FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

Vishwa Narayan Saini

Internship batch-33

ACKNOWLEDGMENT

I would like to extend my sincere gratitude to [Datatrained Academy], and [FLIP ROBO TECHNOLOGY] for their invaluable contributions to this project. Their expertise and support were instrumental in the completion of this project. I would also like to thank Mr. Shwetank Mishra (SME FOR BATCH-33) for providing us with the necessary resources to complete this project.

INTRODUCTION

- ## Business Problem Framing

Flight price prediction can be framed as a business problem by considering the following factors:

1. The demand for flights: This includes factors such as the time of year, holidays, and special events that may affect the number of people looking to travel.
2. The supply of flights: This includes the number of flights available, the routes they operate on, and the capacity of each flight.
3. The competition: This includes the pricing and availability of flights from other airlines, as well as the pricing of alternative forms of transportation such as trains and buses.
4. The cost of operating the flights: This includes fuel costs, maintenance costs, and employee salaries.

By considering these factors, a business can use flight price prediction to optimize pricing strategies and increase revenue by offering competitive prices to customers while also ensuring that they can cover their costs.

- ## Conceptual Background of the Domain Problem

1. Understanding how the prices vary over different days of the week.

2. Analysing the frequency of flights over the route can also help in understanding the pattern of pricing.

3. Different times at which flights depart from the airport

4. Influence of duration and path in determining the price of flight over a route.

- ## Review of Literature

1. Studies on historical flight pricing data, including factors that have been shown to influence prices such as demand, seasonality, and competition.
2. Research on machine learning and artificial intelligence techniques that have been used for price prediction, such as regression analysis, decision trees, and neural networks.
3. Analyses of the performance of existing flight price prediction models, including their strengths and limitations.
4. Studies on how airlines and travel companies currently use pricing data and analytics to make decisions, and how this might be improved with better prediction models.
5. Research the travel industry and economic trends that may impact flight pricing in the future, such as changes in fuel

prices, competition from low-cost carriers, and shifts in consumer preferences.

- ## Motivation for the Problem Undertaken

There are several potential motivations for using flight price prediction as a business problem. One is to help airlines and travel companies make pricing decisions by forecasting demand and understanding how changes in factors such as fuel costs, competition, and economic conditions will affect ticket prices. Another is to help consumers make informed travel decisions by providing them with accurate and up-to-date information about flight prices. Additionally, businesses that operate in the travel industry, such as online travel agencies and travel search engines, can use flight price prediction to optimize their revenue and improve customer satisfaction by providing personalized flight recommendations and deals.

Analytical Problem Framing

- **Mathematical/ Analytical Modelling of the Problem**

Several approaches can be used for a flight price prediction machine learning problem. One common approach is to use a linear regression model, where the dependent variable is the flight price and the independent variables are factors such as the departure and arrival airports, the date and time of the flight, and the number of seats available. Another approach is to use a decision tree or random forest model, which can handle non-linear relationships between the input variables and the output variable.

Another approach is to use time series forecasting techniques, such as ARIMA, which take into account the historical flight prices and the seasonality of the prices. Additionally, you can use deep learning models like LSTM or GRU which are particularly good at modeling time-series data.

To improve the accuracy of the prediction model, it is important to use a large and diverse dataset and to carefully select the input variables based on domain knowledge and feature importance analysis. Additionally, it is important to properly validate the model using techniques such as k-fold cross-validation.

- **Data Sources and their formats**

Flights data i.e., Airlines, Departure Airport, Arrival Airport, Departure Date, Arrival Date, Duration, STOPS, Price, and Number of seats available, Departure Time, Arrival Time are scraped from the website https://www.adanione.com/ using selenium web-scraping.

All the data is then converted into a CSV file.

DATA DESCRIPTION!!

DATE- Date of flight

AIRLINES- The Airliner the flight belongs to

FROM- From where the flight departs

ARRIVAL- To where the flight arrives

DURATION (hrs)-Total duration of the flight

PATH- The route including stops if present

STOPS-The number of stops the flight takes

PRICE(Rs)- The price in rupees of the flight

- **Data Pre-processing Done**

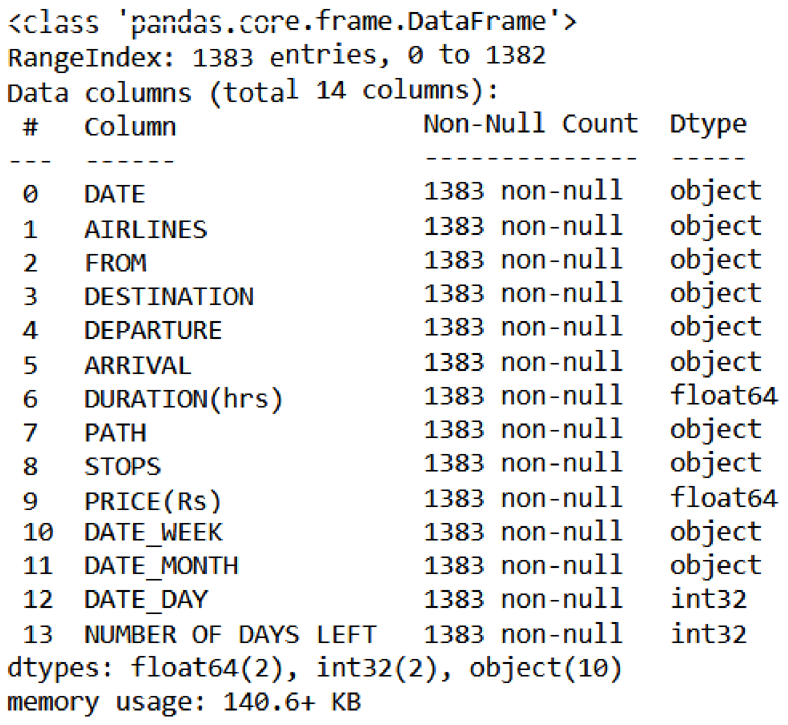
FEATURE ENGINEERING:

Feature DATE is further spitted into DATE_WEEK, DATE_MONTH, and DATE_DAY

Duration which is in the format for example 14h 20m must be converted into complete hours i.e., 14.33

DEPARTURE and ARRIVAL time must be converted into a datetime object

There are no nulls present in the dataset



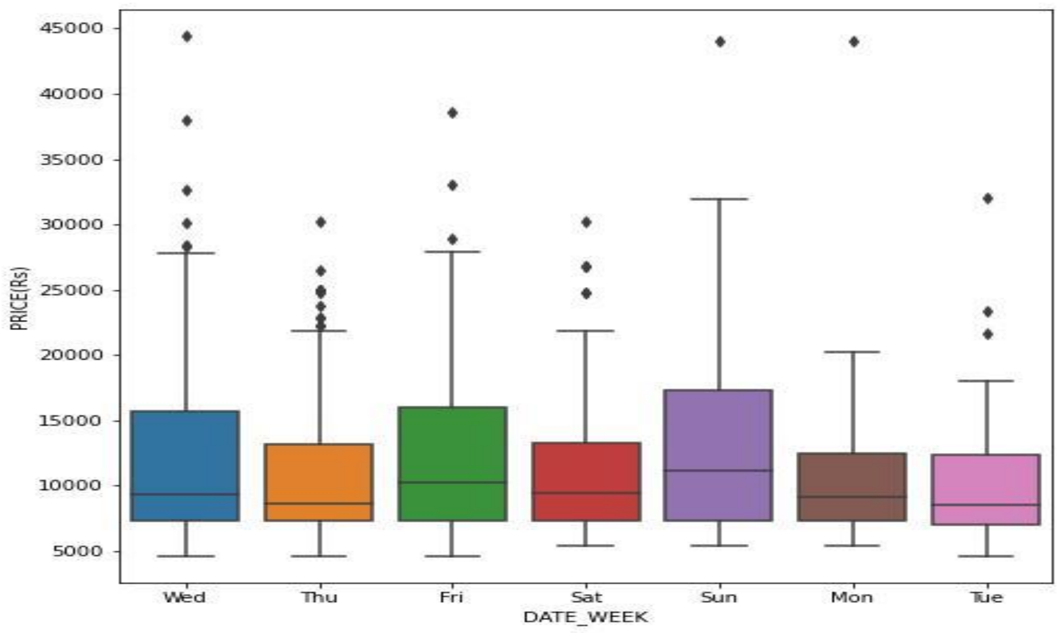
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1383 entries, 0 to 1382
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   DATE                   1383 non-null   object 
 1   AIRLINES               1383 non-null   object 
 2   FROM                   1383 non-null   object 
 3   DESTINATION            1383 non-null   object 
 4   DEPARTURE              1383 non-null   object 
 5   ARRIVAL                1383 non-null   object 
 6   DURATION(hrs)          1383 non-null   float64
 7   PATH                   1383 non-null   object 
 8   STOPS                  1383 non-null   object 
 9   PRICE(Rs)              1383 non-null   float64
 10  DATE_WEEK              1383 non-null   object 
 11  DATE_MONTH             1383 non-null   object 
 12  DATE_DAY               1383 non-null   int32  
 13  NUMBER OF DAYS LEFT    1383 non-null   int32  
dtypes: float64(2), int32(2), object(10)
memory usage: 140.6+ KB
```

Different data types can be seen for different features.

- **Data Inputs- Logic- Output Relationships**

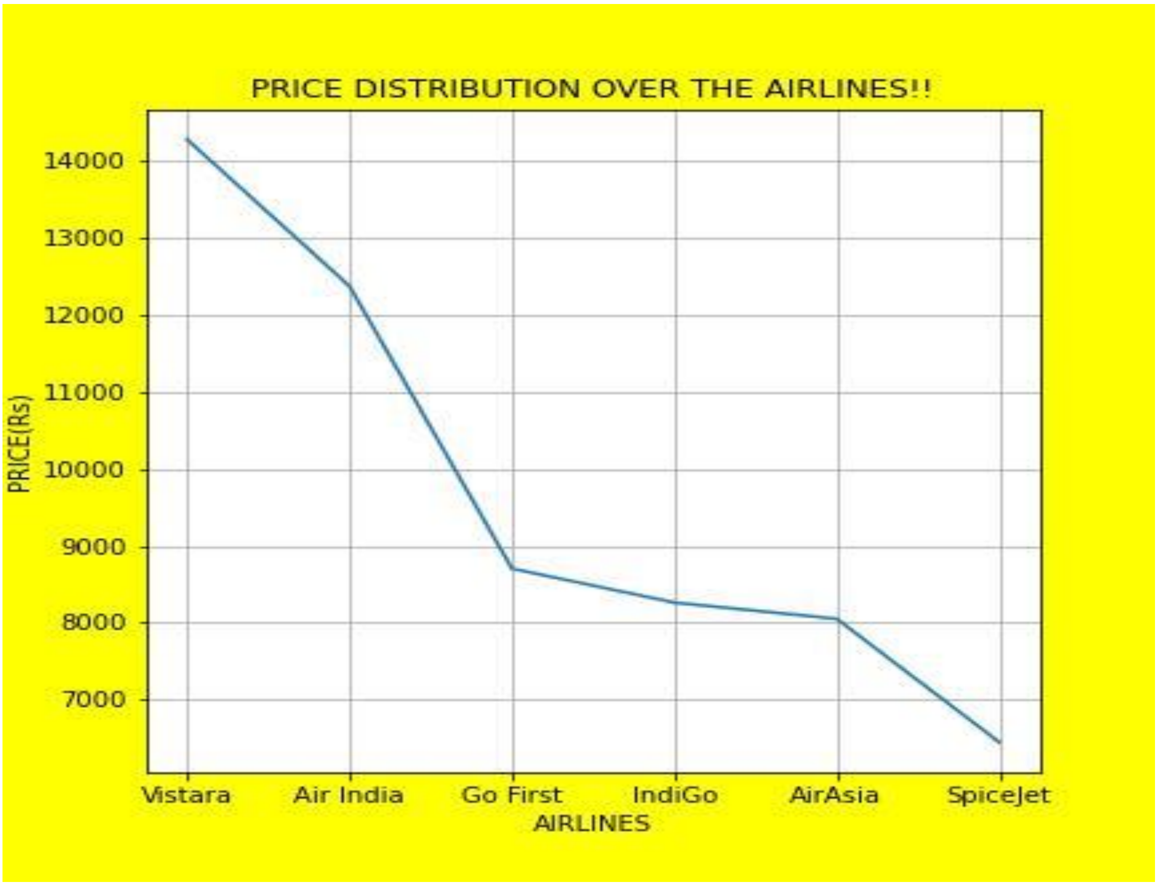
ANALYSIS

HOW PRICES VARY OVER DIFFERENT DAYS OF THE WEEK!



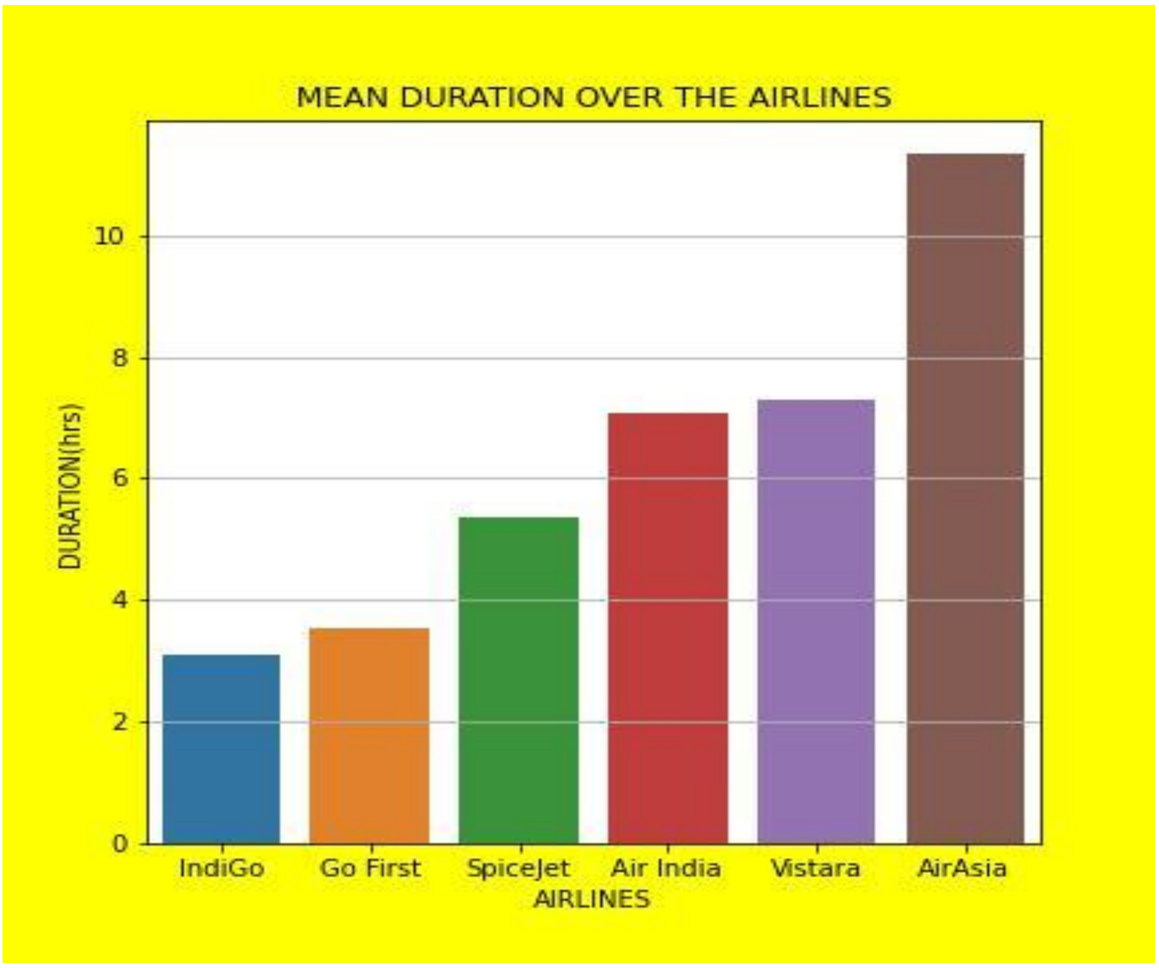
Variations in the prices can be seen over the week.
The inter-quantile range is maximum for Sunday, Friday, and Wednesday.
The maximum number of outliers present for Thursday and Wednesday.

HOW THE PRICE VARIES OVER DIFFERENT AIRLINES



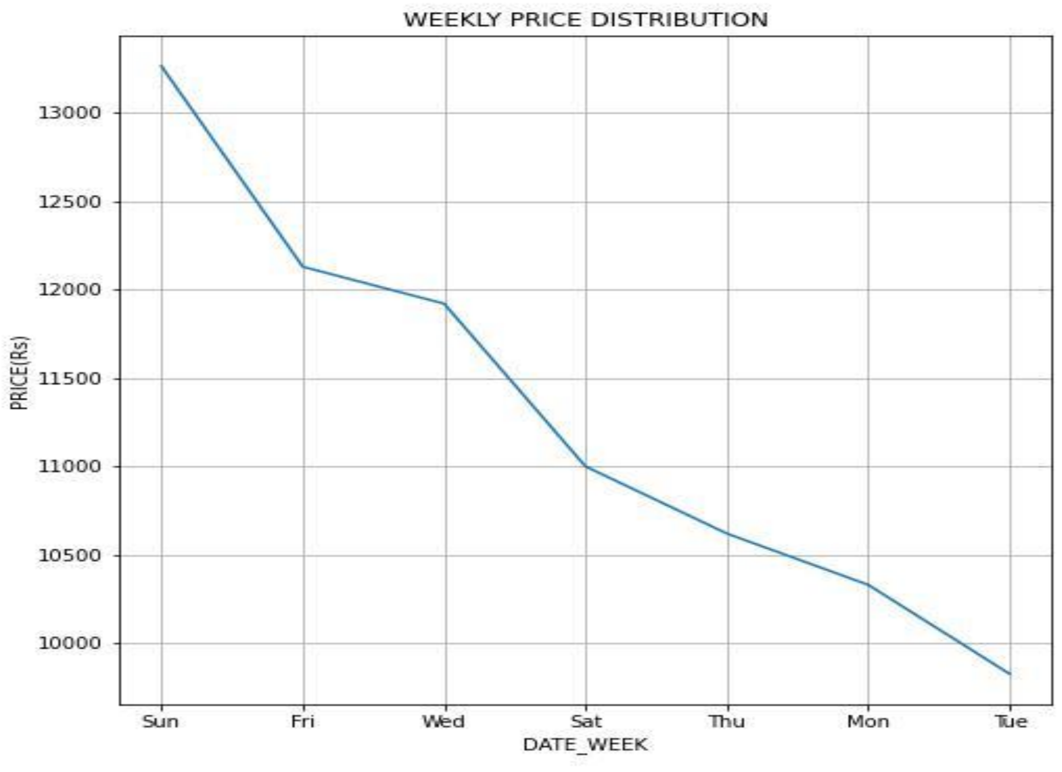
The mean price for the Vistara is the highest reaching around 1,45,000
SpiceJet has the least mean pricing with a price of around 6500rs
Air India and Go First stand second and third on the list.

Mean duration in hours for different airlines over the same path (New Delhi to Mumbai)



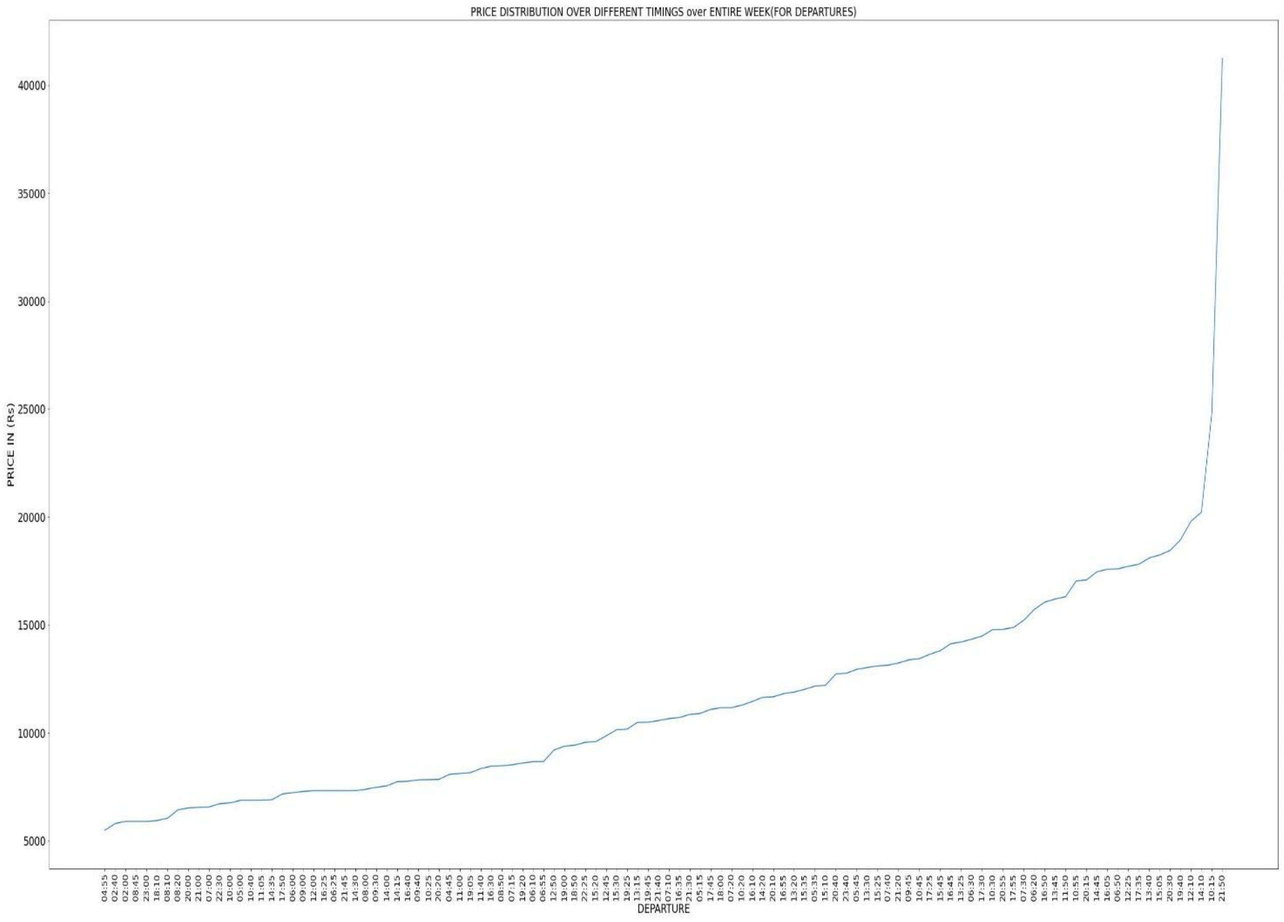
Indigo has the lowest mean duration flights among different airlines.
AirAsia has the longest flights depicting more stops over the route.

Variation of the mean price over different days of the week

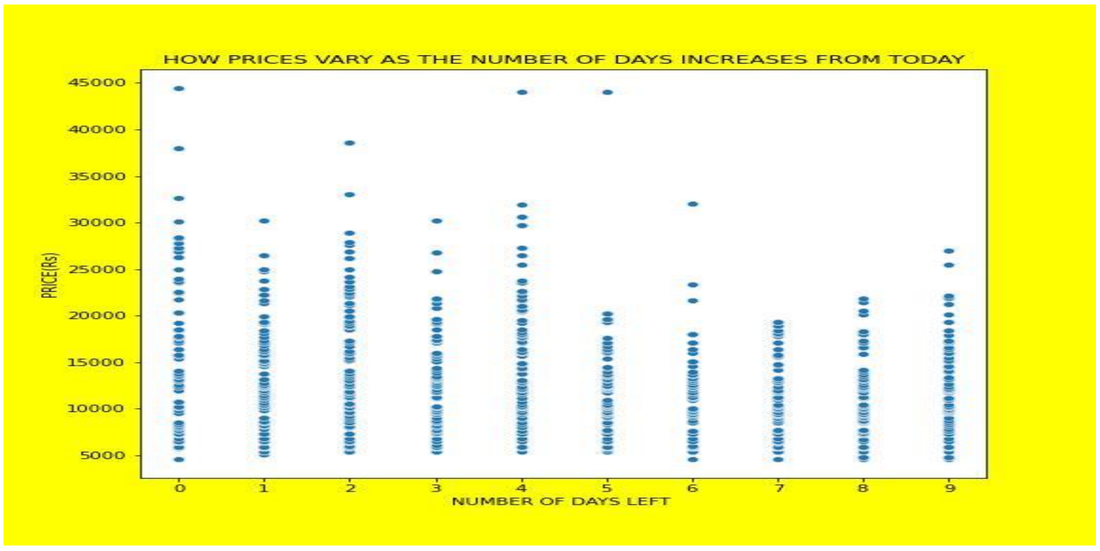


The highest prices can be seen over the weekend i.e., Sunday and Friday
Among them mean prices were least for Monday and Tuesday.

Influence of timing on the price of the flight

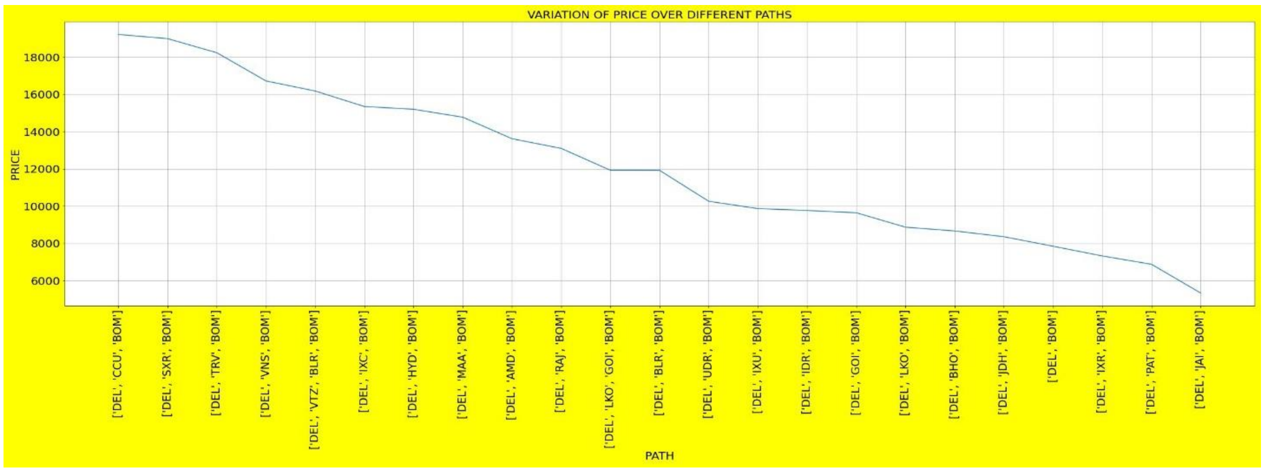


Relationship between the number of days from today and the flight prices

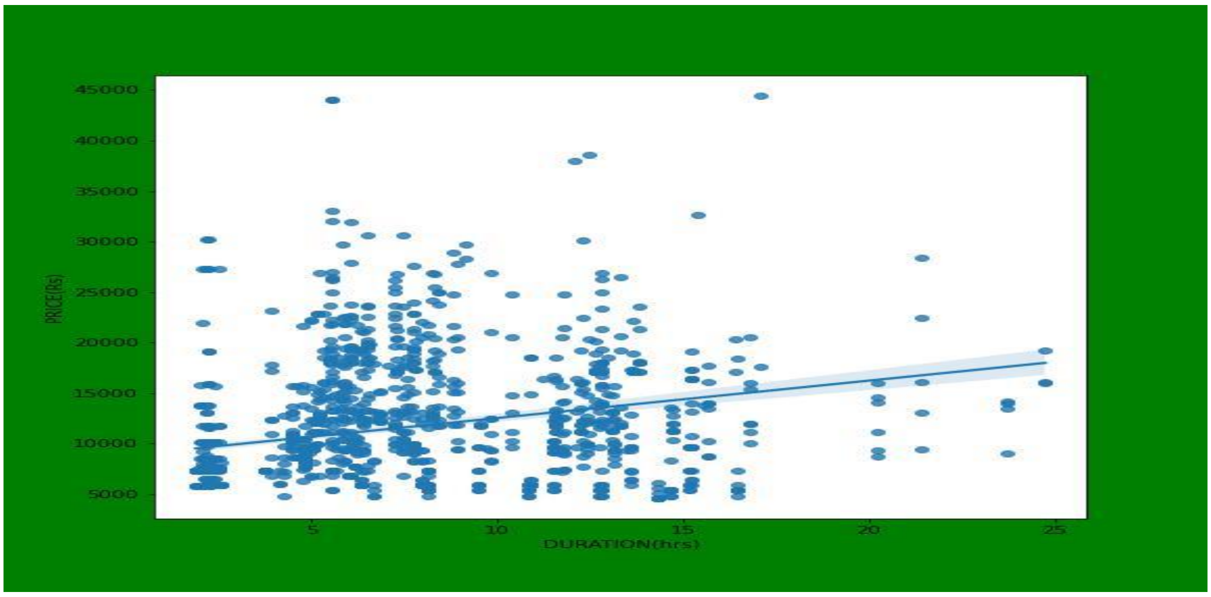


Flight prices tend to decrease as the number of days increases from today.

How the mean prices decrease over different paths



Relationship between duration and flight prices



A significant linear relationship can be seen between duration and price.

- ## Hardware and Software Requirements and Tools Used

Hardware requirements for a machine learning model typically include a computer with a powerful CPU and GPU, as well as sufficient memory and storage capacity. The specific requirements will depend on the complexity of the model and the size of the dataset. Software requirements usually include a programming language such as Python or R, as well as various libraries and frameworks such as TensorFlow, PyTorch, or scikit-learn for implementing the model.

Model/s Development and Evaluation

• Identification of possible problem-solving approaches (methods)

Several analytical and statistical approaches can be used to solve a flight price prediction machine learning problem.

1. Linear Regression: Linear regression is a simple and widely used statistical method for predicting the relationship between a dependent variable and one or more independent variables.
2. Decision Tree: A decision tree is a popular algorithm for both classification and regression problems. It can handle both categorical and numerical data.
3. Random Forest: Random Forest is an ensemble method that combines multiple decision trees to improve the overall performance of the model.
4. Gradient Boosting: Gradient Boosting is also an ensemble method that combines multiple weak models to form a stronger model.
5. Neural Networks: Neural networks are a class of machine learning models that are inspired by the structure and function of the human brain.
6. Time Series Analysis: Time Series Analysis is used to analyse and predict time-related data, such as flight prices. Techniques such as ARIMA, Exponential Smoothing, and LSTM can be used to analyse time series data.

The choice of the analytical and statistical approach will depend on the specific characteristics of the dataset and the problem.

• Testing of Identified Approaches (Algorithms)

1. Linear Regression
2. Decision Tree
3. AdaBoostRegressor
4. RandomForestRegressor

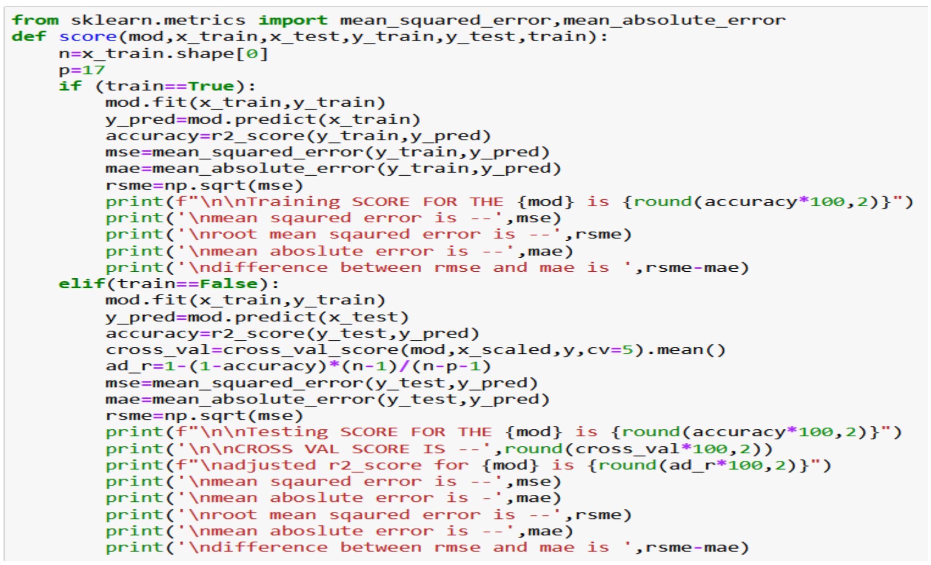
- ## Key Metrics for success in solving the problem under consideration

Several key metrics can be used to evaluate the performance of a flight price prediction machine learning model. These metrics include:

1. Mean Absolute Error (MAE): This measures the average difference between the predicted prices and the actual prices. A lower MAE indicates a better fit of the model to the data.
2. Mean Squared Error (MSE): This measures the average squared difference between the predicted prices and the actual prices. A lower MSE indicates a better fit of the model to the data.
3. Root Mean Squared Error (RMSE): This is the square root of the MSE and is used to interpret the magnitude of the error. A lower RMSE indicates a better fit of the model to the data.
4. R-squared: This measures the proportion of the variance in the dependent variable that is explained by the independent variable. A higher R-squared indicates a better fit of the model to the data.
5. Mean Percentage Error (MPE): This is the average percentage difference between the predicted prices and the actual prices. A lower MPE indicates a better fit of the model to the data.
6. Coefficient of Determination (R2): This is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variable. It ranges from 0 to 1, with 1 being a perfect fit.
7. Correlation coefficient: This measures the strength and direction of a linear relationship between two variables. A correlation coefficient of 1 implies a perfect positive linear relationship, a coefficient of -1 implies a perfect negative linear relationship, and a coefficient of 0 implies no linear relationship.

It is important to use multiple metrics to evaluate the performance of the model and compare it with another model.

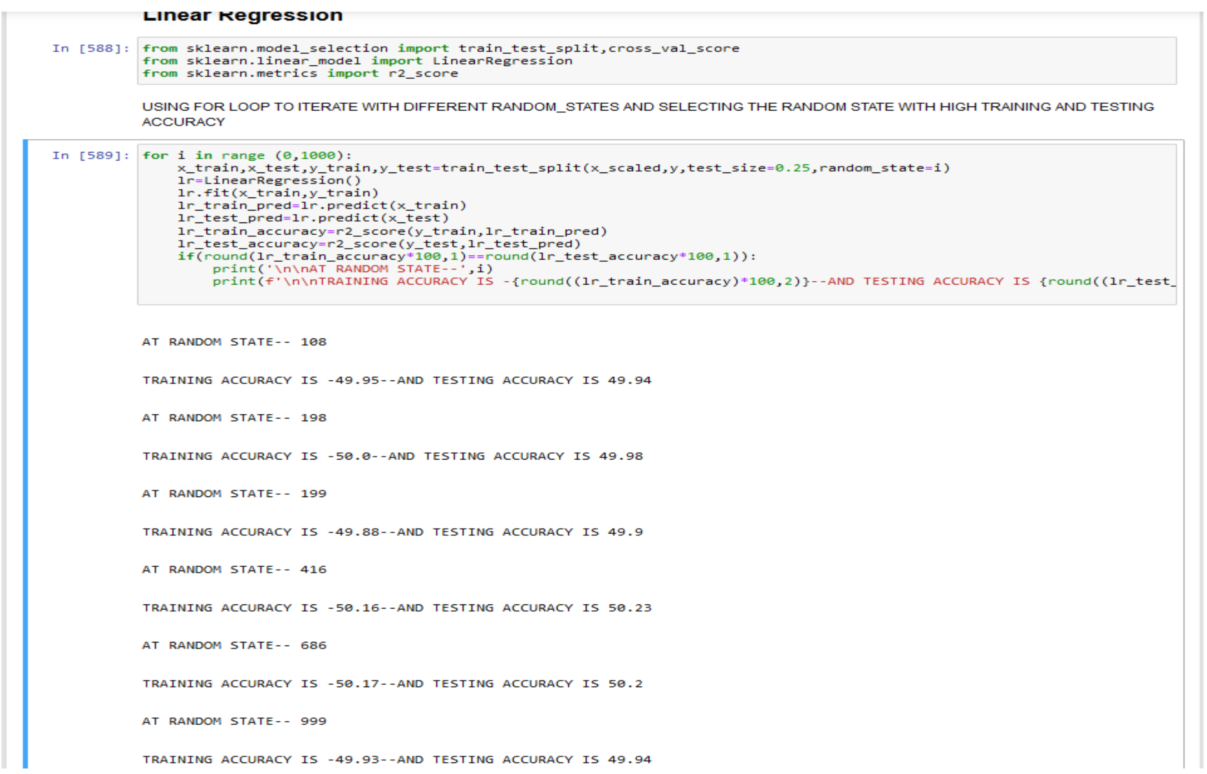
- **Defining a function for evaluating all the metrics**



```
from sklearn.metrics import mean_squared_error,mean_absolute_error
def score(mod,x_train,x_test,y_train,y_test,train):
    n=x_train.shape[0]
    p=17
    if (train==True):
        mod.fit(x_train,y_train)
        y_pred=mod.predict(x_train)
        accuracy=r2_score(y_train,y_pred)
        mse=mean_squared_error(y_train,y_pred)
        mae=mean_absolute_error(y_train,y_pred)
        rsme=np.sqrt(mse)
        print(f"\n\nTraining SCORE FOR THE {mod} is {round(accuracy*100,2)}")
        print('\nmean sqaured error is --',mse)
        print('\nroot mean sqaured error is --',rsme)
        print('\nmean aboslute error is --',mae)
        print('\ndifference between rmse and mae is ',rsme-mae)
    elif(train==False):
        mod.fit(x_train,y_train)
        y_pred=mod.predict(x_test)
        accuracy=r2_score(y_test,y_pred)
        cross_val=cross_val_score(mod,x_scaled,y,cv=5).mean()
        ad_r=1-(1-accuracy)*(n-1)/(n-p-1)
        mse=mean_squared_error(y_test,y_pred)
        mae=mean_absolute_error(y_test,y_pred)
        rsme=np.sqrt(mse)
        print(f"\n\nTesting SCORE FOR THE {mod} is {round(accuracy*100,2)}")
        print('\n\nCROSS VAL SCORE IS --',round(cross_val*100,2))
        print(f"\nadjusted r2_score for {mod} is {round(ad_r*100,2)}")
        print('\nmean sqaured error is --',mse)
        print('\nmean aboslute error is -',mae)
        print('\nroot mean sqaured error is --',rsme)
        print('\nmean aboslute error is --',mae)
        print('\ndifference between rmse and mae is ',rsme-mae)
```

- **Run and evaluate selected models**

1. Linear Regression



Linear Regression

In [588]:
```
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

USING FOR LOOP TO ITERATE WITH DIFFERENT RANDOM_STATES AND SELECTING THE RANDOM STATE WITH HIGH TRAINING AND TESTING ACCURACY

In [589]:
```
for i in range (0,1000):
    x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=i)
    lr=LinearRegression()
    lr.fit(x_train,y_train)
    lr_train_pred=lr.predict(x_train)
    lr_test_pred=lr.predict(x_test)
    lr_train_accuracy=r2_score(y_train,lr_train_pred)
    lr_test_accuracy=r2_score(y_test,lr_test_pred)
    if(round(lr_train_accuracy*100,1)==round(lr_test_accuracy*100,1)):
        print('\n\nAT RANDOM STATE--',i)
        print(f'\n\nTRAINING ACCURACY IS -{round((lr_train_accuracy)*100,2)}--AND TESTING ACCURACY IS {round((lr_test_
```

```


AT RANDOM STATE-- 108


TRAINING ACCURACY IS -49.95--AND TESTING ACCURACY IS 49.94


AT RANDOM STATE-- 198


TRAINING ACCURACY IS -50.0--AND TESTING ACCURACY IS 49.98


AT RANDOM STATE-- 199


TRAINING ACCURACY IS -49.88--AND TESTING ACCURACY IS 49.9


AT RANDOM STATE-- 416


TRAINING ACCURACY IS -50.16--AND TESTING ACCURACY IS 50.23


AT RANDOM STATE-- 686


TRAINING ACCURACY IS -50.17--AND TESTING ACCURACY IS 50.2


AT RANDOM STATE-- 999


TRAINING ACCURACY IS -49.93--AND TESTING ACCURACY IS 49.94
```

CHECKING THE CROSS VAL SCORES AT DIFFERENT CVs

```
|: from sklearn.model_selection import cross_val_score
   for i in range(2,10):
       cross_val=cross_val_score(lr,x_scaled,y,cv=i).mean()
       print(f"\n\nCROSS VAL IS AT CV = {i} is {round(cross_val*100,2)}")
```

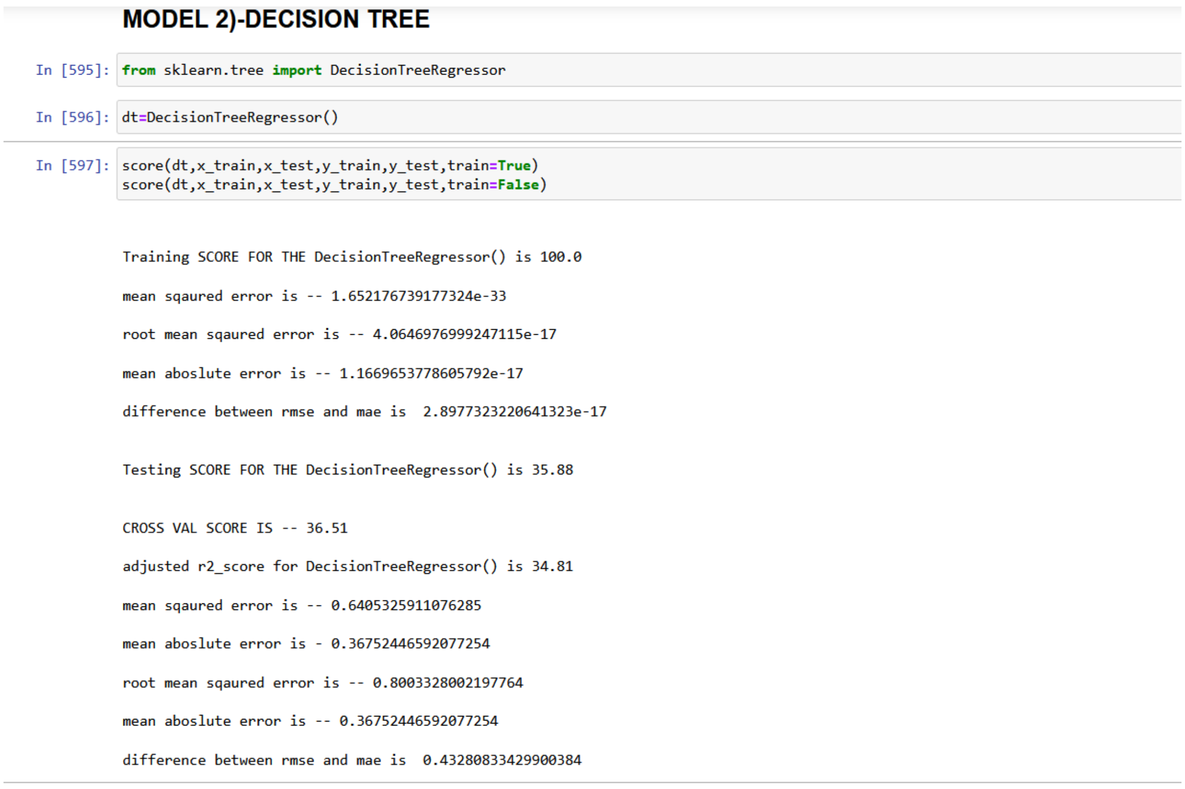
```
CROSS VAL IS AT CV = 2 is 25.2


CROSS VAL IS AT CV = 3 is 42.82


CROSS VAL IS AT CV = 4 is 42.12


CROSS VAL IS AT CV = 5 is 44.57


CROSS VAL IS AT CV = 6 is 44.35


CROSS VAL IS AT CV = 7 is 39.23


CROSS VAL IS AT CV = 8 is 43.17


CROSS VAL IS AT CV = 9 is 43.48
```

CONSIDERING CV AS 5!

```
score(lr,x_train,x_test,y_train,y_test,train=True)
score(lr,x_train,x_test,y_train,y_test,train=False)
```

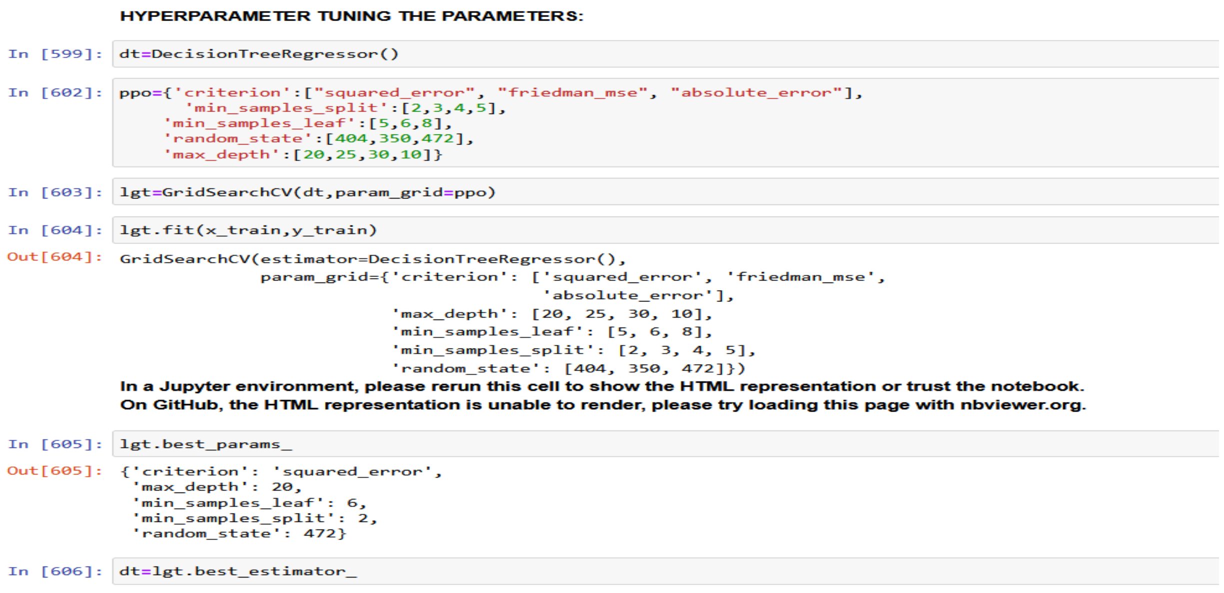
```
Training SCORE FOR THE LinearRegression() is 50.17

mean sqaured error is -- 0.49820697705781036

root mean sqaured error is -- 0.7058377838128321

mean aboslute error is -- 0.4777238503094029

difference between rmse and mae is  0.22811393350342923


Testing SCORE FOR THE LinearRegression() is 50.2


CROSS VAL SCORE IS -- 44.57

adjusted r2_score for LinearRegression() is 49.37

mean sqaured error is -- 0.4974163113722736

mean aboslute error is - 0.46912895080243294

root mean sqaured error is -- 0.7052774711929154

mean aboslute error is -- 0.46912895080243294

difference between rmse and mae is  0.23614852039048245
```

2. Decision Tree



MODEL 2)-DECISION TREE

```
In [595]: from sklearn.tree import DecisionTreeRegressor

In [596]: dt=DecisionTreeRegressor()

In [597]: score(dt,x_train,x_test,y_train,y_test,train=True)
          score(dt,x_train,x_test,y_train,y_test,train=False)


          Training SCORE FOR THE DecisionTreeRegressor() is 100.0

          mean sqaured error is -- 1.652176739177324e-33

          root mean sqaured error is -- 4.0646976999247115e-17

          mean aboslute error is -- 1.1669653778605792e-17

          difference between rmse and mae is  2.8977323220641323e-17


          Testing SCORE FOR THE DecisionTreeRegressor() is 35.88


          CROSS VAL SCORE IS -- 36.51

          adjusted r2_score for DecisionTreeRegressor() is 34.81

          mean sqaured error is -- 0.6405325911076285

          mean aboslute error is - 0.36752446592077254

          root mean sqaured error is -- 0.8003328002197764

          mean aboslute error is -- 0.36752446592077254

          difference between rmse and mae is  0.43280833429900384
```

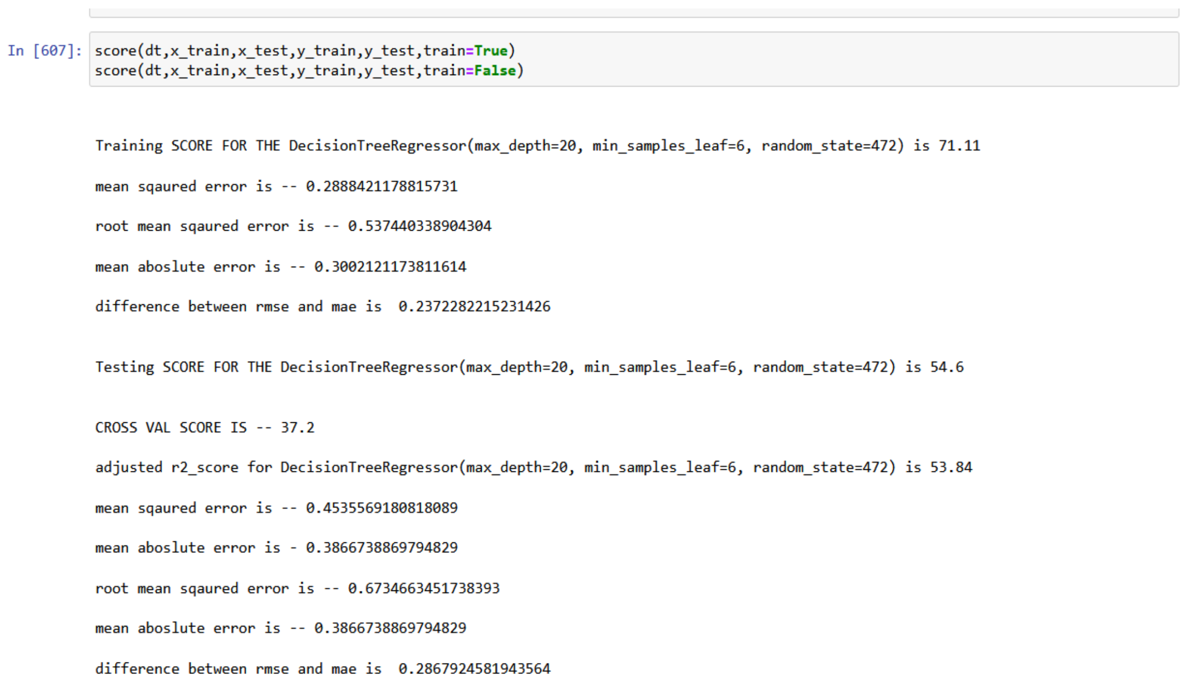


HYPERPARAMETER TUNING THE PARAMETERS:

```
In [599]: dt=DecisionTreeRegressor()

In [602]: ppo={'criterion':["squared_error", "friedman_mse", "absolute_error"],
                 'min_samples_split':[2,3,4,5],
               'min_samples_leaf':[5,6,8],
               'random_state':[404,350,472],
               'max_depth':[20,25,30,10]}

In [603]: lgt=GridSearchCV(dt,param_grid=ppo)

In [604]: lgt.fit(x_train,y_train)
Out[604]: GridSearchCV(estimator=DecisionTreeRegressor(),
                       param_grid={'criterion': ['squared_error', 'friedman_mse',
                                                 'absolute_error'],
                                   'max_depth': [20, 25, 30, 10],
                                   'min_samples_leaf': [5, 6, 8],
                                   'min_samples_split': [2, 3, 4, 5],
                                   'random_state': [404, 350, 472]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

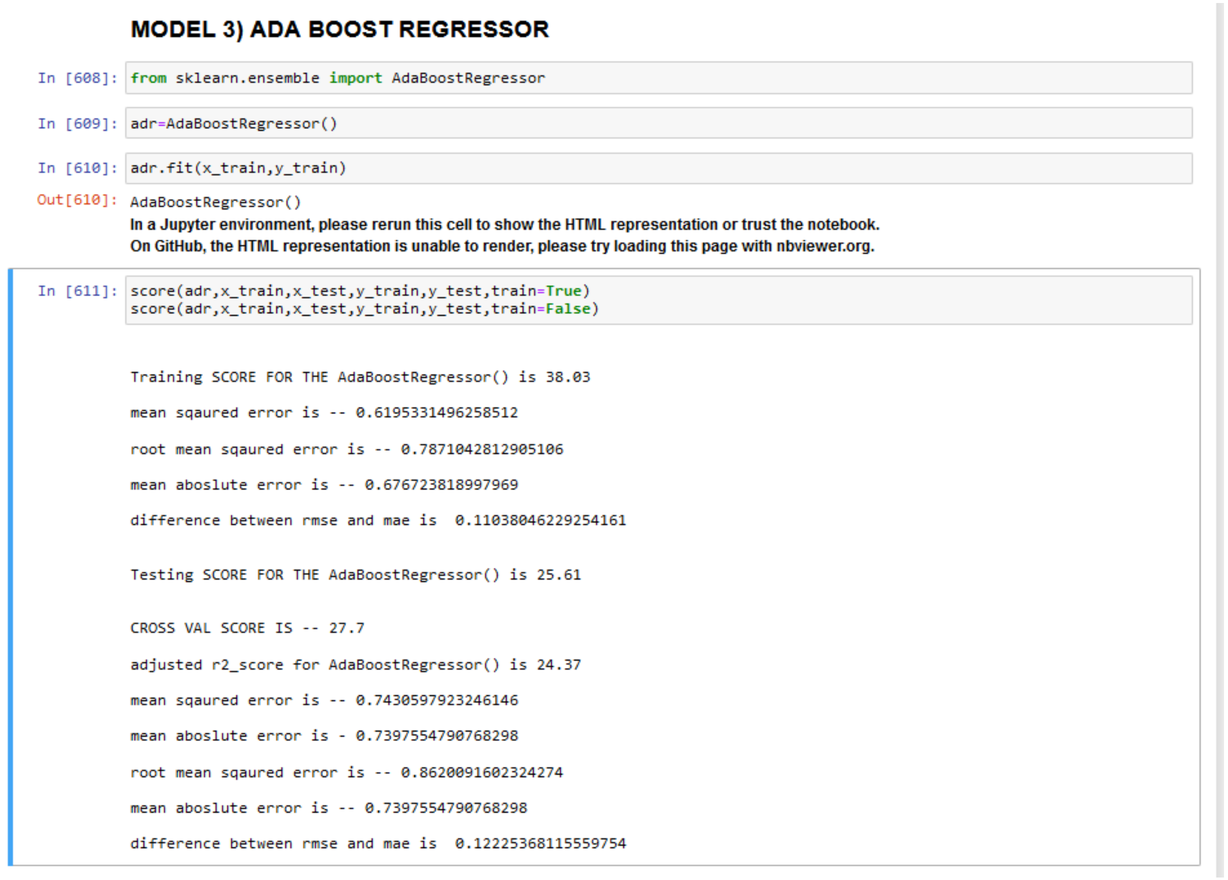
```
In [605]: lgt.best_params_
Out[605]: {'criterion': 'squared_error',
           'max_depth': 20,
           'min_samples_leaf': 6,
           'min_samples_split': 2,
           'random_state': 472}

In [606]: dt=lgt.best_estimator_
```



```
In [607]: score(dt,x_train,x_test,y_train,y_test,train=True)
          score(dt,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE DecisionTreeRegressor(max_depth=20, min_samples_leaf=6, random_state=472) is 71.11

mean sqaured error is -- 0.2888421178815731

root mean sqaured error is -- 0.537440338904304

mean aboslute error is -- 0.3002121173811614

difference between rmse and mae is  0.2372282215231426


Testing SCORE FOR THE DecisionTreeRegressor(max_depth=20, min_samples_leaf=6, random_state=472) is 54.6


CROSS VAL SCORE IS -- 37.2

adjusted r2_score for DecisionTreeRegressor(max_depth=20, min_samples_leaf=6, random_state=472) is 53.84

mean sqaured error is -- 0.4535569180818089

mean aboslute error is - 0.3866738869794829

root mean sqaured error is -- 0.6734663451738393

mean aboslute error is -- 0.3866738869794829

difference between rmse and mae is  0.2867924581943564
```

3. AdaBoostRegressor



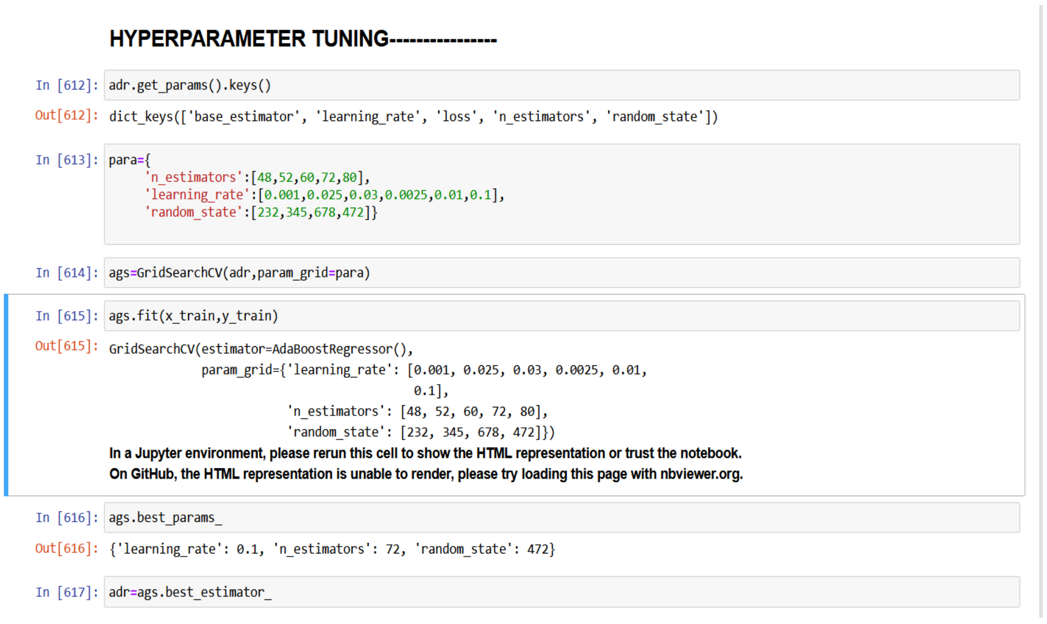
MODEL 3) ADA BOOST REGRESSOR

```
In [608]: from sklearn.ensemble import AdaBoostRegressor

In [609]: adr=AdaBoostRegressor()

In [610]: adr.fit(x_train,y_train)

Out[610]: AdaBoostRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [611]: score(adr,x_train,x_test,y_train,y_test,train=True)
          score(adr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE AdaBoostRegressor() is 38.03

mean sqaured error is -- 0.6195331496258512

root mean sqaured error is -- 0.7871042812905106

mean aboslute error is -- 0.676723818997969

difference between rmse and mae is  0.11038046229254161


Testing SCORE FOR THE AdaBoostRegressor() is 25.61


CROSS VAL SCORE IS -- 27.7

adjusted r2_score for AdaBoostRegressor() is 24.37

mean sqaured error is -- 0.7430597923246146

mean aboslute error is - 0.7397554790768298

root mean sqaured error is -- 0.8620091602324274

mean aboslute error is -- 0.7397554790768298

difference between rmse and mae is  0.12225368115559754
```

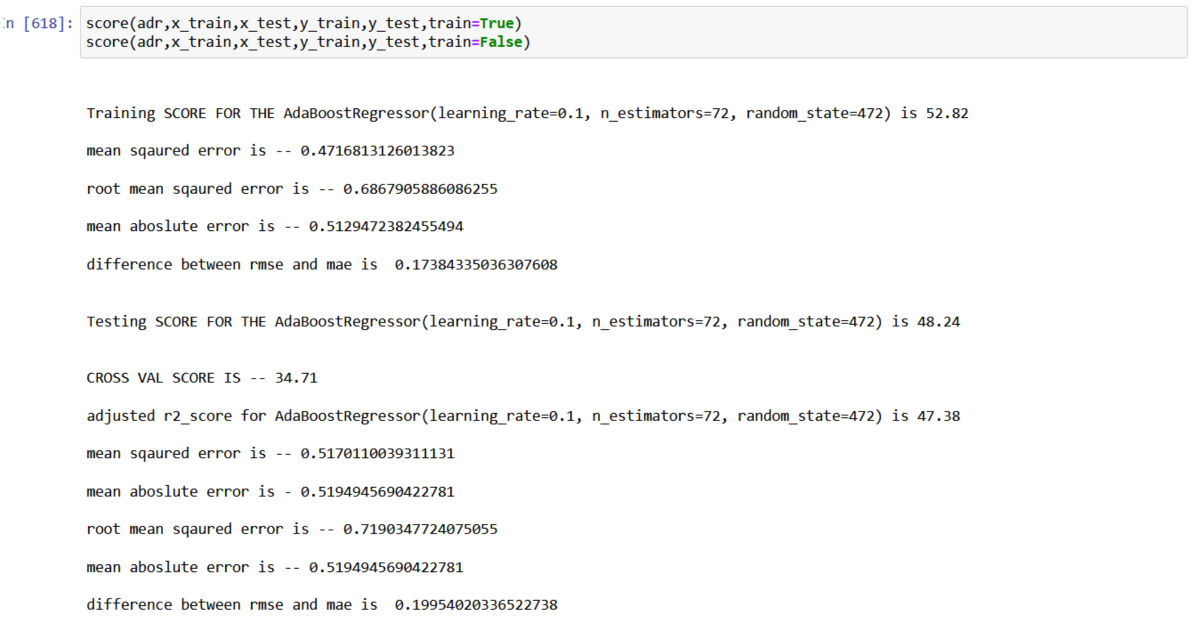


HYPERPARAMETER TUNING----------------

```
In [612]: adr.get_params().keys()

Out[612]: dict_keys(['base_estimator', 'learning_rate', 'loss', 'n_estimators', 'random_state'])

In [613]: para={
               'n_estimators':[48,52,60,72,80],
               'learning_rate':[0.001,0.025,0.03,0.0025,0.01,0.1],
               'random_state':[232,345,678,472]}

In [614]: ags=GridSearchCV(adr,param_grid=para)

In [615]: ags.fit(x_train,y_train)

Out[615]: GridSearchCV(estimator=AdaBoostRegressor(),
                       param_grid={'learning_rate': [0.001, 0.025, 0.03, 0.0025, 0.01,
                                                     0.1],
                                   'n_estimators': [48, 52, 60, 72, 80],
                                   'random_state': [232, 345, 678, 472]})
```

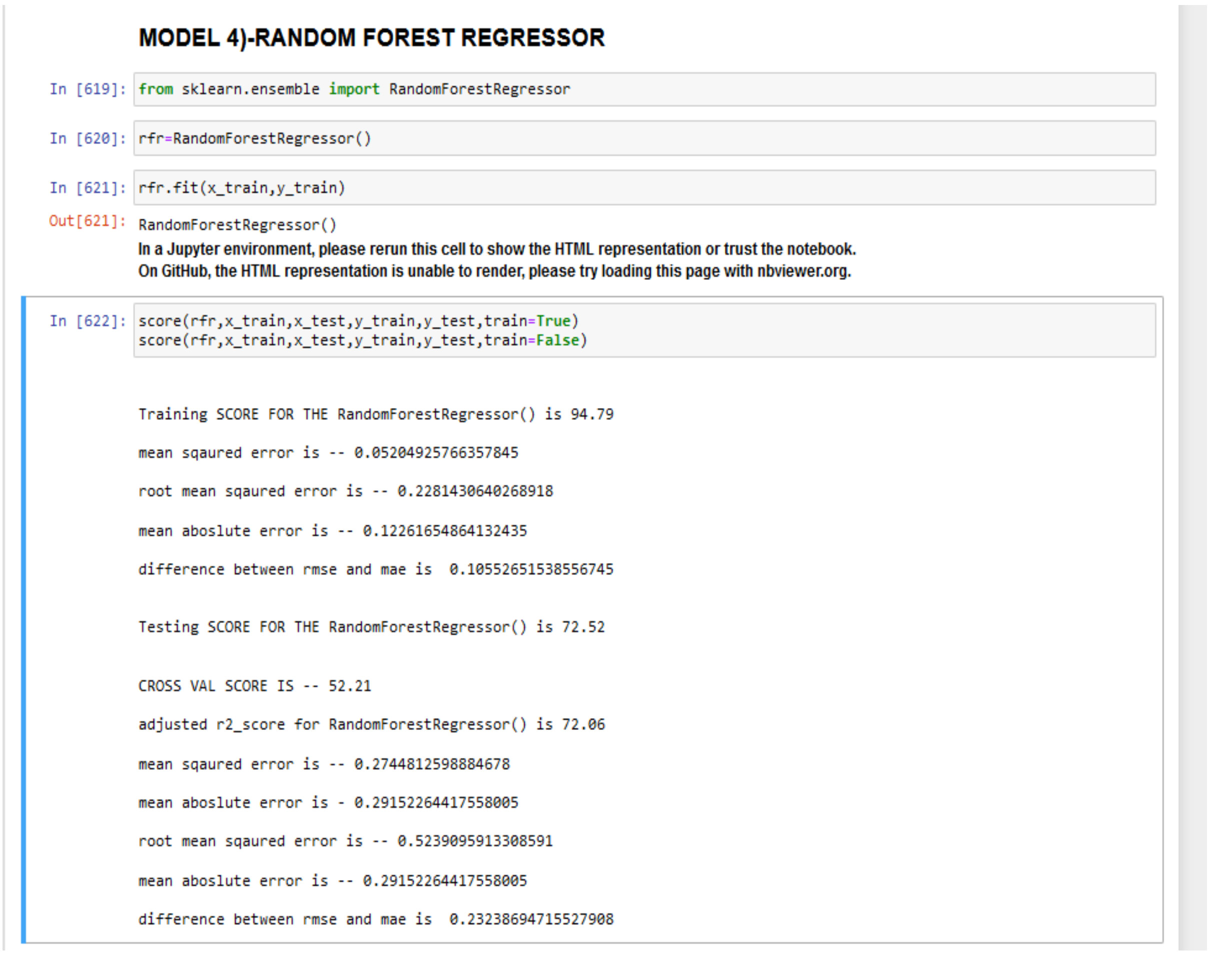
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [616]: ags.best_params_

Out[616]: {'learning_rate': 0.1, 'n_estimators': 72, 'random_state': 472}

In [617]: adr=ags.best_estimator_
```



```
n [618]: score(adr,x_train,x_test,y_train,y_test,train=True)
          score(adr,x_train,x_test,y_train,y_test,train=False)


          Training SCORE FOR THE AdaBoostRegressor(learning_rate=0.1, n_estimators=72, random_state=472) is 52.82

          mean sqaured error is -- 0.4716813126013823

          root mean sqaured error is -- 0.6867905886086255

          mean aboslute error is -- 0.5129472382455494

          difference between rmse and mae is  0.17384335036307608


          Testing SCORE FOR THE AdaBoostRegressor(learning_rate=0.1, n_estimators=72, random_state=472) is 48.24

          CROSS VAL SCORE IS -- 34.71

          adjusted r2_score for AdaBoostRegressor(learning_rate=0.1, n_estimators=72, random_state=472) is 47.38

          mean sqaured error is -- 0.5170110039311131

          mean aboslute error is - 0.5194945690422781

          root mean sqaured error is -- 0.7190347724075055

          mean aboslute error is -- 0.5194945690422781

          difference between rmse and mae is  0.19954020336522738
```

4. Random Forest Regressor



MODEL 4)-RANDOM FOREST REGRESSOR

```
In [619]: from sklearn.ensemble import RandomForestRegressor

In [620]: rfr=RandomForestRegressor()

In [621]: rfr.fit(x_train,y_train)

Out[621]: RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [622]: score(rfr,x_train,x_test,y_train,y_test,train=True)
          score(rfr,x_train,x_test,y_train,y_test,train=False)


Training SCORE FOR THE RandomForestRegressor() is 94.79

mean sqaured error is -- 0.05204925766357845

root mean sqaured error is -- 0.2281430640268918

mean aboslute error is -- 0.12261654864132435

difference between rmse and mae is  0.10552651538556745


Testing SCORE FOR THE RandomForestRegressor() is 72.52


CROSS VAL SCORE IS -- 52.21

adjusted r2_score for RandomForestRegressor() is 72.06

mean sqaured error is -- 0.2744812598884678

mean aboslute error is - 0.29152264417558005

root mean sqaured error is -- 0.5239095913308591

mean aboslute error is -- 0.29152264417558005

difference between rmse and mae is  0.23238694715527908
```

CONCLUSION

1. As we can see root mean squared error in the case of linear regression is quite low – 0.4974
2. Difference between RMSE and Mae is also low 0.236 hinting towards a good fit along the line
3. Testing accuracy and adjusted R2_scores are having closer values
4. Difference between training and testing scores is low depicting a better fit
5. Keeping all these points in mind we can conclude that **LinearRegression** Is the best choice for this problem.