#### Flight Price Prediction



**Problem Statement:**

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travellers saying that flight ticket prices are so unpredictable. Here you will be provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

Size of training set: **10683** records

Size of test set: **2671** records

FEATURES:

**Airline**: The name of the airline.

**Date\_of\_Journey**: The date of the journey

**Source**: The source from which the service begins.

**Destination**: The destination where the service ends.

**Route**: The route taken by the flight to reach the destination.

**Dep\_Time**: The time when the journey starts from the source.

**Arrival\_Time**: Time of arrival at the destination.

**Duration**: Total duration of the flight.

**Total\_Stops**: Total stops between the source and destination.

**Additional\_Info**: Additional information about the flight

**Price**: The price of the ticket

When start the project first of all import the necessary libraries for visualizing reading the whole data some plotting graphs libraries are pandas,numpy,matplotlib,seaborn these are the packages used of datacleaning visualizing , EDA in a dataset.

**Importing the Libraries –**

Graphical user interface, text, application

Description automatically generated

Pandas is open-source library tool which provides high performance data analysis tool by its powerful data structures.

It helps to shorten the procedure of handling the data with extensive set of features.

NumPy is most used package for scientific computing for multi-dimensional array of objects.

Standard Scaler used here to standardize the values to 0 to 1 in order to equalize the range of values as a preprocessing step.

Train Test Split is used to split the train data and test data and we will train our data in the given dataset and use test data to predict the output.

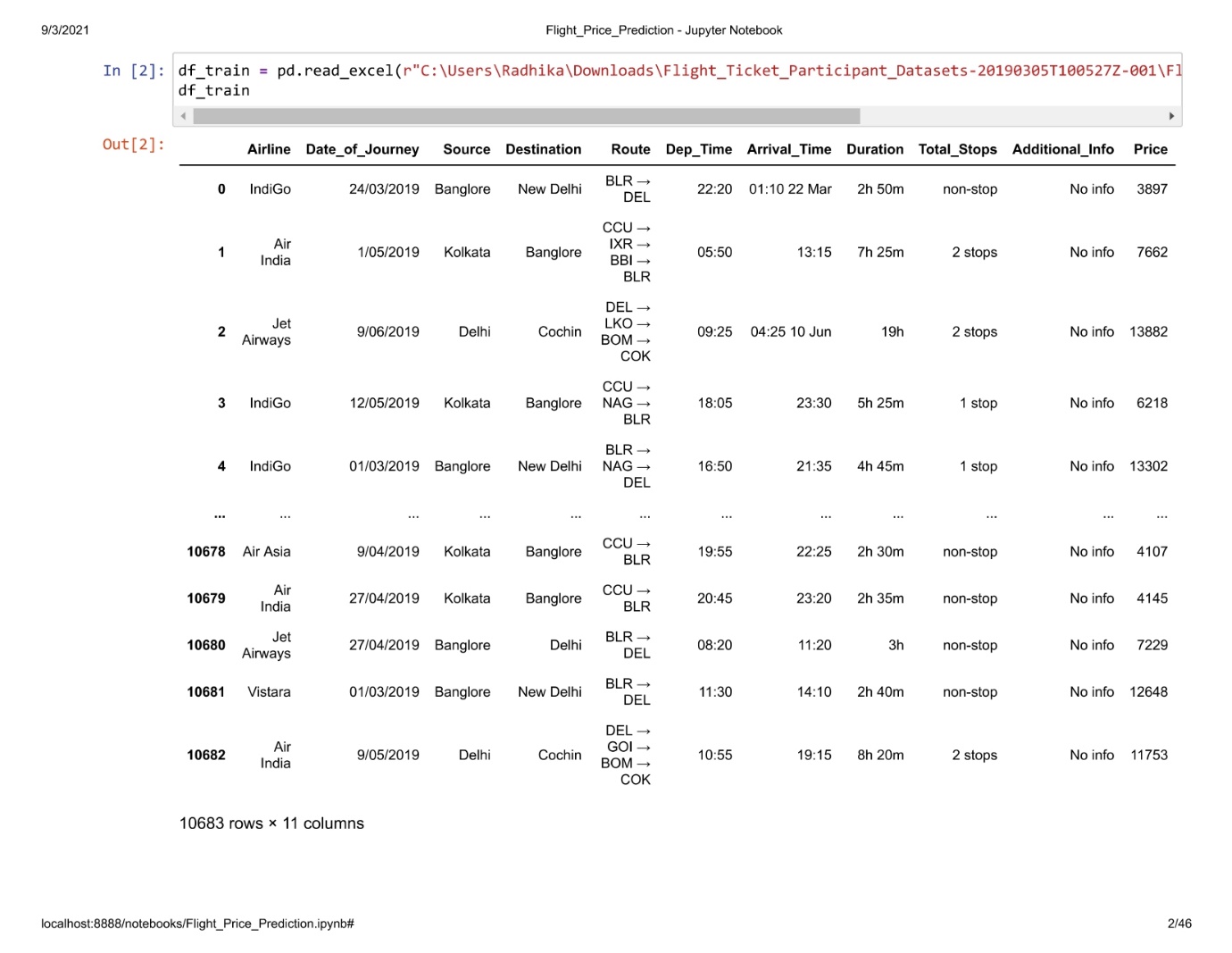
Grid Search CV is used for hyper parameter tuning to increase the model accuracy.

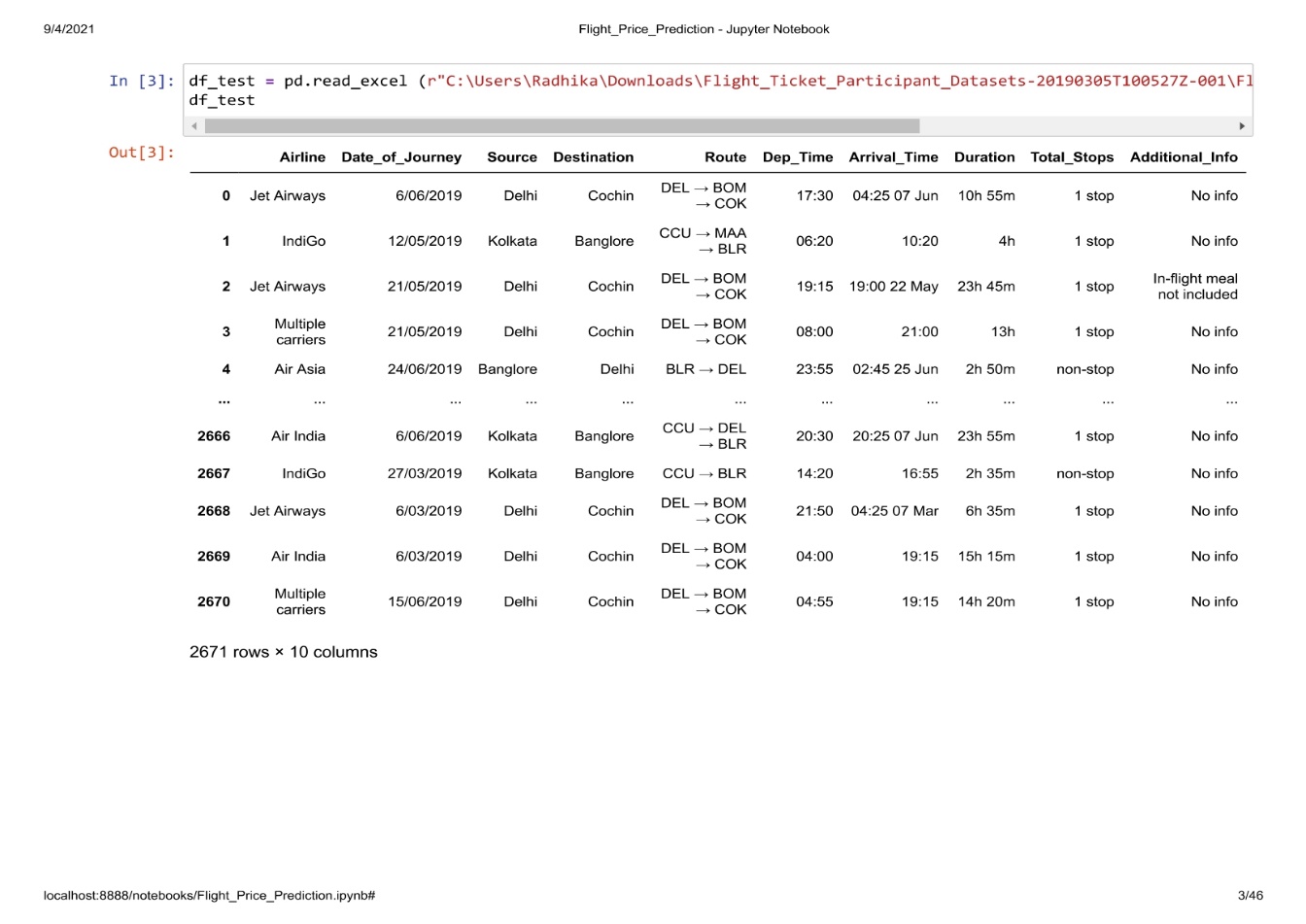
Cross Validation Score is used to check whether the model has been over fit or under fit.

Seaborn and Matplotlib is used here as a visualization library for the stunning plot to understand the data in a better way.

warning occurs when there is some obsolete of certain programming elements, such as keyword, function or class, etc

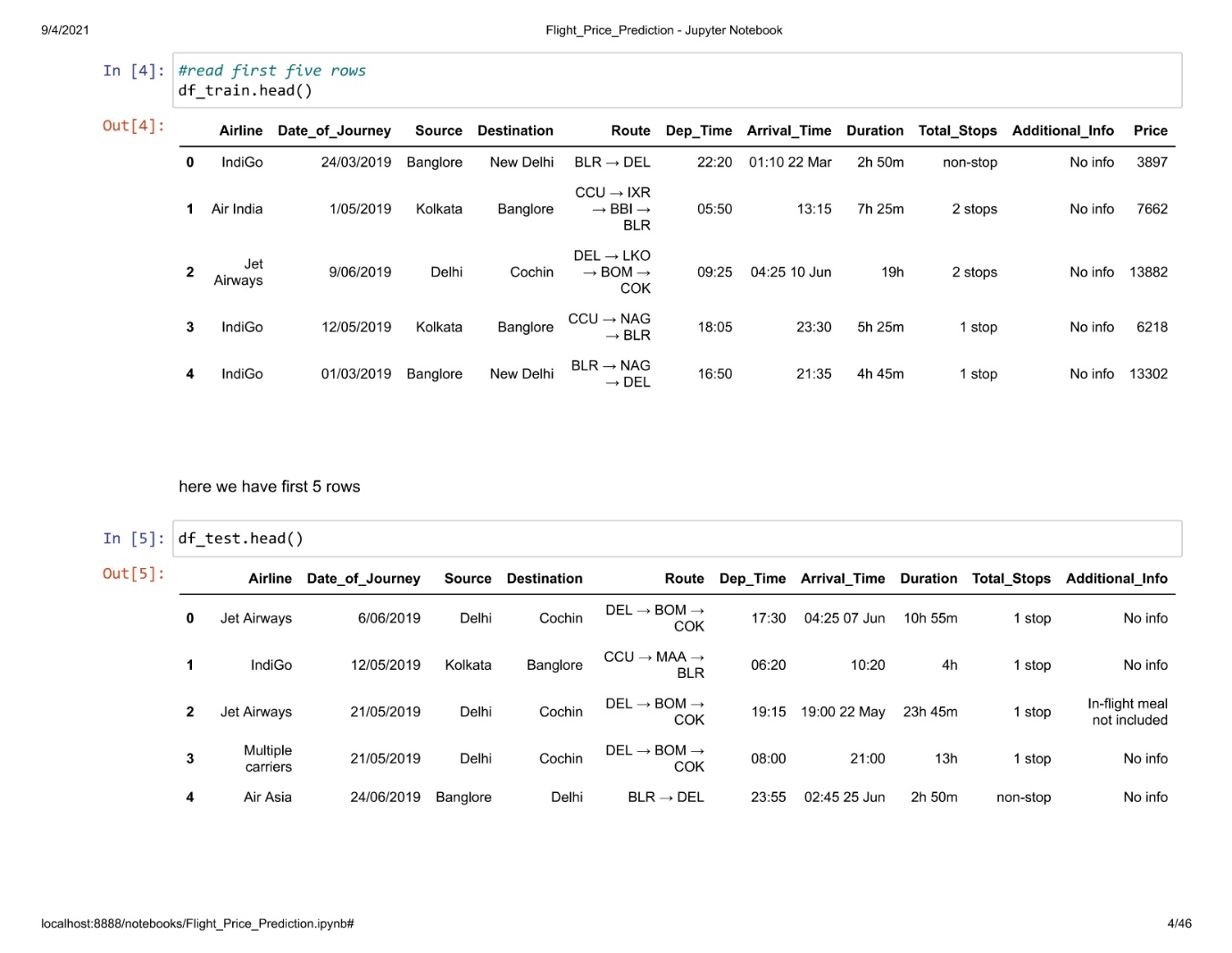
**Loading the Dataset –**



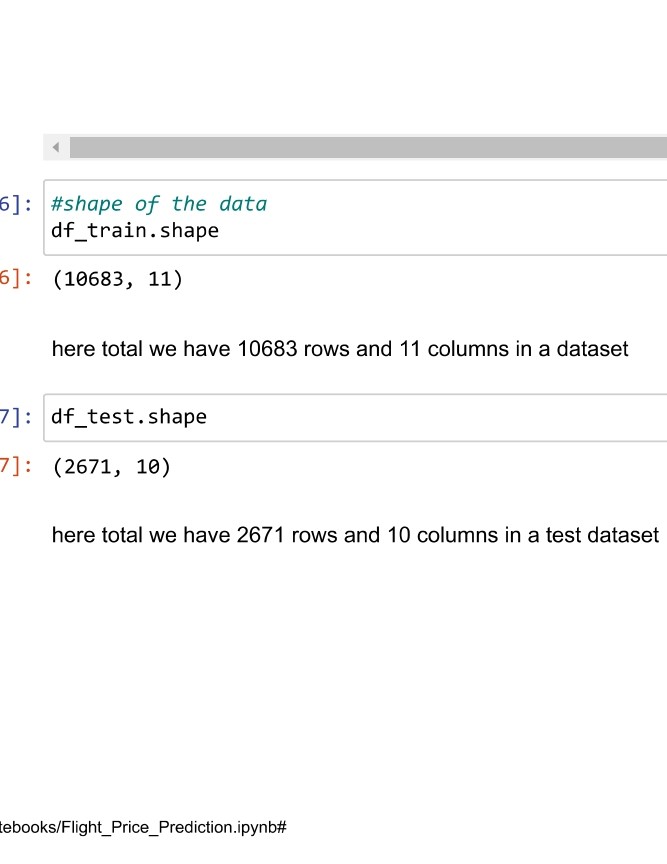


Generally it is a good idea to add both train and test data, perform feature engineering and then divide them later again. This saves time and complexity of performing same step twice once on train data and same on test data. So, Let’s combine them into a data frame ‘data’ with a ‘root’ column specifying where each observation belongs

Loading the dataset and head () method will display the first 5 data from the dataset whereas tail () method displays the last 5 data from the dataset.

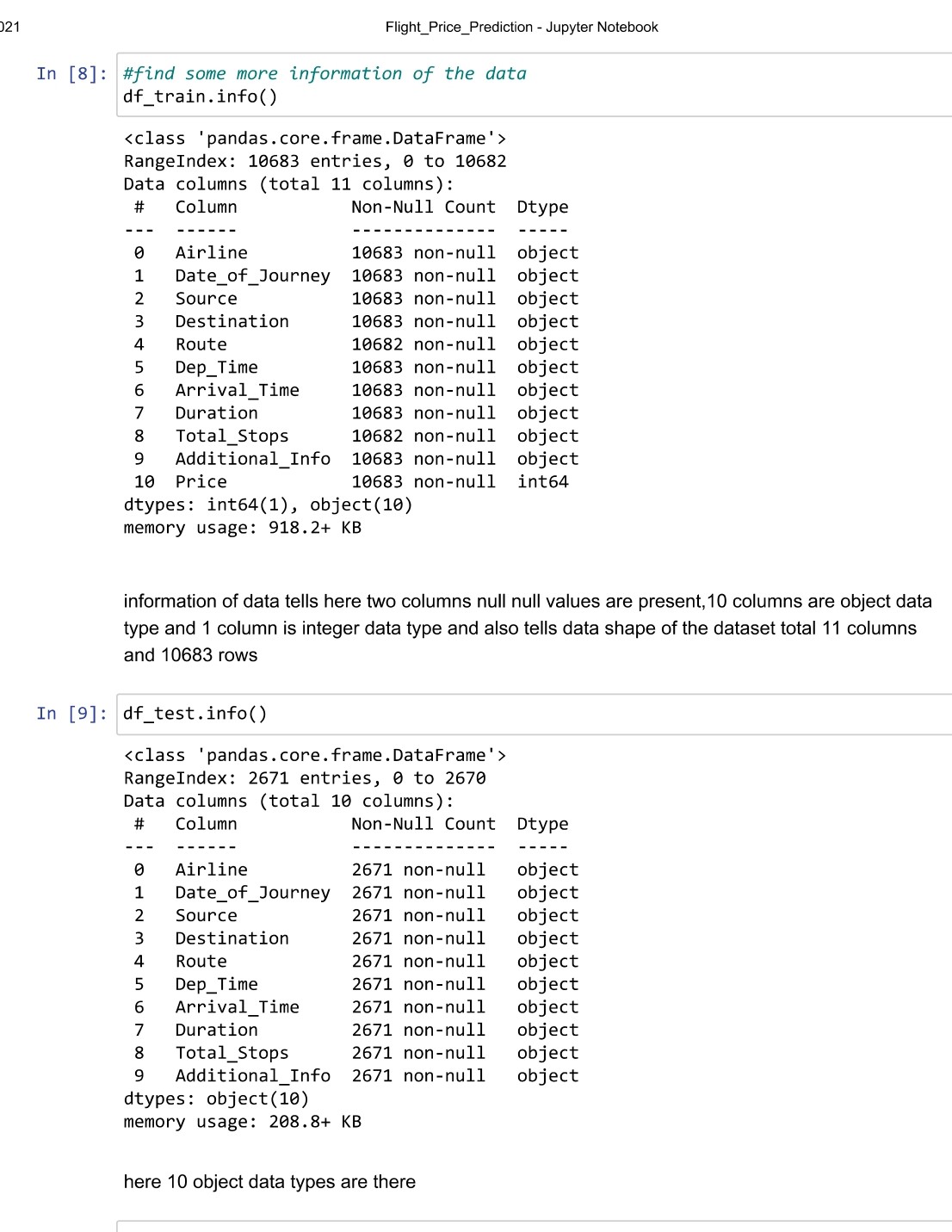


Lets check shape of the dataset



We can see that, training set data have 10683 rows and 11 columns and testing data have 2671 rows and 10 columns. Whereas adding data with both training and testing set has 13354 rows and 11 columns. Test data has 10 columns as it doesn't contain output variable 'Price'.

Checking datatypes with information both test and training data

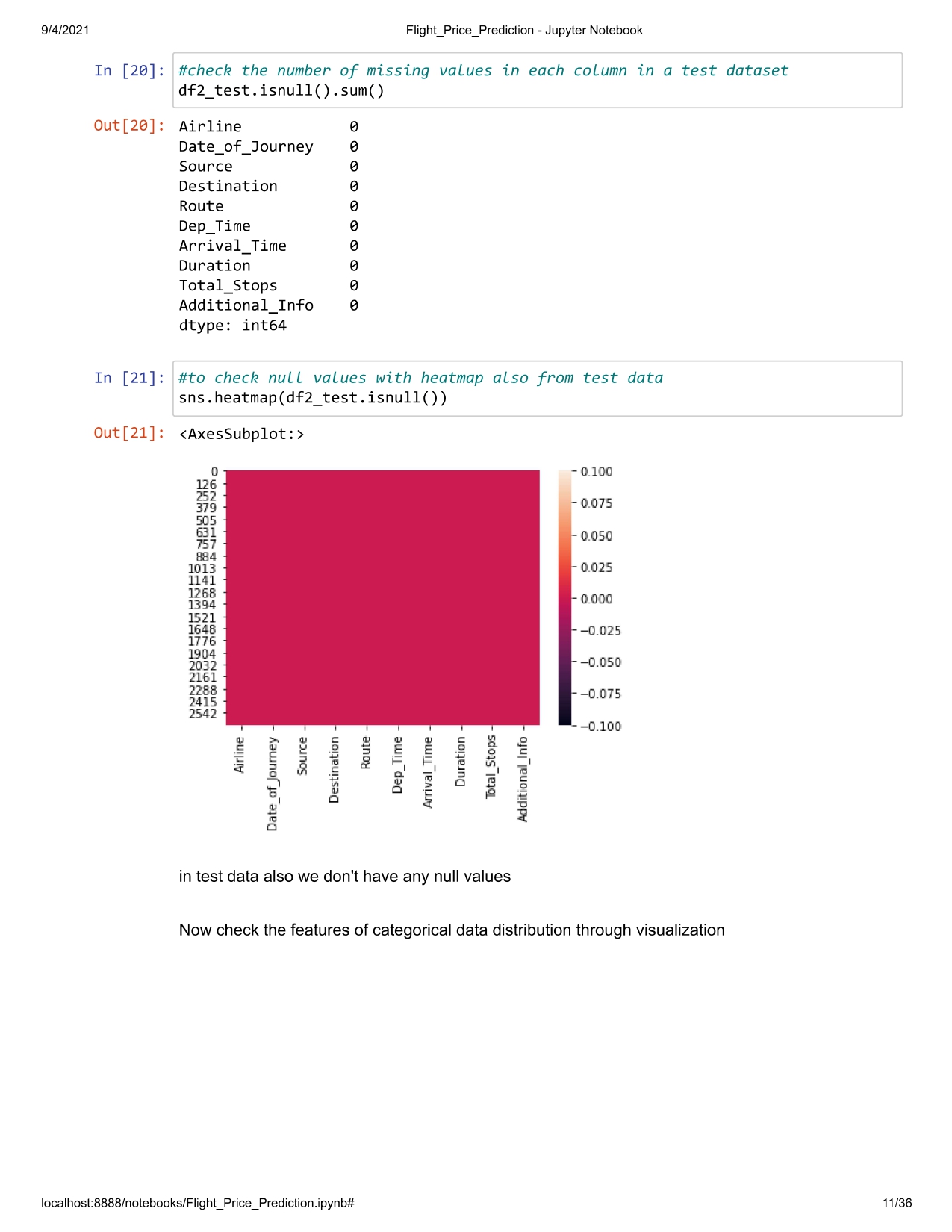


information of data tells here two columns null null values are present,10 columns are object data type and 1 column is integer data type and also tells data shape of the dataset total 11 columns and 10683 rows

10 object data types in test dataset dataset total 10 columns and 2671rows no null values present

Checking null values with isnull.com



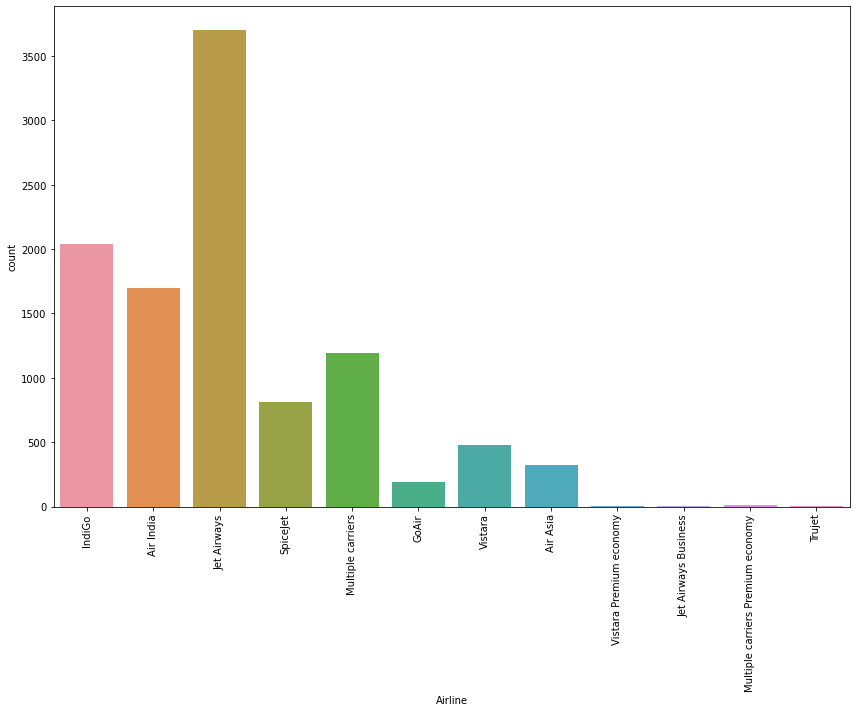


In the train dataset having null values I just drop the null values in the train dataset

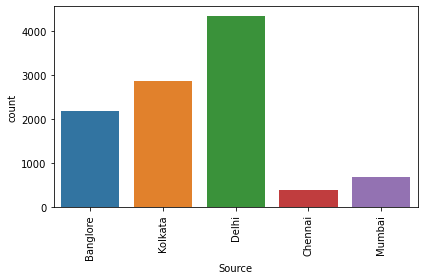
**EDA**

# **Lets do data analysis one by one**

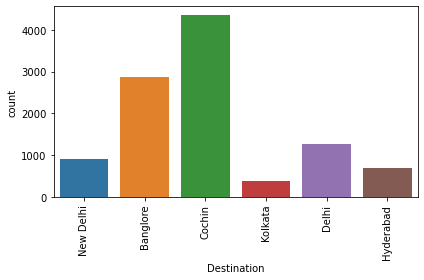
As all the features are categorical data,we will see how the data has been distributed through visualizations,



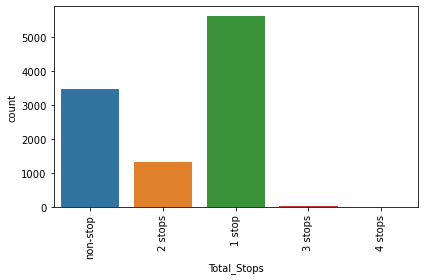
The above plot jet airways airline is more in flights



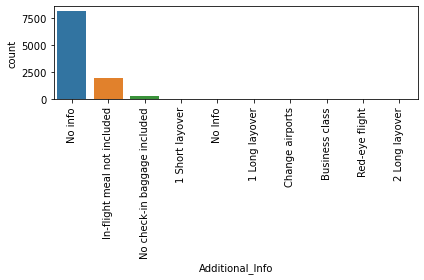
The above plot shows that flights are starts from banglore,kolkata,delhi,chennai and mumbai,in source are high in flight start from delhi and less flights which starts from chennai & mumbai.



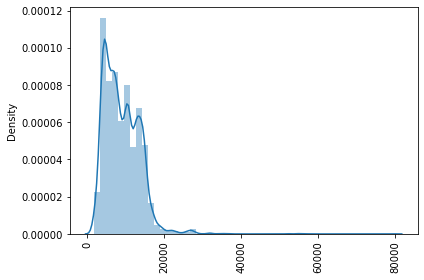
The above countplot shows that most flights has destination to COCHIN and least to KOLKATA.



the above plot, we can tell that most of the flights has atleast 1 stop between source and destination.



Additional\_Info will explains the information about flights and it has more values as no\_info which is no information.



Looking into the target variable PRICE, can see that data is skewed rightly.

* **EDA Concluding Remark:**

Jet Airways Business is the costliest of all airlines. But it can also be seen that, price for Jet airways Business is fixed in respective of parameters like weekends, holidays etc. Trujet has least price. But frequency of Trujet is only 1 for the year 2019. Rest all airlines have variable prices. Prices increase with decreasing days, weekdays, holidays etc.

Most of the flight take off from Delhi has the highest price. Mumbai & Chennai has least of all sources. Prices in Chennai are fixed. Prices vary very drastically from source Delhi and Bangalore. Cochin has destination for many flights with highest of all price. Kolkata and Hyderabad has least price of all destination cities. Prices in kolkata are fixed. Prices vary very drastically from source Delhi and Cochin.

Higher the no. of stops, higher is the flight price. 4 stop flight has higher ticket price of all. Non-Stop flights have least price of all. However 4 stop flights have fixed flight tickets. Whereas for others prices vary drastically.

Flight running in January month has highest price. Month of May, June, September, December have nearly equal price. March month has comparatively lower price. 3rd day of month has the highest price for year 2019. Rest all other days have more or less same amount of flight price.

**PreProcessing The Data**

Actually in dataset we don’t have any null values

We will go ahead and clean the data as we have all data in OBJECT data type and need to do some feature engineering as well.

1) Airline - It contains the name of fligts.

2) date of journey - Has day,month,year.

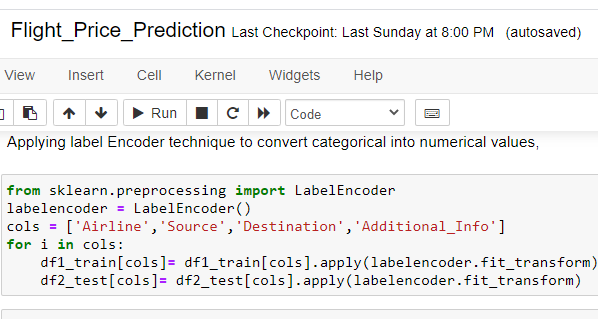
3) source, destination - It has city name on it and need to replace into numerical data.

4) Arrival\_Time, Dep\_Time - This both columns has hours and minutes and we need to split this into 2 columns.

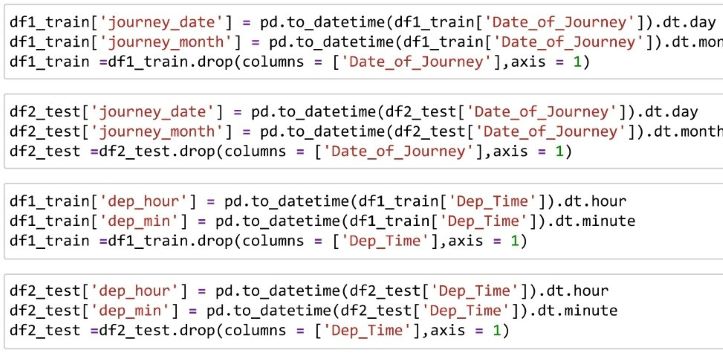
5) Route, No of Stops - These 2 columns are similar and explains the same thing.So we can drop one column.

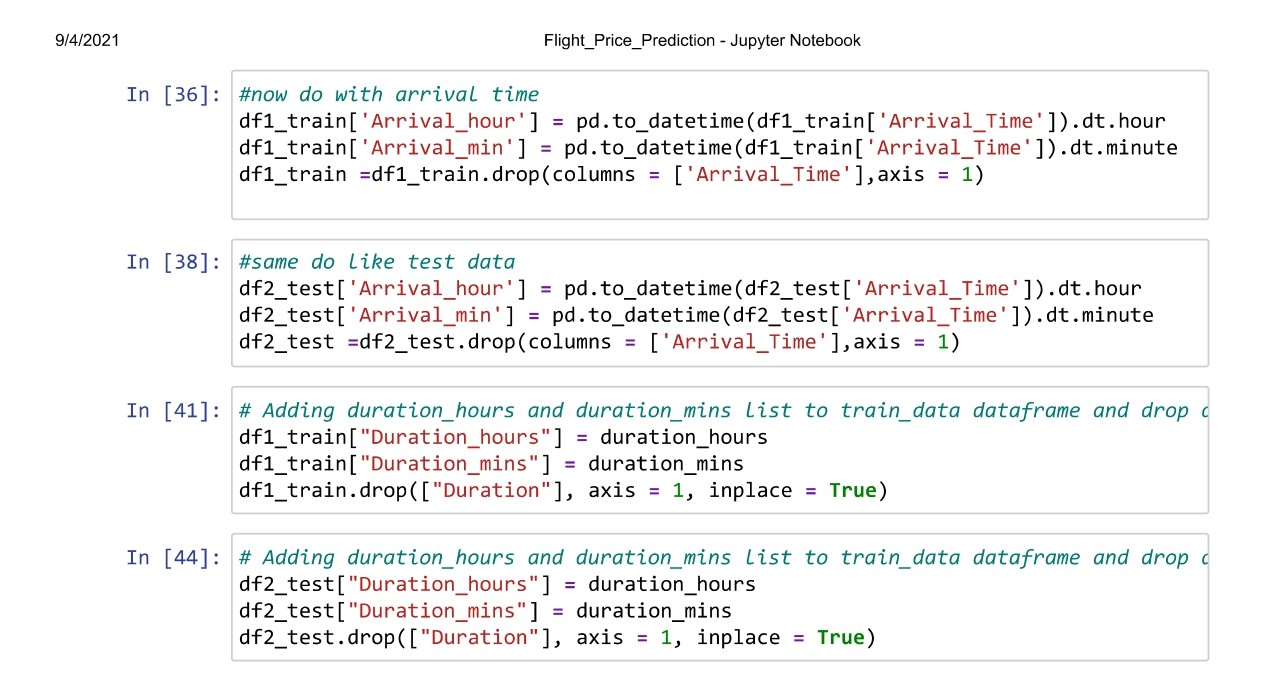
6) Price - It is our target variable and numerical data.

Applying label Encoder technique to convert categorical into numerical values,



Some of the columns like journey date,departure time,arrival time has date time as values,So using datetime index I am separating as day,month and time has hour and minute and dropping the original column from the dataset.





Information of the dataset after preprocessing the data

<class 'pandas.core.frame.DataFrame'>

Int64Index: 10682 entries, 0 to 10682

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Airline 10682 non-null int64

1 Source 10682 non-null int64

2 Destination 10682 non-null int64

3 Total\_Stops 10681 non-null float64

4 Additional\_Info 10682 non-null int64

5 Price 10682 non-null int64

6 journey\_date 10682 non-null int64

7 journey\_month 10682 non-null int64

8 dep\_hr 10682 non-null int64

9 dep\_min 10682 non-null int64

10 Arrival\_hr 10682 non-null int64

11 Arrival\_min 10682 non-null int64

12 duration\_hr 10682 non-null int32

13 duration\_min 10682 non-null int32

dtypes: float64(1), int32(2), int64(11)

memory usage: 1.1 MB

dataset has - 10682 rows and 14 columns

checking null values of the data

Airline 0

Source 0

Destination 0

Total\_Stops 1

Additional\_Info 0

Price 0

journey\_date 0

journey\_month 0

dep\_hr 0

dep\_min 0

Arrival\_hr 0

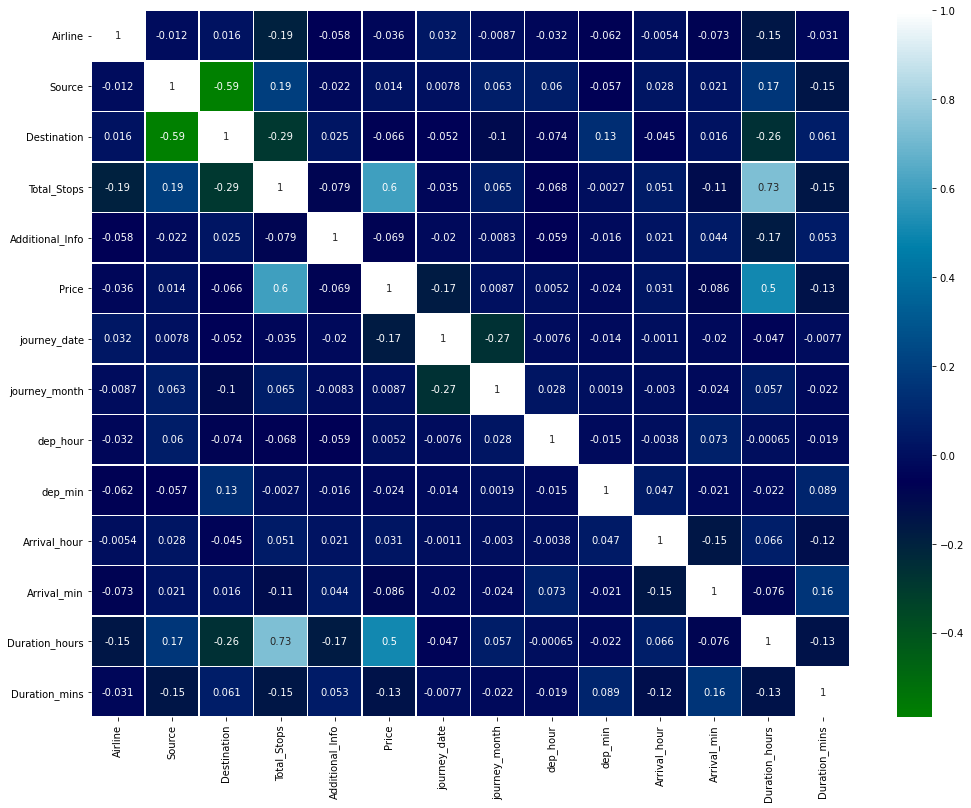
Arrival\_min 0

duration\_hr 0

duration\_min 0

dtype: int64

see dataset has 1 null values and filling that woth mode values.

Checking correlation of the datset using heatmap 

From the above correlation matrix,we can see there is no correlation between features vs features and features vs target.

**Spltting**

separate weght column and features column for model building

**Scaling**

Datset need to get scaled to get maximum accuracy , standard scaler technique was taken help to get out data set properly scaled before model building

**Model Building**

For train test split 75% data for training and remaining 25% for testing purpose

I tried four different models

KNeighborsRegressor : r2\_Score 0.7474191099426459

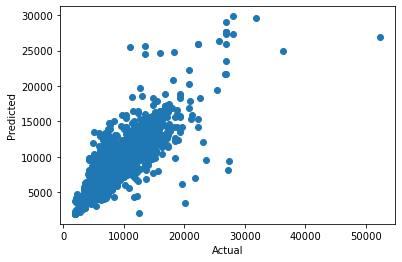
CV Score 0.6196034768374088

MSE 5171927.538251313

RMSE 2274.1872258570343

Train Score 0.8251607880139697

Test Score 0.7474191099426459



DecisionTreeRegressor : r2\_Score 0.8021554189760476

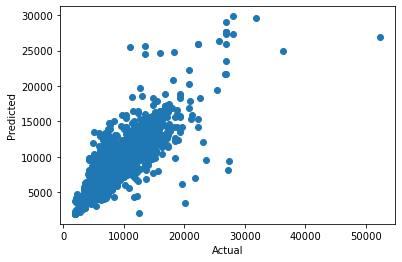
CV Score 0.7837912116485749

MSE 4051129.270544673

RMSE 2012.741729717122

Train Score 0.995926355613127

Test Score 0.8021554189760476



RandomForestRegressor : r2\_Score 0.8897426373033817

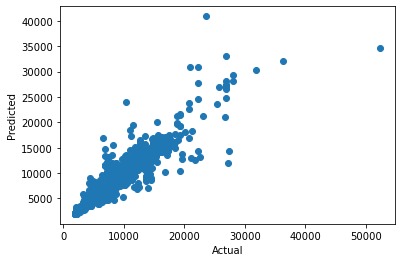
CV Score 0.8702889866913246

MSE 2257665.2188378833

RMSE 1502.552900512286

Train Score 0.9776411128901564

Test Score 0.8897426373033817



XGBRegressor : r2\_Score 0.9108168662081545

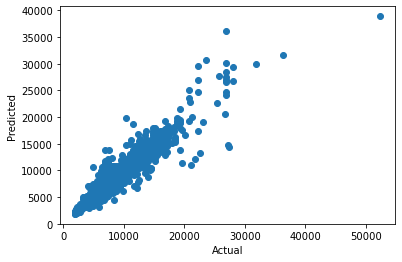
CV Score 0.8919184229293291

MSE 1826142.5300261653

RMSE 1351.3484117821597

Train Score 0.9735905485225135

Test Score 0.9108168662081545



we choosing random forest as best model as decision tree and random forest has same difference but looking upon the training score,Random Forest is considered as best model. Comparing the accuracy r2 score and cv score,the model with less difference is considered as best model and that model is random forest regressior

Using hyperparameter tuning on Random Forest Regressor further increased the accuracy.

# Model Accuracy

Random Forest Regressor :

r2\_Score: 0.8942177732449971

MAE: 707.7730385070416

MSE: 2166030.896033005

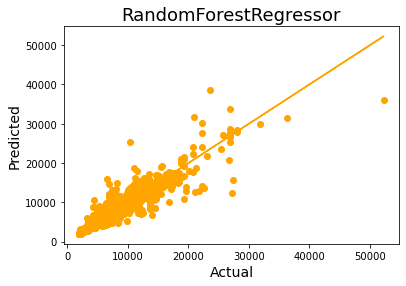
RMSE: 1471.74416799694

r2\_Score: 0.8942177732449971

Train Score: 0.9766737156040055

Test Score: 0.8942177732449971

after hyperparameter tuning increased one precent of accuracy,model score accuracy is 90% it is good score.



We can see that model is linear and distance between residuals and best fit line is less and that proves that model is good,even though we can see some points far from the best fit line but very few.

* **Concluding Remarks:**

We started with the data exploration where we checked on information about the data set, its data types, shape of data, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data pre processing part, we computed missing values with mean/median/mode of the data by checking distribution plot, converted features into numeric ones using encoding, grouped values into categories.

Next we trained 4 different machine learning models which included Decision Tree regressor, random Forest regressor, Gradient Boost regressor and KNeighborsRegressor checked for its accuracy score/r2\_score and mean square error and picked one of them (random forest) and applied cross validation on it to fix problem of under fitting/ over-fitting with high performance

Of course there is still room for improvement, like doing a more extensive feature engineering, by comparing and plotting the features against each other and identifying and removing the noisy features. Another thing that can improve the overall result would be a more extensive hyper-parameter tuning on several machine learning models.