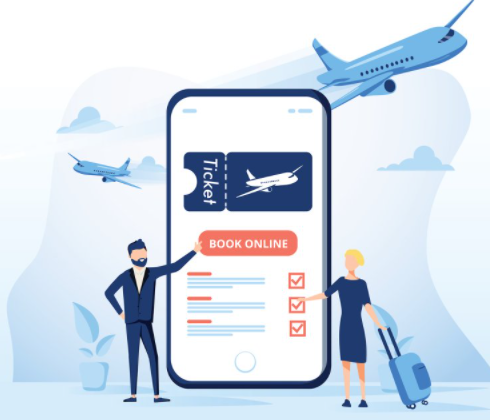
**Flight Price Prediction**



**Author:**

**Rupali Bisen**

**Blog on Flight Price Prediction**

**Problem Definition:**

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 traveller’s 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

**Train Dataset Description:**

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

**Test Dataset Description:**

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.

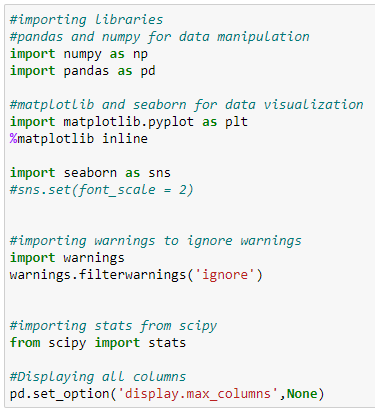
Additional\_Info : Additional information about the flight

The principle objective is to build a model that can predict prices of flight tickets on the basis of the details provided in the data-set.

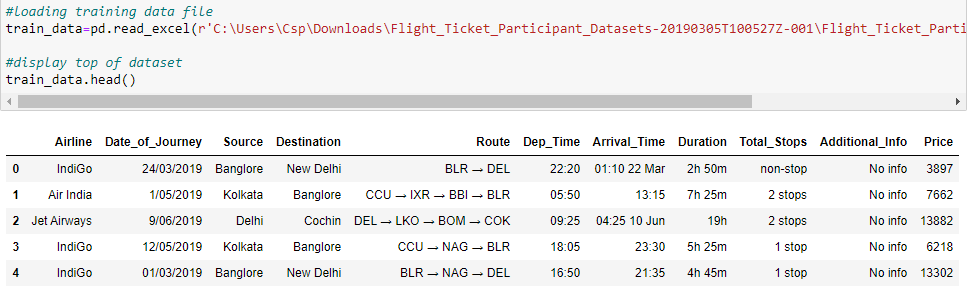
In this dataset two different datasets are given: train and test. Training dataset has dependent variable i.e. Price.

**Data Analysis:**

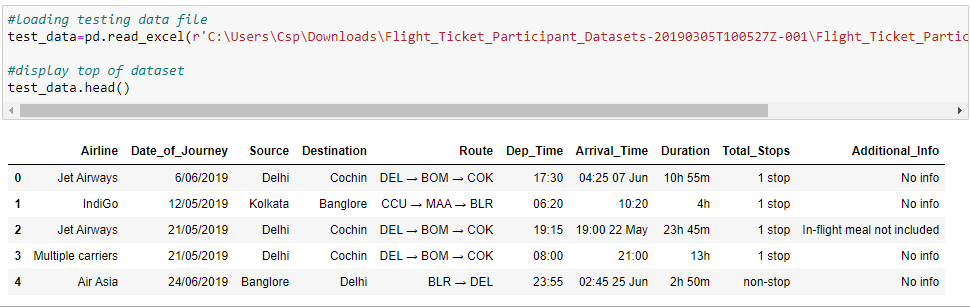
Firstly, import all libraries and load both train and test data-set into jupyter notebook using pandas.read\_csv function.



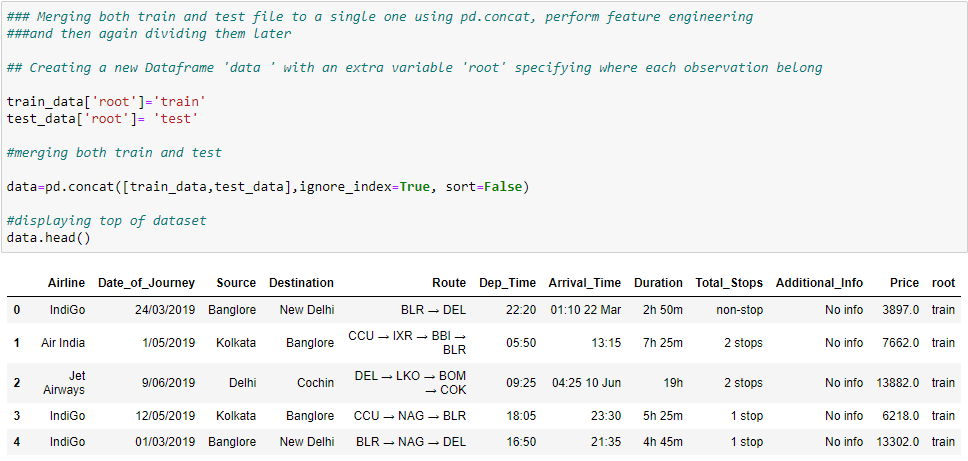
Loading train dataset:

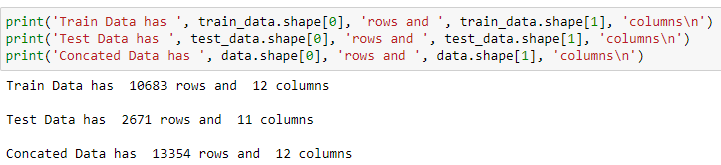


Loading test dataset:



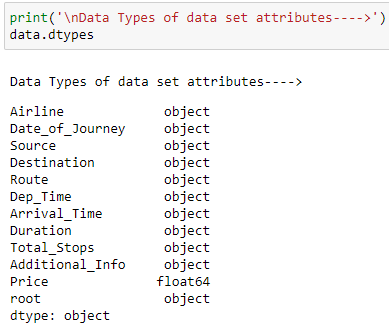
Generally it is a good idea to concat both train and test data, perform feature engineering and then divide them later again. This saves time and complexity of performing same step twice once on train data and same on test data. So, Lets combine them into a data frame ‘data’ with a ‘root’ column specifying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



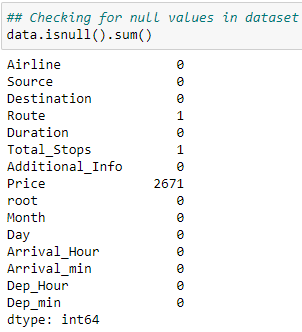


We can see that, training set data have 10683 rows and 12 columns and testing data have 2671 rows and 11 columns.Whereas, concated data with both training and testing set has 13354 rows and 12 columns. Test data has 11 columns as it doesn't contain output variable 'Price'.

On checking for data type of concated dataset, we observed that All the attributes has data types as object except for Price.However, Date cannot be object type. Attribute 'Date\_of\_Journey','Dep\_Time','Arrival\_Time' are object type which is a problem over here, as date can not be object type.

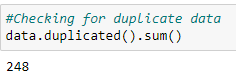


On Checking for null values/ missing values in the data-set it was observed that data set has many null values .



Attributes: route, Total\_stops and Price has null values. Hence dealing with null values in data pre processing section.

Next, we will check for duplicates in dataset.



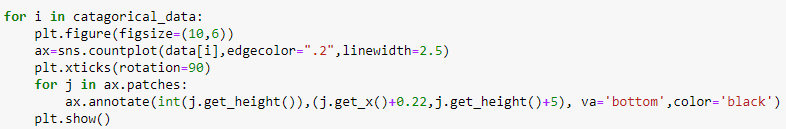
On checking for duplicate data, it is observed that dataset has 248 duplicated rows which needs to be removed in data pre processing.

In attribute Destination we have Delhi and New Delhi , However destination is same.Hence replacing new delhi to delhi. Also, Additional info has mis Interpreted No info as No Info. Hence replacing No Info to No info

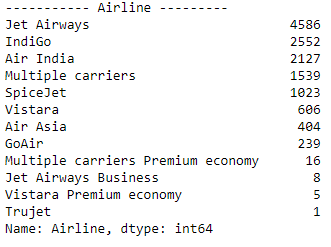
#### **Exploratory data analysis for Nominal/categorical type of data:**

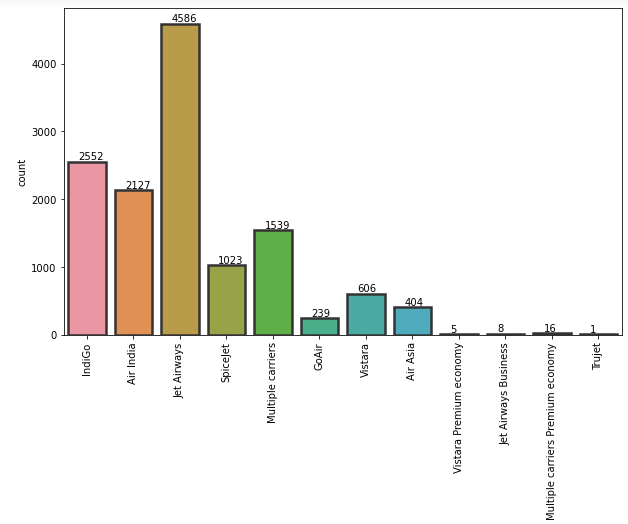
**Univariate Analysis:**

categorical\_data=['Airline','Source','Destination','Total\_Stops','Additional\_Info','Month','Day']

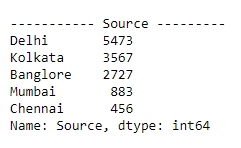


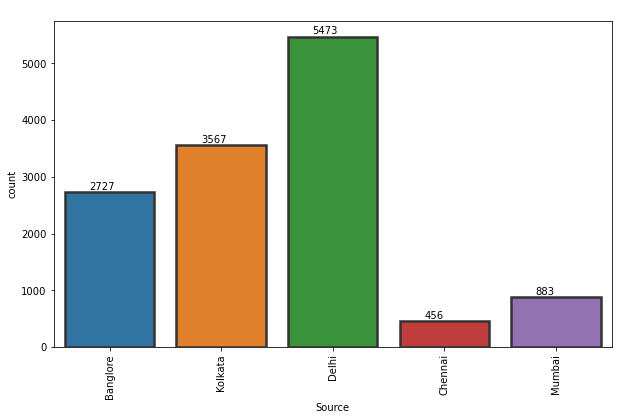
1. **Airline:**



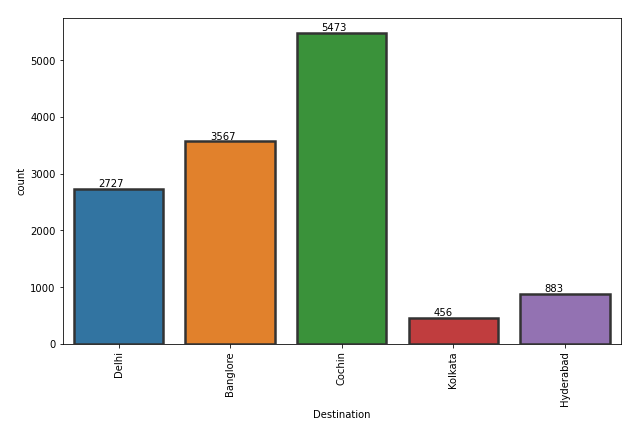


1. **Source:**

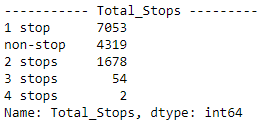


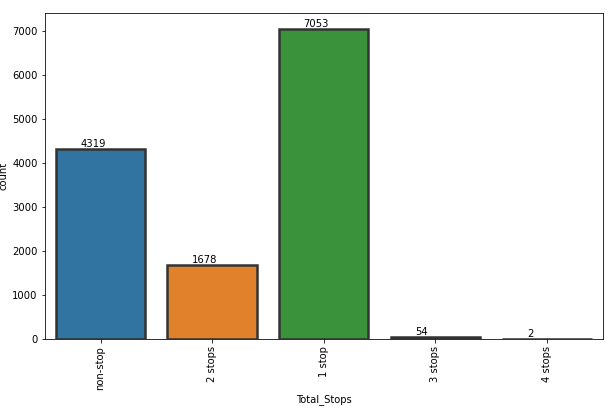


1. **Destination:**

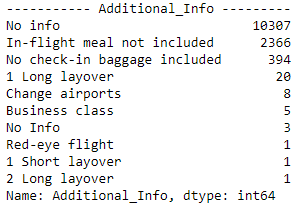


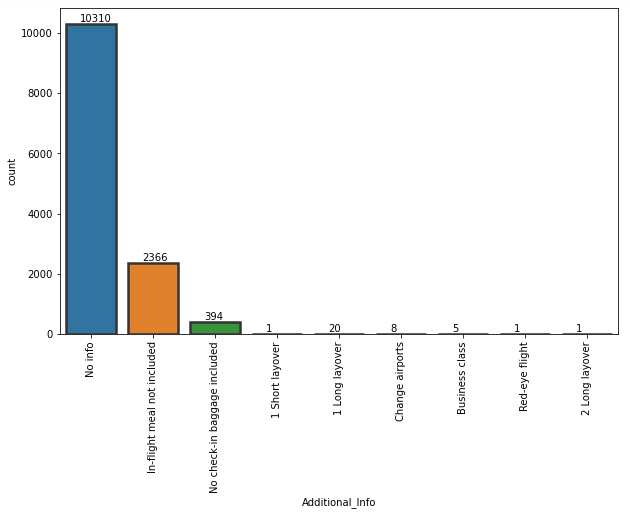
1. **Total\_Stops:**



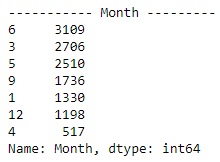


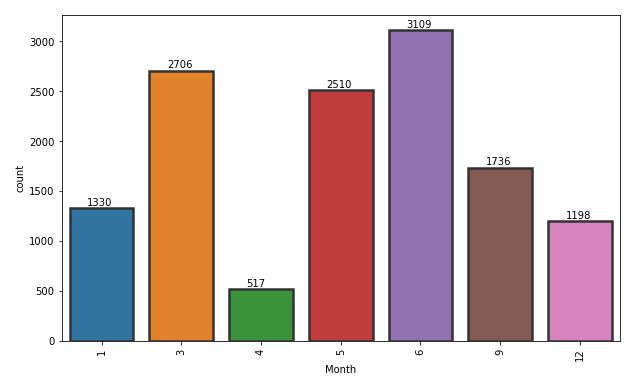
1. **Additional\_Info:**



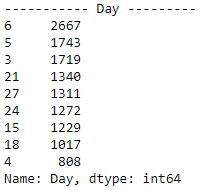


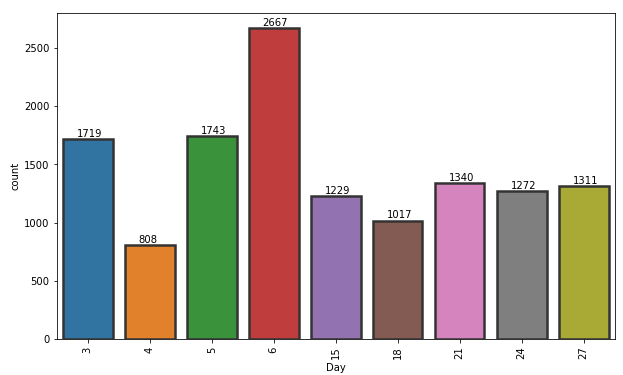
1. **Month:**





1. **Day:**





Observations for count plot::

1. Most of flights take off on 6th day of month.

2. Most of the flights take off on in the month of June, March and May.

3. Almost 79% of flight have no additional info.

4. Most of the flights running have 1 stop(54%), non-stop-33%, 2-stops-13%, 3-stops-0.4%.

5. Around 42% flights running have destination as Cochin, 27%-Bangalore, 21%-Delhi, 7%-Hyderabad & 3%-Kolkata.

6. Around 42% flights running have source as Delhi, 27%-Kolkata, 21%-Bangalore, 7%-Mumbai & 3%-Hyderabad.

7. Most of flights running are of Jet Airways Airline(35%). Only one flight is of Trujet Airline. Indigo has 19% flights.

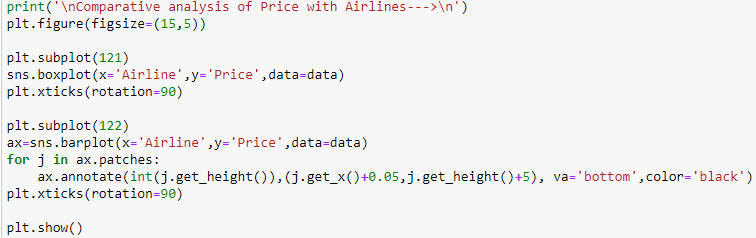
8. Air India have around 16% flights running, Multiples carriers with 12%, Spice-jet have 8%, Vistara have 5%.

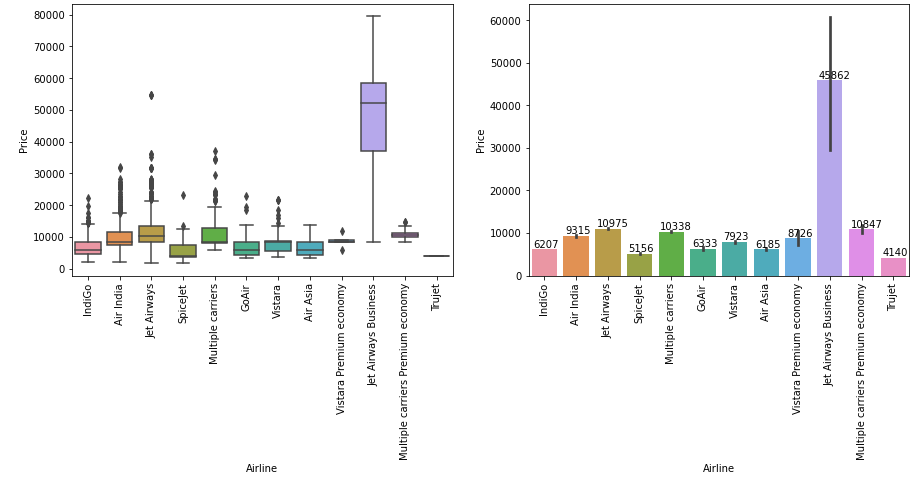
9. Air Asia have 3% flights running. Go air has only 2% flights running.

10.Multiple carriers Premium economy, Jet Airways Business, Vistara Premium economy together have 0.2% flights running of all airlines.

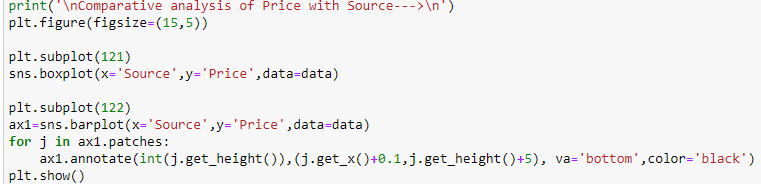
**Bivariate Analysis:**

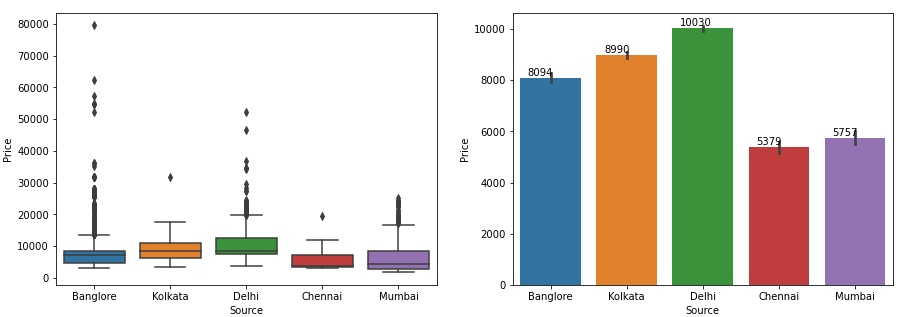
1. **Comparative analysis of Price with Airlines:**



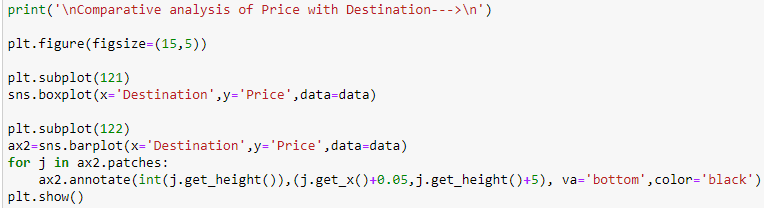


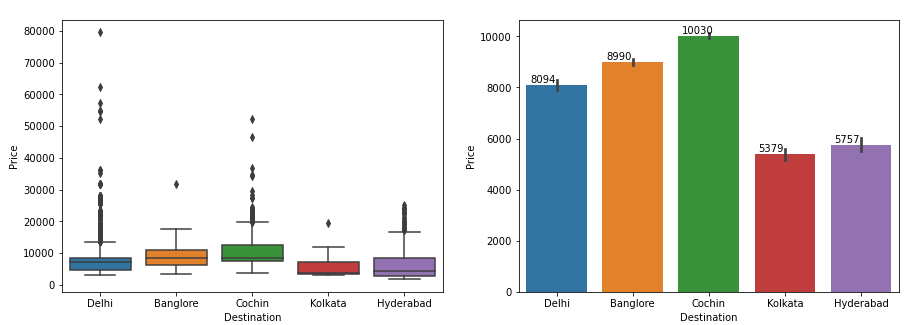
1. **Comparative analysis of Price with Source**



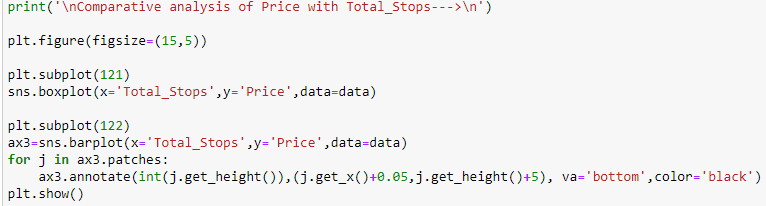


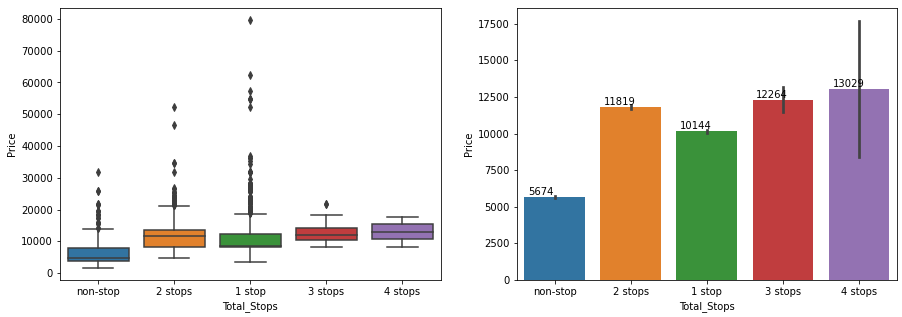
1. **Comparative analysis of Price with Source**





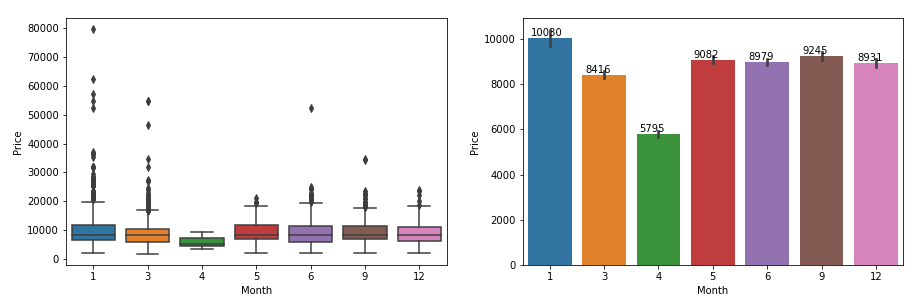
1. **Comparative analysis of Price with Source**

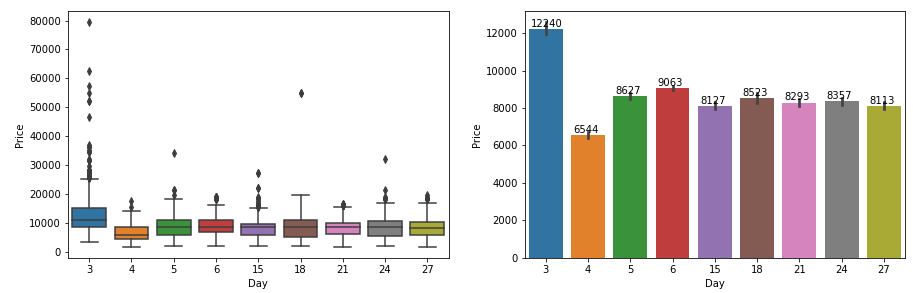




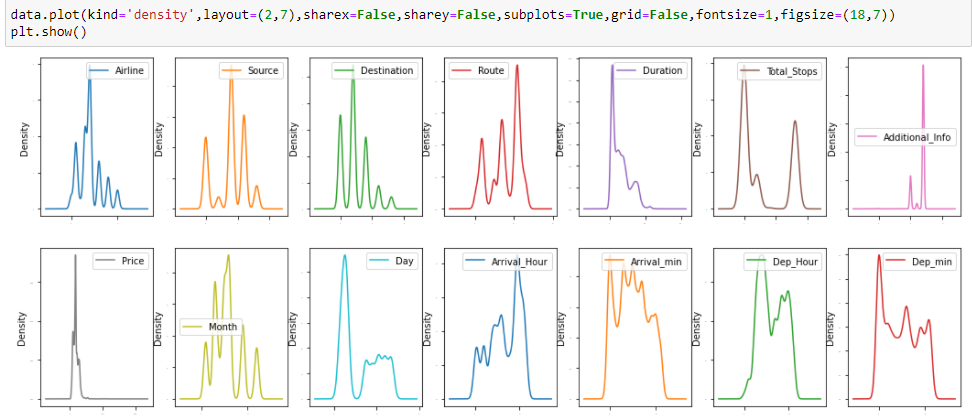
1. **Comparative analysis of Price with date and Month**







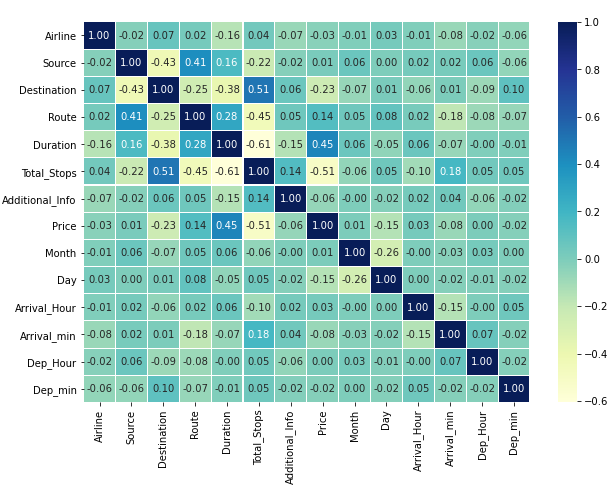
**Density visualization for all attributes**



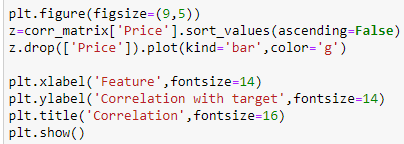
Above density plot shows skewness for numerical attribute Duration and Density which needs to be removed as it affects performance of the model.

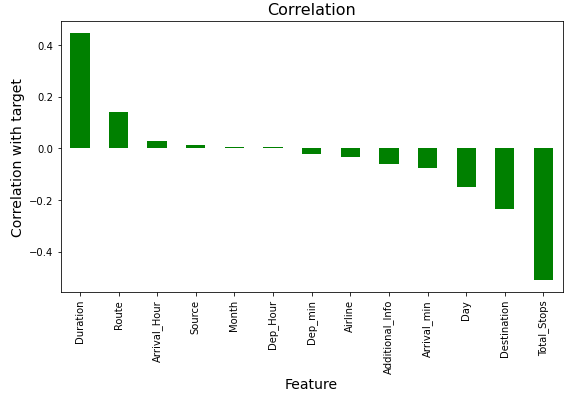
**Checking for correlation of output variable with other attributes:**





Sorting attributes based on correlation with target variable. Sorting values gives more clear idea in a simplified manner on how independent variables correlate with dependent variable. Also visualization of the same makes the picture very crystal clear.





From correlation plot it is observed that, dataset has very less(near to zero) correlation with attribute "Month" & "Dep\_Hour".

dataset has very high negative correlation with Total\_stops and high positive correlation with Duration.

**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.

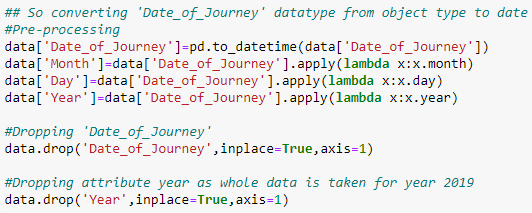
Higher the no. of stops, higher is the flight price. 4 stop flight has higher ticket price of all. Non-Stop flights have least price of all. However 4 stop flights have fixed flight tickets. Whereas for others prices vary drastically.

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.

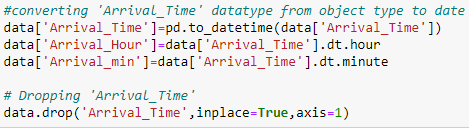
**Pre-Processing Pipeline:**

On checking for data type of dataset, All the attributes has data types as object except for Price. However, Date cannot be object type. Attribute 'Date\_of\_Journey','Dep\_Time','Arrival\_Time' are object type which is a problem over here and we need to convert it to date-time stamp.

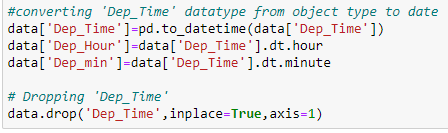
So, firstly converting 'Date\_of\_Journey' from object type to datetime. And performing feature engineering on the same by creating new attributes of month, day and year. And finally deleting base attribute date\_of\_journey.



Repeating the same procedure for Arrival\_time and dep\_time. Creating two new attributes from arrival\_time named Arrival\_hour and Arrival\_min and deleting the former one.



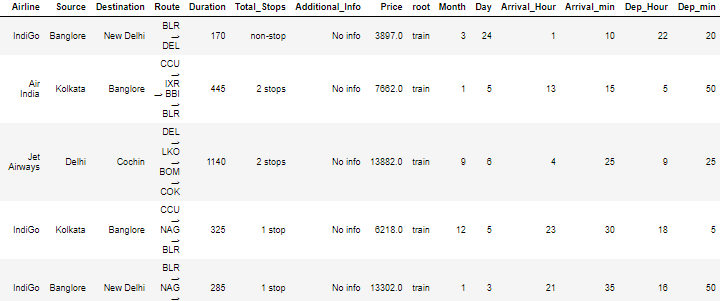
Performing feature engineering on attribute Dep\_Time by creating two new attributes from the same named Dep\_Hour and Dep\_Min. And finally deleting the former one.



Next, Attribute duration is also problem over here as it is object type due to characters in between. Hence converting duration into minutes form by replacing ‘h’ with ‘\*60’ , replacing ‘ ‘ by ‘+’ and ‘m’ by ‘\*1’ and applying eval function to the same. This converts string to numeric form.



Displaying head of dataset after pre processing of date type variables:



On checking for null values, Attributes: route, Total\_stops and Price has null values. Hence dealing with null values first.

we need to fill these null values as it will effect model performance.

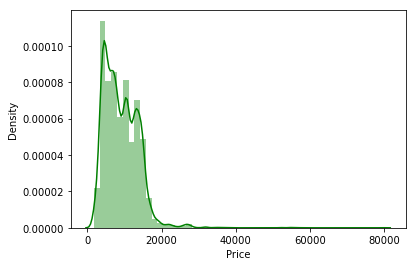
1. Route is categorical type of data. Hence replacing null values with it's mode.

2. Total\_stops is categorical type of data. Hence replacing null values with it's mode.

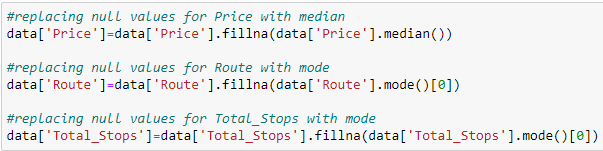
3. Price is numerical type of data. Hence replacing null values with it's median/mean.

In-order to replace null values in attribute Price, we need to first check for distribution plot of attribute 'Price'.

Checking for distribution of attribute Price

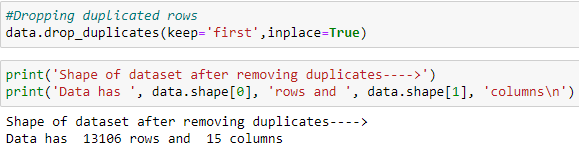


Data for attribute Price is not normally distributed. Hence, replacing null values with median of data.



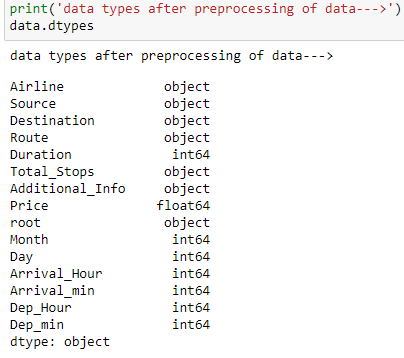
All the null values are handled.

Dataset has 248 duplicated rows which also need to be removed. Hence dropping duplicated rows and checking for shape of data.



After deleting duplicated rows, new shape of dataset is 13106 rows and 15 columns.

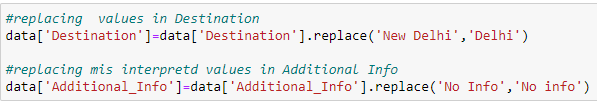
Checking for data types after pre processing of data



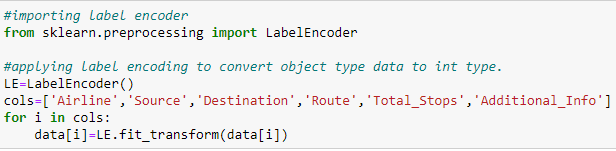
In attribute Destination we have Delhi and New Delhi , However destination is same.

Hence replacing New Delhi to Delhi.

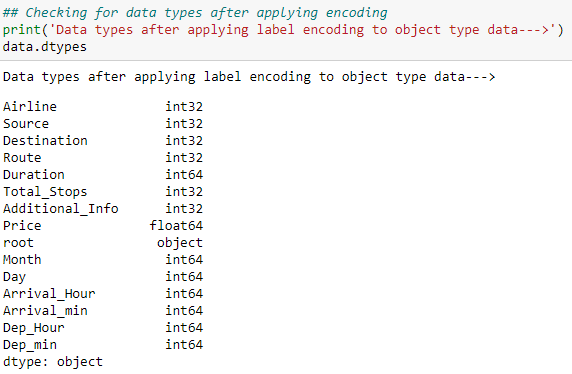
Also, Additional info has mis Interpreted No info as No Info. Hence replacing No Info to No info



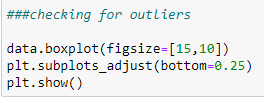
Next step is encoding categorical(object type data type) to numeric form before fitting and evaluating the model. Encoding is a required Pre-Processing step when working with categorical data for machine learning algorithms. Machine learning models require all input and output variables to be numeric. Using label encoding, each unique categorical value will be assigned a integer value. Every object type data is converted into int type value.

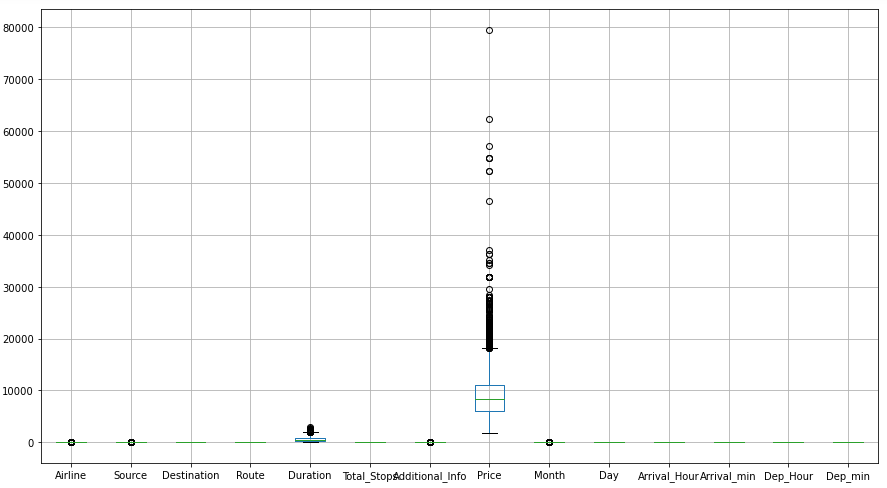


Checking for data types after applying encoding

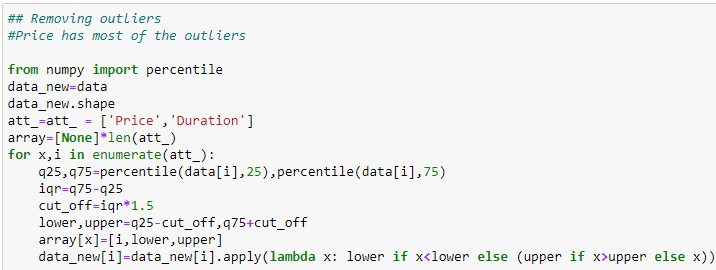


Checking for outliers and removing the same

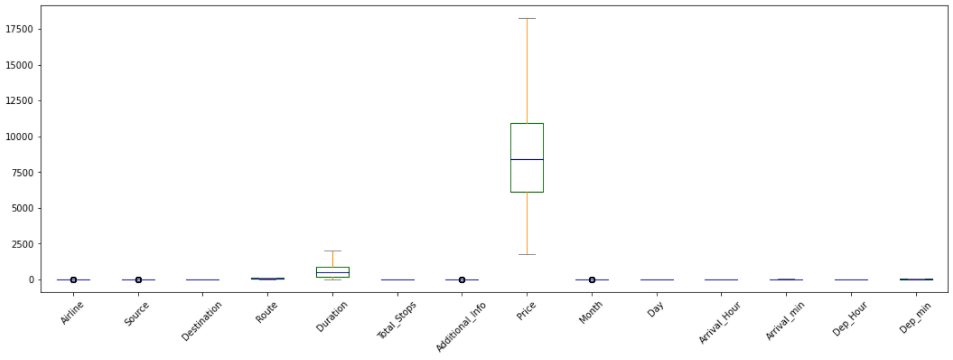




Removing outlier as attribute Price has most of the outliers.



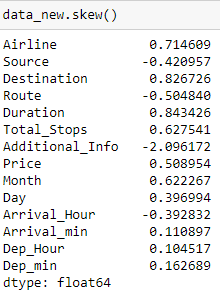
Checking for Outliers after removing outliers.



We can see that all the outliers are removed in attribute price.

From above density plot, skewness was observed in most of the attributes.

Hence checking for skewness and removing the same.

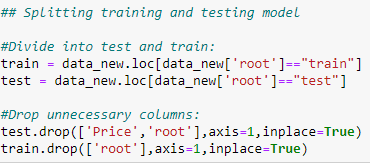


After removing outliers, skewness is also reduced for all numerical data.

Hence , Skewness and Outliers removed successfully

**Building Machine Learning Models:**

Formerly we have concated train and test set into one dataset ‘data’ using an extra attribute ‘root’ to identify where each observations belong. Now dividing dataset back into train and test dataset and dropping uncessary columns Price and root form test set and root from train set.

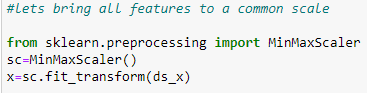


Splitting train dataset into independent and dependent variable.

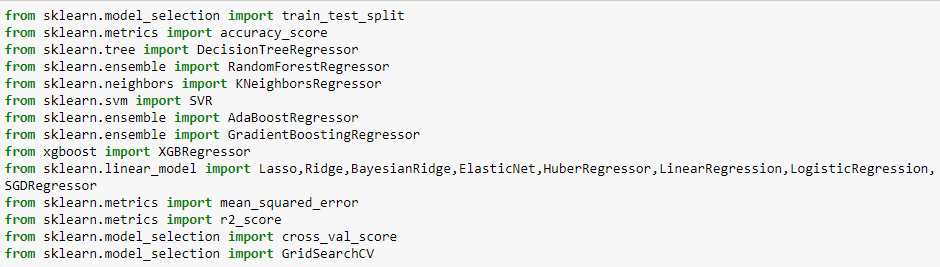


Using Min Max scaler to Transform features by scaling each feature to a given range.

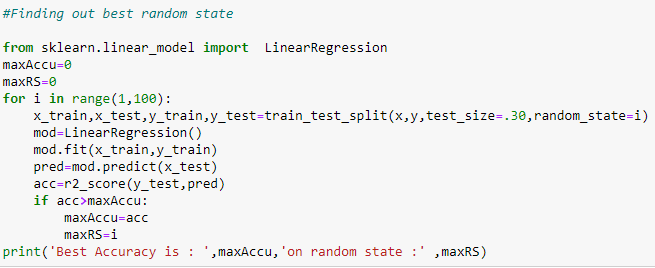
This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.



Importing libraries:

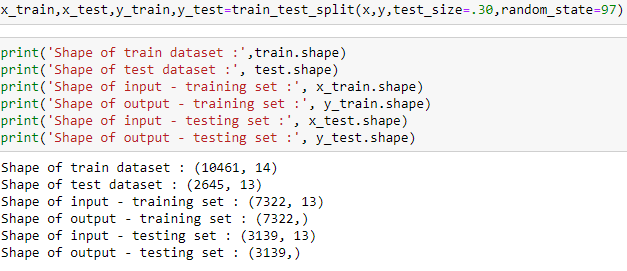


In order to find out best random state we apply for loop on Decision Tree Classifier Model by splitting 30% of data for test case. Firstly we fit the train data into the model and then predict for test data. On checking for best random state we get 97 as best random state.

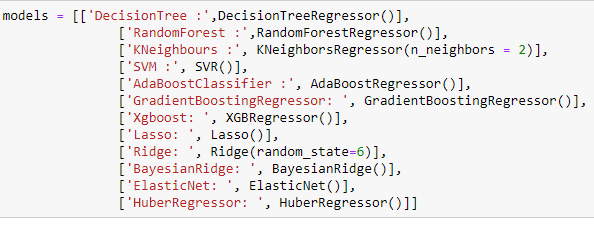


Assigning 70% training and 30% testing test using train\_test\_split.

Next we will check for shape of actual dataset and input-output training and testing set.

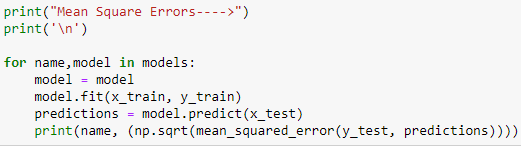


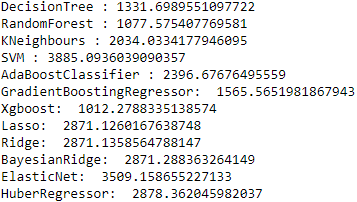
Defining machine learning models:



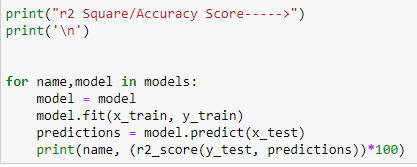
Since this is a regression type of problem hence we will find mean square errors and r2 score of the defined models.

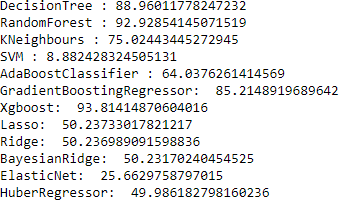
Checking for Mean square errors of defined models:





Checking for r2\_score of defined models:

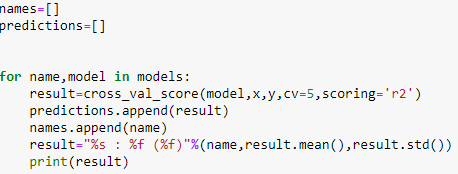


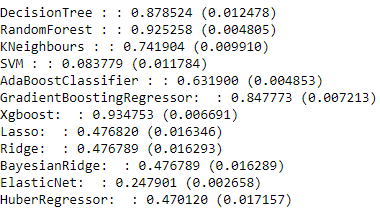


Among-st all the models, XGBoost regressor has the highest r2\_score and least mean square erroe.

But first, let us check, how models performs, when we use cross validation. Cross validation on model eliminates problem of over fitting / under fitting of the data in data set.

Checking for cross validation score of each model:



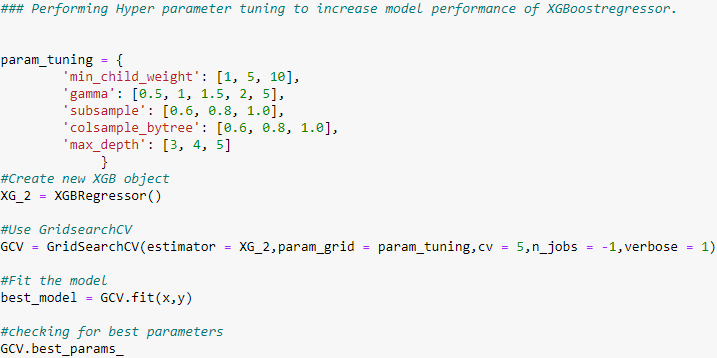


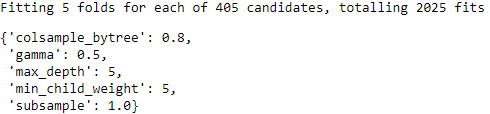
From above predictive analysis on defined models using MSE, r2\_score and CV\_scores XGBoost has the highest accuracy with 93.4% and standard deviation of 0.7%.

This means in our case that the accuracy of our model can differ + — 0.7%.

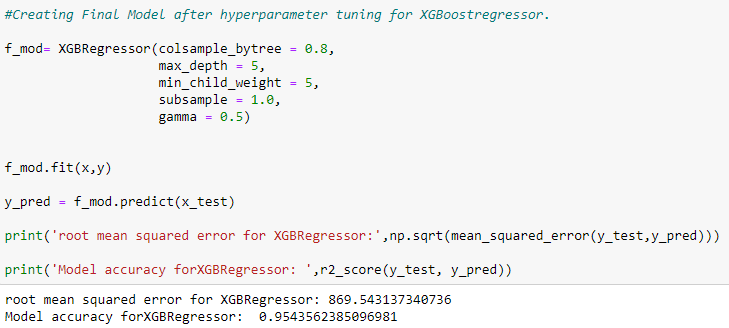
In order to get better accuracy and to improve performance of model hyper parameter tuning is done.

**Hyper parameters** are crucial as they control the overall behaviour of a machine learning model. The ultimate goal is to find an optimal combination of hyper-parameters that minimizes a predefined loss function to give better results. **Tuning** is the process of maximizing a model's performance without over-fitting or creating too high of a variance.





Testing new parameters:



Model was underfitting which was resolved using hyper parameter tuning.

### **After hyper parameter tuning, model accuracy is 95.43%**

**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 12 different machine learning models which included Decision Tree regressor, random Forest regressor, support vector machine, Ada Boost regressor, Gradient Boost regressor and XG Boost regressor, Lasso, Ridge, Bayesian Ridge, ElasticNet, HuberRegressor checked for its accuracy score/r2\_score and mean square error and picked one of them (XGBoost ) and applied cross validation on it to fix problem of under fitting/ over-fitting. After that we tuned it’s performance through optimizing it’s hyper-parameter values.After hyper parameter tuning, model accuracy with high performance is 95.43%.

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.