
# FLIGHT PRICE PREDICTION PROJECT INTERNSHIP - 15

Submitted by:

JIJIN .P

# 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. Sapna Verma for providing the projectand 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.

## REVIEW OF LITERATURE

This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amount of data that includes more flight related features, we can do better data exploration and derive some interesting features using the available columns.

The goal of this project is to build an application which can predict the flight prices with the help of other features. In the long term, this would allow people to better explain and reviewing their purchase with each other in this increasingly digital world.

# 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 classification.

This project is done in two parts:

- ➔ Data Collection phase
- ➔ Model building phase

**Data Collection phase:**

We have to scrape at least 1500 rows of data. We can scrape more data as well, it's up to us, More the data better the model

In this section we 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..

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

➔We collected the data from difference websites like make my trip and yatra. The data is scrapped using Web scraping technique and the framework used is Selenium.

➔We scrapped nearly 10000+ 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:

```
data=pd.read_excel("FlightTrain.xlsx")
```

```
data
```

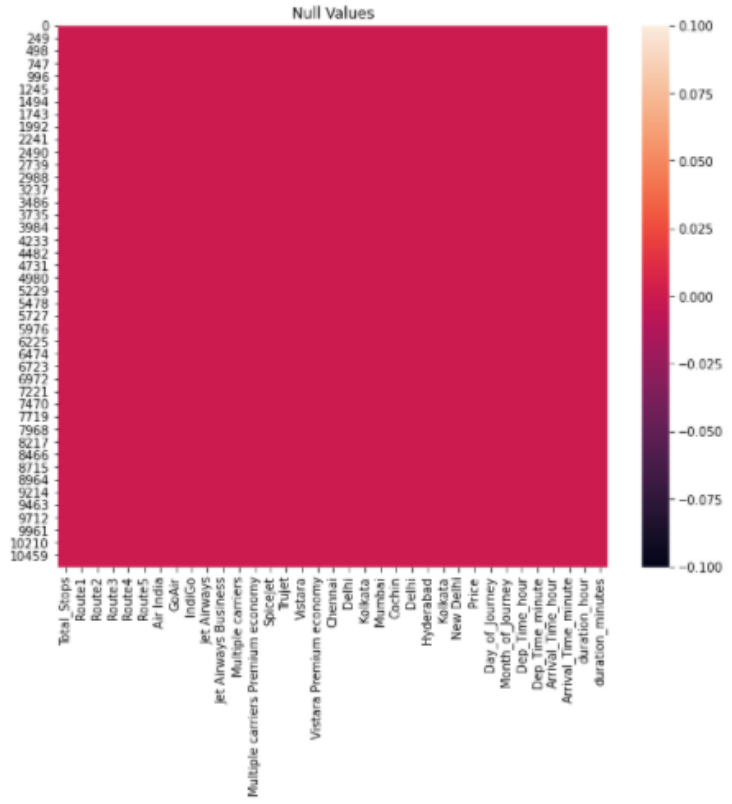
| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Duration | Total_Stops | Additional_Info | Price |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | IndiGo | 24/03/2019 | Banglore | New Delhi | BLR → DEL | 22:20 | 01:10 22 Mar | 2h 50m | non-stop | No info | 3897 |
| **1** | Air India | 1/05/2019 | Kolkata | Banglore | CCU → IXR → BBI → BLR | 05:50 | 13:15 | 7h 25m | 2 stops | No info | 7662 |
| **2** | Jet Airways | 9/06/2019 | Delhi | Cochin | DEL → LKO → BOM → COK | 09:25 | 04:25 10 Jun | 19h | 2 stops | No info | 13882 |
| **3** | IndiGo | 12/05/2019 | Kolkata | Banglore | CCU → NAG → BLR | 18:05 | 23:30 | 5h 25m | 1 stop | No info | 6218 |
| **4** | IndiGo | 01/03/2019 | Banglore | New Delhi | BLR → NAG → DEL | 16:50 | 21:35 | 4h 45m | 1 stop | No info | 13302 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **10678** | Air Asia | 9/04/2019 | Kolkata | Banglore | CCU → BLR | 19:55 | 22:25 | 2h 30m | non-stop | No info | 4107 |
| **10679** | Air India | 27/04/2019 | Kolkata | Banglore | CCU → BLR | 20:45 | 23:20 | 2h 35m | non-stop | No info | 4145 |
| **10680** | Jet Airways | 27/04/2019 | Banglore | Delhi | BLR → DEL | 08:20 | 11:20 | 3h | non-stop | No info | 7229 |
| **10681** | Vistara | 01/03/2019 | Banglore | New Delhi | BLR → DEL | 11:30 | 14:10 | 2h 40m | non-stop | No info | 12648 |
| **10682** | Air India | 9/05/2019 | Delhi | Cochin | DEL → GOI → BOM → COK | 10:55 | 19:15 | 8h 20m | 2 stops | No info | 11753 |

10683 rows × 11 columns

# DATA PRE-PROCESSING

## Checking for null-values

In [278]:
```
# heatmap on null values
plt.figure(figsize=[10,8])
sns.heatmap(data.isnull())
plt.title("Null Values")
plt.show()
```




This clearly shows there are no missing values in the dataset

## Data Cleaning

- Removing words from object data and converting them into int data type using lambda operation.
- Removing ‘,’, replacing missing data with highest weightage data.
- Removing unnecessary features

## Converting categorical data into numeric data

In [262]:
```
#Applying Label encoder for encoding extracted data from route

from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()

for i in ['Route1','Route2','Route3','Route4','Route5']:
    categorical[i]=LE.fit_transform(categorical[i])
```

In [263]:
```
categorical.head()
```

Out[263]:

| | Airline | Source | Destination | Total_Stops | Additional_Info | Route1 | Route2 | Route3 | Route4 | Route5 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | IndiGo | Banglore | New Delhi | non-stop | No info | 0 | 13 | 29 | 13 | 5 |
| 1 | Air India | Kolkata | Banglore | 2 stops | No info | 2 | 25 | 1 | 3 | 5 |
| 2 | Jet Airways | Delhi | Cochin | 2 stops | No info | 3 | 32 | 4 | 5 | 5 |
| 3 | IndiGo | Kolkata | Banglore | 1 stop | No info | 2 | 34 | 3 | 13 | 5 |
| 4 | IndiGo | Banglore | New Delhi | 1 stop | No info | 0 | 34 | 8 | 13 | 5 |

In [264]:
```
#we can drop additional_info as it doesn't add any impact. single type of data mostly
drop_col(categorical,'Additional_Info')
```

In [265]:
```
categorical['Total_Stops'].unique()
```

Out[265]:
```
array(['non-stop', '2 stops', '1 stop', '3 stops', '4 stops'],
      dtype=object)
```

In [266]:
```
#Encoding Total stops
dict={'non-stop':0,'2 stops':2,'1 stop':1, '3 stops':3, '4 stops':4}
```

In [267]:
```
categorical['Total_Stops']=categorical['Total_Stops'].map(dict)
```

In [268]:
```
categorical.head()
```

Out[268]:

| | Airline | Source | Destination | Total_Stops | Route1 | Route2 | Route3 | Route4 | Route5 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | IndiGo | Banglore | New Delhi | 0 | 0 | 13 | 29 | 13 | 5 |
| 1 | Air India | Kolkata | Banglore | 2 | 2 | 25 | 1 | 3 | 5 |
| 2 | Jet Airways | Delhi | Cochin | 2 | 3 | 32 | 4 | 5 | 5 |
| 3 | IndiGo | Kolkata | Banglore | 1 | 2 | 34 | 3 | 13 | 5 |
| 4 | IndiGo | Banglore | New Delhi | 1 | 0 | 34 | 8 | 13 | 5 |

In [269]:
```
#we can drop Airline Source and Destination as we have already extracted it separately
drop_col(categorical,'Airline')
drop_col(categorical,'Source')
drop_col(categorical,'Destination')
```

We have pre processed all data and the data is clean now. Let's join back all the columns

In [270]:
```
final_data = pd.concat([categorical,Airline,Source,Destination,data[cont_col]],axis=1)
```

## Statistical summary of the dataset

### Statistical summary

[280]:
```
data.describe()
```

[280]:

| | Total_Stops | Route1 | Route2 | Route3 | Route4 | Route5 | Air India | GoAir | IndiGo | Jet Airways | Jet Airv Busi |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.000000 | 10682.00 |
| mean | 0.824190 | 2.019378 | 11.897959 | 13.232166 | 11.792080 | 4.983056 | 0.163921 | 0.018161 | 0.192192 | 0.360326 | 0.00 |
| std | 0.675229 | 1.206239 | 8.006427 | 11.302832 | 2.948124 | 0.263913 | 0.370221 | 0.133541 | 0.394042 | 0.480117 | 0.02 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| 25% | 0.000000 | 1.000000 | 7.000000 | 4.000000 | 13.000000 | 5.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| 50% | 1.000000 | 2.000000 | 8.000000 | 6.000000 | 13.000000 | 5.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00 |
| 75% | 1.000000 | 3.000000 | 14.000000 | 29.000000 | 13.000000 | 5.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.00 |
| max | 4.000000 | 4.000000 | 44.000000 | 29.000000 | 13.000000 | 5.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.00 |

Since most of the columns are categorical type. we have to deal only with Price column which is continous. There are possible outliers in the column as there is a huge difference between 75th percentile and maximum values

As Price is in continous data. The model will be of Regression type

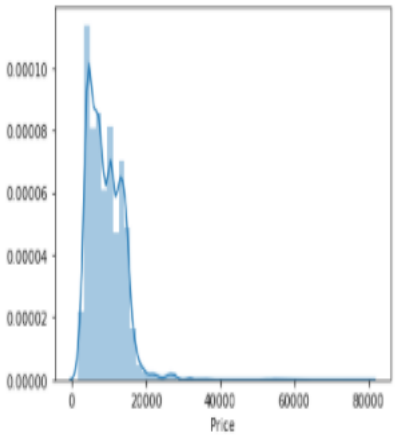
## Checking outliers

## Checking skewness and data distribution

### Target Variable Analysis - Checking outliers

```
[285]: sns.distplot(data.Price)
```

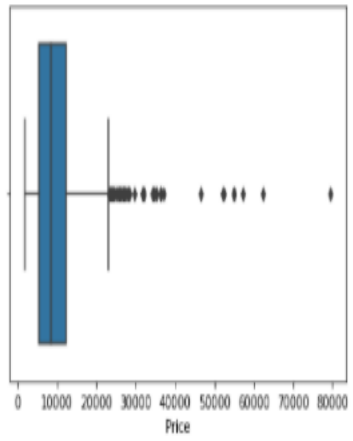
```
t[285]: <matplotlib.axes._subplots.AxesSubplot at 0x2926248fd90>
```




The price higher than 40,000 should be replaced by median to remove the outliers and skewness. The data distributed is rightly skewed

```
[286]: sns.boxplot(data.Price)
```

```
t[286]: <matplotlib.axes._subplots.AxesSubplot at 0x29262c00760>
```




Conclusion is The price higher than 40,000 should be replaced by median to remove the outliers and skewness.

## Handling outliers

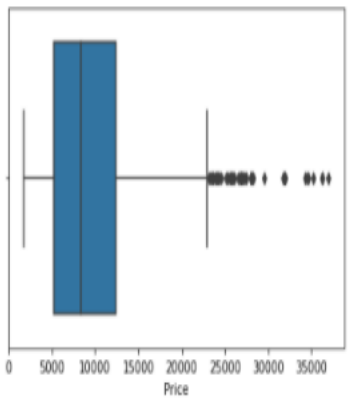
### Handling Outliers ¶

we have some outliers in Price column. we can replace t by median

In [287]:
```
data['Price']=np.where(data['Price']>=40000,data['Price'].median(),data['Price'])
```

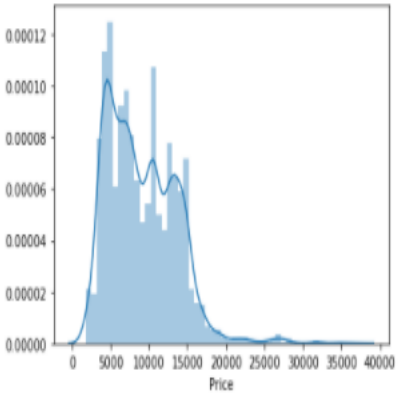
In [288]:
```
sns.boxplot(data.Price)
```

Out[288]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29262c731f0>
```




In [289]:
```
sns.distplot(data.Price)
```

Out[289]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29262cba850>
```




Now data looks much better

# 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 required libraries

```
#Basic libraries
import pandas as pd
import numpy as np

#Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#Importing required metrices and model for the dataset
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.linear_model import LinearRegression,Lasso,ElasticNet,Ridge
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV

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

# MODEL/S DEVELOPMENT AND EVALUATION

## Preparing dataset for model training

In [329]:
```
#initialization of x and y
x=data.drop('Price',axis=1)
y=data['Price']
```

In [330]:
```
x.head()
```

Out[330]:

| | Total_Stops | Route1 | Route2 | Route3 | Route4 | Route5 | Air India | GoAir | IndiGo | Jet Airways | Jet Airways Business | Multiple carriers | Multiple carriers Premium economy | SpiceJet | Vistara | Vistara Premium economy | Chennai | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 13 | 29 | 13 | 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 2 | 2 | 25 | 1 | 3 | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 2 | 2 | 3 | 32 | 4 | 5 | 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 3 | 1 | 2 | 34 | 3 | 13 | 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 4 | 1 | 0 | 34 | 8 | 13 | 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

In [331]:
```
x.shape
```

Out[331]: (10682, 33)

In [332]:
```
y.shape
```

Out[332]: (10682,)

## Treating skewness and scaling the data

```
#We are treating skewness by using square root transform
for col in df_x.skew().index:
    if col in df_x.describe().columns:
        if df_x[col].skew()>0.55:
            df_x[col]=np.sqrt(df_x[col])
        if df_x[col].skew()<-0.55:
            df_x[col]=np.sqrt(df_x[col])
```

```
df_x.skew()   #Checking skewness after treating it
```

```
year           -0.656801
km_driven      -0.173734
fuel            0.156582
transmission   -2.214140
dtype: float64
```

We can see that skewness has been treated and we can proceed further model building process

### Scaling the data

Sometimes model can be biased to higher values in dataset, so it is better to scale the dataset so that we can bring all the columns in common range. We can use StandardScaler here.

```
#Scaling the dataset using StandardScaler
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(df_x)
x=pd.DataFrame(x,columns=df_x.columns)
```

# Model Building

Finding best random state and r2_score

Out[332]: (10682,)

In [333]:
```
#finding best random_state
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,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=.33,random_state=i)
    LR=LinearRegression()
    LR.fit(x_train,y_train)
    predrf=LR.predict(x_test)
    r2=r2_score(y_test,predrf)
    if r2>maxAccu:
        maxAccu=r2
        maxRS=i
print("Best accuracy is ",maxAccu," on random_state ",maxRS)
```

```
Best accuracy is  0.6373826462672242  on random_state  154
```

After initializing the instances of the model, we will run a for loop with the required metrics and other regression related scores and we will append the obtained values to empty lists. The code will be as follows:

:
```
# different algorithm going to use
from sklearn.ensemble import GradientBoostingRegressor

lr=LinearRegression()
dtr=DecisionTreeRegressor()
svr=SVR()
rfr=RandomForestRegressor()
adr=AdaBoostRegressor(n_estimators=200)
gdr=GradientBoostingRegressor()
knn=KNeighborsRegressor()
```

:
```
#code for Training and Prediction

def eval(x):
    mod=x
    print(mod)
    mod.fit(x_train,y_train)
    pred=mod.predict(x_test)

    print("r2_score(predicted_train) is ",r2_score(y_train,mod.predict(x_train)))
    print("\n")
    print("r2_score is :",r2_score(y_test,pred))
    print("\n")
    print("Score of trained data :",mod.score(x_train,y_train))
    print("\n")
    print("Score of test data :",mod.score(x_test,y_test))
    print("\n")
    print("Error")
    print("mean absolute error (MAE): ",mean_absolute_error(y_test,pred))
    print("mean squared error (MSE): ",mean_squared_error(y_test,pred))
    print("RMSE",np.sqrt(mean_squared_error(y_test,pred)))
    sns.distplot(y_test-pred)
```

Random Forest Regressor is considered to be the best model with a learning percentage of 95.53% and Accuarcy of 82.93% and 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

```
# hyperparameter tuning of RandomForest

#no. of tress random_forest
n_estimators = [int(x) for x in np.linspace(start=150,stop=1000,num=6)]
#max levels in tree
max_depth = [int(x) for x in np.linspace(start=6,stop=30,num=5)]
#min no. of splitting required to split a node
min_samples_split = [2,7,10]
#min no. of sample required at each leaf node
min_samples_leaf = [2,5,10]
#max_features
max_features=['auto','sqrt']
# generate a dictionary of all the Hyper Parameters
rand_params = {'n_estimators':n_estimators,
               'max_depth':max_depth,
               'min_samples_split':min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'max_features':max_features,
              }
```

```
from sklearn.model_selection import RandomizedSearchCV

rand_rfr= RandomizedSearchCV(estimator=rfr,param_distributions=rand_params,cv= 5,n_jobs=-1)
```

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

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [6, 12, 18, 24, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [2, 5, 10],
                                        'min_samples_split': [2, 7, 10],
                                        'n_estimators': [150, 320, 490, 660,
                                                         830, 1000]})
```

```
best_parameters=rand_rfr.best_params_
```

```
best_parameters
```

```
{'n_estimators': 320,
 'min_samples_split': 2,
 'min_samples_leaf': 2,
```

After hypertuning, The accuracy of the model is increased by almost 2 percent that is from 82% to 84%. RandomForestRegressor (hypertuned model) is chosen to be the best model with almost 84.35% accuracy

# FINAL THE MODEL

## Comparing original and predicted price

## Saving the model

### Conclusion - Saving the model for future use¶

54]:
```
joblib.dump(rfr_hyp,"rfr_hyp_flightprice_prediction.obj")
```

54]:
```
['rfr_hyp_flightprice_prediction.obj']
```

55]:
```
#Lets Check loading the file
flightprice_prediction=joblib.load("rfr_hyp_flightprice_prediction.obj")
```

56]:
```
flightprice_prediction.score(x_test,y_test)
```

56]:
```
0.8435107890447374
```

57]:
```
pred=rfr_hyp.predict(x_test)
Conclusion=pd.DataFrame([flightprice_prediction.predict(x_test)[:],pred[:]],index=["Predicted","Original"])
```

59]:
```
Conclusion.round(2)
```

59]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Predicted | 12913.21 | 3824.94 | 6810.51 | 5150.08 | 8910.72 | 13804.32 | 10424.41 | 4884.42 | 2375.99 | 17248.65 | 27464.75 | 11759.56 | 3881.42 | 16478.48 | 12898.98 | 12402 |
| Original | 12913.21 | 3824.94 | 6810.51 | 5150.08 | 8910.72 | 13804.32 | 10424.41 | 4884.42 | 2375.99 | 17248.65 | 27464.75 | 11759.56 | 3881.42 | 16478.48 | 12898.98 | 12402 |

The model is working well

After Training and Testing 7 algorithm model. The best accuracy model was determined as random forest regressor with almost 84% accuracy after all the data cleaning, pre-processing, training and prediction as well as evaluation phase.

The Fligh price is predicted for the test dataset provided

### Random Forest Regressor is the final model with 84.35% accuracy

# 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, analysing the data and building a model.

**->** First, we collected the used cars data from different websites like mmt, yatra.com and 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 10000 plus of data which contained the selling priceand 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 analyse 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 outliers, 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 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 the model in obj format and then saved the predicted values in a csv format.

**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.

**********
**********