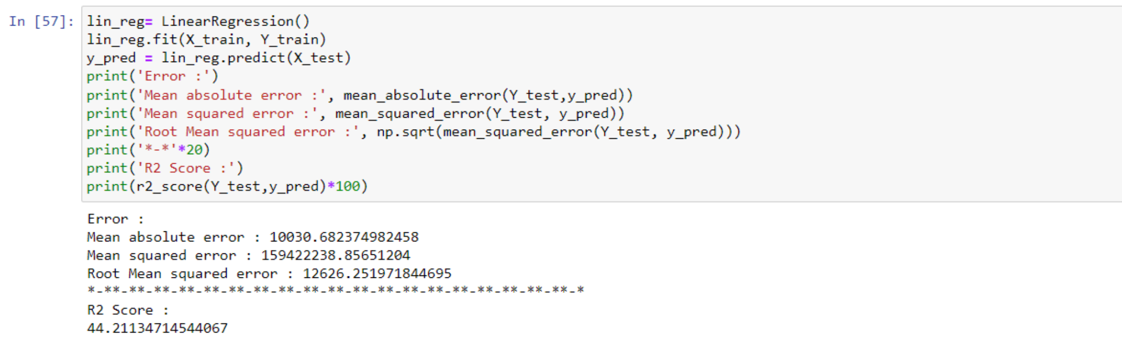
to solve this problem using an analytic approach to predict flight prices and by plotting various graphs like a pie plot, a count plot, and others. Use correlation to find multicollinearity problems, plot heatmaps, and find the best feature to aid in label prediction.

- Testing of Identified Approaches (Algorithms)

Despite having a large number of categorical variables, only one of them in the data set provides continuous data. Think, I On this dataset, a non-linear regression model will perform better.

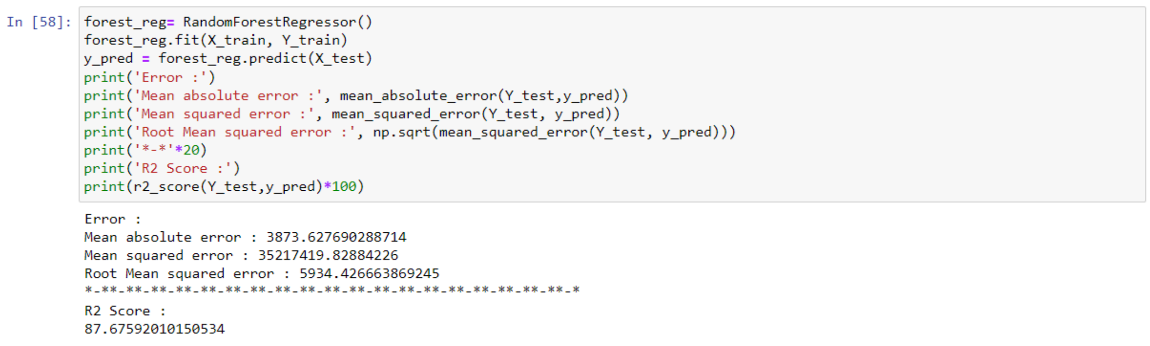
- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- XGB Regressor
- SVM

- Run and Evaluate selected models
  - Linear Regression



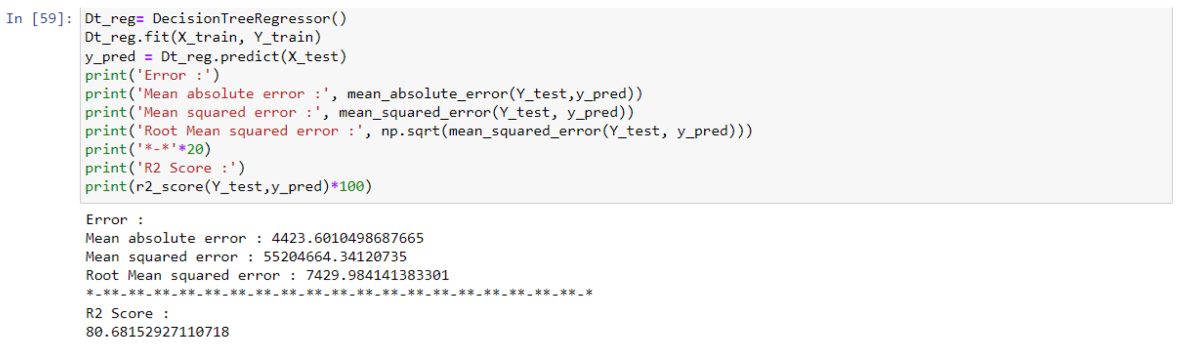
```
In [57]: lin_reg= LinearRegression()
         lin_reg.fit(X_train, Y_train)
         y_pred = lin_reg.predict(X_test)
         print('Error :')
         print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
         print('Mean squared error :', mean_squared_error(Y_test, y_pred))
         print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
         print('*-*'*20)
         print('R2 Score :')
         print(r2_score(Y_test,y_pred)*100)

         Error :
         Mean absolute error : 10030.682374982458
         Mean squared error : 159422238.85651204
         Root Mean squared error : 12626.251971844695
         *-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-*
         R2 Score :
         44.21134714544067
```

  - Random Forest Regressor



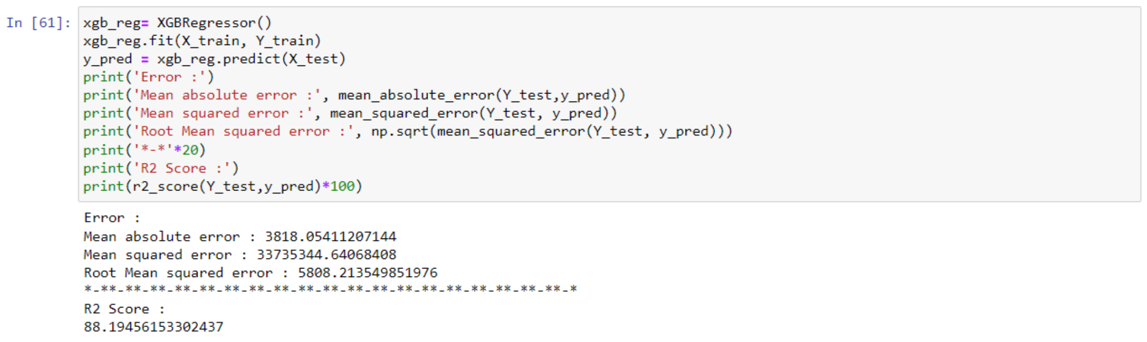
```
In [58]: forest_reg= RandomForestRegressor()
         forest_reg.fit(X_train, Y_train)
         y_pred = forest_reg.predict(X_test)
         print('Error :')
         print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
         print('Mean squared error :', mean_squared_error(Y_test, y_pred))
         print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
         print('*-*'*20)
         print('R2 Score :')
         print(r2_score(Y_test,y_pred)*100)

         Error :
         Mean absolute error : 3873.627690288714
         Mean squared error : 35217419.82884226
         Root Mean squared error : 5934.426663869245
         *-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-*
         R2 Score :
         87.67592010150534
```

  - Decision Tree Regressor



```
In [59]: Dt_reg= DecisionTreeRegressor()
         Dt_reg.fit(X_train, Y_train)
         y_pred = Dt_reg.predict(X_test)
         print('Error :')
         print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
         print('Mean squared error :', mean_squared_error(Y_test, y_pred))
         print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
         print('*-*'*20)
         print('R2 Score :')
         print(r2_score(Y_test,y_pred)*100)

         Error :
         Mean absolute error : 4423.6010498687665
         Mean squared error : 55204664.34120735
         Root Mean squared error : 7429.984141383301
         *-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-*
         R2 Score :
         80.68152927110718
```

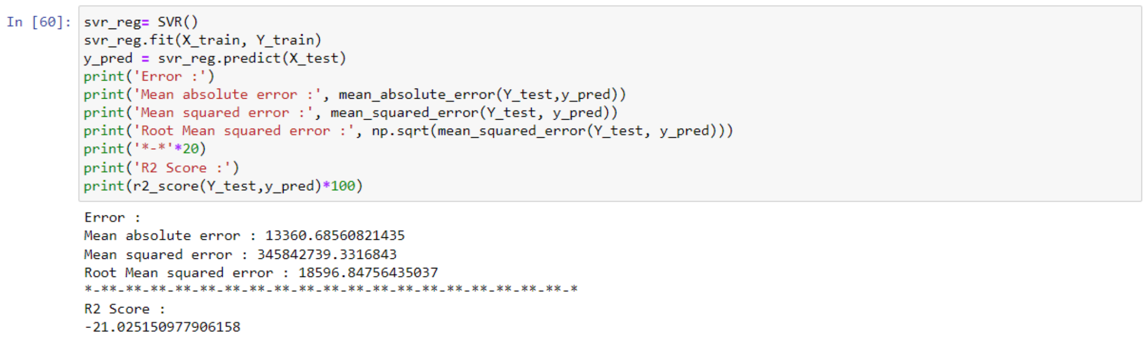
- XGB Regressor



```
In [61]: xgb_reg= XGBRegressor()
         xgb_reg.fit(X_train, Y_train)
         y_pred = xgb_reg.predict(X_test)
         print('Error :')
         print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
         print('Mean squared error :', mean_squared_error(Y_test, y_pred))
         print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
         print('*-*'*20)
         print('R2 Score :')
         print(r2_score(Y_test,y_pred)*100)
```

```
Error :
Mean absolute error : 3818.05411207144
Mean squared error : 33735344.64068408
Root Mean squared error : 5808.213549851976
*-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-*
R2 Score :
88.19456153302437
```

- SVR



```
In [60]: svr_reg= SVR()
         svr_reg.fit(X_train, Y_train)
         y_pred = svr_reg.predict(X_test)
         print('Error :')
         print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
         print('Mean squared error :', mean_squared_error(Y_test, y_pred))
         print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
         print('*-*'*20)
         print('R2 Score :')
         print(r2_score(Y_test,y_pred)*100)
```

```
Error :
Mean absolute error : 13360.68560821435
Mean squared error : 345842739.3316843
Root Mean squared error : 18596.84756435037
*-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-*
R2 Score :
-21.025150977906158
```

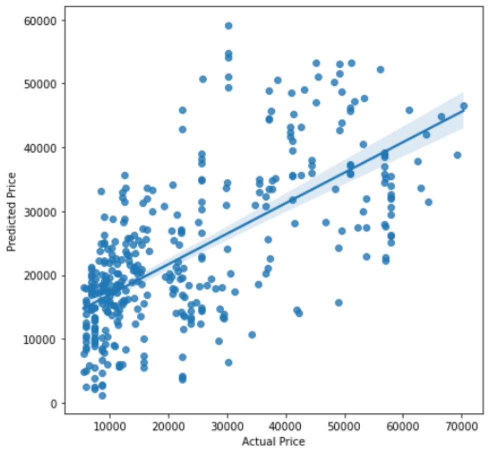
- Key Metrics for success in solving problem under consideration

Use the following metrics to choose the best model:

- Mean absolute error
- Mean Squared error
- Root mean squared error
- R2 score
- Visualization

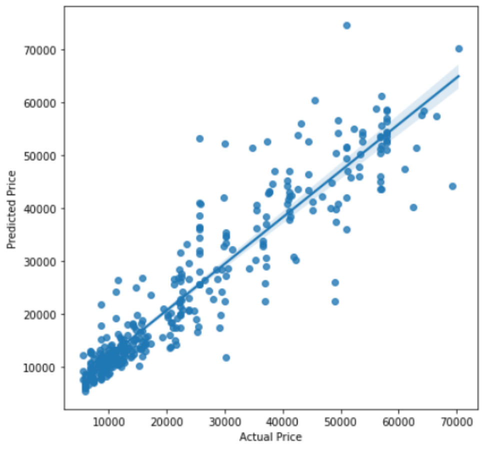
- Linear Regression

```
In [47]: plt.figure(figsize=(7,7))
         sns.regplot(x=Y_test,y=y_pred)
         plt.xlabel('Actual Price')
         plt.ylabel('Predicted Price')
         plt.show()
```



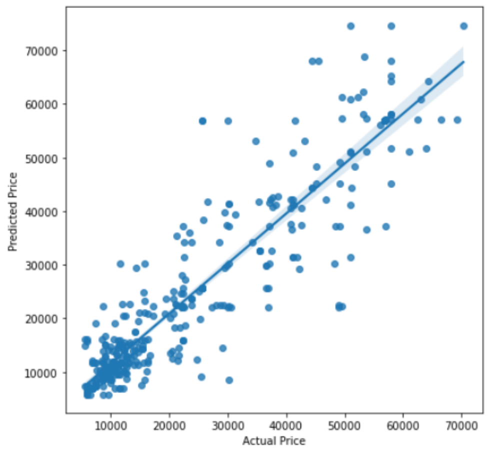
- Random Forest Regressor

```
In [49]: plt.figure(figsize=(7,7))
         sns.regplot(x=Y_test,y=y_pred)
         plt.xlabel('Actual Price')
         plt.ylabel('Predicted Price')
         plt.show()
```



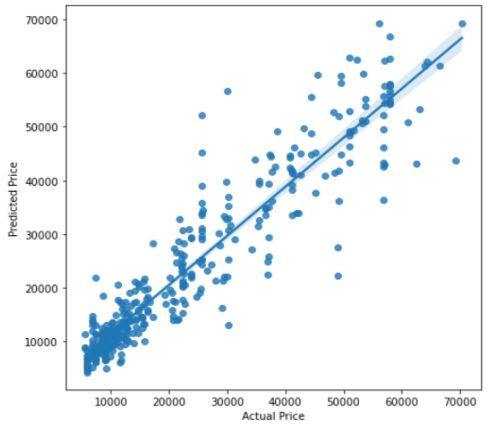
- Decision Tree Regressor

```
In [51]: plt.figure(figsize=(7,7))
         sns.regplot(x=Y_test,y=y_pred)
         plt.xlabel('Actual Price')
         plt.ylabel('Predicted Price')
         plt.show()
```



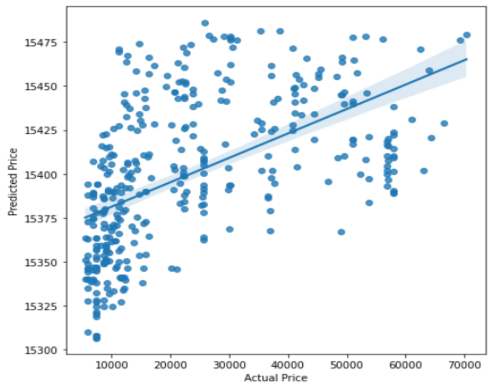
- XGB Regressor

```
In [55]: plt.figure(figsize=(7,7))
         sns.regplot(x=Y_test,y=y_pred)
         plt.xlabel('Actual Price')
         plt.ylabel('Predicted Price')
         plt.show()
```



- SVR

```
In [53]: plt.figure(figsize=(7,7))
         sns.regplot(x=Y_test,y=y_pred)
         plt.xlabel('Actual Price')
         plt.ylabel('Predicted Price')
         plt.show()
```



- Interpretation of the Results

Algorithm	R2 Score	CV score
Linear Regression	44.21	0.27
Random Forest	87.33	0.65
Decision Tree Regressor	81.75	0.50
XGB Regressor	88.19	0.65
SVR	-21.02	-0.21
After Hyper parameter tuned		
XGB Regressor	88.71	
Random Forest	87.23	

CONCLUSION

- Key Findings and Conclusions of the Study
- The following are the top airlines that offer regular flights from Mumbai to Delhi.

1. Vistara Premium Economy	352
2. Vistara Business	300
3. Vistara	278
4. IndiGo	271
5. Air India Business	263

- 4. 56% of seats are in economy class, and business and premium economy class seats are 25% and 18%, respectively.
- Sunday flight tickets are more expensive than other days.

- An airline flight ticket has no relationship between duration and price.
- Nonstop flights are less expensive than one-stop flights.

• Learning Outcomes of the Study in respect of Data Science

Learn about the operation of the aircraft sector. The cost of flights depends on a number of variables.

A non-linear algorithm will work better on a dataset if the majority of the project's variables are categorical data.

• Limitations of this work and Scope for Future Work

In this study, we focus only on flights from Mumbai to Delhi.

This project can be expanded by incorporating additional routes.

present investigation.

This investigation concentrated on a short period of time (7 days before flights).

(take off), which can be extended as a variation over a larger period.