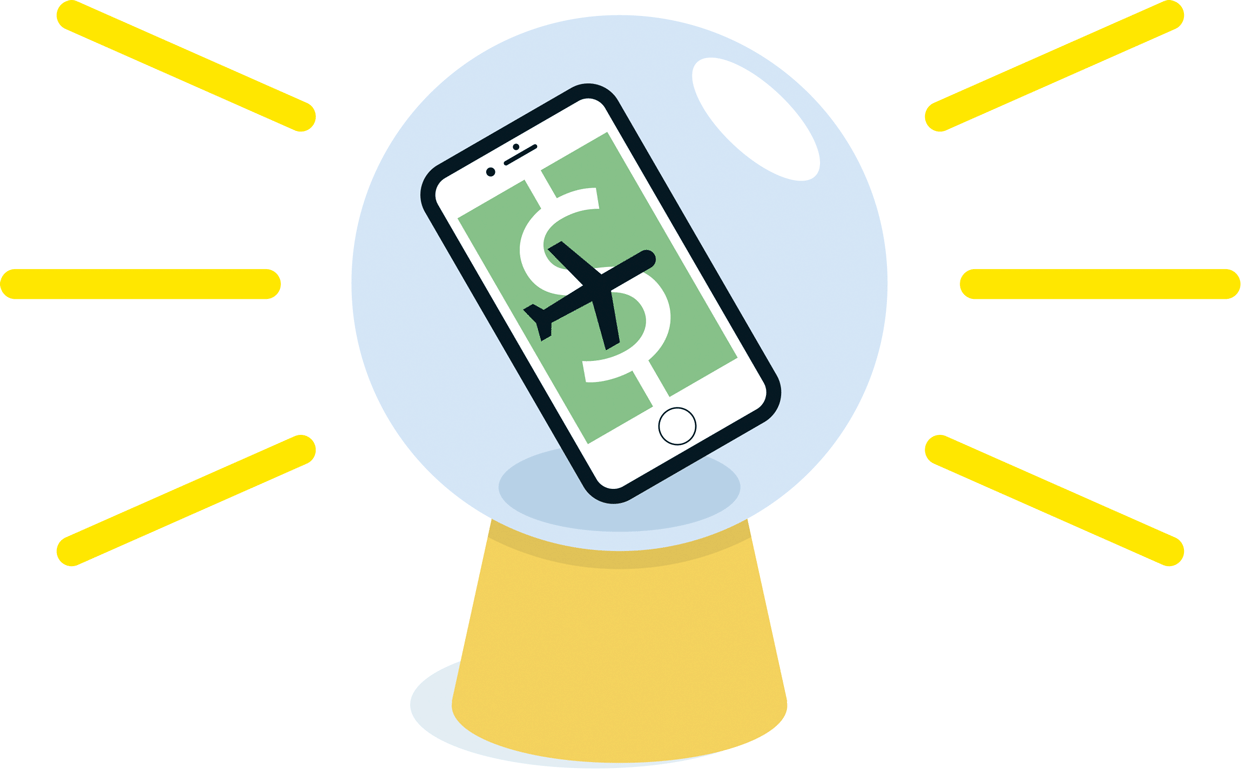
**Flight Price Prediction**

A blog on Data Analysis and Predictions.



**Presented by :**

Chamlin Najir Rahman

In association with DataTrained Academy

Batch No. 1838

2022

1. **Introduction :**

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 travelers saying that flight ticket prices are so unpredictable.

In recent times, the number of people using flights has increased notably. It is also difficult for the airlines to maintain prices since prices change dynamically due to different conditions. That’s why we will try to use machine learning to solve this problem. It can help both airlines as well as customers to predict future flight prices and plan their journey accordingly.

1. **Objective :**

The objective of this blog is to analysis and predict flight prices, using machine learning models and the given data and parameters.

This project will be the case of regression, since our target variable is ***Price***, which is continuous numeric.

1. **Prerequisites :**

### Hardware requirement :

* Processor : core i5 or above
* RAM : 4 GB or above
* ROM/SSD : 250 GB or above

**Software requirement** :

* Jupyter Notebook

1. **Data Used :**

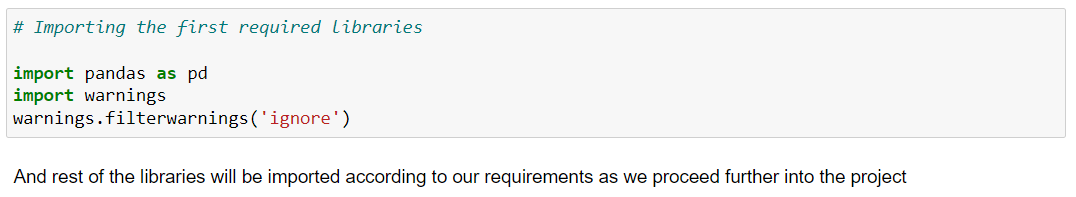
The data used in this project was provided by the academy. We are provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

In this project we are provided with 2 separate datasets, one for training purpose and one for testing purpose, for the machine learning model.

***Source :*** [*https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects*](https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects)

1. **Importing dataset :**

We’ll start our project by importing the required datasets into our juypter notebook :

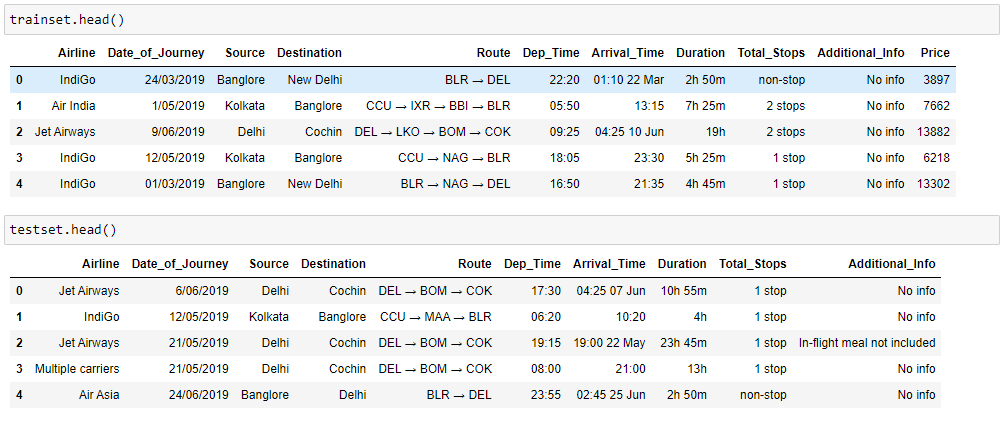


* Imported pandas library, named as pd, to create dataframes.
* Imported warnings library to ignore warnings.



These are 2 datasets we have :

* trainset contains the dataframe for “Flight\_Train.xlsx” dataset. This is the dataset that will be used for training our machine learning models.
* testset contains the dataframe for “Flight\_Test.xlsx” dataset. This is the dataset that will be used for predicting the Price using the machine learning model.



Now these are our datasets for the project, I have displayed the first 5 rows of each dataset using the ***df.head()*** method.

Now let’s understand the various features of the dataset :

**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.

***“Price”*** feature is not present in the testing dataset, as we will predict the price using the best machine learning model.

**NOTE: -** We also cannot combine both the datasets, or else there will be data leakage. That means our training model will get to know some of our testing values and we might not get the correct predictions.

# 6. Exploratory Data Analysis for Train Dataset :

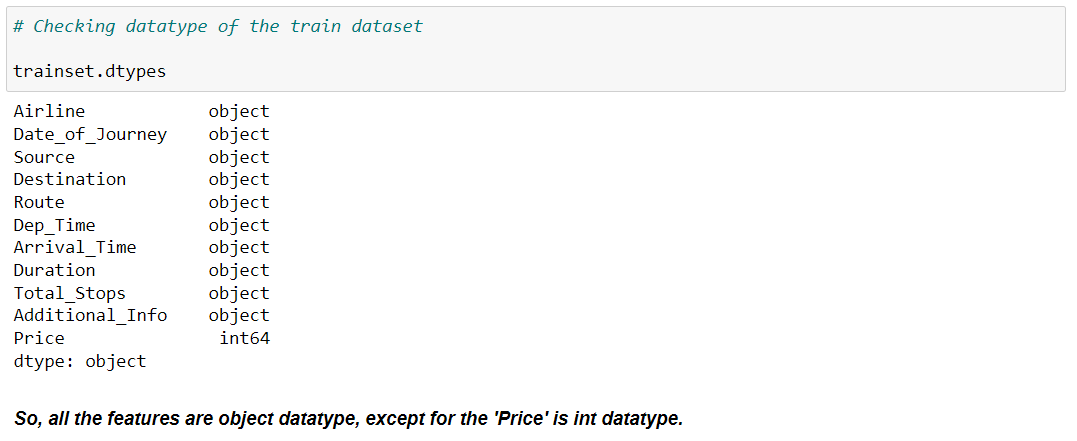
First we will start with the EDA process for the training dataset :



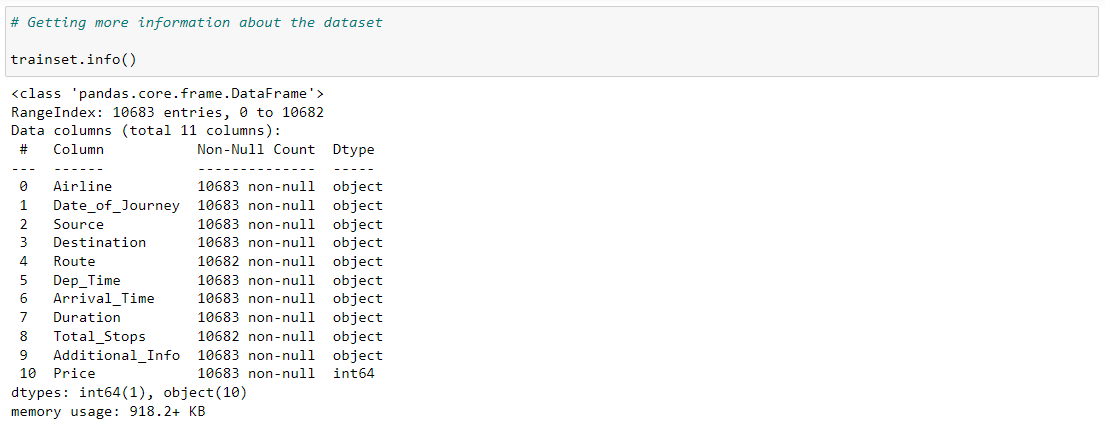
Using ***trainset.columns*** command we got all the features present in the dataset.

Using ***trainset.set*** to retrieve the information :

Number of variables = 11  
Number of rows = 10683



Here we are checking the data type of each feature in the dataset.

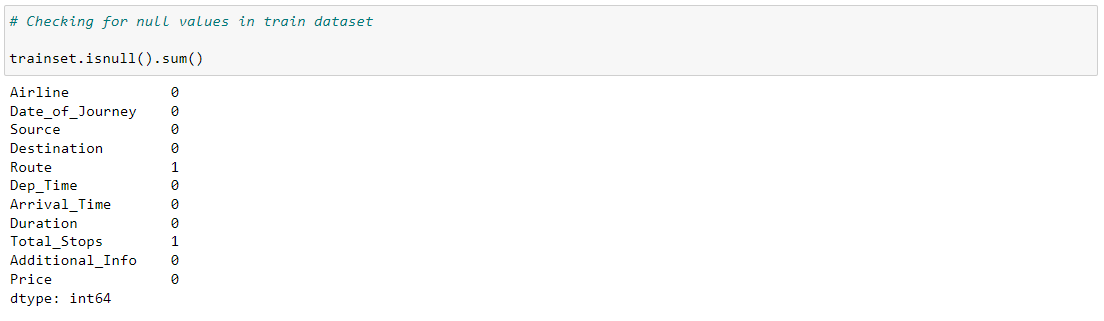


Using ***trainset.info()*** we can retrieve an overall information about the dataset.

It also displays the non null count of each feature in the dataset, Ex. 10683 non null means there are no null values in that feature, but in “**Route”** feature there are 10682 non null values, that means there is one null value

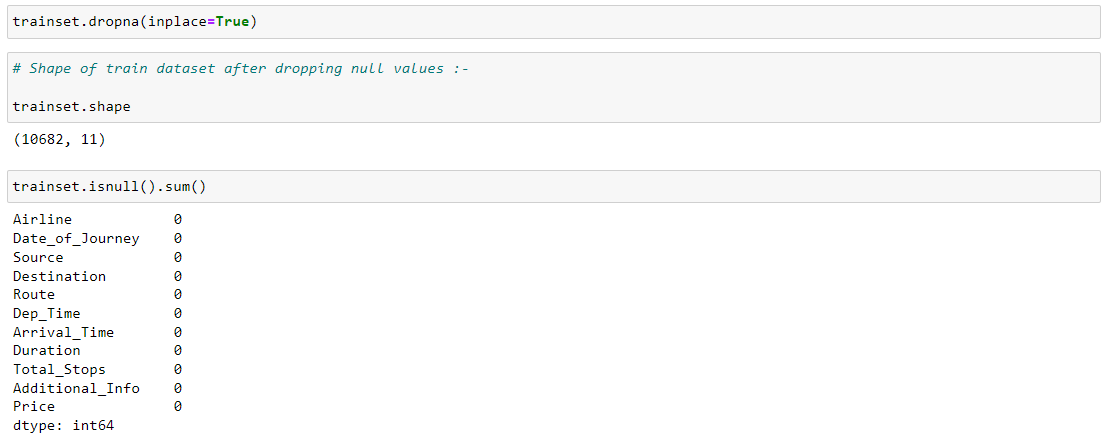
And memory used by the dataset is 918.2+ KB.

**Check for null values :**

******

**trainset.isnull().sum()** command is used to check null values for the whole dataset.

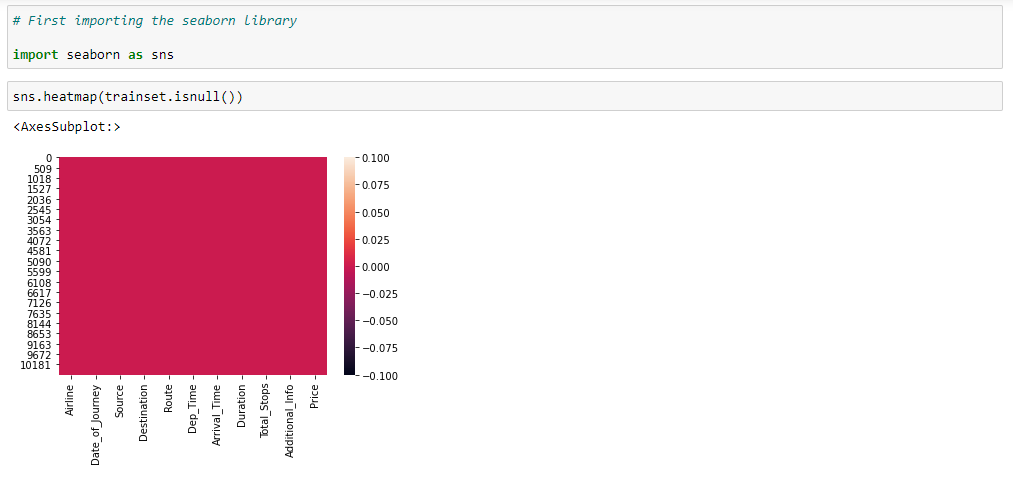
There are only 2 null values in the train dataset, so it can be dropped easily.

**

***trainset.dropna(inplace=True)*** command is used to drop all the null values present in the dataset.

And then we observe that all the null values are successfully dropped.

We can also visualize and check using heatmap :

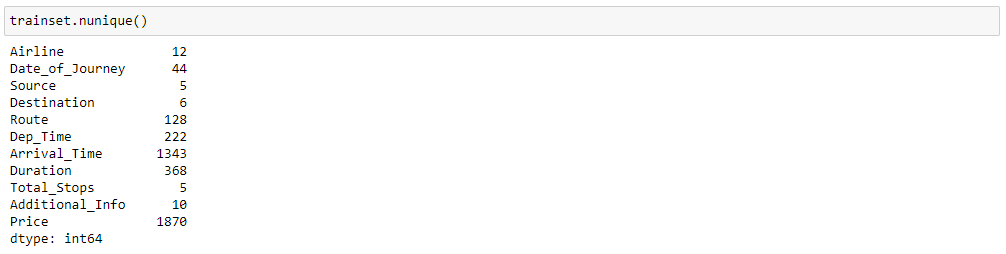


Here first the seaborn library is imported, to use heatmap.

Since there are no white spaces in the heatmap, we can conclude that there are no null values in the dataset.

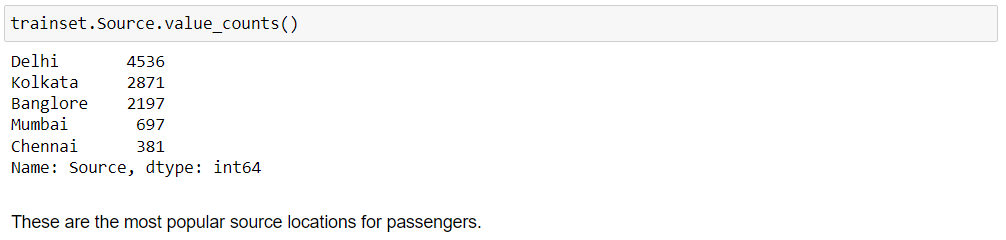
Proceeding further into the analysis :

**Checking unique categories for each feature :**

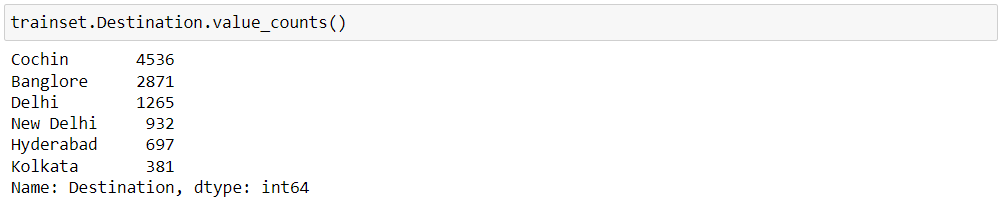
****

***trainset.nunique()*** is the command used to check unique categories of each feature.

We have some categorical data here, let’s check their value counts :

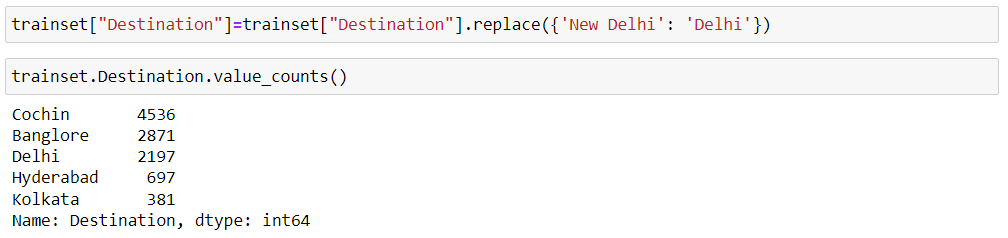


Here we have checked unique values for the **“Source”** feature, where Delhi has the highest count.

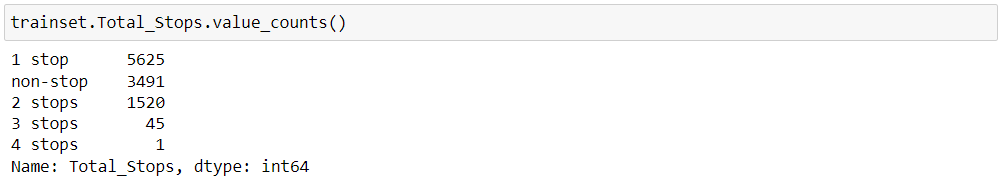


Here we have checked unique values for the **“Destination”** feature.

These are the popular destinations passengers are travelling to. But there are repeated values for 'Delhi' and 'New Delhi', which are basically the same locations. Hence it can be replaced with one feature.

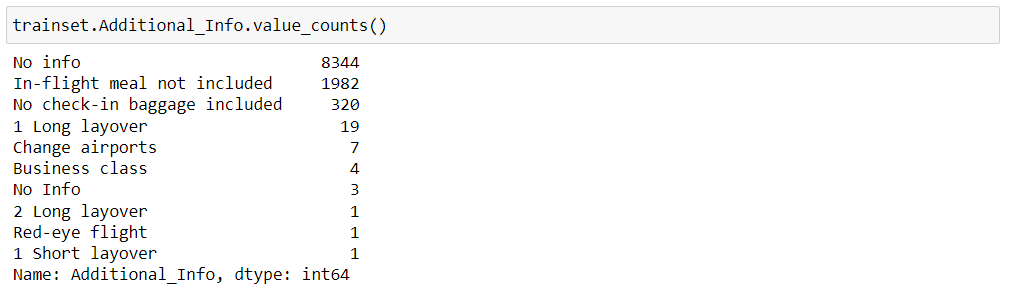


So in this way I replaced the value “Delhi” into “New Delhi”.

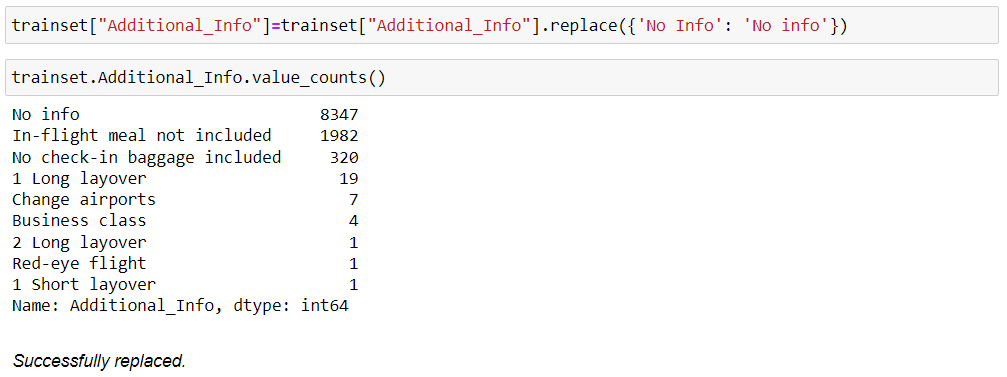


Here we have checked unique values for the **“Total\_Stops”** feature.

Passengers mostly prefer the 1 stoppage route.



Here we have checked unique values for the **“Additional\_Info”** feature.

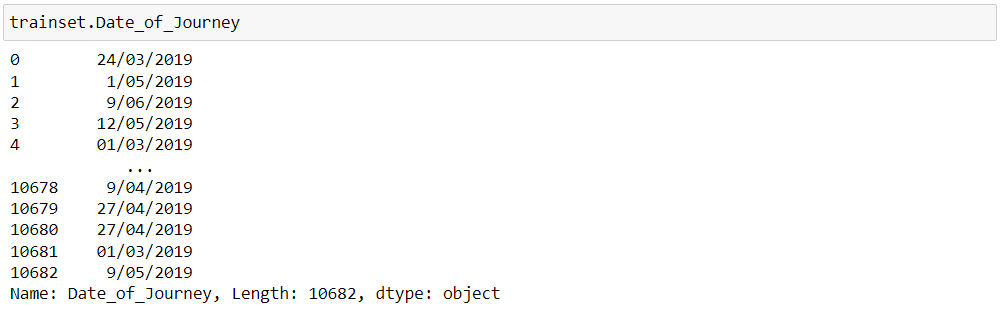
Here also we can observe repeated values for 'No info'. And hence these will be replaced into one value.

# 7. Feature Engineering :

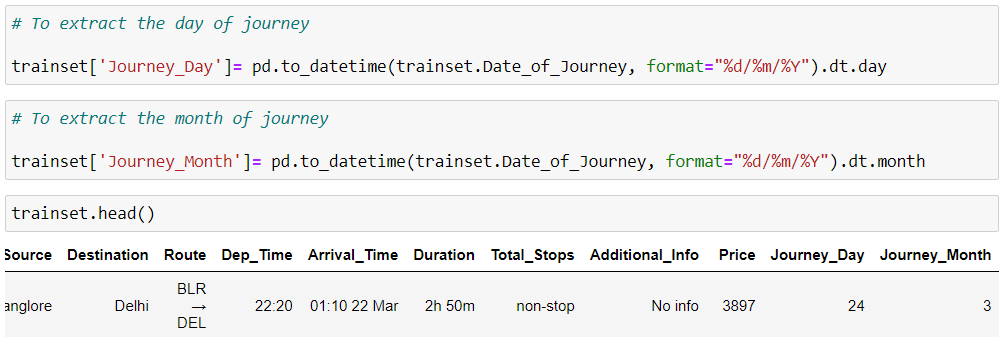
In the dataset we can observe columns such as **"Date\_of\_Journey", "Route", "Dep\_Time", "Arrival\_Time", "Duration"** are not in a proper format which can be used for analysis and machine learning models.

Therefore we need to convert them into timestamps using **datetime** library for proper analysis and prediction.

**Date\_of\_Journey :**

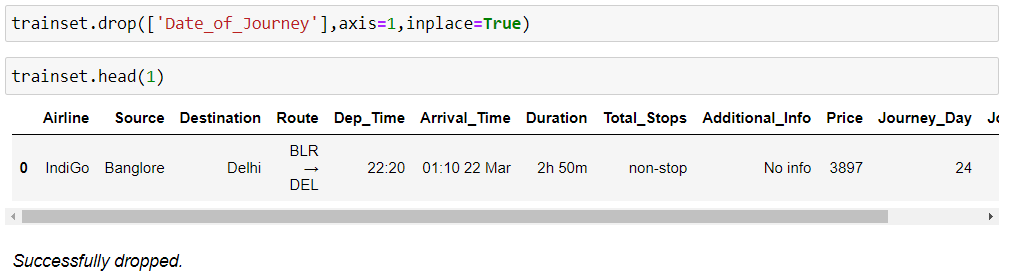


From "Date\_of\_Journey", we will extract only the date and month. This data is specifically for the year 2019 only, so no need to extract year.

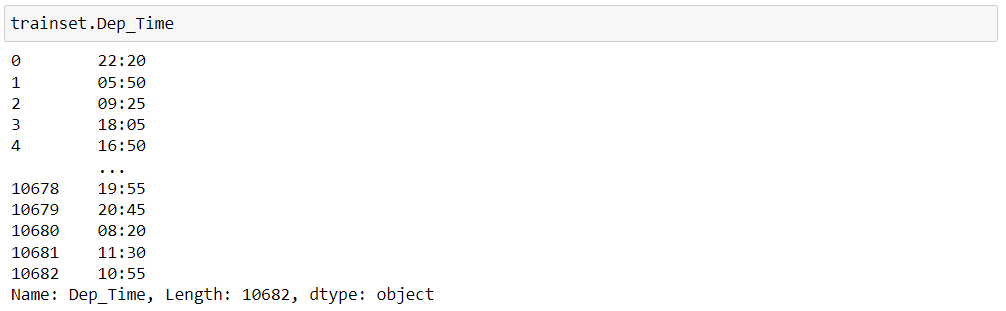


Separate columns for day and month of journey are added to the train dataset.

Hence, **"Date\_of\_Journey"** can be dropped.



**Now similarly, we can extract values from "Dep\_Time" :**

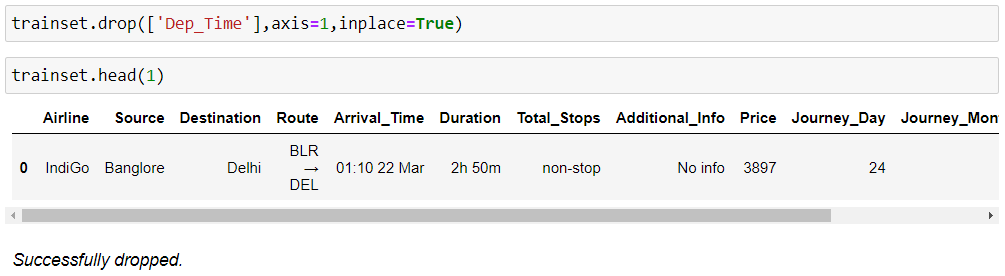


From here we have to extract the hour value and minute value separately.



Hence 2 separate columns for hour and minute of departure are added to the train dataset.

Hence, **“Dep\_Time”** can be dropped.

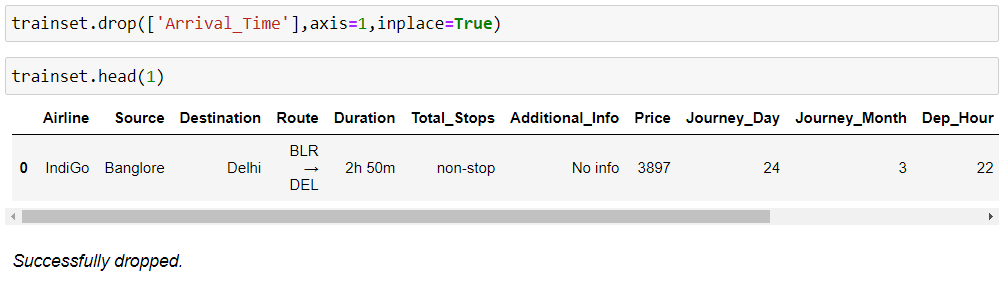


**Now similarly, we can extract values from "Arrival\_Time" :**



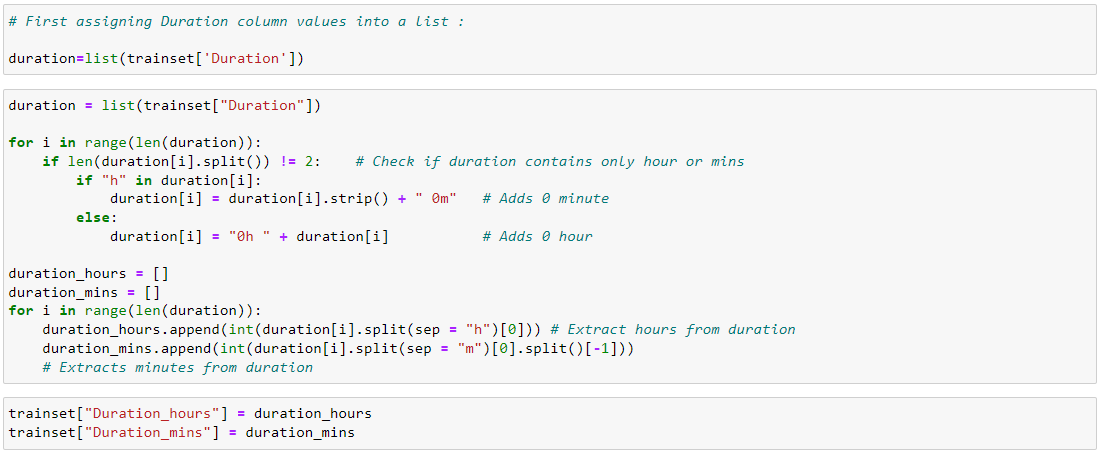
Hence 2 separate columns for hour and minute of arrival are added to the train dataset.

Hence, **“Arrival\_Time”** can be dropped.

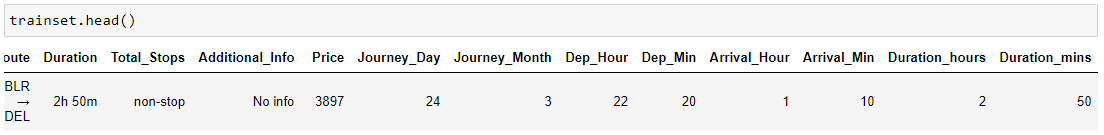


**Now we have to extract values from "Duration" :**

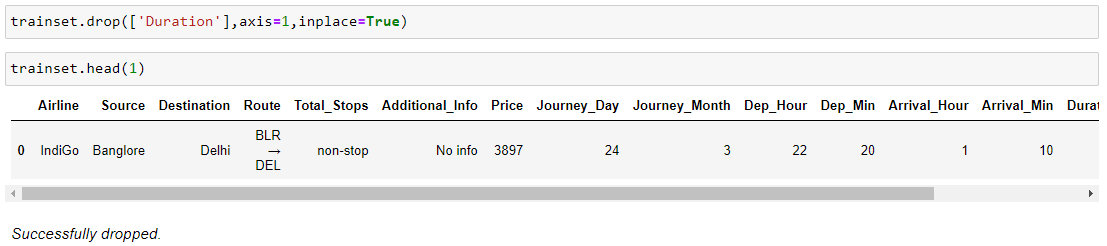
In this “Duration” feature, most of the rows has 2 values, i.e. the hour and the minute. But some rows also have one value only, either the hour value or the minute value. Hence here the extraction process will be different. First we will check if all the rows has both the values, by running a for loop. If both values not present, then we will either add a “0m” wherever minute value is not present, or else add “0hr” wherever hour value is not present.



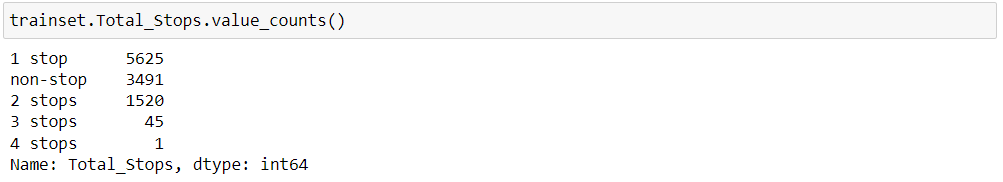
Hence 2 separate columns for hour and minute of “Duration”, are added to the dataset.



Hence **“Duration”** column can be dropped.

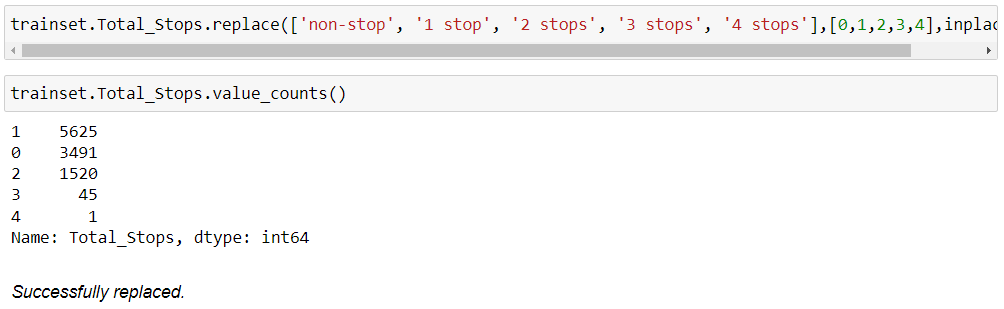


**Handling "Total\_Stops" column :**



Here we can observe 4 unique values : non-stop, 1 stop, 2 stops, 3 stops, 4 stops.

We can replace these values with single digits for easy analysis, because while using label encoder to convert them, it might assign random values.



**These features are handled successfully, hence we can move into further analysis.**

**Handling Categorical Data :**

We have 3 categorical features to handle :

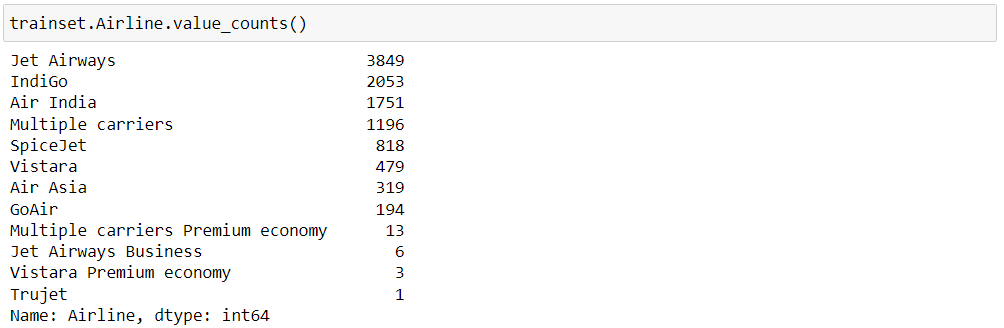
1. Airline

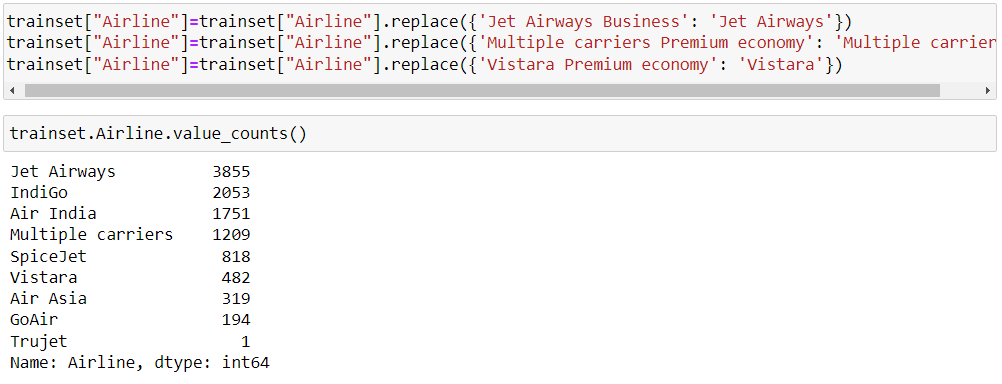
2. Source

3. Destination

**Airline :**

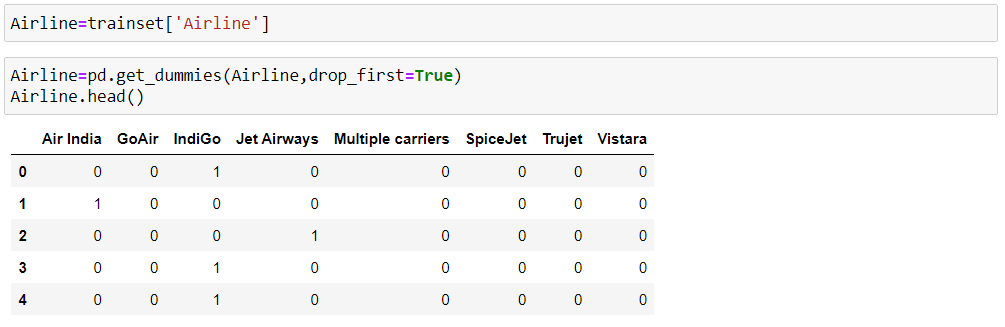
Here we can observe repeated values for some airlines, which are Jet Airways, Multi-ple carriers, and Vistara. Hence these can be replaced with one airline only.





Hence the repeated values are successfully replaced.

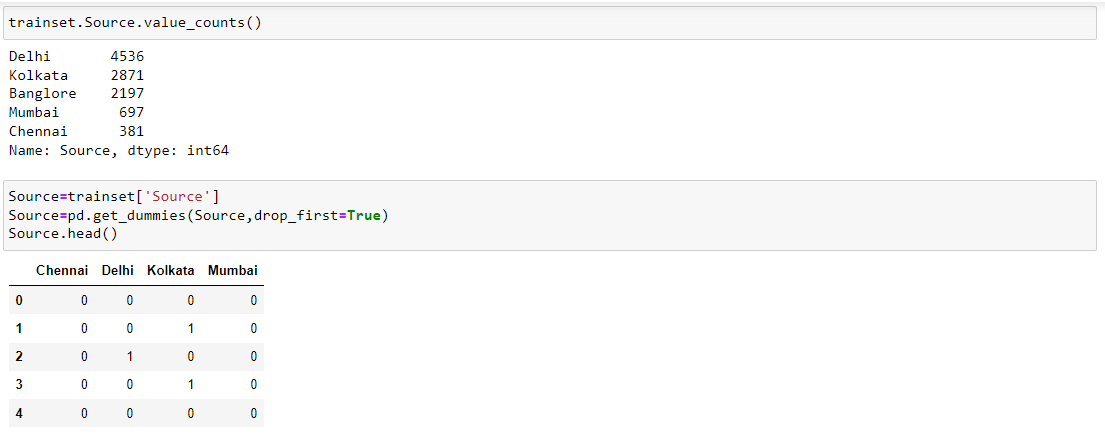
Now we need to convert this column into numeric data type for further analysis. Here Airline is nominal data, that is, data is not in any specific order, hence we use OneHotEncoder.



And we obtained separate columns for all the airlines. These 1 and 0 value in all the columns represent the entries wherever they were present in the main feature i.e. “Airline”. 1 means that airline was present for that entry in the main feature and 0 means that airline was not present for that row entry.

**Source :**

Here also we need to convert this column into numeric data type for further analysis. Here “Source” is nominal data, that is, data is not in any specific order, hence we use OneHotEncoder.

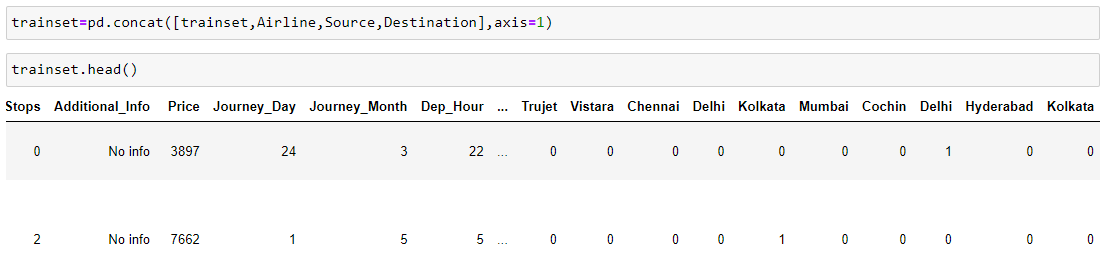
****

And we obtained separate columns for the source places.

**Destination :**

Similarly here also we need to convert this column into numeric data type for further analysis. Here “Destination” is nominal data, that is, data is not in any specific order, hence we use OneHotEncoder.

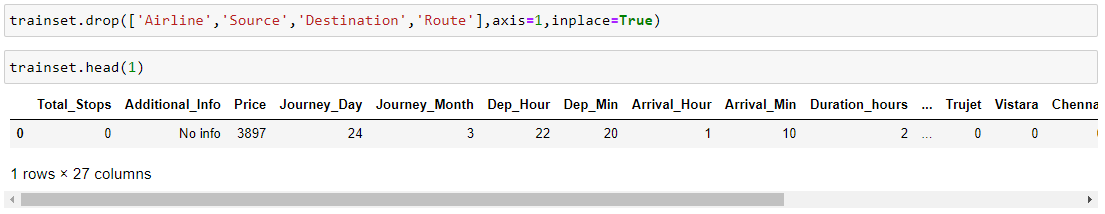
**Now let’s merge all the above 3 dataframes into our train dataset :**

****

Successfully merged and all the columns from the 3 dataframes are here.

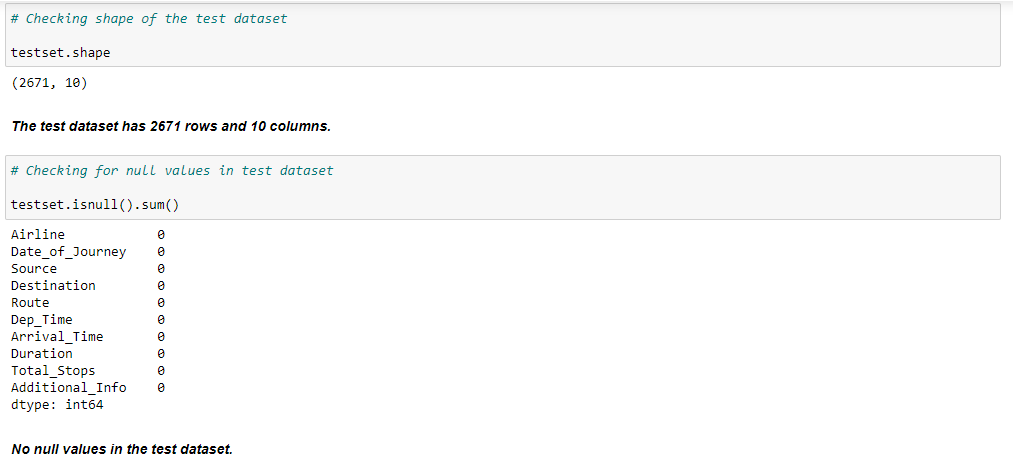
Therefore now we can drop those 3 columns from the dataset, as we already encoded them.

'Route' feature is similar to 'Total\_Stops', as the route also gives total number of stoppages during the whole journey, so it can be dropped.

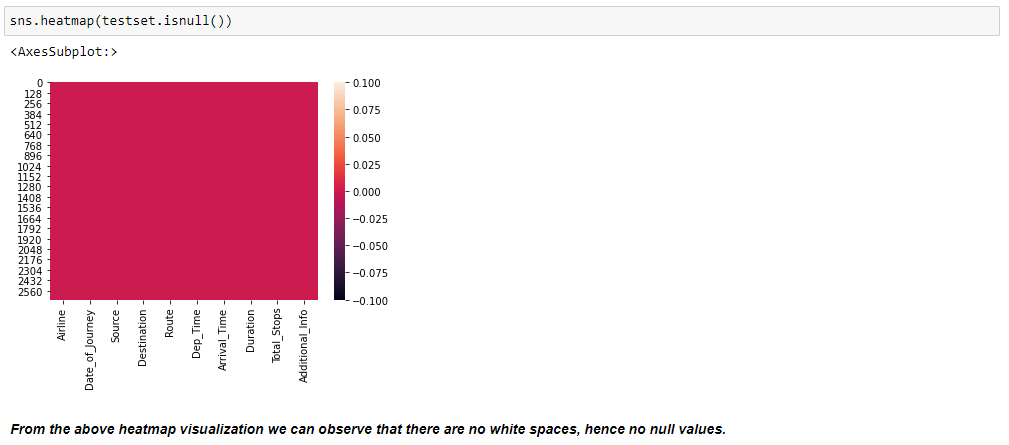


Hence successfully dropped.

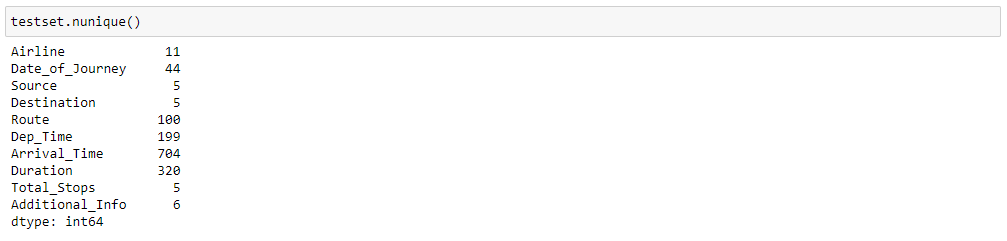
# 8. Exploratory Data Analysis for Test Dataset :

All the same EDA steps, as above, will be performed for the test dataset too.

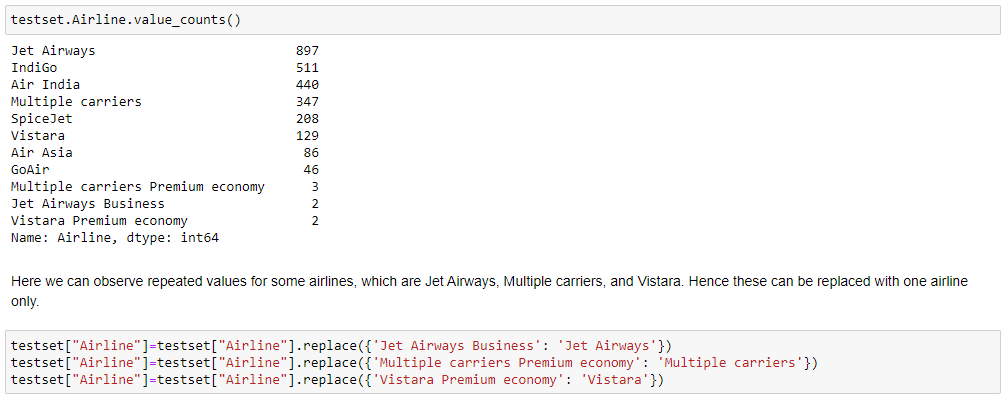
**Checking null values :**

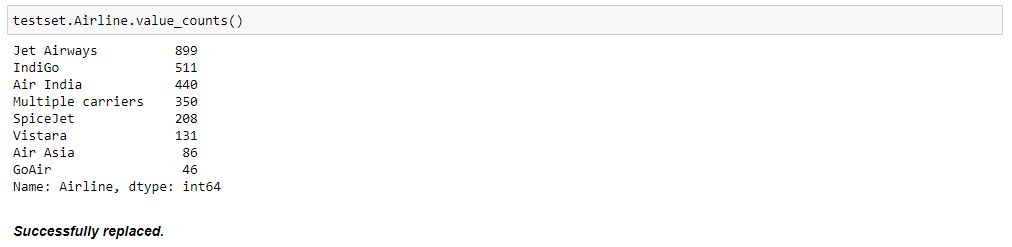
****

**Checking unique categories fo each column :**

****

**Replacing repeated values :**

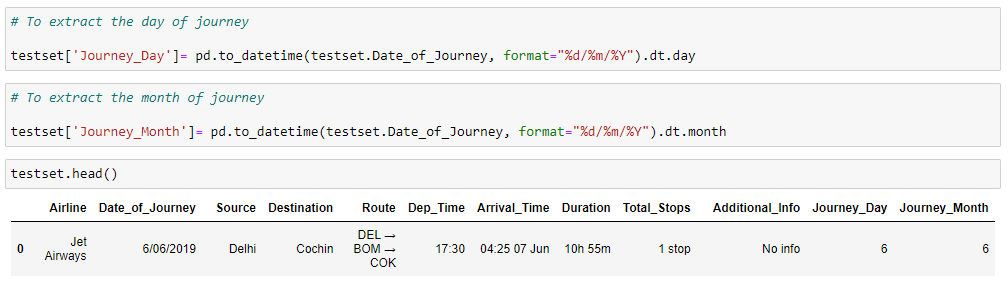
****

****

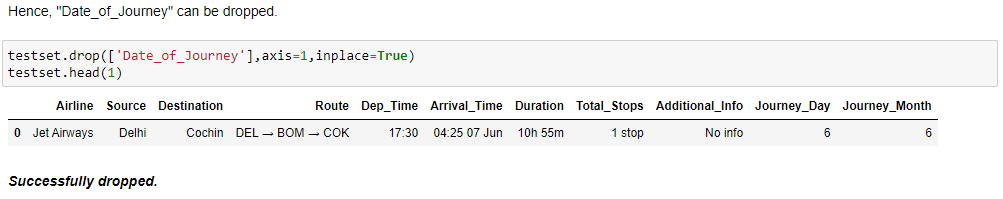
**9. Feature Engineering :**

In this dataset too, we can observe columns such as **"Date\_of\_Journey", "Route", "Dep\_Time", "Arrival\_Time", "Duration"** are not in a proper format which can be used for analysis and machine learning models.

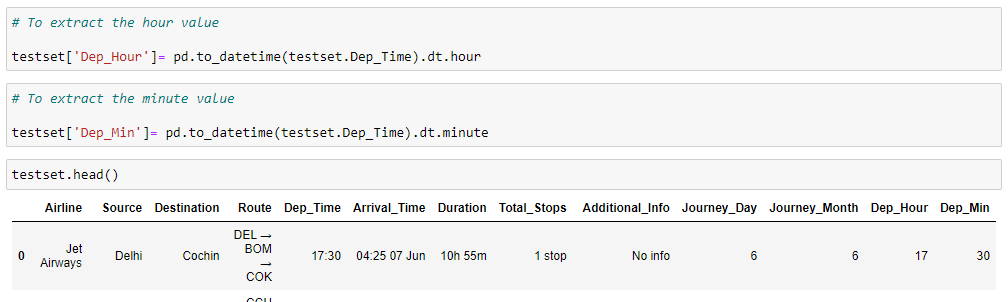
Therefore we need to convert them into timestamps using **datetime** library for proper analysis and prediction.

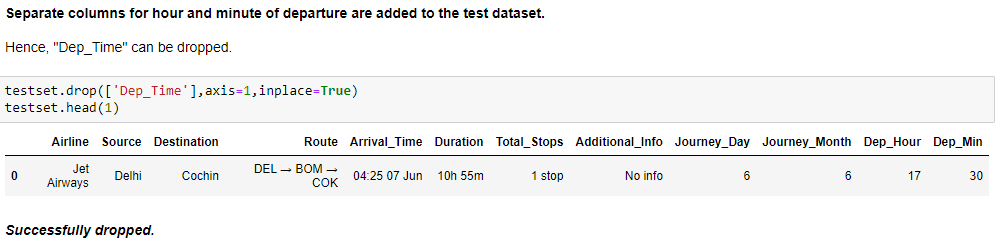


Separate columns are added for the day and month of journey in test dataset.



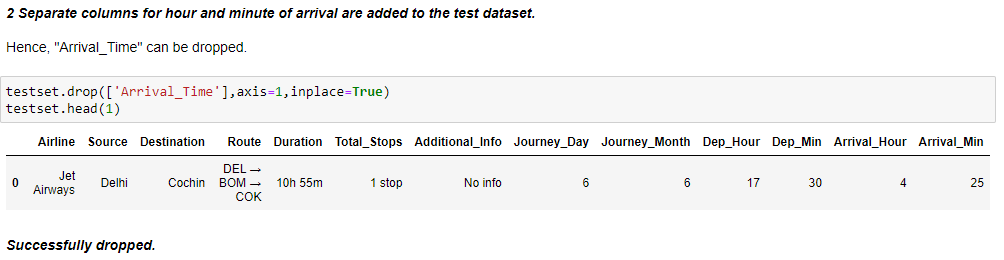
**Now similarly, we can extract values from "Dep\_Time"**





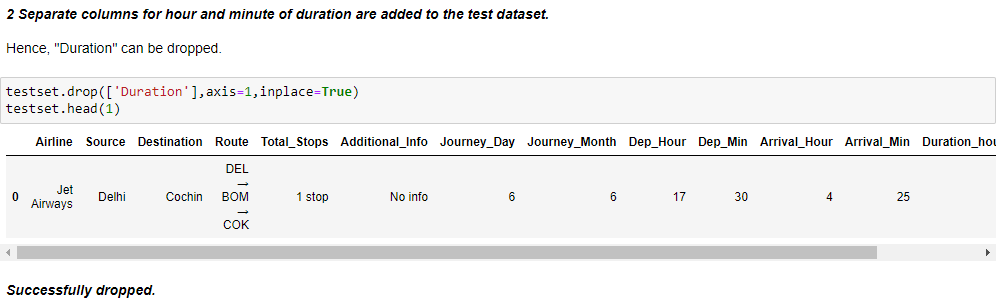
**Now similarly, we can extract values from "Arrival\_Time"**

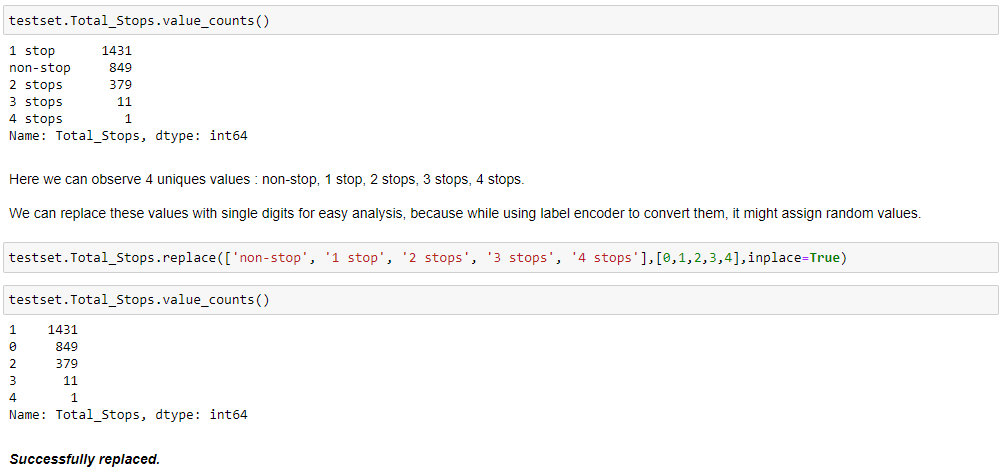




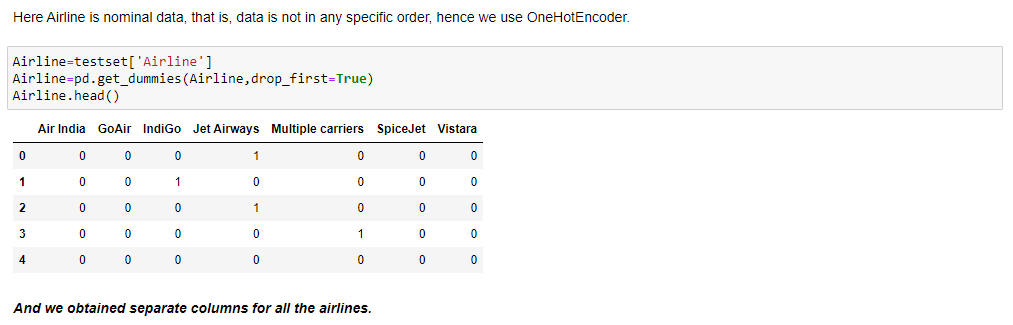
**Now we have to extract values from "Duration"**



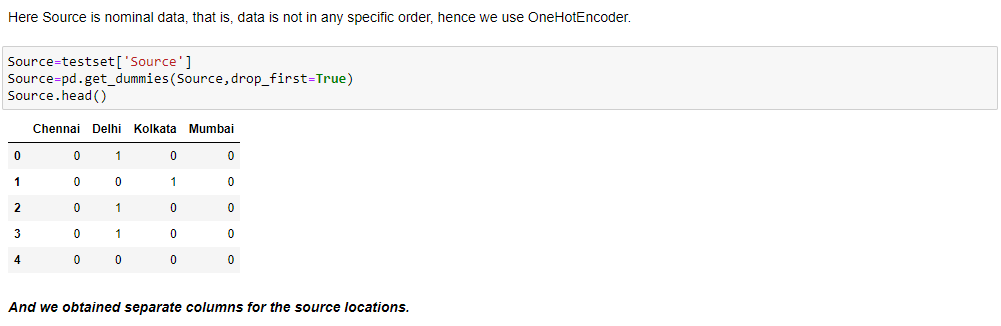


**Handling "Total\_Stops" column :**

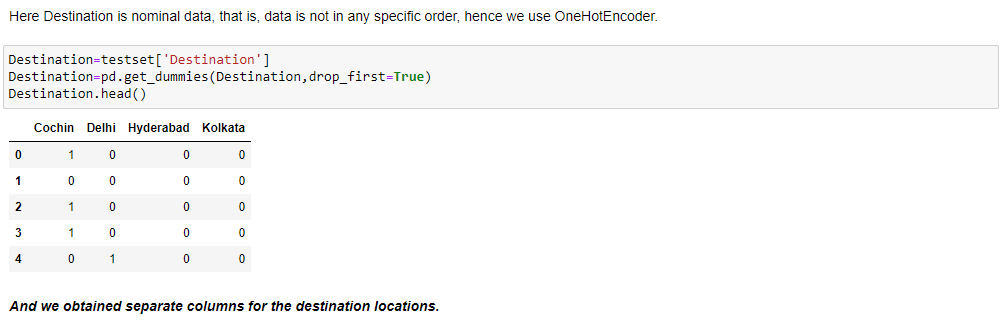
**Handling "Airline" column :**



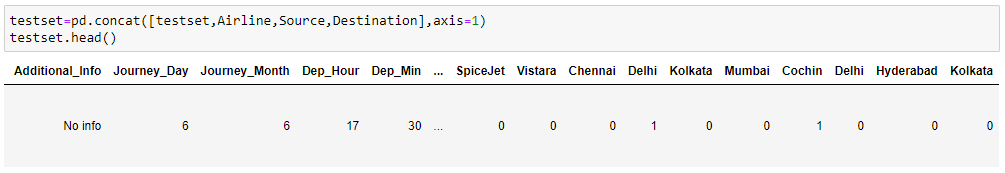
**Handling “Source” column :**

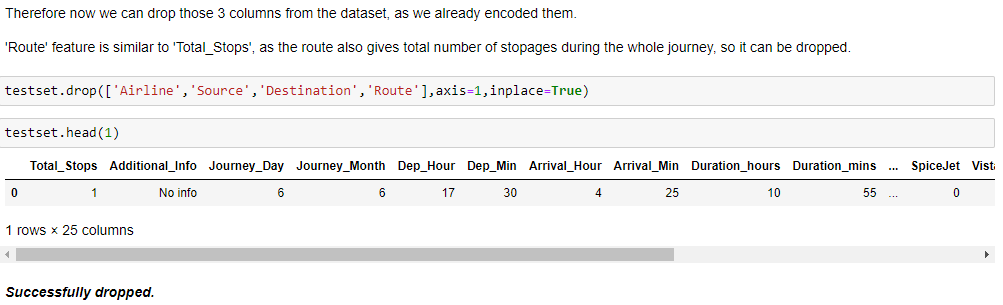
****

**Handling “Destination” column :**



**Now let’s merge all the 3 dataframes into our test dataset :**





**10. Statistical Information :**

***trainset.describe().T*** command is used to get all the statistical information about the dataset, like mean, maximum value, minimum value, count, standard deviation, 25th, 50th (median) and 75th percentile.

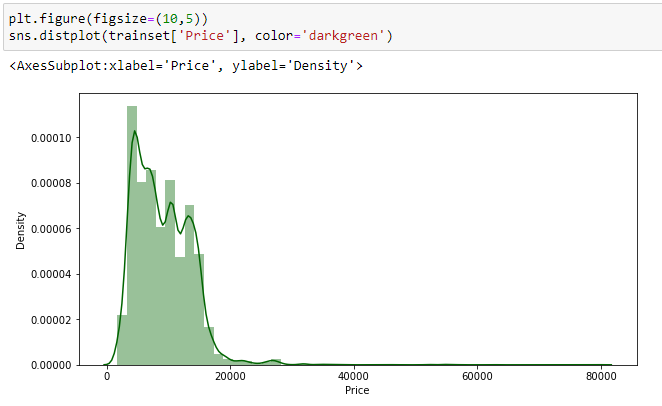
**Observations :**

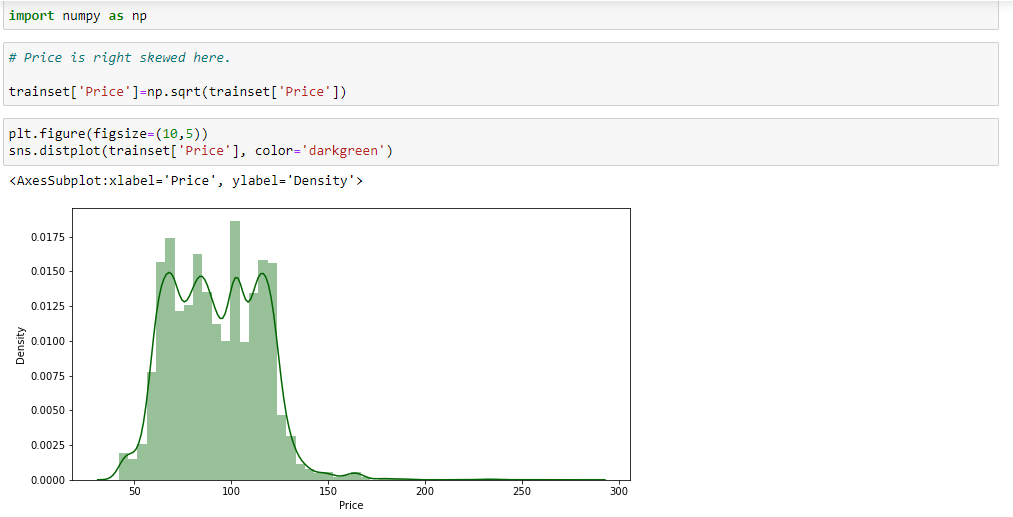
1. Most of the features have mean value greater than its median, which indicates presence of skewed data.
2. Some of the features have much difference between its 75th percentile and max value, which indicates presence of outliers.
3. Minimum price of a flight ticket is Rs. 1759 and the maximum price can reach upto Rs. 79512.

**11. Univariate Analysis :**

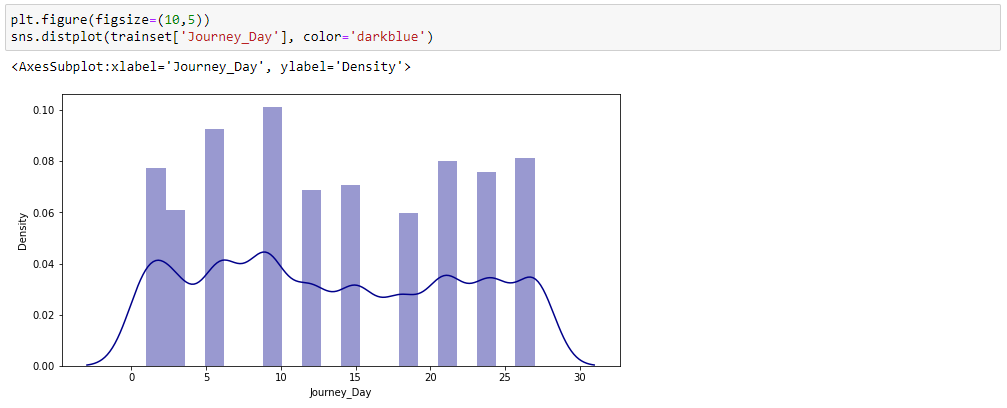
****

Most passengers prefer routes with only 1 stoppage during the journey.

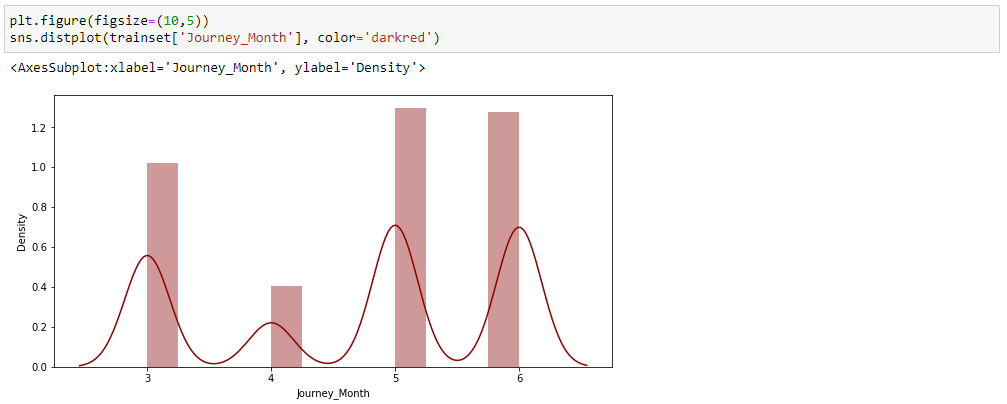




Skewness resolved upto some extent.

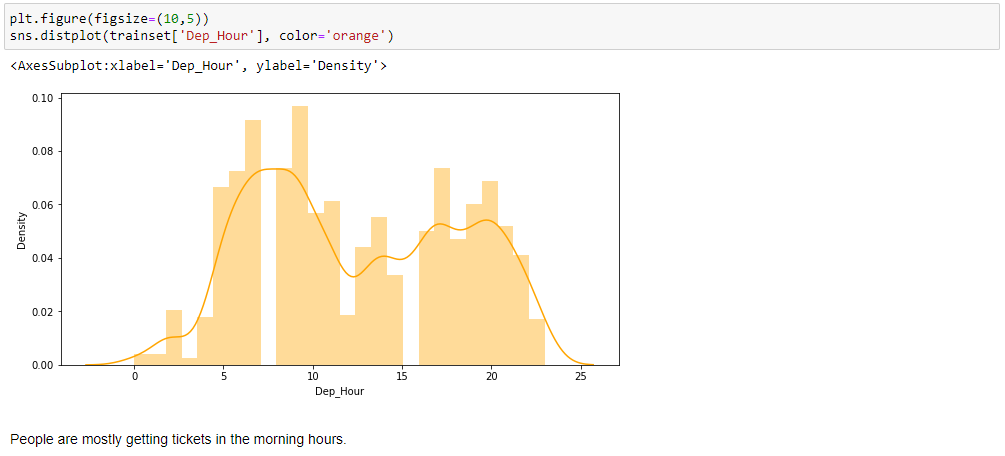


Distribution plot for “Journey\_Day”.



Distribution plot for “Journey\_Month”.

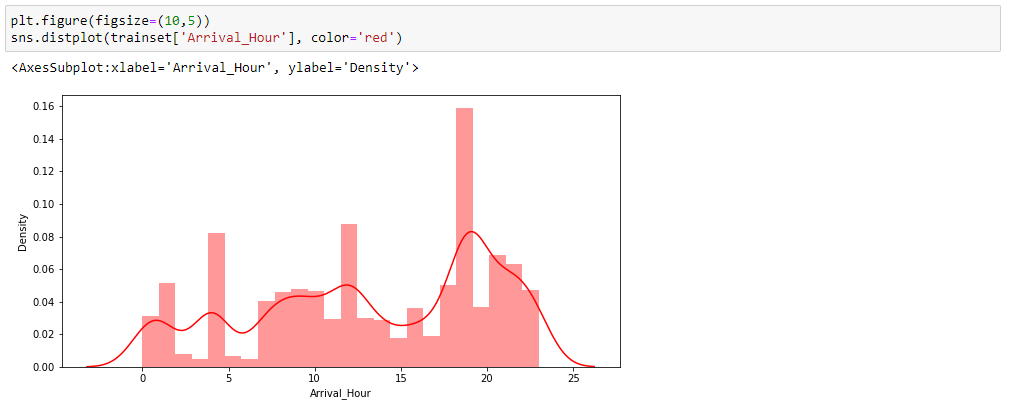
People are mostly travelling in the month of May.



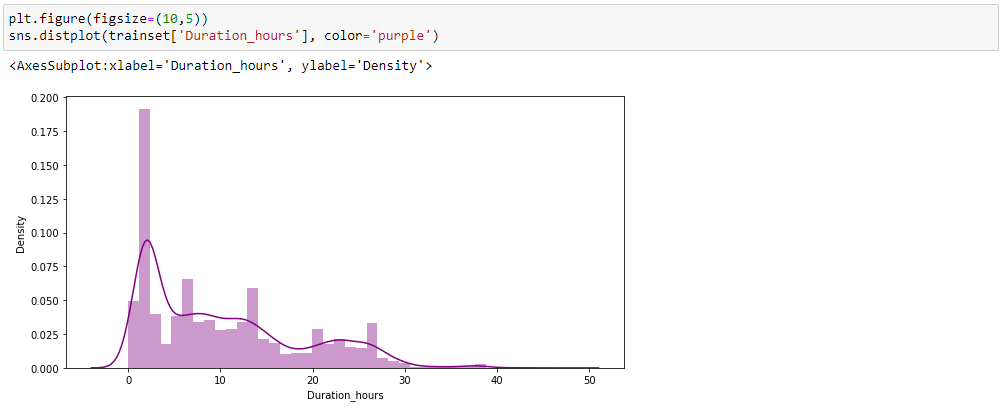
Distribution plot for “Dep\_Hour”.

People are mostly getting tickets in the morning hours.

Distribution plot for “Arrival\_Hour”.

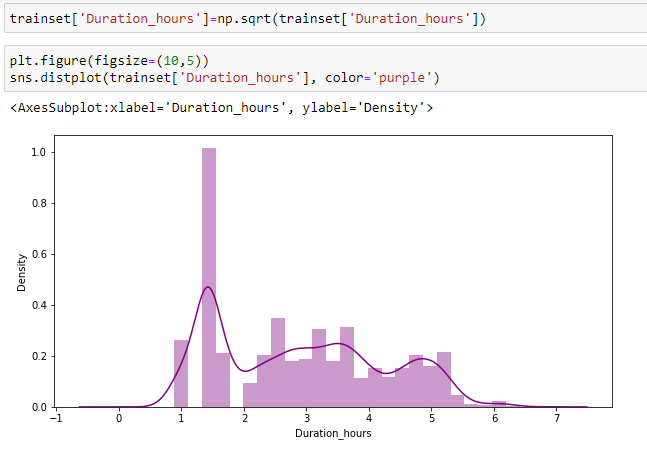


Passengers are arriving at late hours and early departures[as seen in the above graph], which indicates longer journeys.

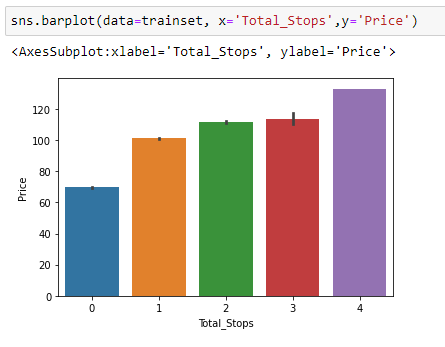


Distribution plot for “Duration\_hours”.

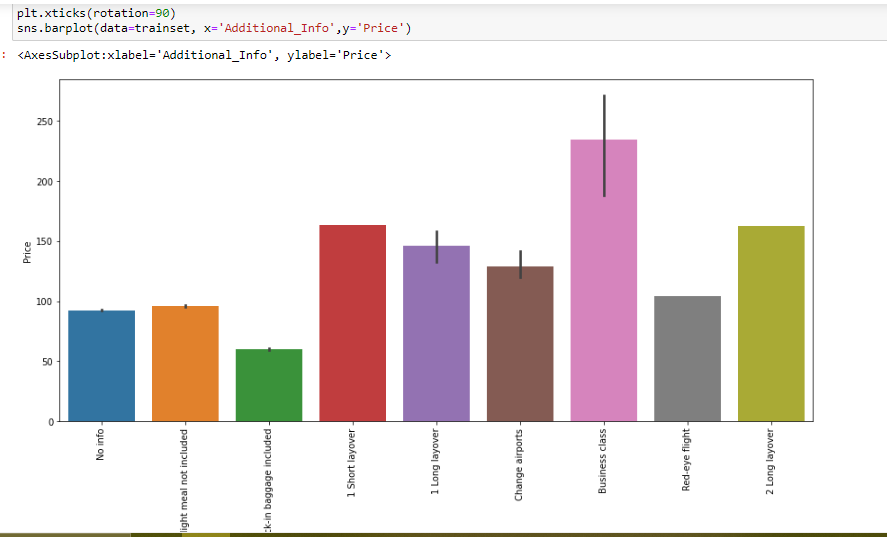
Here we can observe some right skewed data.



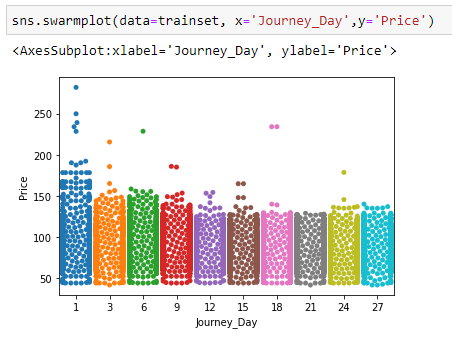
Skewness reduced to some extent.

**12. Bivariate Analysis :**

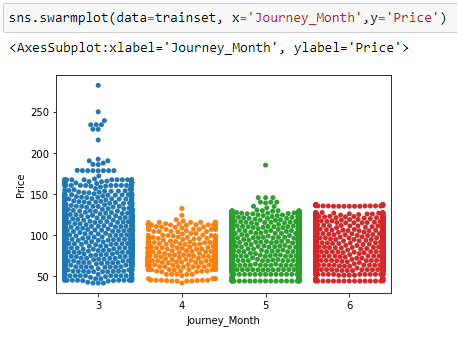
With the increase in layovers during the journey, price also increases. Price of tickets is lowest with nonstop flights.

****

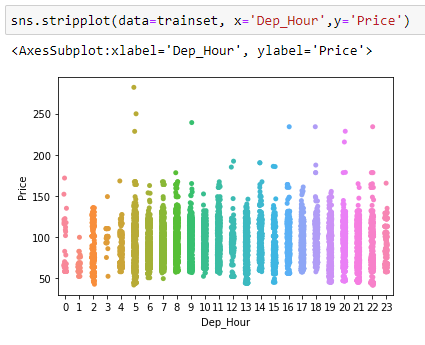
Business class tickets are having highest price.



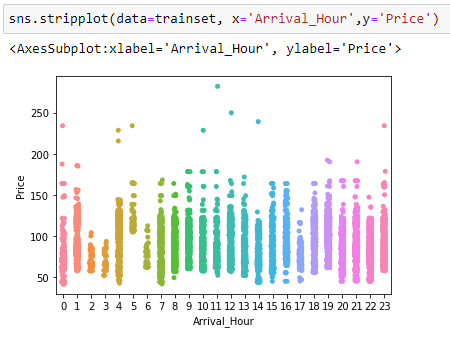
People are mostly travelling in the first week of the month and due to the spread of datapoints we can say that outliers can be present.



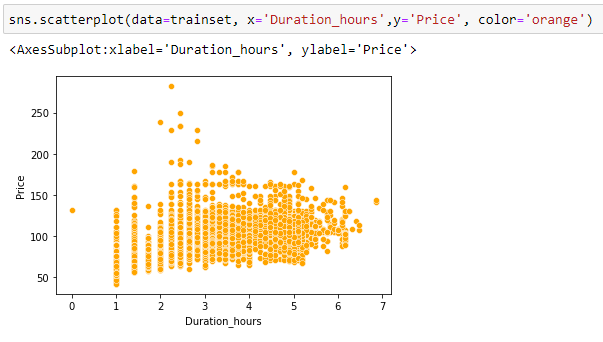
People are most travelling in the month of March



Price can be little less with early departure hours, and outliers can be observed here.



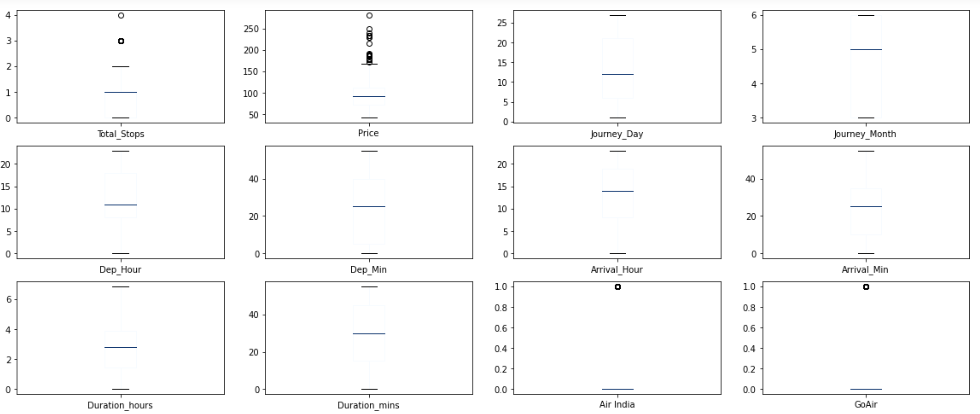
Price of the tickets can be less for early arrival hours. And outliers can also be observed here.

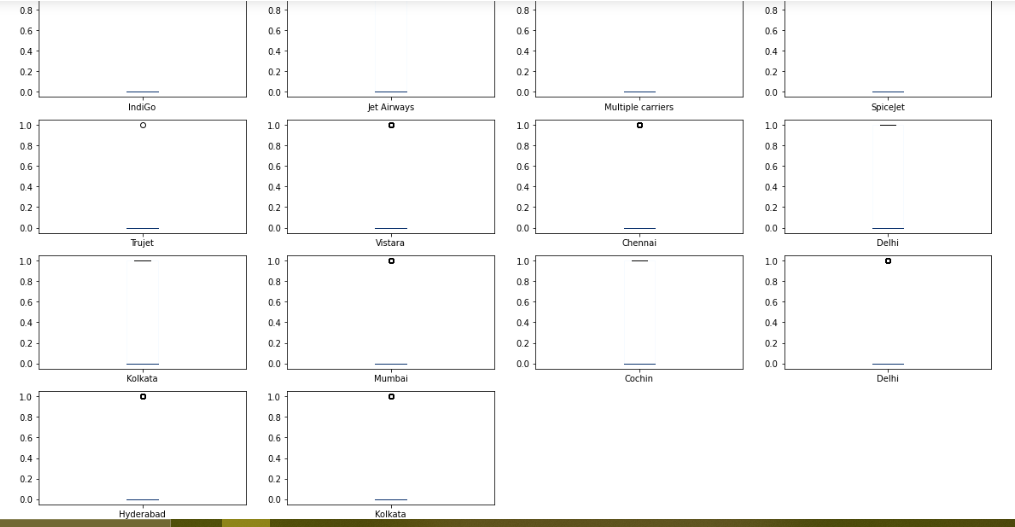


Mostly flight journeys takes 3 to 5 hours of duration.

**13. Detecting Outliers :**

1.png



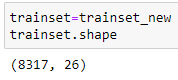


Through this visualizations, the outliers are clearly visible.

And we need to remove them.

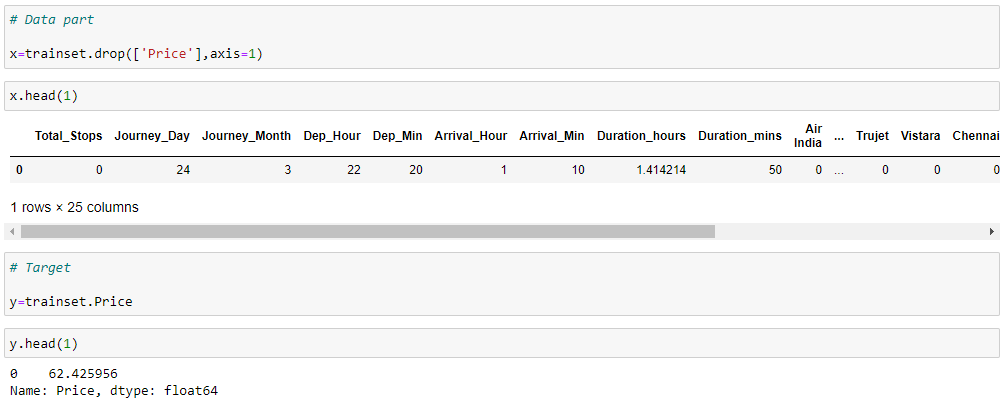


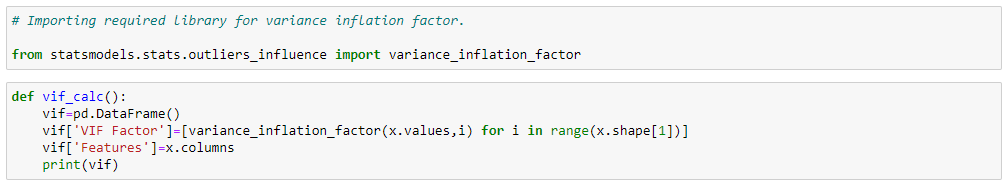
Here we can see the difference between our original df shape and df\_new shape, and the percentage of data loss. Much of the outliers are removed with 22% and we can't afford to lose more data and will proceed further with these values.

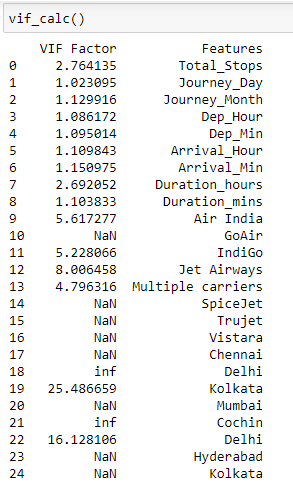
 Assigned the new outliers freee dataframe to the original one.

**14. Variance Inflation Factor :**

**It helps us to eliminate multi co-linearity.**

**First we need to split the dataset into data part and target.**

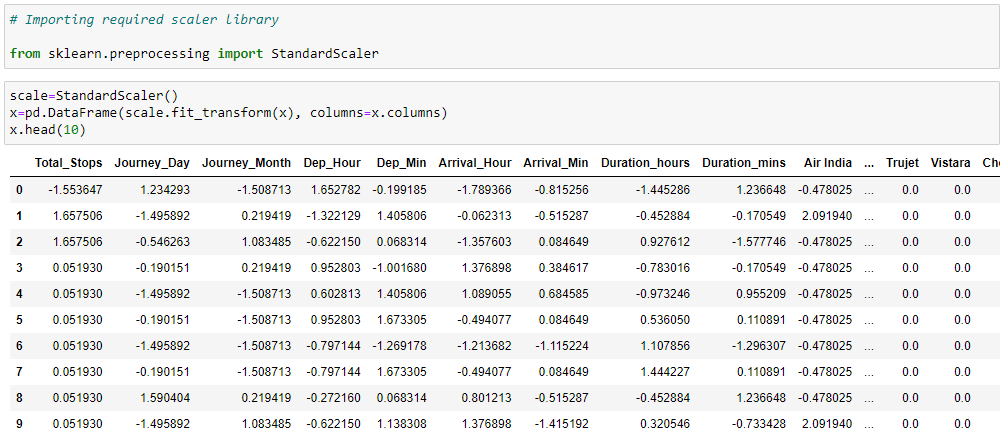
****



As we can observe here, the VIF factors are not balanced. We will recheck them after feature scaling and power transformation.

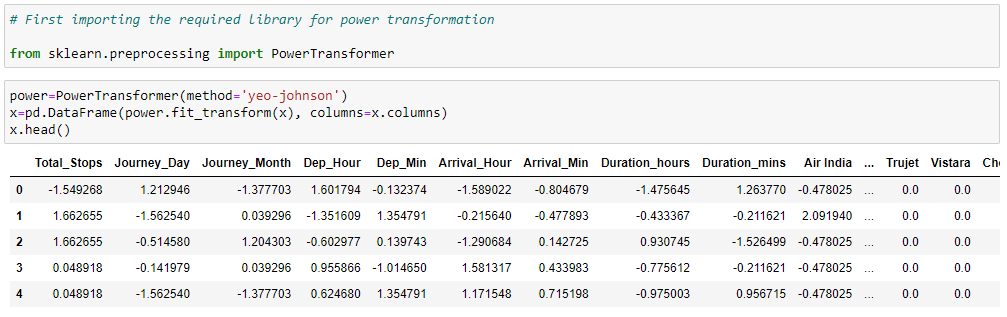
**15. Feature Scaling :**

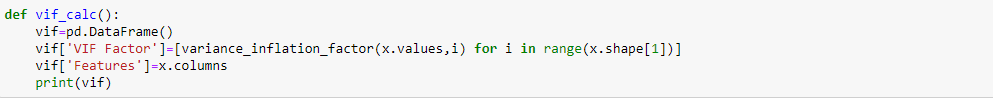
**To transform the dataset into same format, so that we have a uniform dataset.**

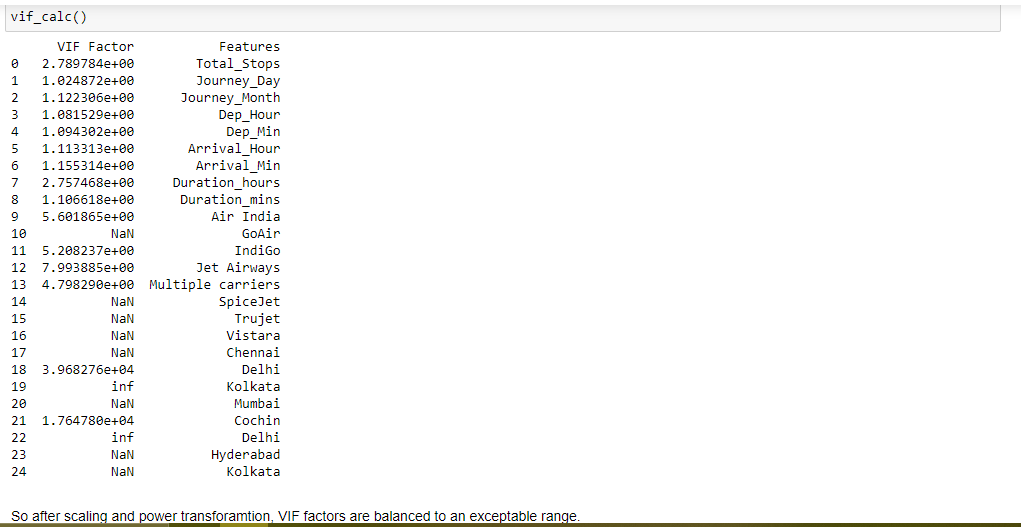
****

**16. Power Transformation :**

I will use Yeo-Johnson to transform this dataset so that the resulting features looks more normally distributed. And also reduce skewness and outliers.

****

**Rechecking VIF :**

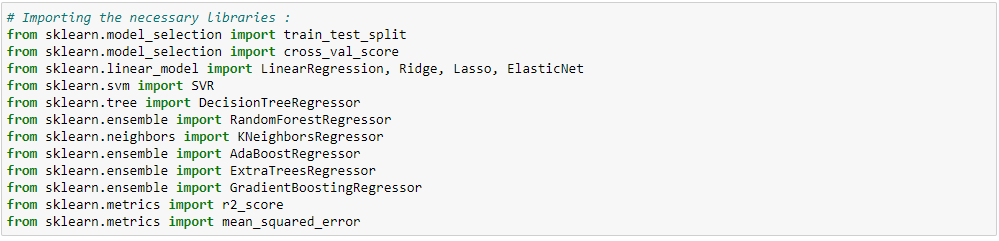


So, we are done with all the analysis and visualization, and now we can proceed with the machine learning models, testing, training and predictions.

**17. Machine Learning Models :**

Here we have a linear case, hence we'll go with Linear Regression and various other regression models.

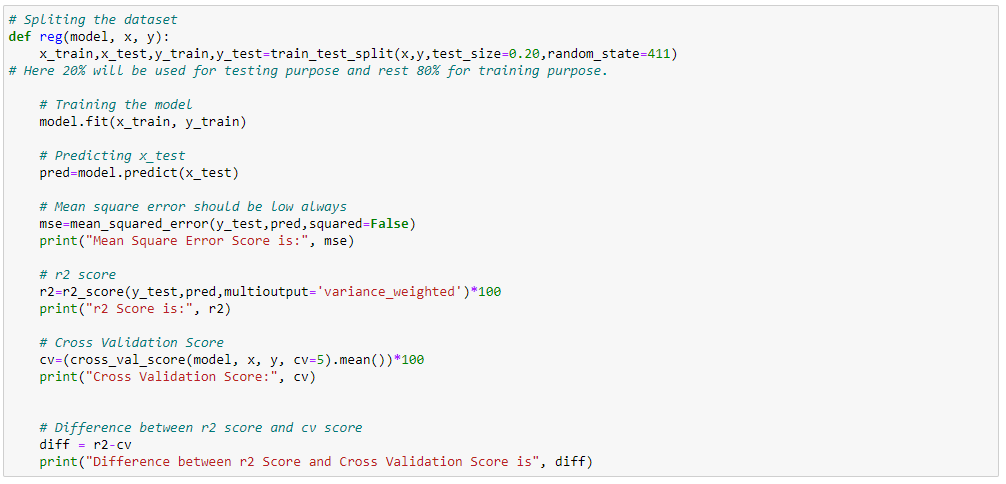
Firstly starting with importing the required libraries :

****

Next we will find the best random state using for loop, to use in our models :

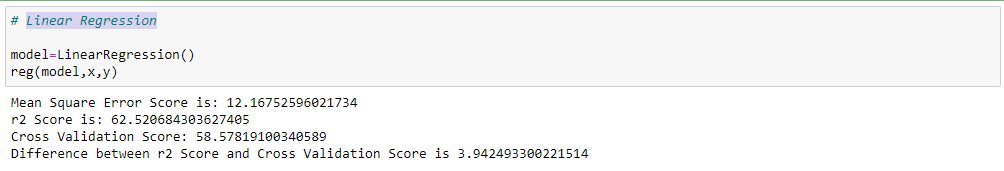


Then we will create a function containing all the required evaluation metrices :

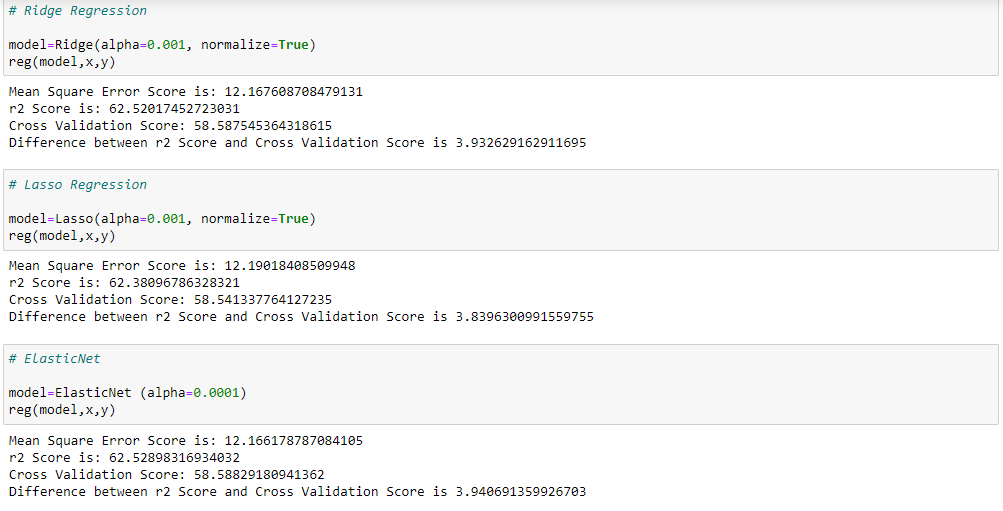


Now we can call all the models one by one using this function :

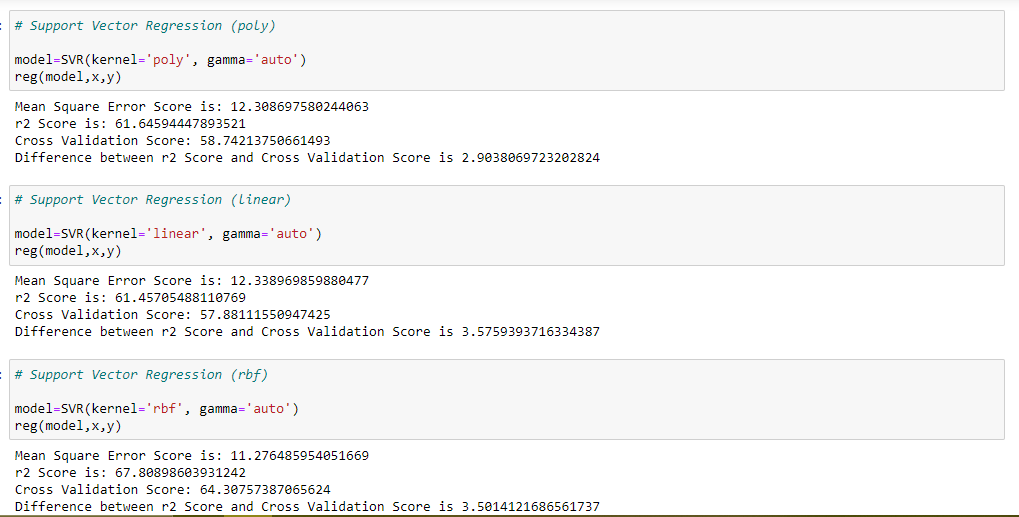
**Linear Regression Model :**

****

**Ridge Regression, Lasso Regression and ElasticNet Models :**

****

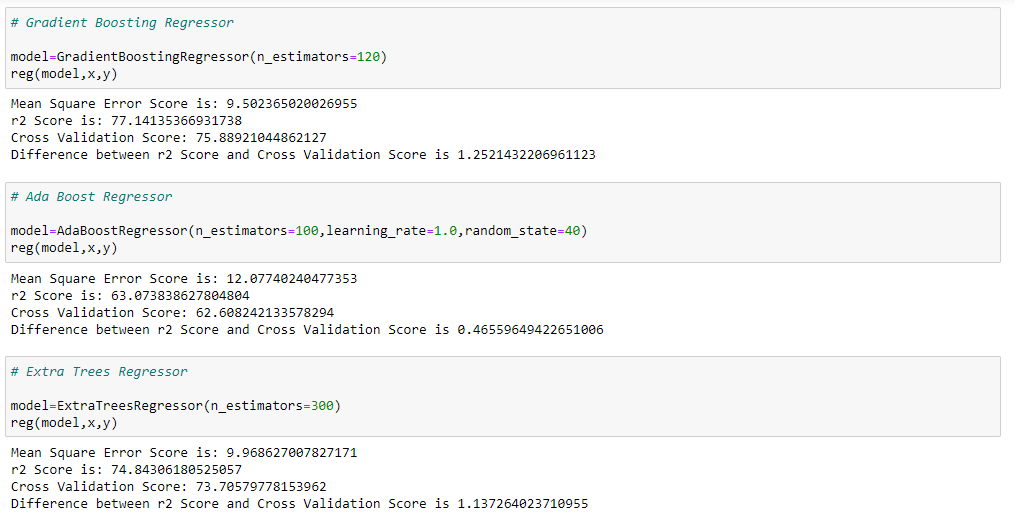
**Support Vector Regression(poly, linear, rbf) :**

****

**DecisionTree Regressor, RandomForest Regressor and KNeighbors Regressor :**

****

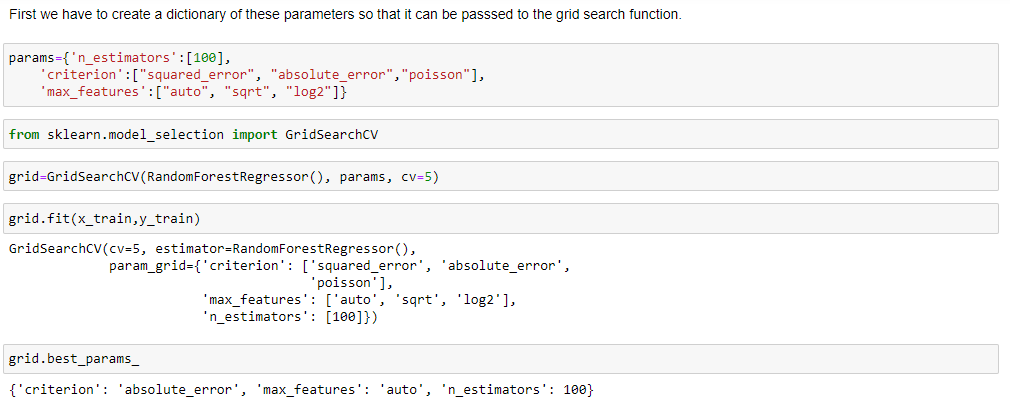
**GradientBoosting Regressor, AdaBoost Regressor, and ExtraTrees Regressor :**

****

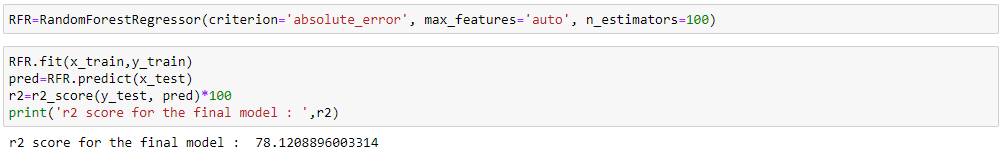
After finding all the scores of various models, we found that Random Forest Regressor model gives the highest r2 score, and less difference between r2 score and cross validation score. Hence we choose this model and proceed further with the process.

# 18. Hyper Parameter Tuning :

Applying hyper parameter tuning using Grid Search CV method, to find the best parameters of Random Forest Regressor Model.



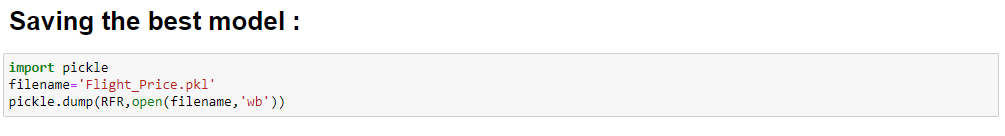
So here we have found the best parameters for our model, and now we can finally train our model.

**

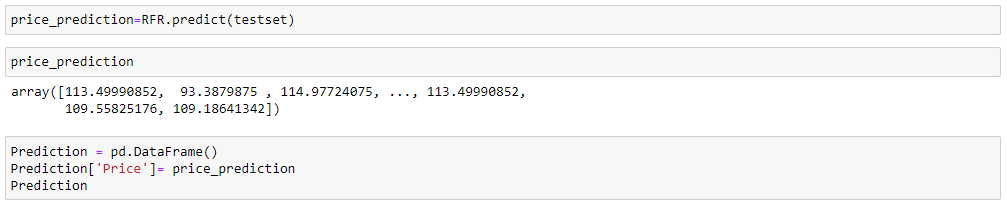
So this the best model we have achieved, by doing all the analysis and training and testing.

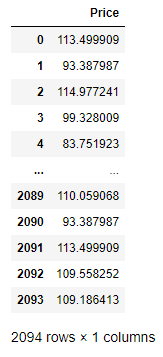
Next we will save this model using pickle.

**19. Saving the best model :**

****

**20. Prediction using Test dataset :**

****

Predicted Values :

**Conclusion Remarks :**

In this blog, we saw how to apply our analytical skills, visualization and machine learning models, to predict the price of flight tickets.

In every data science project, EDA process, feature engineering, visualization play a very important role. That’s how we can clean tha data and then ultimately train our macine learning models, to derive the required predictions.

In the first part of the process, we cleaned the date time columns which was very crucial, because it was not in the format that ML algorithms can understand.

Then we also dropped some extra columns. We converted the object columns into numeric ones, because ML models can only understand numeric values.

Feature scaling and power transformations play a vital role, to transform the whole dataset into an uniform format. It also helps us in removing skewness.

Then finding the best random state and training our models. Used hyper parameter tuning to find the best parameters for our model and improving the accuracy.

Hence, using this Machine Learning Model, we people and the airlines can easily predict the flight ticket price, and plan their happy journey.

