**Flight Price Prediction – Machine learning Project**

**With 17years of working in the airline industry and moving into the field of data science, this is a project where I can use my domain knowledge and combine it with my newly acquired skill of machine learning. Let’s how it goes.**

1. **Problem definition**

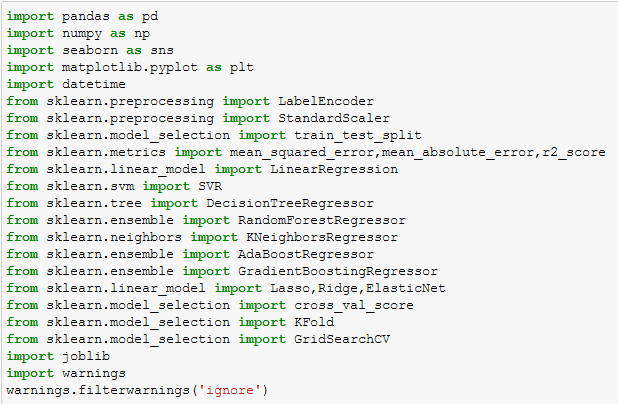
Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travellers saying that flight ticket prices are so unpredictable.

In this blog post, I will go through the whole process of creating a machine learning model on the dataset. The dataset has the information on the price the ticket was purchased along with source, destination and route the aircraft has taken.

*Flight prices are based on demand and availability. You may see that prices are high during weekend or during vacations as there is an increase in demand for holiday or leisure travel. Sometimes we notice that airline will offer promotional fare as they are expecting low availability during that period and want to lure the customer by offering discounts.*

Since the price is continuous we will require a regression model to predict the price.

**Importing the libraries we will use.**



We have train and test data. Let’s load the data.







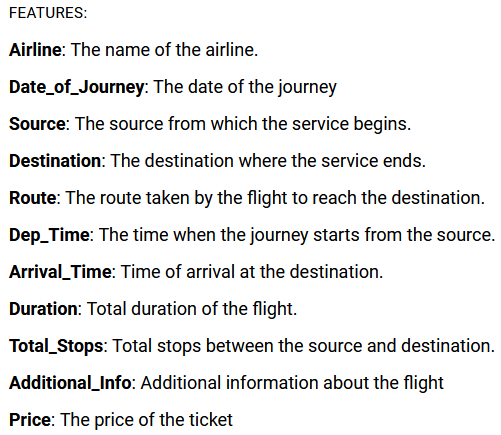
Train dataset has 10683 rows and 11 columns.

Test dataset has 2671 rows and 10 columns.

The one column less in the test dataset is the price column that we have to predict using the machine learning model.

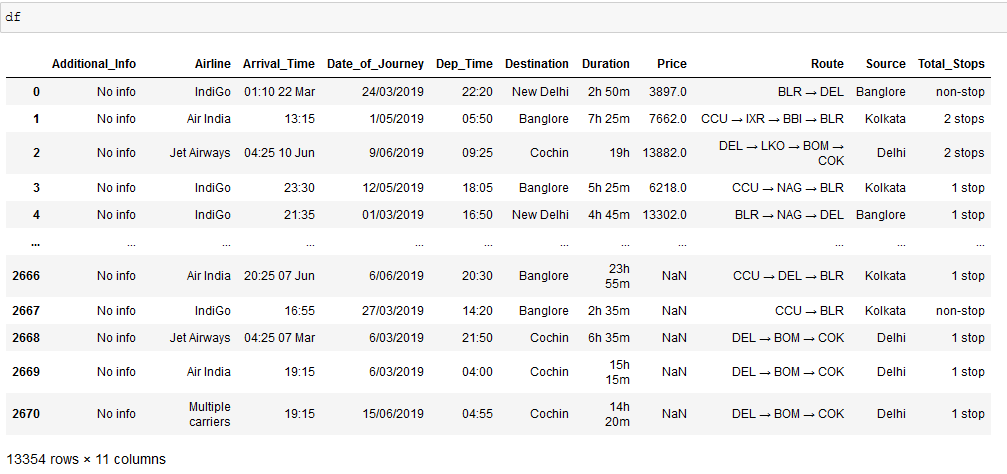
**2. Data Analysis**

Let’s see all the features we have in the dataset:



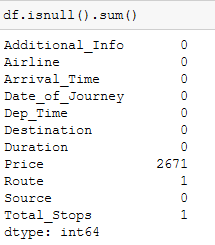
There is lot data cleaning and data processing we are going to perform, in order that the task is carried out simultaneously on training and test dataset, we are going to merge it.





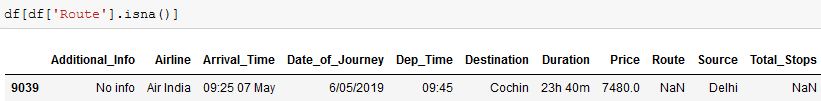
So now our new dataset has 13354 rows and 11 columns.

Let’s check for any missing values.

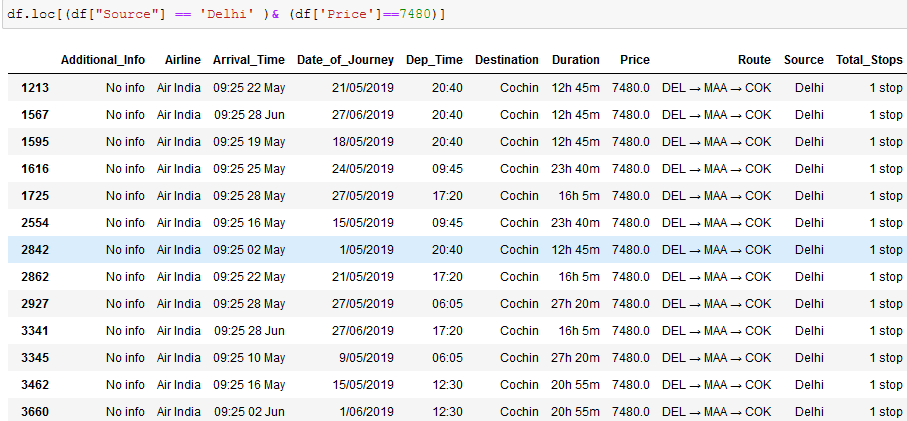


2671 null value in Price is of test dataset, the price which we need to predict. Hence we will not treat it.

There is 1 missing value both for Route and Total\_Stops.



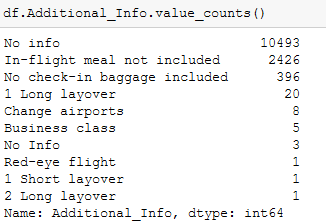
Form the above we can observe that missing value from Route and Total\_Stops is from the same row. To fill his missing value, I will check if we have any other row which has Source as Delhi and Price as 7480.



We can see that DEL-MAA-COK is the route where the same fare 7480 is applicable. Hence we will fill the missing values with this route and Total\_stops as 1 stop.

After treating the missing value we will check each feature and see if it requires any cleaning.

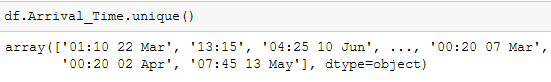
**3. Data Pre-processing and EDA**



We can observe that ‘No info’ has come twice, the other one with ‘Info’, hence we will correct the same.

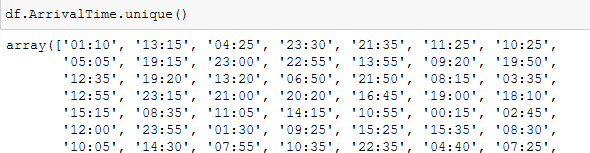


There is no correction require for ‘Airline’ feature. We will move forward to check for ‘Arrival\_Time’.



Some values in arrival time is along with the date. We will have to split the date from the time. We can use the split function as the time and date is separated by space and create the new column named as ‘ArrivalTime’





This is done. So we will drop the ‘Arrival\_Time’ column from the dataset. The ‘Dep\_Time’ has no such issues hence no correction is required there.

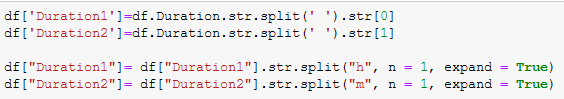
If we observe our dataset, the ‘Destination’ and ‘Source’ has the full city name mentioned like ‘Mumbai’ ‘Cochin’ and if we check the ‘Route’ column they have updated the code of the city used by the airline, like ‘BOM’ for Mumbai and ‘COK’ for Cochin. Being from the airline domain help me to identify the code I can apply for each city. Price of the ticket is highly dependent on the route it takes, hence for our machine to predict the price correctly, I will replace the name of the city with its code in ‘Source’ and ‘Destination’ column.



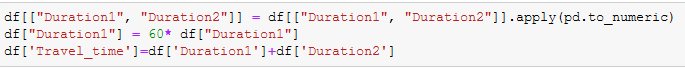


Moving on now we will check the Duration column. The duration column contains the total travelling time in hours and minute i.e. 2h 50m. I will change the value of total traveling time into minutes. Means the value 2h 50m will be changed to 170. This is to bring the Duration column on one scale.

To go about it, I will split the Duration columns and extract 2 columns, ‘Duration1’ with hours value and ‘Duration2’ with the value of minute if any.



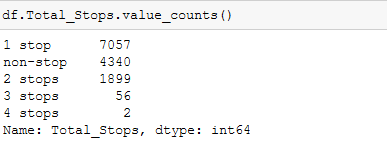
Now we will drop the Duration column and change the extracted columns to numeric. Once this is done we will multiply the value of ‘Duration1’ with 60 to convert the hours into minutes and then extract another column ‘Travel\_time’ by adding the Duration1 and Duration2 column which will give us the total travel time in mins.





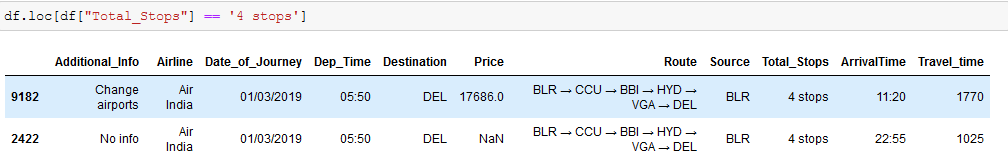
Since the new feature has been extracted, we will drop all the duration columns.

Let’s now observe the ‘Total\_Stops’ feature.



There is no data cleaning required in this column but we it will help us to extract another features.

What we are observing is that the some of the flight has maximum 4 stops going to its final destination. This stop can also be termed as VIA points. The details of the via points are given in the ‘Route’ column.



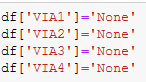
Let me break the Route details.

If Total\_Stops = 4 stops. That means we have 4 via points indicated by numbers below.

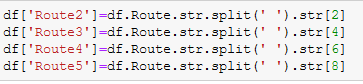


Similarly if total stops are 3 then the flight will have 3 via points.

So to extract the via columns from the route we have to split the route columns but before that we will create the 4 via columns with None as a value.



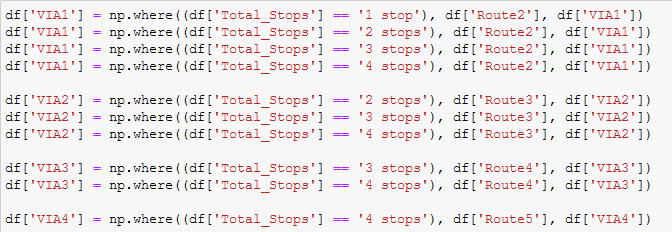
Now we will split the Route column. The 1st value in the Route column will always be the source. Hence we will not split that value. Now if Route has 4 via points then we will require 4 routes column to fill it.



Now comes the tricky part is to fill the VIA points.

Remember these points:

1. If the flight is non-stop that means all the values in via points will be ‘None’. Hence we will not have to do any processing here.
2. If the flight has 1 stop or more, then the values in Route 2 will go in VIA1.
3. If the flight has 2 stops or more, then the values in Route 3 will go in VIA2.
4. If the flight has 3 stops or more, then the values in Route 4 will go in VIA3.
5. Finally if the flight has 4 stops, then the values in Route 5 will go in VIA4.

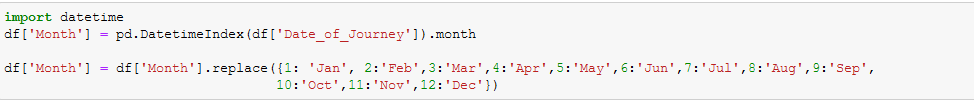


Some may ask, why don’t we just split the route from 1-6 and why update the via points?

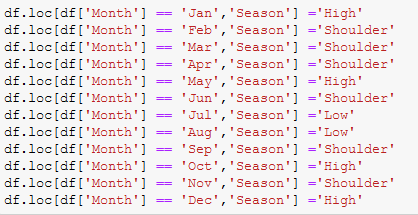
The reason is for the machine to learn, what the destination is. If I just split the route, some values in Route can either be the flight destination or via points, hence there is no clarity.

One the VIA points are updated we will the drop all the Route columns.

We will now extract the Month column from Date\_of\_Journey mentioned in the dataset and map each month with the name of the month.



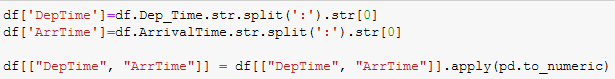
Remember I said in the beginning, that generally the flight prices are higher during vacation or weekend and lower during the non-vacation period. We will extract the season column to understand what season the flight was taken. Vacations in India are during Christmas, Diwali and summer holidays. Hence we will update them as high season. Rainy season will be marked as Low and remaining period we will update that as Shoulder season.



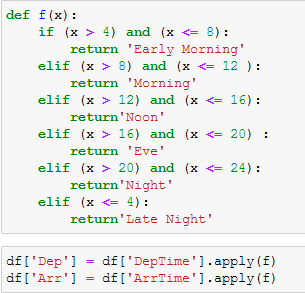
Now lastly let’s look at the Dep\_Time and Arrival\_Time column.

I will change the time to zones, like early morning, morning, noon, evening, night and late night. This is again a very important feature as sometimes the flight price is higher for early morning and night flights.

First we split the time to extract the new feature



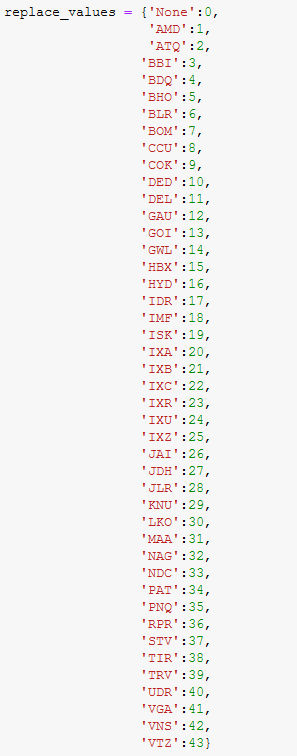
To change this to zone we will create a function and apply it to new extracted feature.

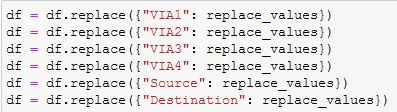


Now we will drop the other Departure and arrival time columns.

Remember we only have price and Traveltime as numerical feature and rest others are categorical. We need to change all categorical features to numerical. We will use LabelEncoder to all the features except Via points. Yes we are not yet done with via points.

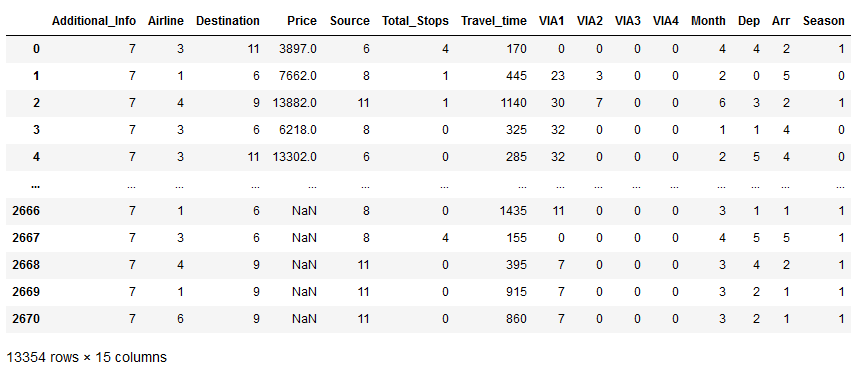
I want each city with the specific numeric value e.g. AMD should be shown as 1 in any given feature it is present. This will classify my source, destination and via points with correct numerical value. Using LabelEncoder will not give me this liberty.





And now we will run the LabelEncoder for remaining categorical feature.

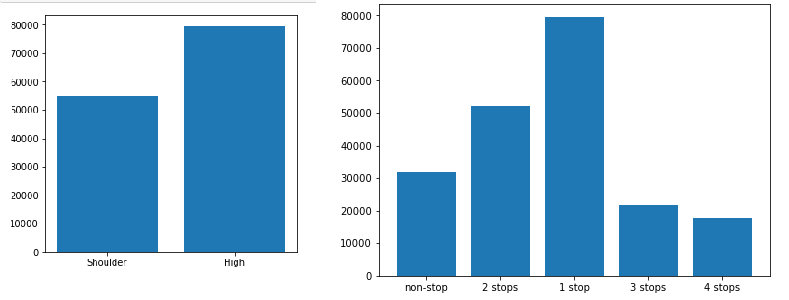
Let’s have a look at our final dataset.



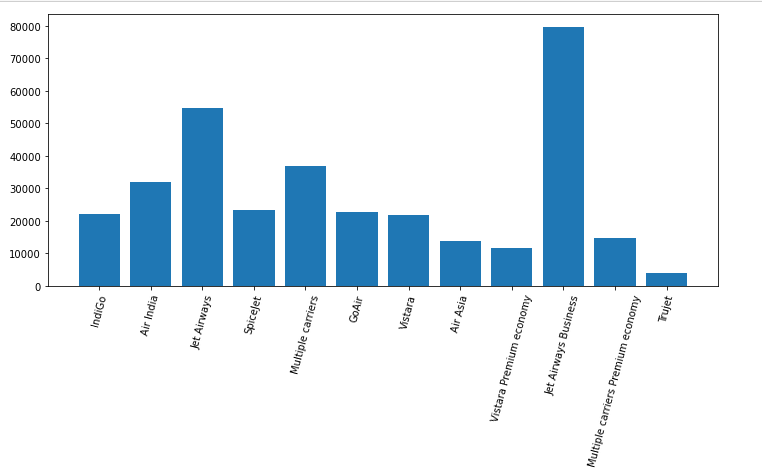
Before we move forward with model fitting we will visualize the data.

Let’s check some of the feature with our target variable ‘Price’

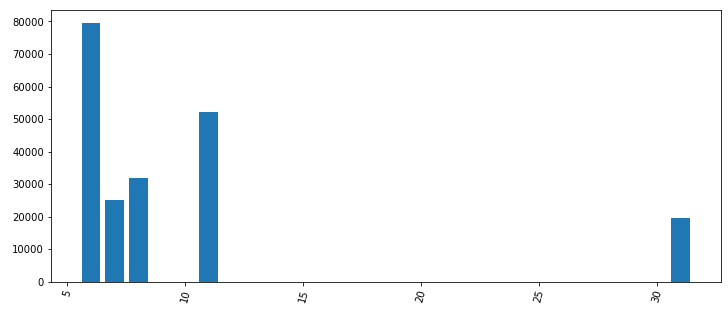
**Season and Total\_Stops**



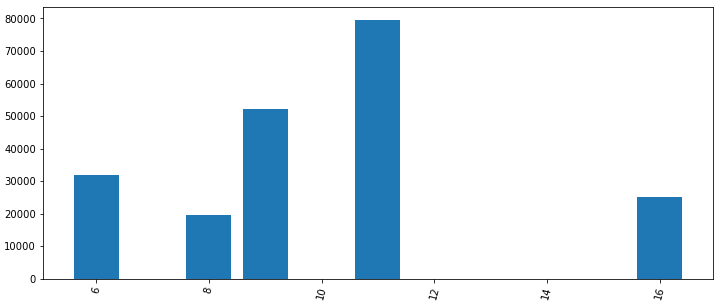
**Airline**



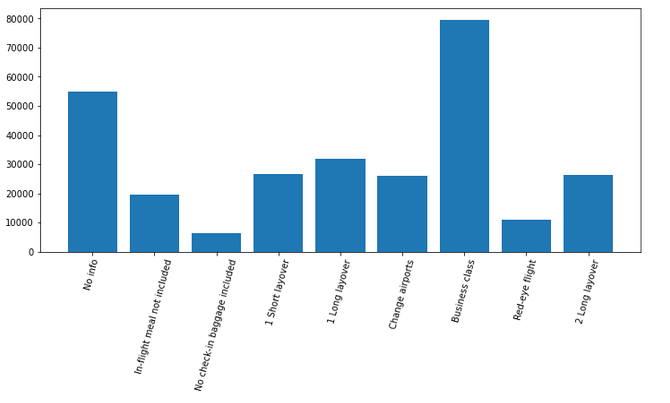
**Source v/s Price**



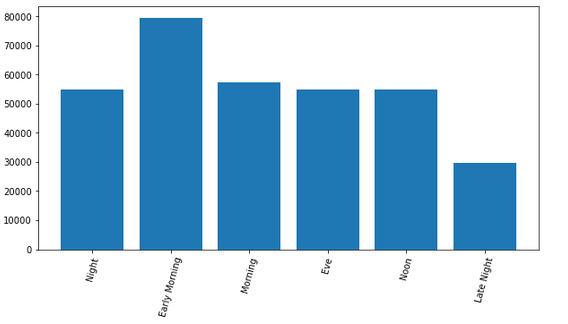
**Destination v/s Price**



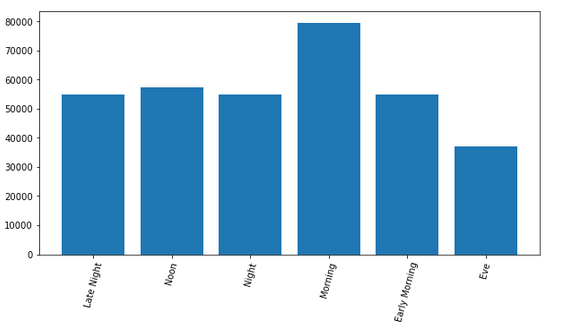
**Additional info**



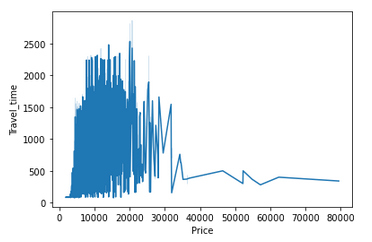
**Departure time**



**Arrival time**

****

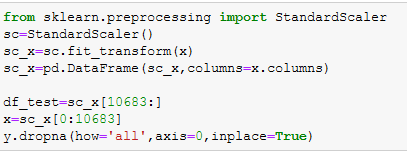
**Travel Time**

****

**4. EDA concluding remarks:**

1. Since there is no data for the month of Low season we can clearly see that the price is higher during the High season.
2. Flight with 1 stop has the higher price. This could be because of the cabin customer is flying on. Maybe business class.
3. We can observe that fares are high for Jet airways business. True Jet has the lowest price.
4. Price will be higher if the Source is BLR.
5. Destination Delhi will have the higher ticket Price.
6. Additional Info also show that price is higher for business class and the lowest for No check-in baggage
7. Flights departing early morning tend to have the higher price. Lowest price is available for the flights departing late night.
8. Flights arriving in the morning have higher price.
9. There is no correlation we can find with travel time and price.
10. We should have more information on the type of fare customer has purchased and also the cabin the customer is using.
11. **Building Machine Learning Models.**

To build the machine learning model we need will 1st split the input and target variable. Later we will bring the data to a common scale using the StandardScaler.



We have also splitted the train and test data.

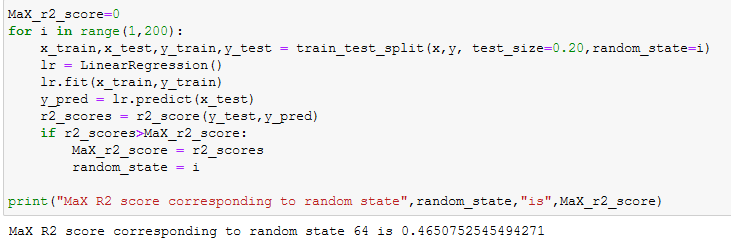
We will train our model using below Algorithtm

1. LinearRegression
2. SVR
3. DecisionTreeRegressor
4. RandomForestRegressor
5. KNeighborsRegressor
6. AdaBoostRegressor
7. GradientBoostingRegressor
8. Lasso
9. Ridge

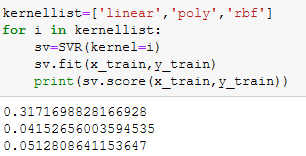
We will also use K-fold cross validation process to check the under fitting or over fitting of our trained dataset.

K-Fold Cross Validation randomly splits the training data into K subsets called folds. Let’s image we would split our data into 5 folds (K = 5). Our model will be trained and evaluated 5 times, using a different fold for evaluation everytime, while it would be trained on the remaining 4 folds.

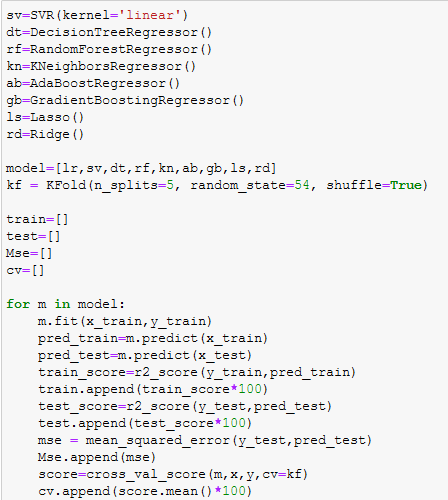
We will 1st select the best random state to run our model.



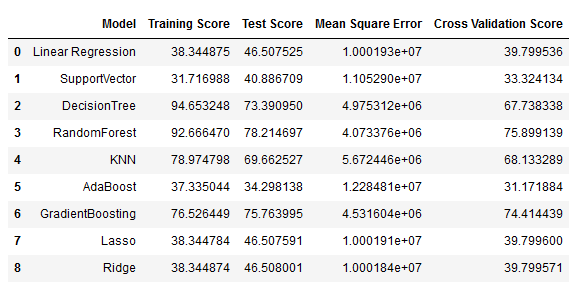
After splitting the data with the above random state we will find the best kernel list for Support vector Regressor model.



Now lets the train the data with remaining model.



Let’s check all the model performance.



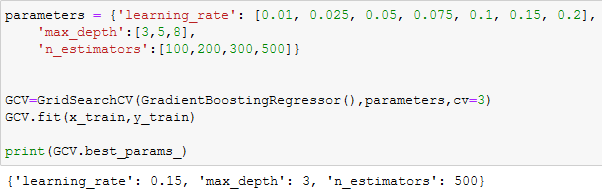
To select the final model we don’t only have to look at the test accuracy score, we also need to check the Cross validation (CV) score. The least difference between the test and CV score indicates that the model is performing well without being undefit or overfit.

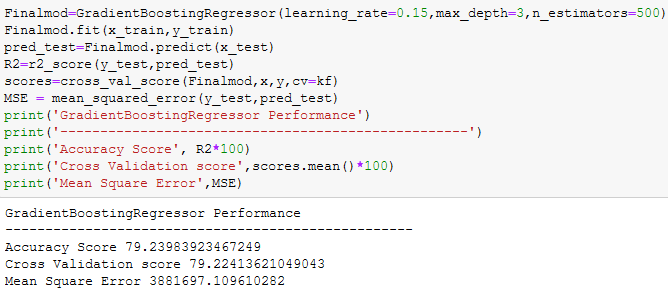
From the above performance metrics we can conclude that Random forest Regressor (RFR) has performed well with test accuracy of 78% and CV score of 75% . Although the test accuracy of GradienBoosting is 76% it has the CV score of 74% indicating less underfitting then RFR.

We will hypertune both the model and see if there is an improvement in the model performance.

**HyperParameter Tuning**

Hyperparameters are crucial as they control the overall behaviour of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.

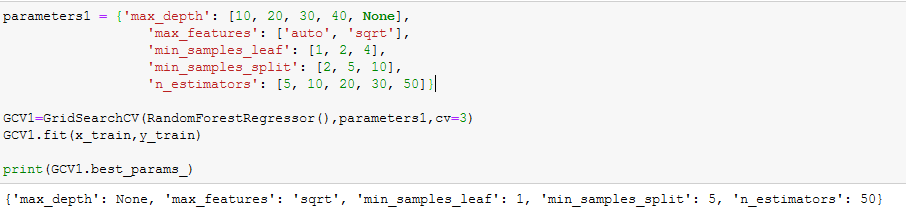


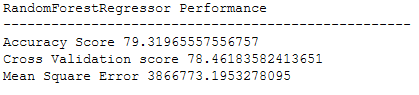


Bingo!!!

From the best parameters we got after running the GridSearchCV our model performance has improved considerably. We got both the test accuracy and CV score at 79% for GradientBoosting.

Let’s check the performance of RFR after hypertuning the model.

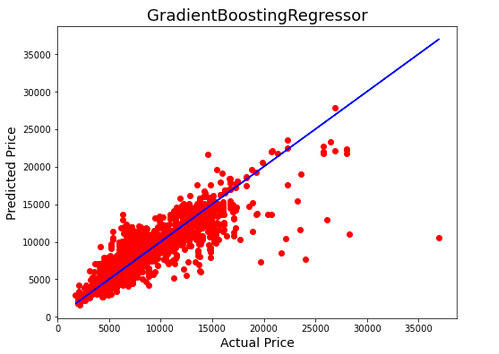




We can see that even RFR performance has improved after HyperTuning. The RFR has got the test accuracy of 79% and CV score of 78%.

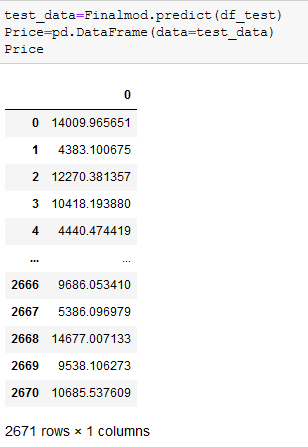
As I said we have to select the model that gives us the least difference on CV score with test accuracy.

**Hence we will select GradientBoostingRegressor as our final model since it has no difference between test and CV score.**



This is how our model is fitting.

Let’s predict the value of price from the test dataset.



We are now able to predict the price from test dataset with 79% accuracy.

**Summary**

We started with the data exploration where we got a feeling for the dataset, 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 preprocessing part, we computed missing values, converted features into numeric ones, grouped values into categories and created a few new features. Afterwards we started training 9 different machine learning models, picked two of them(RFR and GradientBoostingRegressor) and tuned it’s performance through optimizing it’s hyperparameter values and finally selected the GradientBoostingRegressor.

1. **Concluding Remarks**

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. Also more details about the flight price will certainly help. Like the cabin customer is flying and fare category, special fare has lot of discount but we don’t enjoy free meal on board or maybe no flexibility to change or refund the booking. The additional info feature did have for some of the flights but we should have that for most of the flights.

**Thank you.**