FLIP ROBO

FLIGHT PRICE PREDICTION PROJECT

Submitted by:

Mrs. Chaitali Mohit Nakade

ACKNOWLEDGMENT

- Towards data science– A medium publication sharing concepts, idea and codes
- Analytics Vidya — Flight Price prediction/ Nikita verma
- Code to Express – Snehanshu Sengupta
- Yatra.com
- Easemytrip.com

Article Resources

- Notebook and Data: GitHub, Jupyter Notebook
- Libraries: numpy, pandas, sklearn, scipy, joblib, matplotlib, seaborn, statsmodels

INTRODUCTION

In this story we will deliberate the broad information about model structure of "Flight Price Prediction Project" through linear regression/Random Forest regressor/Decision tree regressor/Root mean squared error and find the best model from that. The data consists of records of randomly 4104 rows and 11 features. Now, discuss about the model building.

- ## Business Problem Framing

Flight ticket price prediction is very hard to guess. We can check the price of tickets will goes low and high very frequently. The price of ticket we check today might be different tomorrow but with the help of machine learning we can predict the price of tickets. HOW....?

So, Let's work on flight price prediction project.

- ## Conceptual Background of the Domain Problem

There are various factors which affects the prices of flights > Date, Distance, flight time, number of stops, Duration etc. These factors help create a pattern to decide the price of a flight, and the machine learning models get trained on this pattern to make the predictions in future.

This project contains three phases: -

1. ### Data collection phase:

   In data collection process I had scrap the data from **yatra.com** with selenium and converting that data into structured format. I had selected 4104 rows and 11 columns out of which 10 features are independent variable s and "PRICE" is a target variable. In model building we will go through those rows and columns.

2. ### Data Analysis:

   After cleaning the data, I had done some analysis on the data.

   i. Do airfares change frequently?
   ii. Do they move in small increments or in large jumps?
   iii. Do they tend to go up or down over time?
   iv. What is the best time to buy so that the consumer can save the most by taking the least risk?
   v. Does price increase as we get near to departure date?
   vi. Is Indigo cheaper than Jet Airways?
   vii. Are morning flights expensive?

3. Model Building phase:

Following are the complete life cycle of data science model building. Include all the steps like.

a. Data Cleaning (Identifying null values, filling missing values and removing outliers)
b. Exploratory Data Analysis
c. Data Pre-processing
d. Model Building
e. Model Evaluation
f. Selecting the best mode

• Review of Literature

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

With the help of this project, we will find out when we need to book the tickets and which airlines provide us tickets with very less costing. How people will save our money?

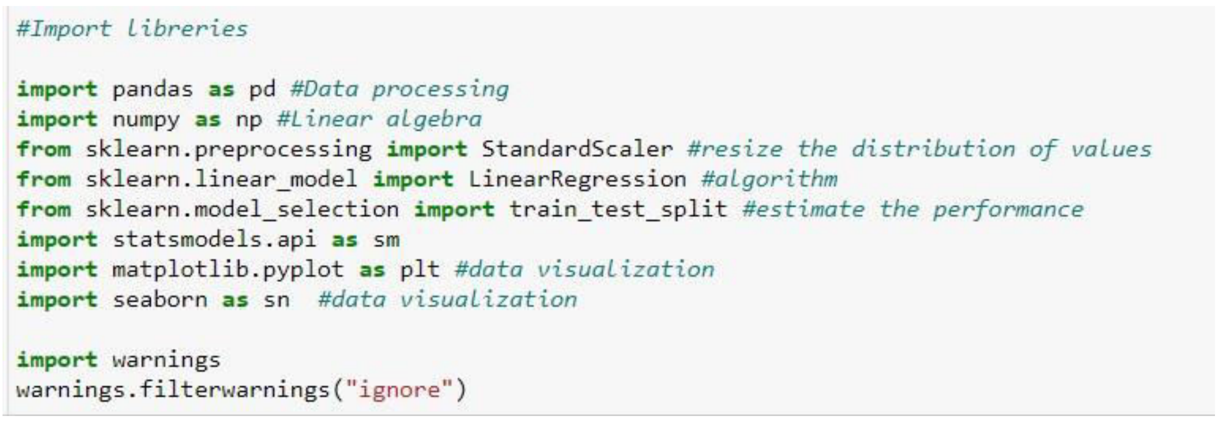
The prediction will help a traveller to decide a specific airline as per there budget. Single entries of current or previous data can be made. This training set is used to train the algorithm for accurate predictions. Traveller get the fare prediction handy using which it’s easy to decide the airlines.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

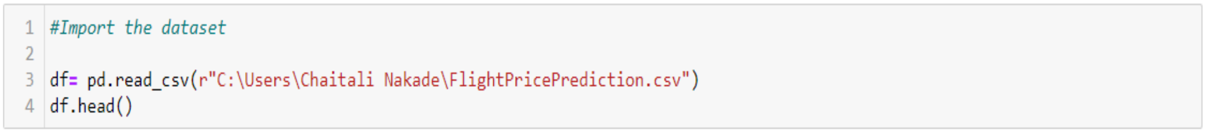
  Now we will work on model building, we are using machine learning techniques to predict Used car prices. Here we are having dataset named as df. The dataset contains 4104 rows and 11 columns.

  Start work with dataset with importing necessary libraries and dataset.

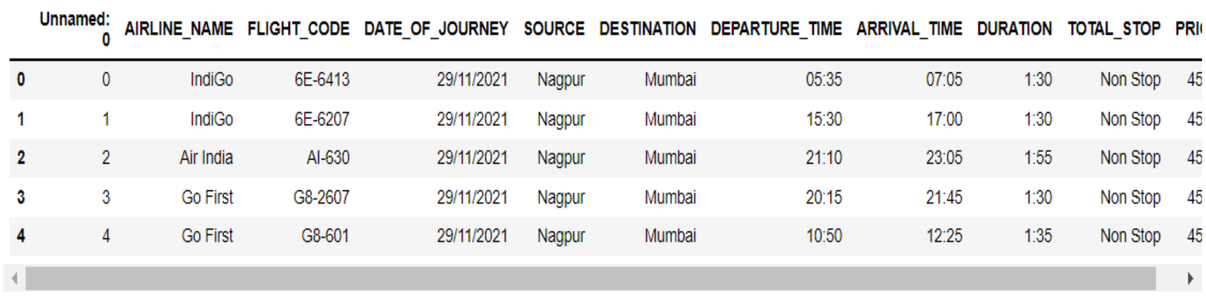


```
#Import libreries

import pandas as pd #Data processing
import numpy as np #Linear algebra
from sklearn.preprocessing import StandardScaler #resize the distribution of values
from sklearn.linear_model import LinearRegression #algorithm
from sklearn.model_selection import train_test_split #estimate the performance
import statsmodels.api as sm
import matplotlib.pyplot as plt #data visualization
import seaborn as sn  #data visualization

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

All these libraries are used for model building and data processing.



```
#Import the dataset

df= pd.read_csv(r"C:\Users\Chaitali Nakade\FlightPricePrediction.csv")
df.head()
```

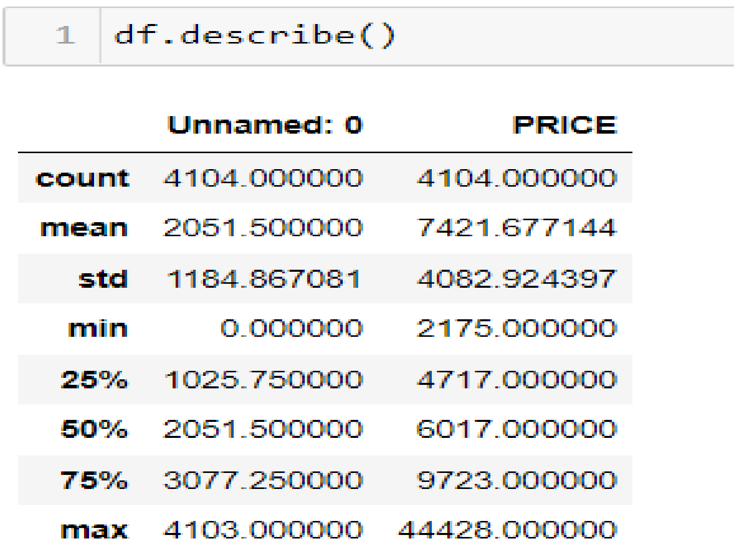


	Unnamed: 0	AIRLINE_NAME	FLIGHT_CODE	DATE_OF_JOURNEY	SOURCE	DESTINATION	DEPARTURE_TIME	ARRIVAL_TIME	DURATION	TOTAL_STOP	PRI
0	0	IndiGo	6E-6413	29/11/2021	Nagpur	Mumbai	05:35	07:05	1:30	Non Stop	45
1	1	IndiGo	6E-6207	29/11/2021	Nagpur	Mumbai	15:30	17:00	1:30	Non Stop	45
2	2	Air India	AI-630	29/11/2021	Nagpur	Mumbai	21:10	23:05	1:55	Non Stop	45
3	3	Go First	G8-2607	29/11/2021	Nagpur	Mumbai	20:15	21:45	1:30	Non Stop	45
4	4	Go First	G8-601	29/11/2021	Nagpur	Mumbai	10:50	12:25	1:35	Non Stop	45

The complete dataset is imported in variable name 'df' and we can see there are some columns are need to alter them into numeric data because machine learning will not work with object data type but before that need to analyse the data.

Exploratory Data Analysis

- Data Sources and their formats



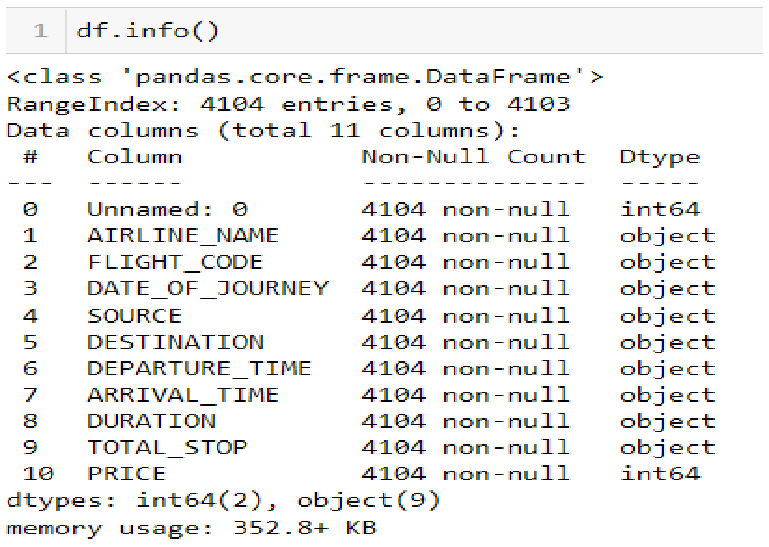
```
df.describe()
```

	Unnamed: 0	PRICE
count	4104.000000	4104.000000
mean	2051.500000	7421.677144
std	1184.867081	4082.924397
min	0.000000	2175.000000
25%	1025.750000	4717.000000
50%	2051.500000	6017.000000
75%	3077.250000	9723.000000
max	4103.000000	44428.000000

Express more about dataset:
Maximum columns are having object datatype only Unnamed: 0 and PRICE column is shown in above table,

a. As we can see minimum price of ticket is 2175Rs. and maximum fare of ticket is 44428rs. Maximum price is very large as compare to minimum fare.

Similarly, we can conclude so many things by just observing at the describe function.



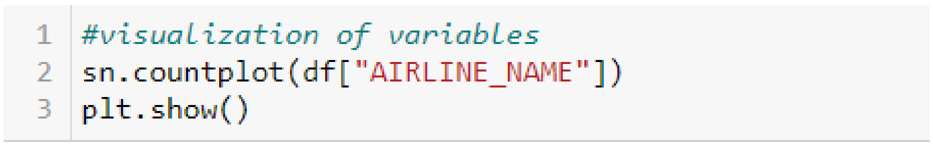
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4104 entries, 0 to 4103
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Unnamed: 0       4104 non-null   int64 
 1   AIRLINE_NAME     4104 non-null   object
 2   FLIGHT_CODE      4104 non-null   object
 3   DATE_OF_JOURNEY  4104 non-null   object
 4   SOURCE           4104 non-null   object
 5   DESTINATION      4104 non-null   object
 6   DEPARTURE_TIME   4104 non-null   object
 7   ARRIVAL_TIME     4104 non-null   object
 8   DURATION         4104 non-null   object
 9   TOTAL_STOP       4104 non-null   object
 10  PRICE            4104 non-null   int64 
dtypes: int64(2), object(9)
memory usage: 352.8+ KB
```

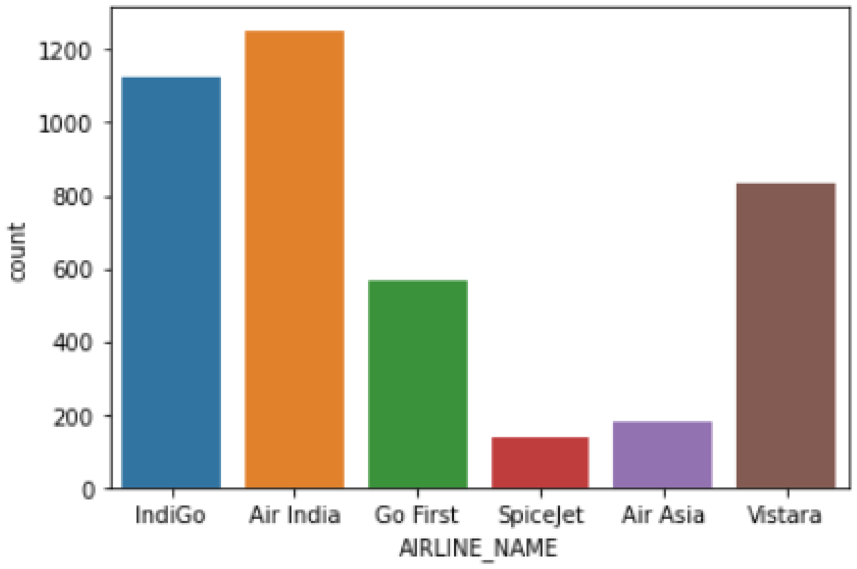
- There are 4104 Rows and 11 Columns in dataset
- We can see no columns are having null values.
- We can find here nine columns are having object data types

and rest two columns are having integer datatype.

- Data Inputs- Logic- Output Relationships (Visualization of data)
  Univariate analysis: Proceeding with the plotting and analysing the data using seaborn, matplotlib libraries –
  Plotting count plots for categorical data –

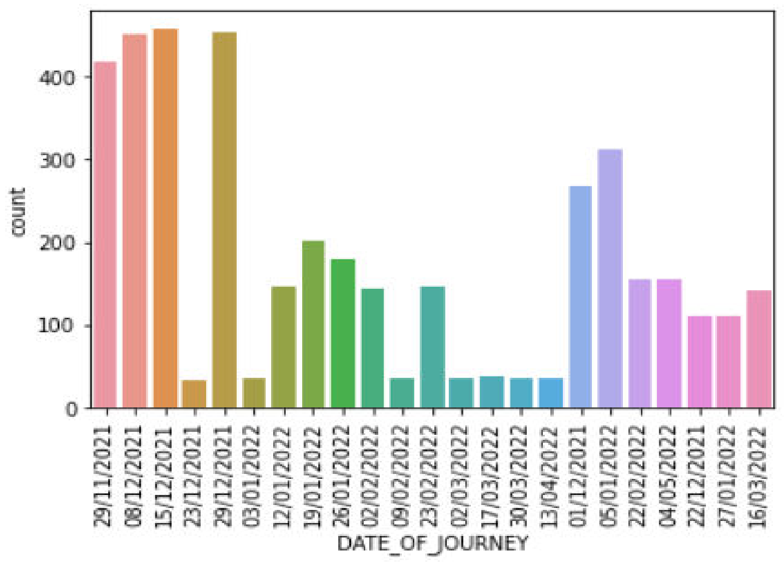


```
#visualization of variables
sn.countplot(df["AIRLINE_NAME"])
plt.show()
```

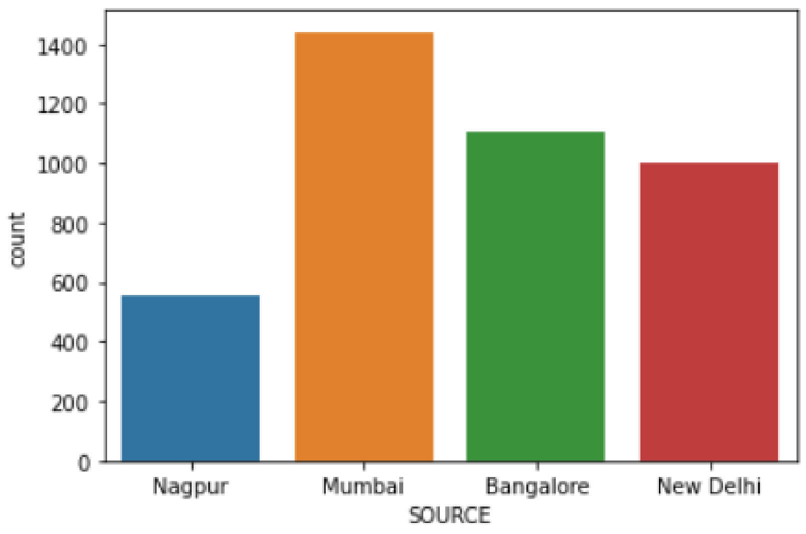


1. **AIRLINE_NAME:**
   - Air India is the most preferred airline with the highest row count, followed by Indigo and Vistara.
   - Count for Go First, SpiceJet, and Air Asia business is quite low.

2. **Date of journey:**
   As compare to other majority of flights are departed on 15/12/21 followed by 29/12/21 and 8/12/21



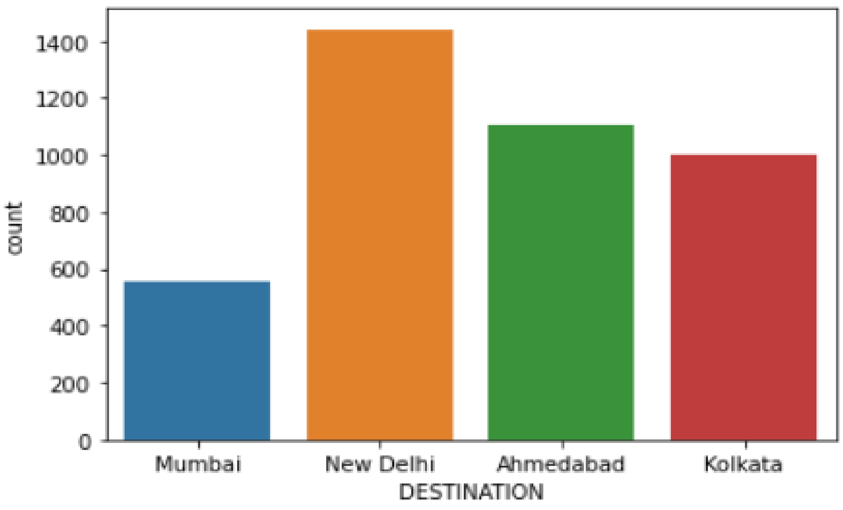
- **<u>SOURCE</u>:**



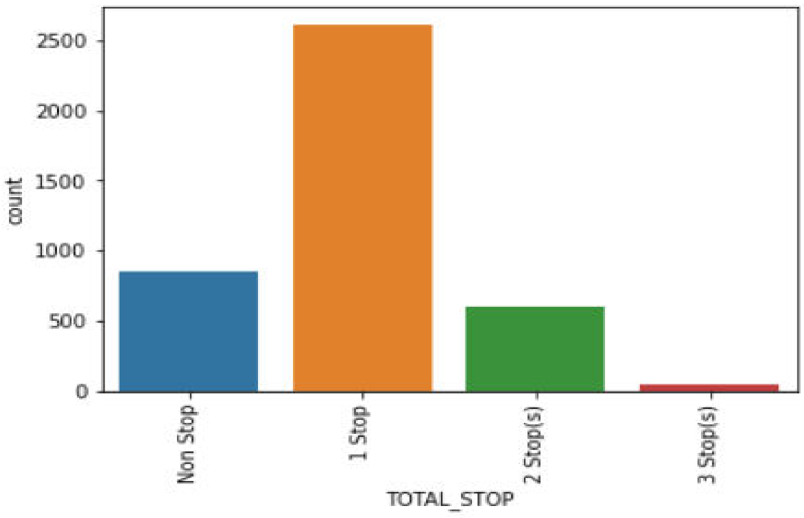
  - Majority of the flights take off from Mumbai followed by Bangalore.
  - Even I had collected maximum dates data for Nagpur column but still we can see very less flights are take-off from Nagpur.

- **<u>Destination:</u>**

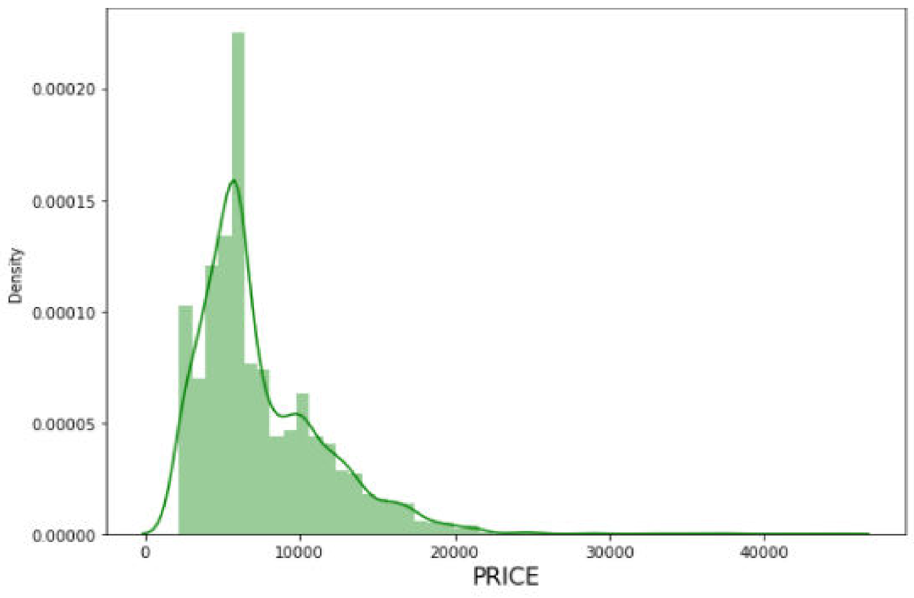
  - Maximum flights land in New Delhi.
  - Nagpur has the lowest count of receiving the flights compare to other.



- **<u>TOTAL STOPS</u>**:
  - Majority of the flights have stops as 1, flights with 2 and 3 stops are quite low.

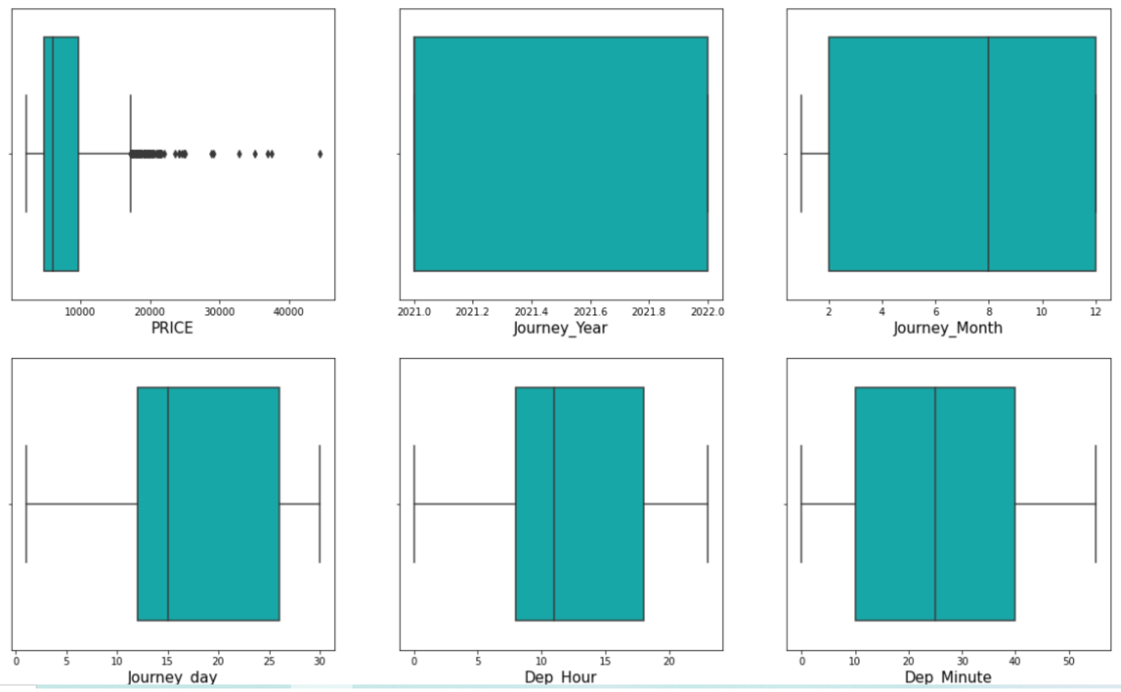


- **<u>Distribution of 'Price' column:</u>**

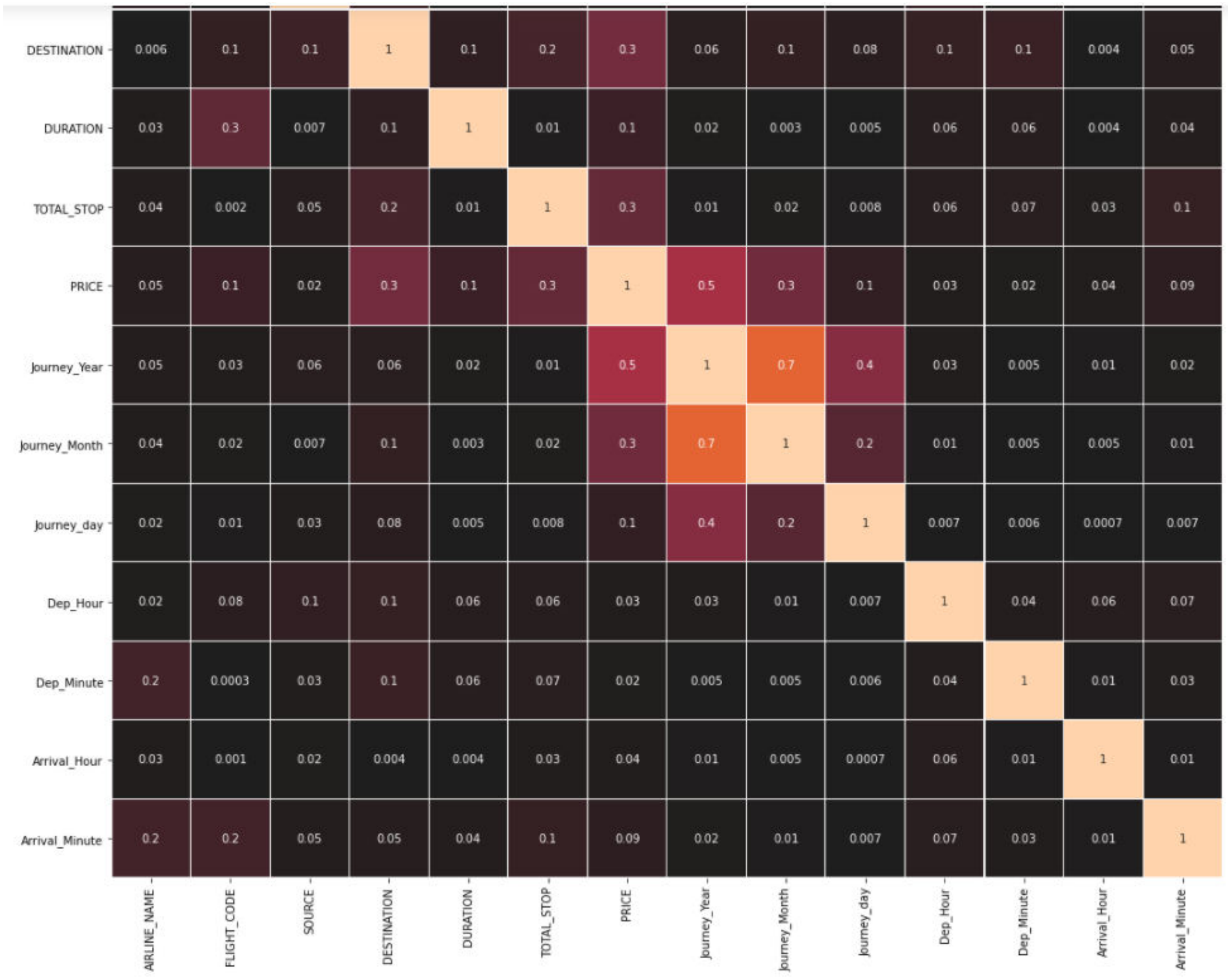


- The price column contains the minimum value as 2175 and maximum value as 44428. Majority of the flights have price range between 1759–20k, and number of flights having prices greater than 20k are quite less. Price range is skewed towards right.

- <span style="color:red">Check is their outliers are present in dataset?</span>
  - As we can see in figure below there are no outliers present in whole dataset only PRICE column contains outliers but PRICE is a target variable, so no need to remove outliers from that.

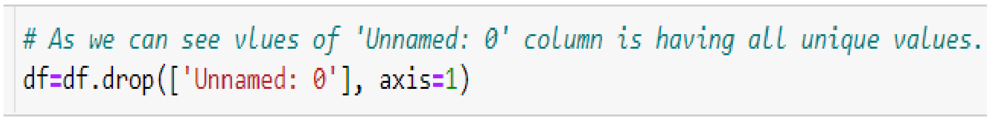


As shown in heatmap journey_year is highly correlated with target variable i. e. Price. Hence, it is an important variable to predict the ticket fare followed by total_stops and Destination.



- Data Pre-processing Done

a. Drop unnecessary columns



```
# As we can see vlues of 'Unnamed: 0' column is having all unique values.
df=df.drop(['Unnamed: 0'], axis=1)
```

'Unnamed:0' which are having poor correlation with target variable will be deleted or we can drop that column from the dataset to avoid overfitting problem.

b. Encode the columns which are having object data type

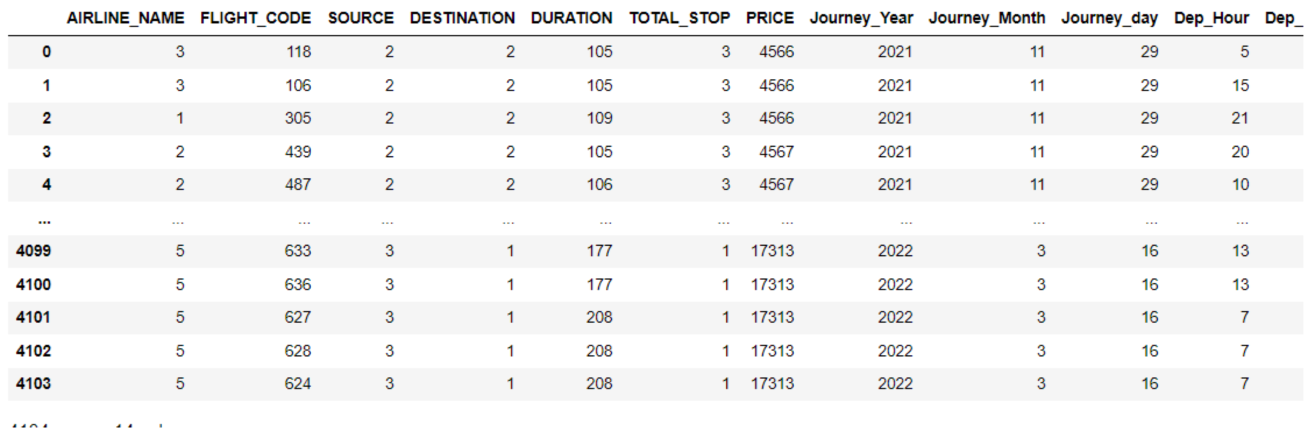
Encoding data is very important for model because data will not work with object data type. In this project I will encode the object columns with label encoder.

Label encoder is work on alphabetical order and encode the column. I am encoding all the columns which are having object data type.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
for i in df.columns:
    if df[i].dtypes=="object":
        df[i]=le.fit_transform(df[i].values.reshape(-1,1))
```

After encoding all columns, the dataset will be in numeric format and it will look like,



	AIRLINE_NAME	FLIGHT_CODE	SOURCE	DESTINATION	DURATION	TOTAL_STOP	PRICE	Journey_Year	Journey_Month	Journey_day	Dep_Hour	Dep_
0	3	118	2	2	105	3	4566	2021	11	29	5	
1	3	106	2	2	105	3	4566	2021	11	29	15	
2	1	305	2	2	109	3	4566	2021	11	29	21	
3	2	439	2	2	105	3	4567	2021	11	29	20	
4	2	487	2	2	106	3	4567	2021	11	29	10	
...	...	...	...	...	...	...	...	...	...	...	...	
4099	5	633	3	1	177	1	17313	2022	3	16	13	
4100	5	636	3	1	177	1	17313	2022	3	16	13	
4101	5	627	3	1	208	1	17313	2022	3	16	7	
4102	5	628	3	1	208	1	17313	2022	3	16	7	
4103	5	624	3	1	208	1	17313	2022	3	16	7	

- Hardware and Software Requirements and Tools Used

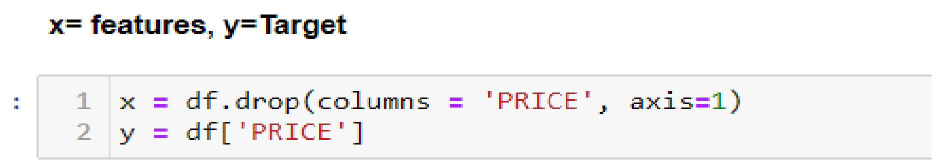
a. **Notebook and Data:** GitHub, Jupyter Notebook

b. **Libraries:** numpy, pandas, sklearn, scipy, joblib, matplotlib, seaborn, statsmodels

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

Now we will train several machine learning models and compare their results. We need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation. After that I will compare the difference of accuracy score and cross validation score of all models. The model which is having minimum difference will be consider as a best model and further procedures will done on that model. Following are the basic steps of model building.

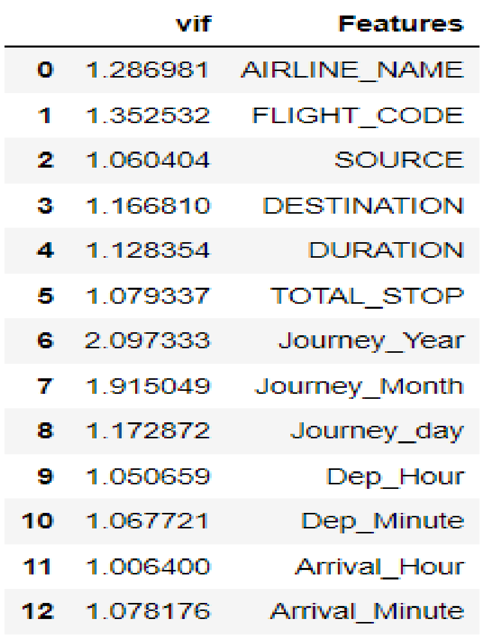


x= features, y=Target

```
x = df.drop(columns = 'PRICE', axis=1)
y = df['PRICE']
```

The next step is to bring the data to a common scale, since there are certain columns with very small values and some columns with high values. This process is important as values on a similar scale allow the model to learn better.

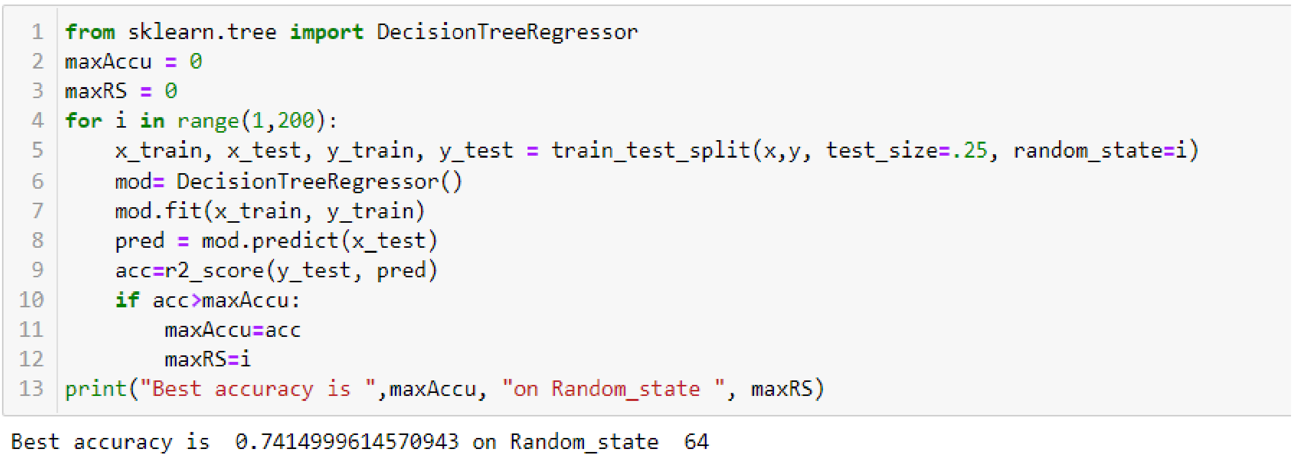
Now, check the variance inflation factor to avoid multicollinearity problem. In this model I am having all the values of variance inflation factor is less than 10 so safe to proceed further.



	vif	Features
0	1.286981	AIRLINE_NAME
1	1.352532	FLIGHT_CODE
2	1.060404	SOURCE
3	1.166810	DESTINATION
4	1.128354	DURATION
5	1.079337	TOTAL_STOP
6	2.097333	Journey_Year
7	1.915049	Journey_Month
8	1.172872	Journey_day
9	1.050659	Dep_Hour
10	1.067721	Dep_Minute
11	1.006400	Arrival_Hour
12	1.078176	Arrival_Minute

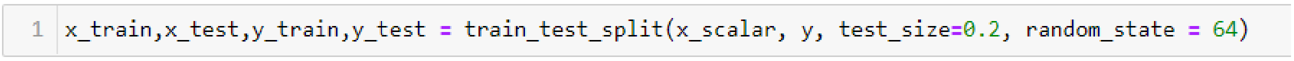
- ## Testing of Identified Approaches (Algorithms)
- Find best random state: first of all, find the best random state

  And then, splitting our dataset by train test split with random state: 64



```
from sklearn.tree import DecisionTreeRegressor
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=.25, random_state=i)
    mod= DecisionTreeRegressor()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu, "on Random_state ", maxRS)
```

```
Best accuracy is  0.7414999614570943 on Random_state  64
```



```
x_train,x_test,y_train,y_test = train_test_split(x_scalar, y, test_size=0.2, random_state = 64)
```

- ## Run and Evaluate selected models

  We now proceed to the main step of our machine learning, fitting the model and predicting the outputs. We fit the data into multiple regression models to compare the performance of all models and select the best model –

  1. Random forest regressor:

     Random forest regressor is a type of supervised machine learning algorithm. It gives the single result which is made with combination of multiple decision tree output.

✓ After applying Random Forest regression method, the output of R2 score is: **82.89%**

✓ And cross validation score is**: 44.51%**

  2. Decision Tree Regressor

     Decision tree uses the tree illustration to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

     ✓ After applying Decision Tree regression method, the output of R2 score is: **72.42%**

     ✓ And cross validation score is: **65.08%**

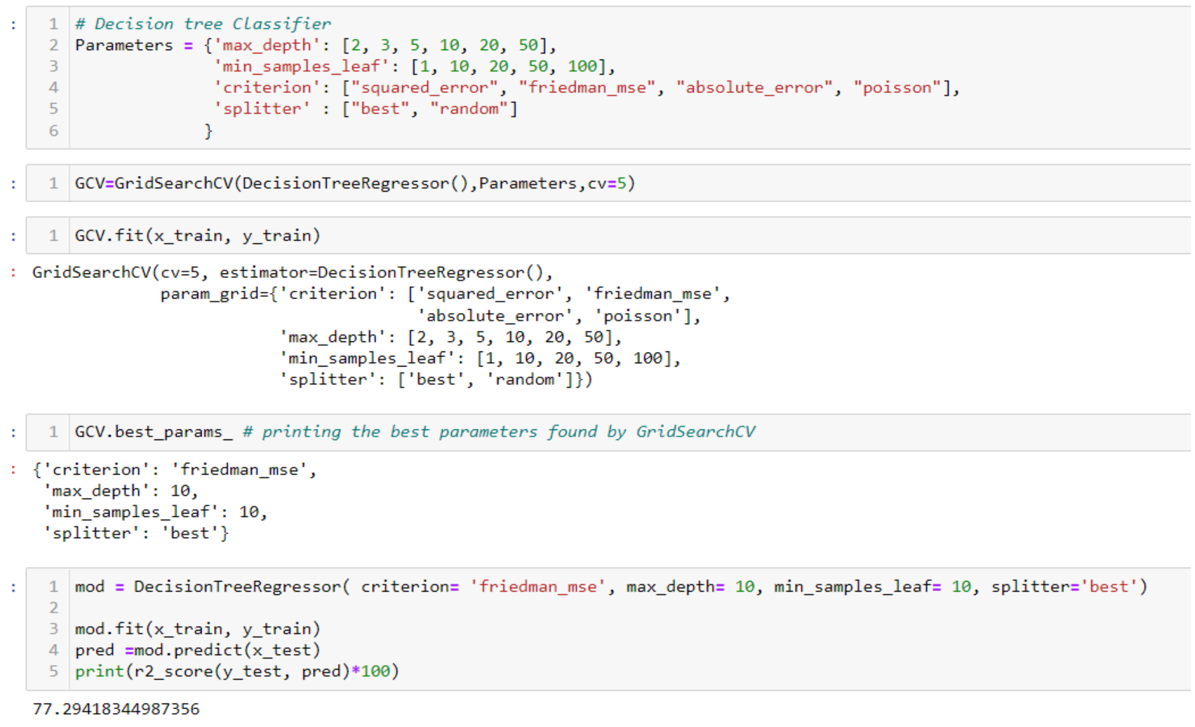
- **Interpretation of the Results**
- *Comparison of all models:*

Sr. No.	Algorithms	Accuracy score (a)	Cross -Validation Score(b)	Difference (a-b)
1.	Random forest regressor	82.89%	44.51%	32.42 %
2.	Decision Tree Regressor	72.42%	65.08%	6. 34%

As shown in above table Decision Tree regressor is having minimum difference, so Decision Tree regressor is a best model. Now, I will work on best model with hyper parameter tunning.

• Hyper Parameter Tuning:

Grid search cross validation (GCV) in hyperparameter tuning will help us to finding the best hyperparameter and tune it with training data set and give best R2 score:



```
# Decision tree Classifier
Parameters = {'max_depth': [2, 3, 5, 10, 20, 50],
              'min_samples_leaf': [1, 10, 20, 50, 100],
              'criterion': ["squared_error", "friedman_mse", "absolute_error", "poisson"],
              'splitter' : ["best", "random"]
             }
```

```
GCV=GridSearchCV(DecisionTreeRegressor(),Parameters,cv=5)
```

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

```
GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['squared_error', 'friedman_mse',
                                       'absolute_error', 'poisson'],
                         'max_depth': [2, 3, 5, 10, 20, 50],
                         'min_samples_leaf': [1, 10, 20, 50, 100],
                         'splitter': ['best', 'random']})
```

```
GCV.best_params_ # printing the best parameters found by GridSearchCV
```

```
{'criterion': 'friedman_mse',
 'max_depth': 10,
 'min_samples_leaf': 10,
 'splitter': 'best'}
```

```
mod = DecisionTreeRegressor( criterion= 'friedman_mse', max_depth= 10, min_samples_leaf= 10, splitter='best')

mod.fit(x_train, y_train)
pred =mod.predict(x_test)
print(r2_score(y_test, pred)*100)
```

```
77.29418344987356
```

In above figure we can see the default hyperparameter which I was tunned with Decision tree regression with GCV and then find the best parameters, now fit it in to model and find the R2 score:

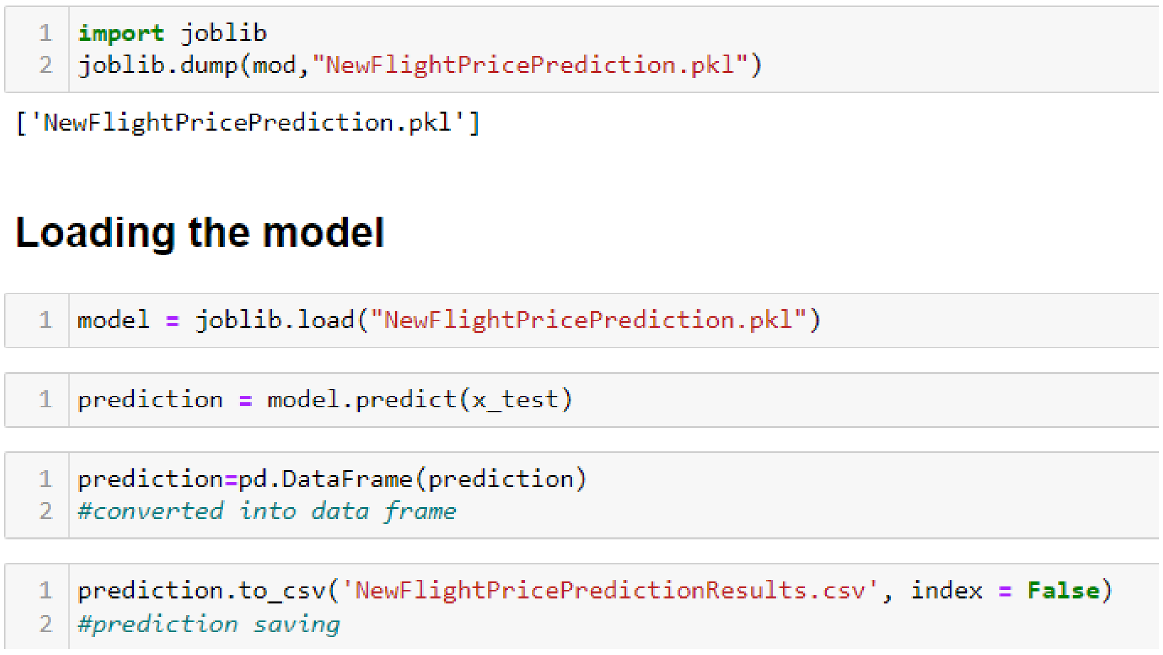
After hyperparameter tuning the r2 score is raise to: 77.29%

Boosting of model: -

With the help of Gradient Boosting Regressor, I will boost my model and then my R2 score raise to:77.25%

Saving and loading the model:

I save this model with name "NewFlightPricePrediction.pkl"



```
import joblib
joblib.dump(mod,"NewFlightPricePrediction.pkl")
```

```
['NewFlightPricePrediction.pkl']
```

Loading the model

```
model = joblib.load("NewFlightPricePrediction.pkl")
```

```
prediction = model.predict(x_test)
```

```
prediction=pd.DataFrame(prediction)
#converted into data frame
```

```
prediction.to_csv('NewFlightPricePredictionResults.csv', index = False)
#prediction saving
```

Conclusion:

Hence, at the end, we were effectively able to train our regression model 'Gradient Boosting Regressor' to predict the flights of prices with an r2_score of 77%, and have achieved the required task successfully.

Learning Outcomes of the Study in respect of Data Science:

- Best model is Decision tree regression, R2 score and cross validation score is having small difference compare to other algorithms.