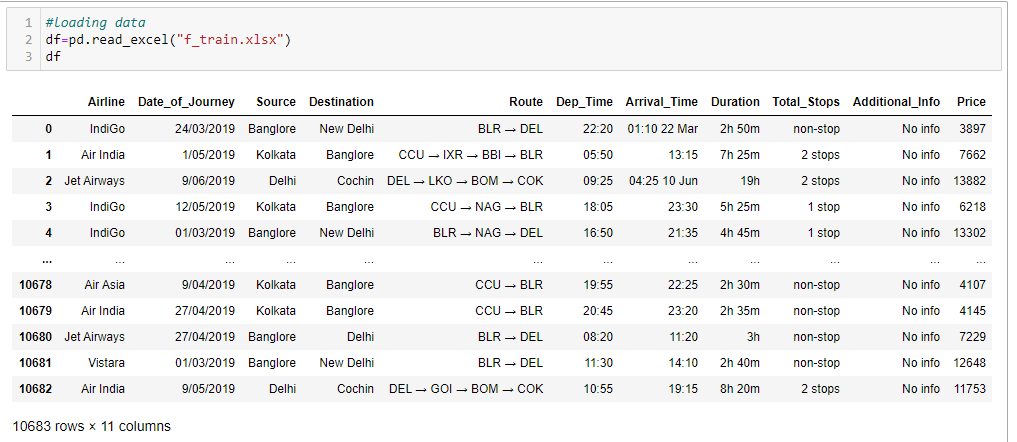
**Flight Price Prediction**

**Introduction**

At the time, the invention of the airplane was the fulfillment of a dream. "If I were a bird…" was the very first fantasy of human beings. Even more than providing unprecedentedly rapid delivery of passengers and goods across the globe, developing multiple industries to build, support, and market this enormous economic sector, and inspire billions to dream and achieve more. Nevertheless, all this cost is also a factor, as we know air travel is one of the expensive means of transportation. Flight ticket prices are something hard to guess. Today we might see a price; check out the same flight price tomorrow; it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable. The price of an airline ticket varies on several factors, such as ﬂight distance, purchasing time, airline, and many more. Since the deregulation of the aviation industry, airfare pricing strategy has developed into complex rules and mathematical models that drive airfare pricing strategies. Although still primarily held in secret, studies have found that these rules are widely known to be affected by various factors. Traditional variables such as distance, although still playing a signiﬁcant role, are no longer the sole factor that dictates the pricing strategy. Elements related to economic, marketing, and societal trends have played increasing roles in dictating airfare prices. In this paper, we will discuss the process of Flight price prediction with the help of machine learning.

**Dataset:**

The data here is originally is provided by Kaggle. The dataset contains information about the Indian domestic flight of various airlines between March and June of 2019 and between various cities.

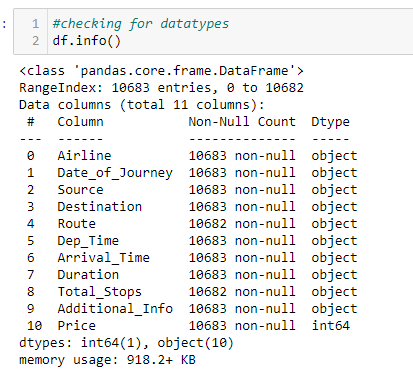


As we can see, the data contains 10683 rows and 11 columns in total. The information this dataset contains is:

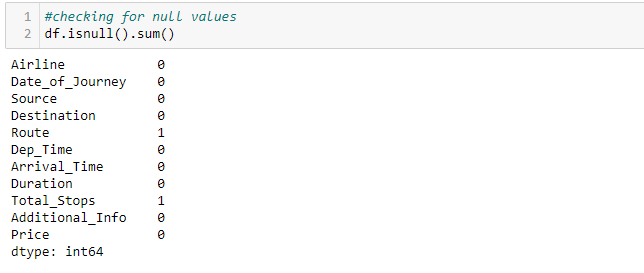
* Name of the Airline
* Date of journey
* Source (take-off city)
* Destination (touch-down city)
* Route
* Departure Time
* Arrival Time
* Duration
* Total Stops
* Additional info
* Price

**EDA**

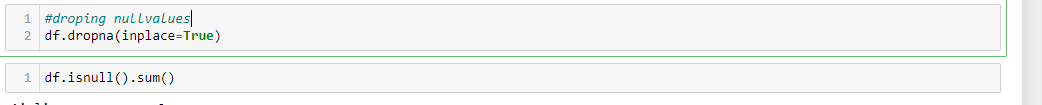
it is a good practice to explore the given data set with various exploratory methods, specifically when they can be compared. The objective of EDA is to obtain confidence in your data to a point where you are confident enough to engage with a machine learning algorithm. The first thing to check in the data given is the type of the given variable.



As we can see in the image the all the columns are object types except price. Also, the output indicates that there are some Nan values in the dataset. So next, we will check for null values and try to rectify them. For this we will use "**isnull().sum()”** function.



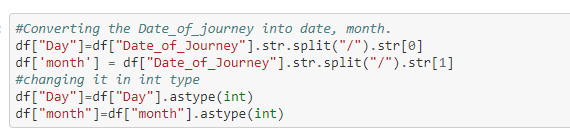
The image shows that there are Nan values in two columns (Route, Total\_Stops), and both the columns contain a single Nan value. Since the Nan value is nearly nil compared to the dataset, we will drop the Nan value with the "**dropna()** “function.



The function will drop both the Nan value and our dataset is now Nan-free. As we got the basic idea about the given dataset, we will proceed with the Data Preparation part as necessary for the analysis and model building part. The more the data is clean, our model will perform more efficiently as we can see that the given data is not clean. For example, the data which should be float or int type is object type (Date\_of\_Journey , Dep\_Time, Arrival\_Time, Duration, Total stops). Also, we need to check Additional info to find out what all information we can extract from the given column.

**Fixing Date\_of\_journey**

As we are aware that the data given here is from March to June in 2019. We need to convert it into numerical data to analyze better and feed it to the training model.



As shown in the image here, we have used the "split" method to extract month and day detail and created two new variables (month, Day). We did not use the command in the year because the data provided for 2019 only. It won provide any insight. After this, we will not need the Date\_of\_journey column, so that we will drop that column.



**#Fixing arrival and departure time:**

In the departure and arrival data, the time is in 24-hour format. Also, in some of the columns of the Arrival date, there is a date mentioned. To clean this data, we will create a function that will convert the data four times (morning, afternoon, evening, midnight). By this process, we can analyze the arrival and departure times more efficiently.



The function above with taking the first two values of the variable will convert those in int type data format. As the data currently is an object type. Then given the conditions, it will convert the data into Morning, Afternoon, evening, midnight format. We will apply this function to Dep\_Time and Arrival\_time and create two new columns Departure\_time and aril\_time, respectively. For this, we will use the ".apply” function.



**Treating Total\_stops column:**

As we have managed to clean departure and arrival time. Now it is time to fic Total\_stops columns. As we can see, Total\_stops are a numerical representation of the Route variable. However, Total\_stops is also an object time and contains unwanted data. To fix this, we will create a function that will filter the numbers of stops a plane has made during the flight. The minimum number of stops is zero, while the maximum number of stops is 4.

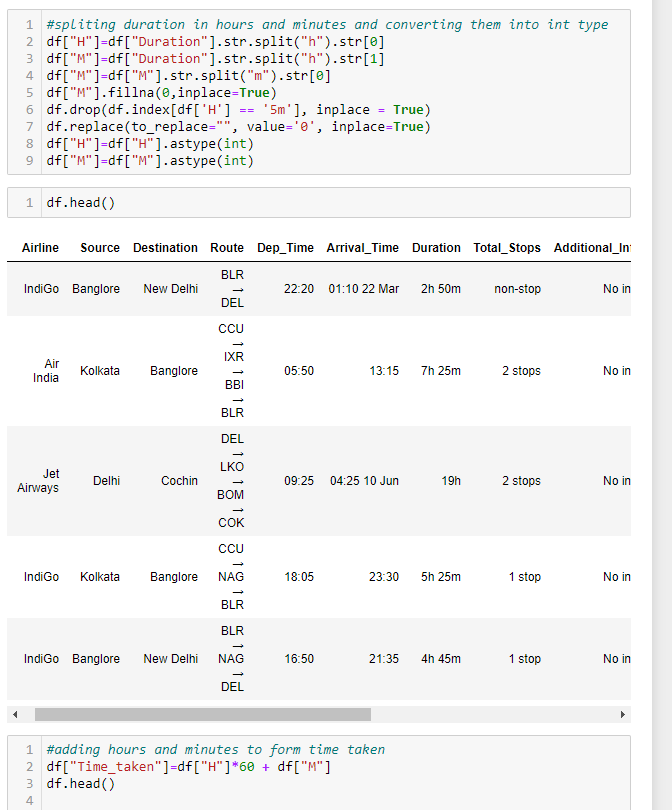


Further, we will apply this function on the Total\_time variable, and with the help of the ".apply()" function, we will create a new variable, Stops.

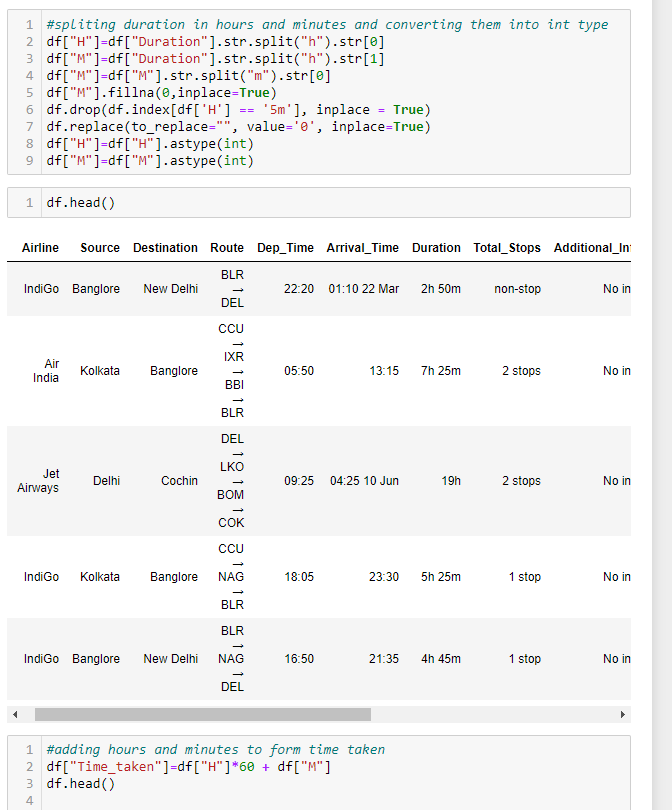


**Fixing Duration:**

Duration is the variable that explains the total time taken during the flight. In the given data, the duration column is also the object type. The data is in an hour and minute format. We will use the split method to split the hour and minute data separately and convert them into integers.



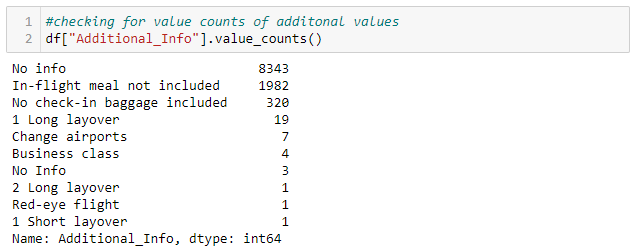
We have used column “H” to store hour data and column “M” to store minute data. Also, we have to keep in mind that there is only one hour in some of the data provided in the duration column. So that will create Nan values in the "M" columns. We will replace those Nan values with zero as the duration was in hours only. Further, a column in "H" contains a value "5m," which is impossible for the Hour column, so that we will drop this row. Then we have converted the "H" and" M" columns into int type. Now we need to convert hour data into minutes and add this to minute columns to know the total time taken by the flight in minutes.



Here we have multiplied every row of the H column with 60, then added it with "M" columns and stored that in Time\_taken columns to have total time in a minute.

**Fixing Additional\_Info column:**

The data is an object type; first, we will analyze how much unique value these columns hold and what sort of information it carries. We will then decide how to proceed with the data. For this, we will use the value count method.



This column contains information about the inflight meal, baggage checking, layovers, class. However, more than 80 percent of the data of this column is Nan. So, we cannot consider this column; we have to drop this column.

This is the final step of the Data preparation; we have cleaned all the data, now we need to drop columns which is of no use. For this, we will use the "drop" function.

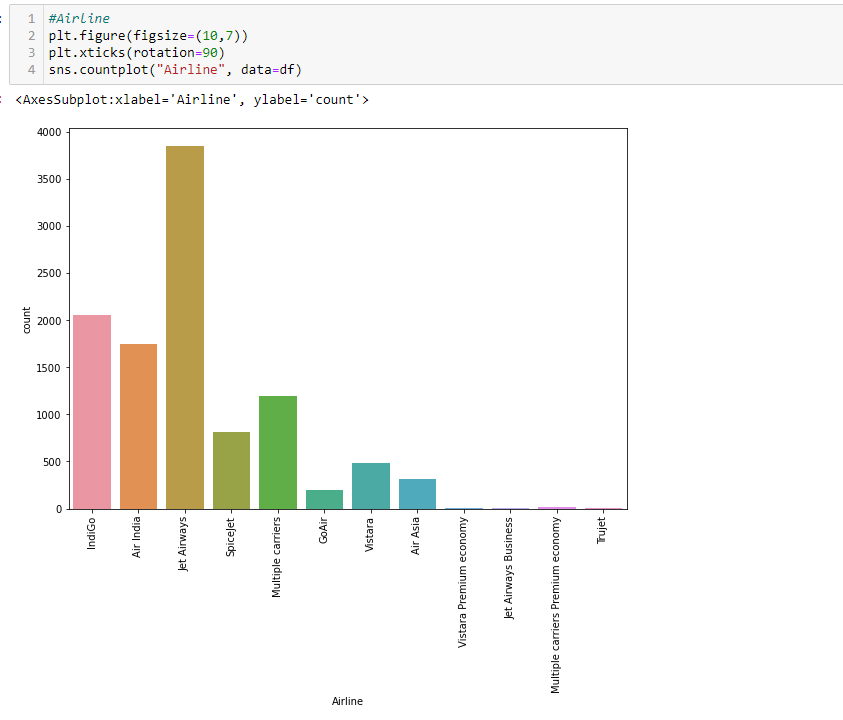
df.drop(['Route','Dep\_Time','Arrival\_Time','Duration','Total\_Stops','Additional\_Info','year','H','M'],axis=1, inplace=True)

**Visualization:**

As we have clean data now, we can work on the visualization part to gain insight into the data.

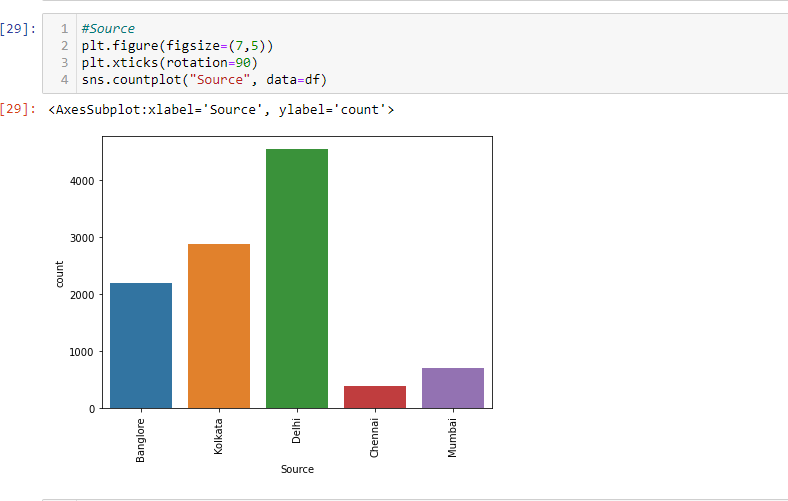
Bivariate analysis:

#Airline



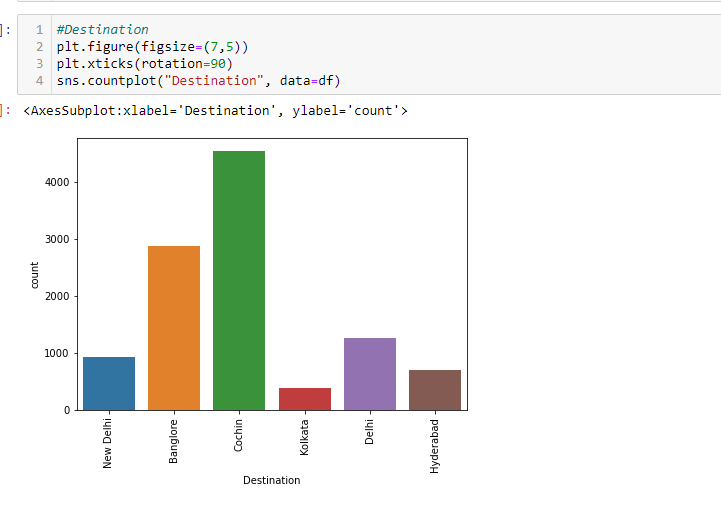
The image above shows that Jet airways are the most used airline among all the airlines. Indigo and Air India have also covered a good number of flights. However, Trujet, Go Air, Air Asia is the most preferred airlines. Their premium and business class is also the least favorite among passengers.

**#Source**



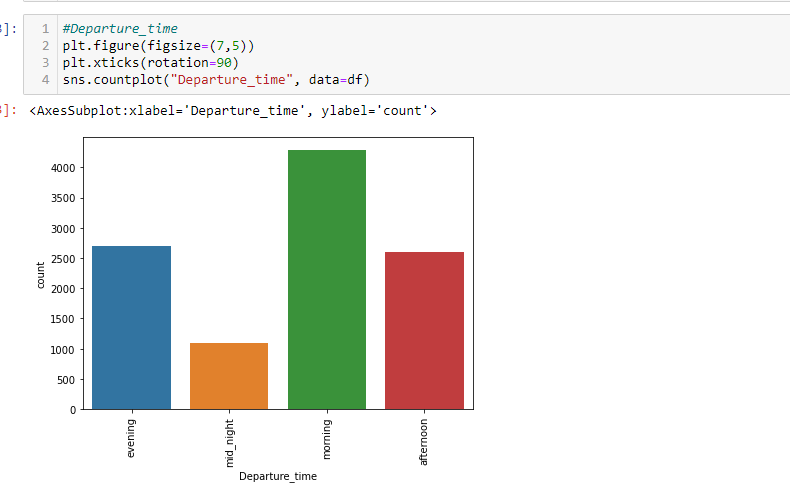
The majority of the flights' take off from Delhi, while Chennai sees the least number of take-offs.

**#Destination**



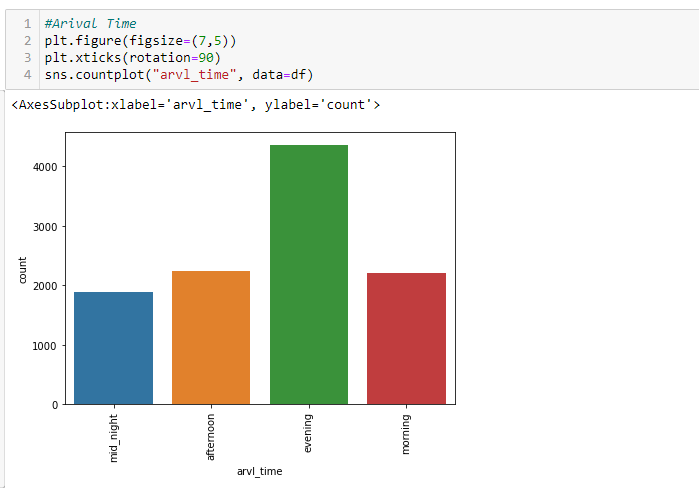
The majority of the flights touch down at Cochin, while Kolkata has seen the least number of touch-down. Also, there are two airports mentioned here Delhi and New Delhi.

**#Departure time**



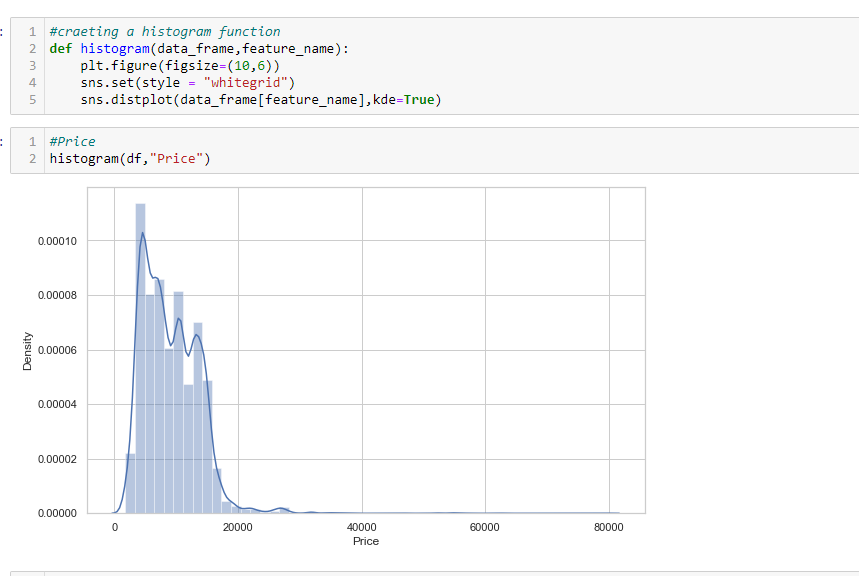
The majority of the flights' take off in the morning, while midnight sees the least number of flights.

**#Arrival time**

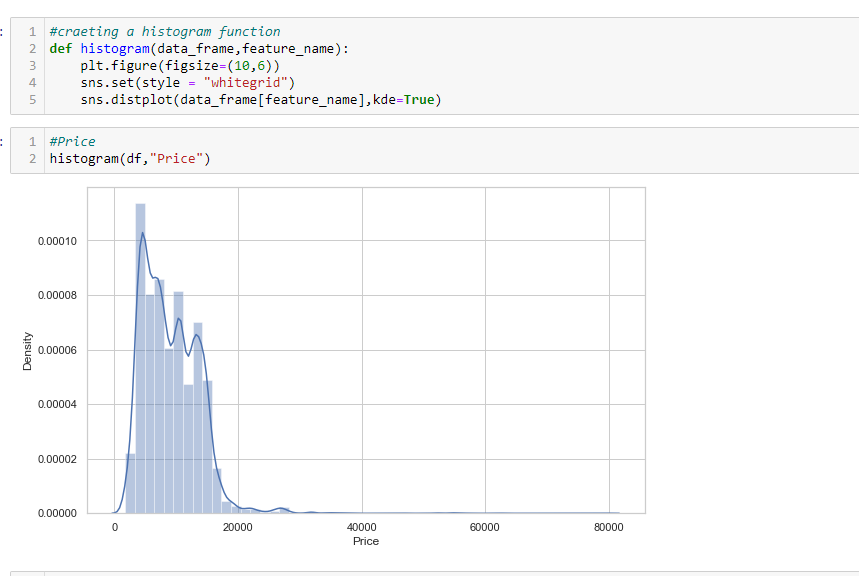


The majority of the flights touch down in the evening, while others see a comparable amount of touch down.

To analyze the int and float variable, we are creating a histogram function using a distplot.

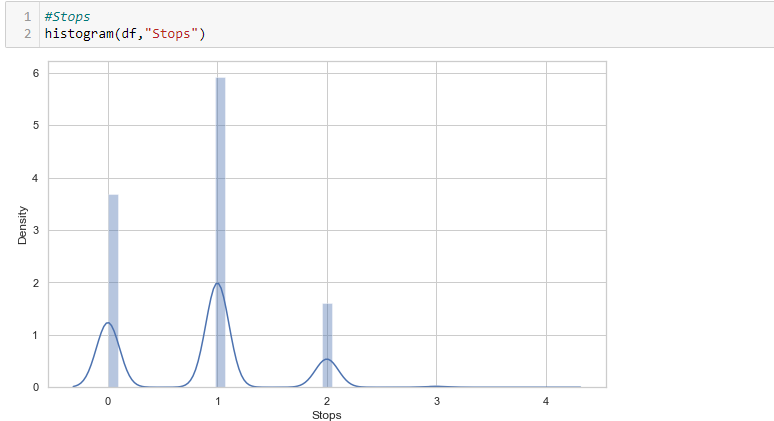


**#Price**



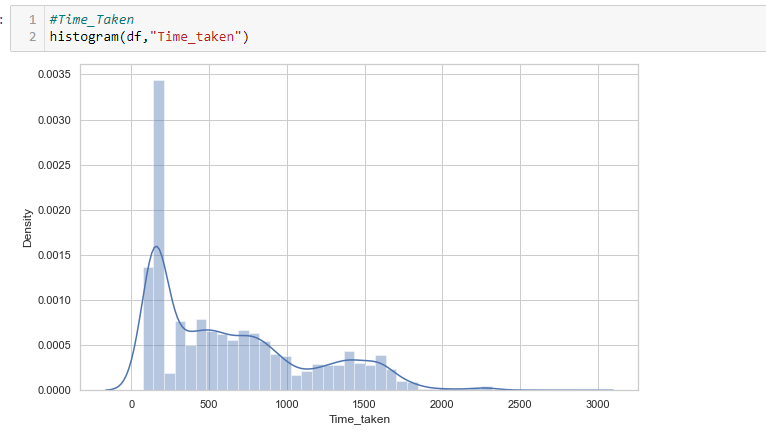
As the majority of the prices are under 10k, there are few prices which more than 80k.

**#Stops**



The majority of the flights take one stop. However, the frequency of 3 and 4 is shallow.

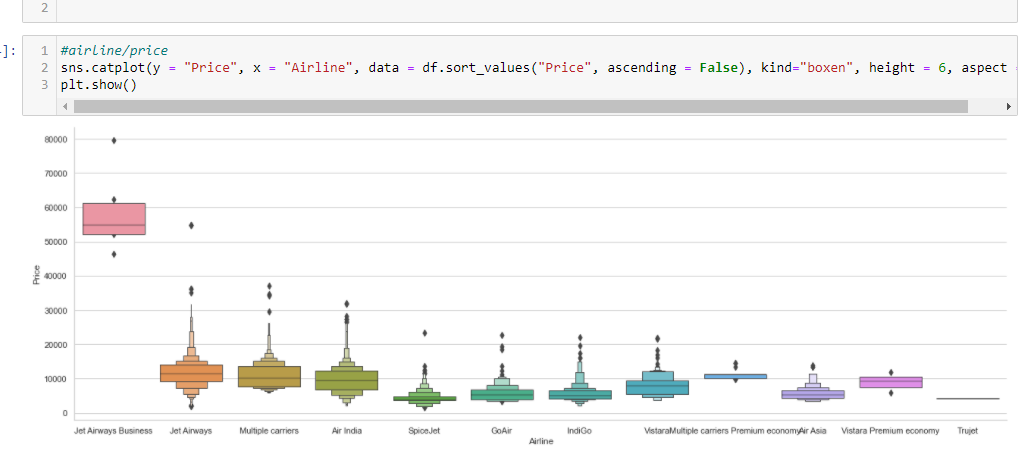
**#Tme taken**



The majority of the total flight time taken is around 2-3 hours; however, there are very long layovers on some of the flights.

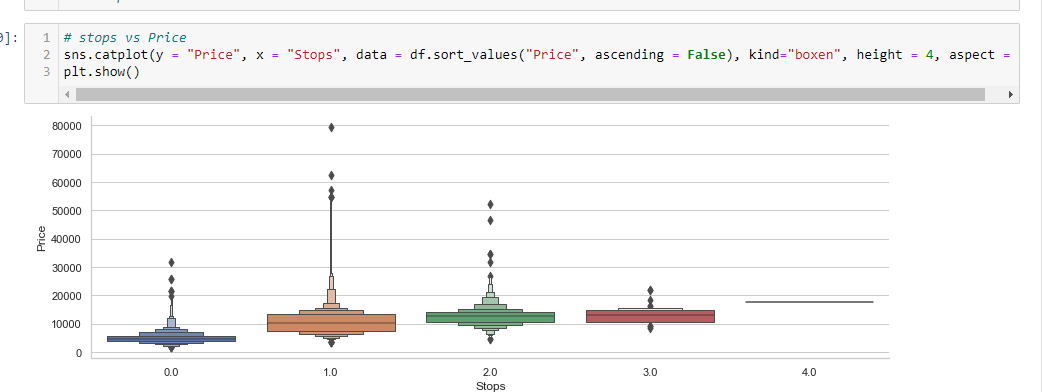
**Bivariate analysis:**

**#Airline/Price**



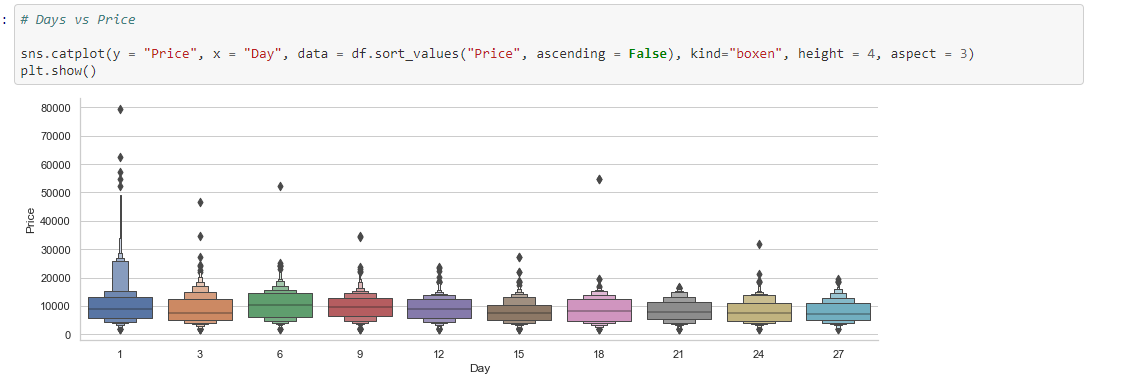
The business class of jet airways has the highest price. Also, the price of jet airways standard ticket is high in comparison to other airlines. The price of Trujet is the lowest among all.

**#Stops/Price**



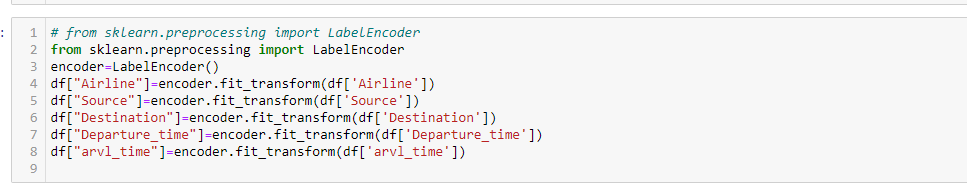
The stops and price are showing a Linear correlation. As we can see, the prices are increasing with an increase in the number of stops.

**#Days/Price**



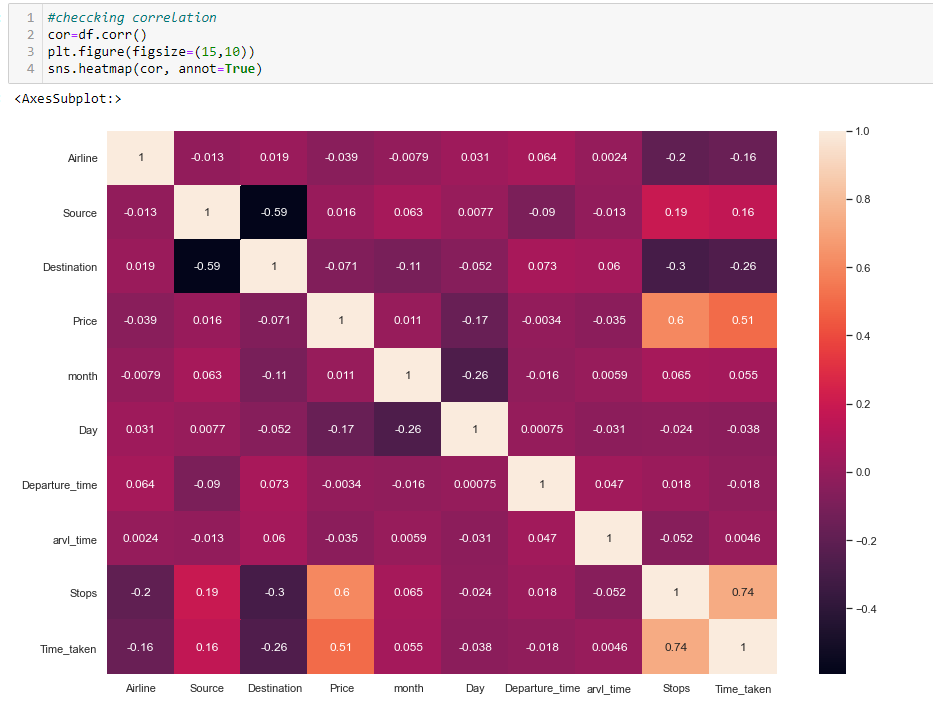
The prices are high at the beginning of the month. Also, we can see considerable outliers in the 1st,3rd, and 6th Days. So, we can say people prefer to travel in business and premium class at the beginning of the month.

As we completed the analysis part, It is time to go for feature selection. We will use the correlation method to get the importance of the feature concerning the dependent variable. So, before proceeding, we need to encode all of the categorical data.



Here we have used LabelEncoder to transform the value of the given variables.

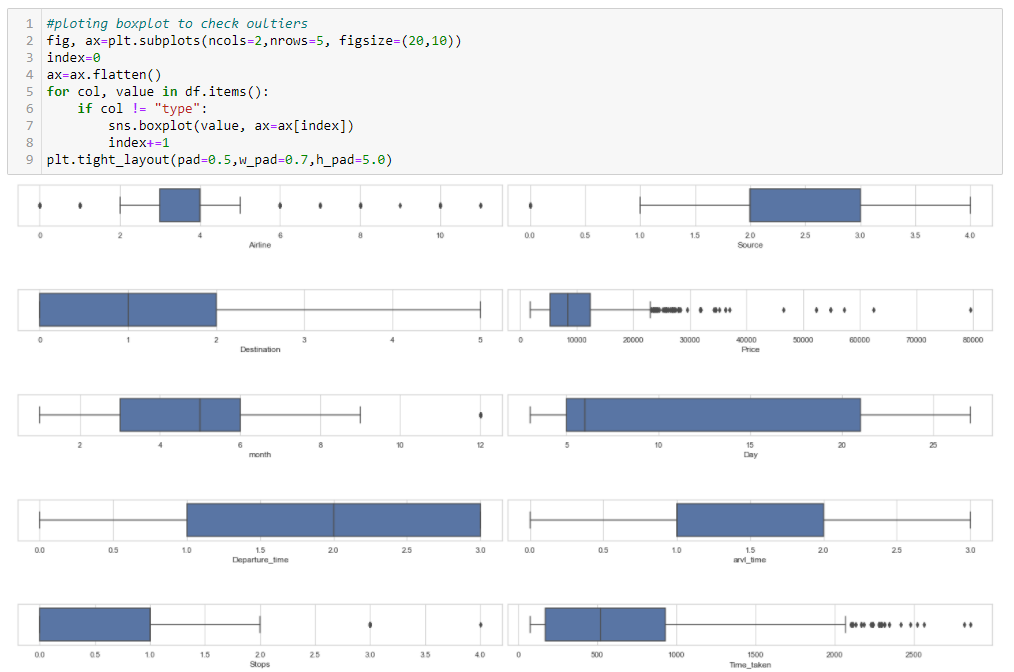
**#Correlation/Feature selection**



We can see price is highly correlated with stops and time taken while most negatively correlated with Day. Stops are also highly correlated with the time taken. Source and stops are also correlated. We will keep all the values; if the prediction is good, we will not drop any column. If the model does not turn out well, we will drop departure time as it correlates approx to zero.

**Checking outliers:**

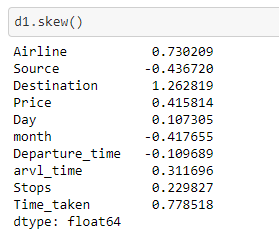
The next step is to see whether our data contains outliers or not.



There are outliers in Price and Time taken. The airline is our categorical variable though it shows that it contains outliers. It is nothing to worry about. Now to treat this, we will use the Zscore method. We need to apply this step before splitting the data because this process removes the row which contains outliers.



As we can see, we have used three as the threshold value; beyond that, all values will be treated as outliers. Further, before feeding the data for the prediction, we need to check the skewness of the data and treat it.



We will consider any value as skewness that is more than 0.5. As we can see, there is skewness in some columns. Price is our dependent variable, so we will not be changing anything in that column. So before proceeding further, we need to data in independent(x) and dependent(y) datasets.

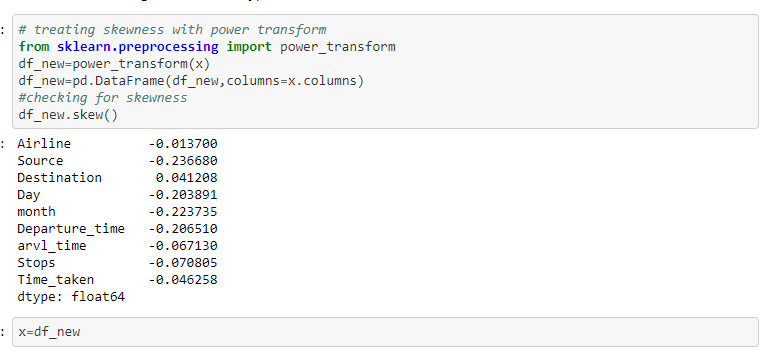


Except for the price, we have stored all the data in x.



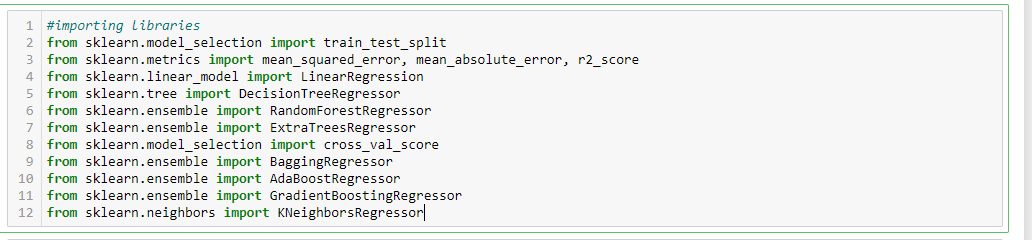
We have stored price in y.

Now we will treat skewness with the help of the PowerTransform method.

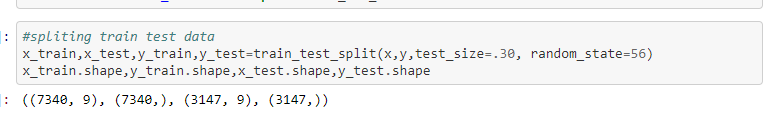


All the variable has skewness less than 0.5 which implies the problem of the skewness has been treated. Now we can proceed with the training part.

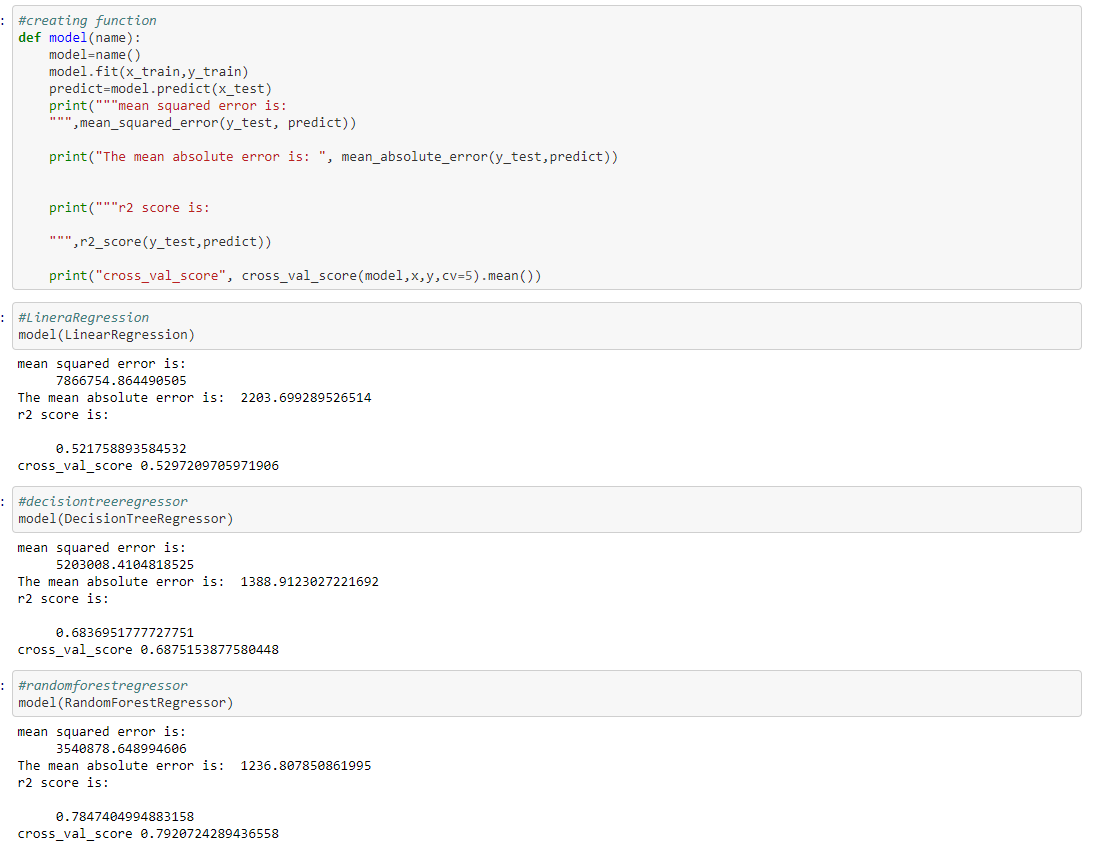
Important libraries for the model building



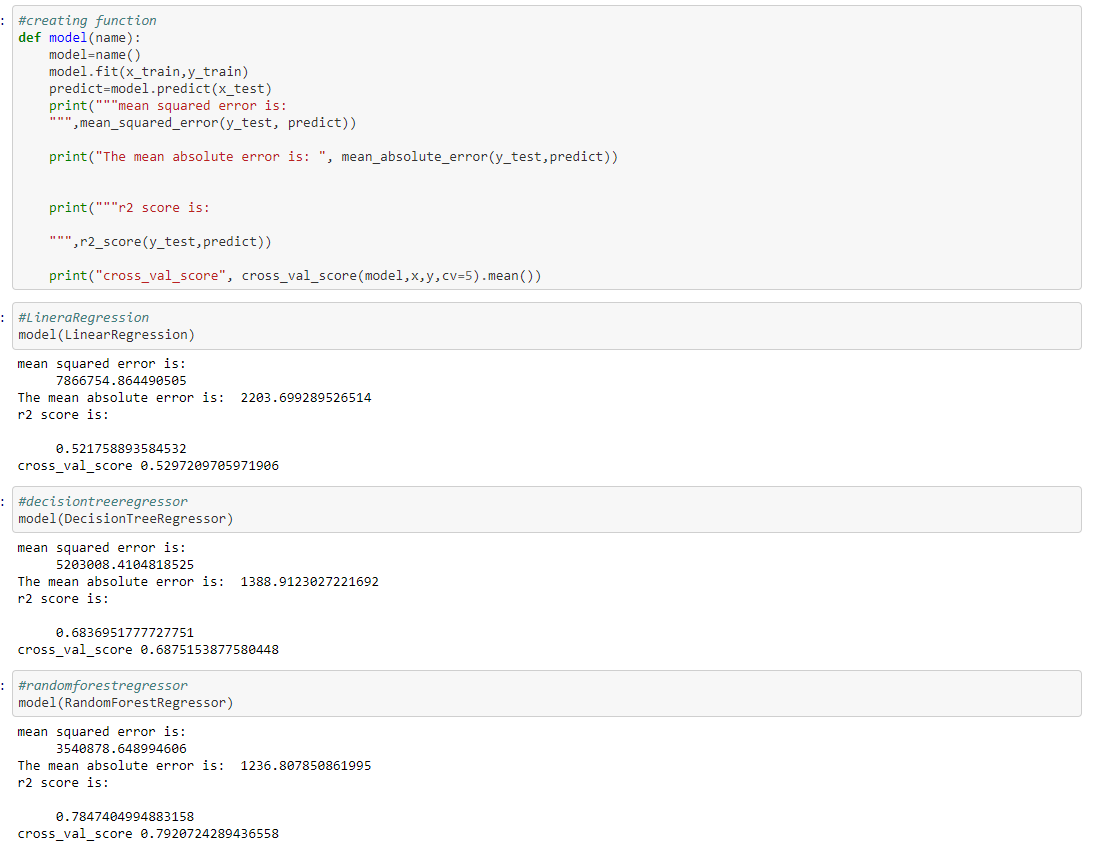
Apart from the model and train\_test\_split, we have imported some metrics to check the effectiveness of our model. Here we are using *the r*2 score, mean squared error and mean absolute error. Also, we are using cross\_val \_score for cross verification to detect any underfitting and overfitting in our data. Now we need to split our data into train and test variables.



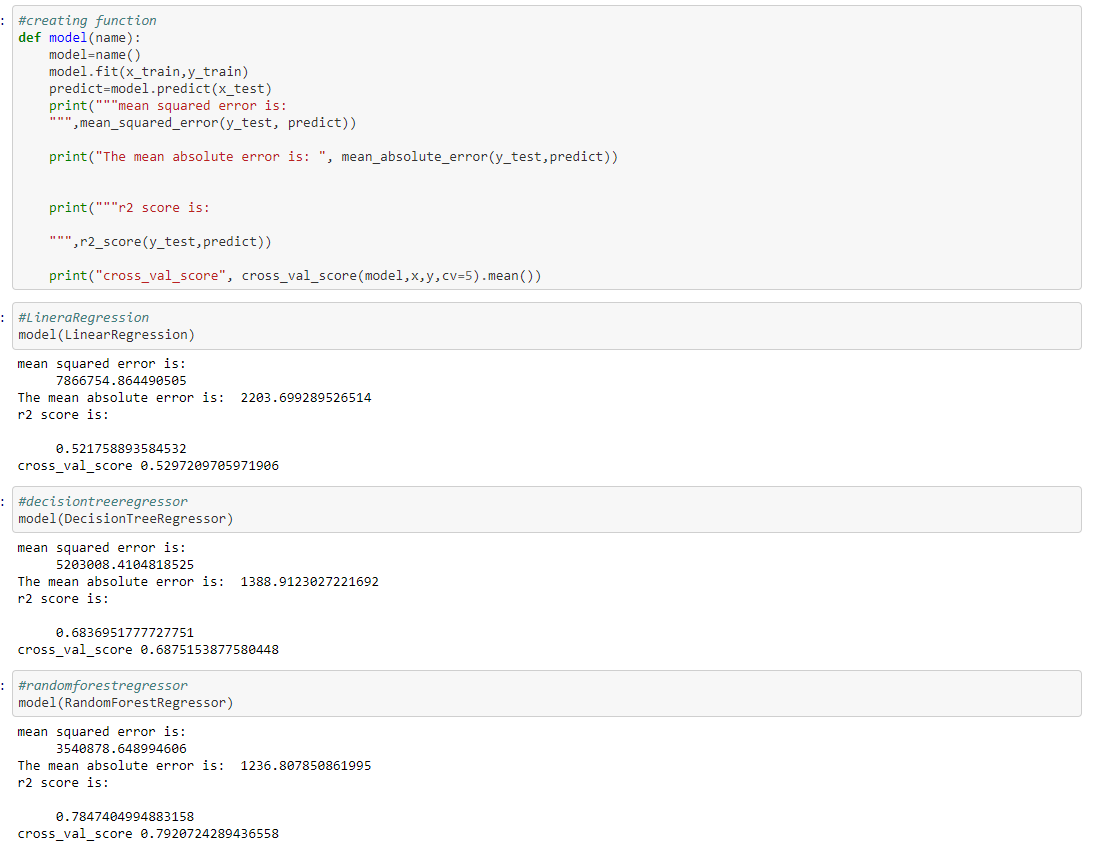
As we can see in the image, we have 7430 rows for training the model and 3147 rows for testing the model.



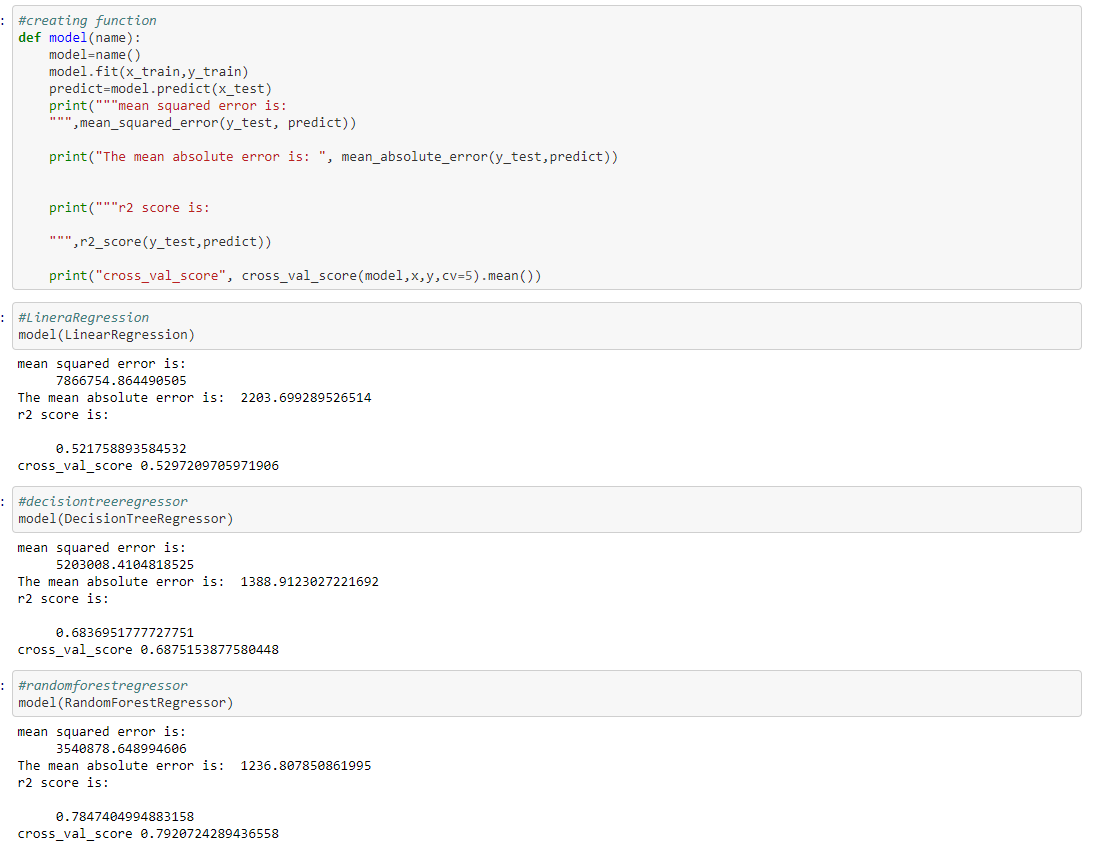
We have created a function name model, which will take the regressor name and apply the whole training process and give us all the statical information.



The LinearRegression model is giving 52 percent accuracy; we want it a good model.



The DecisionTreeRegressor model is giving 68 percent accuracy.



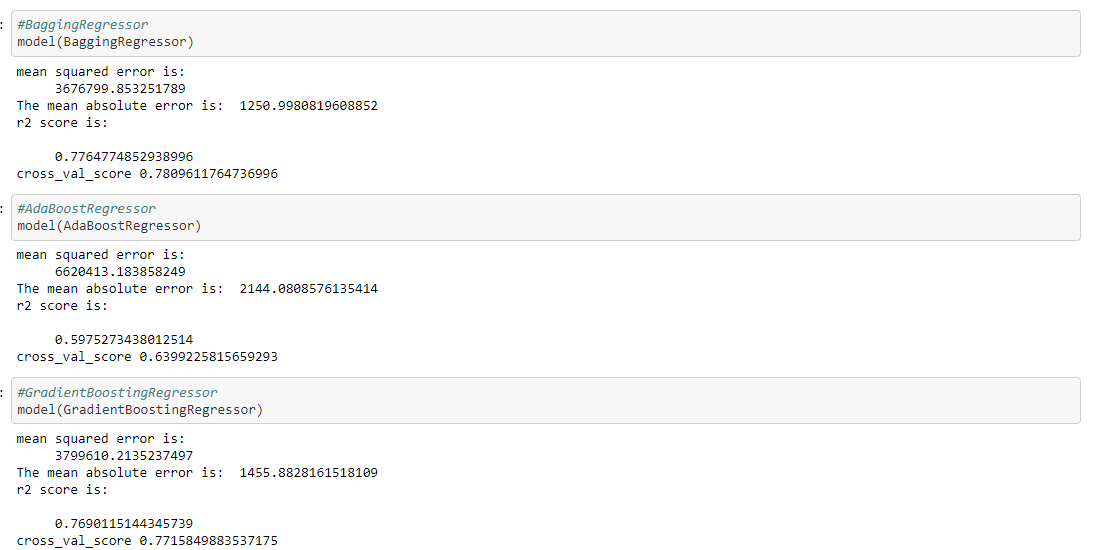
RandomForestRegressor model is giving 78 percent accuracy.



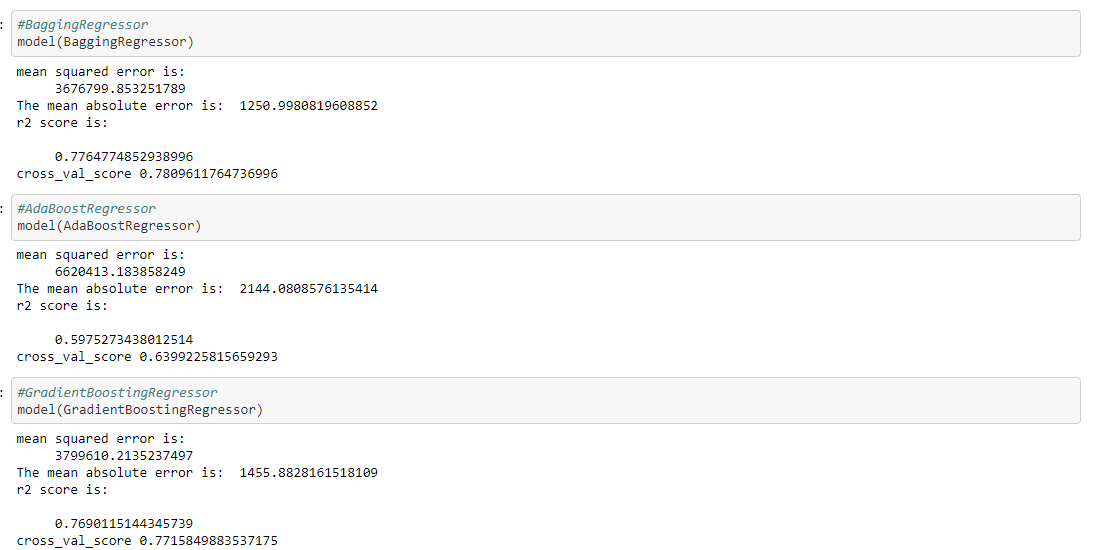
ExtraTreesRegressor model is giving 73 percent accuracy.



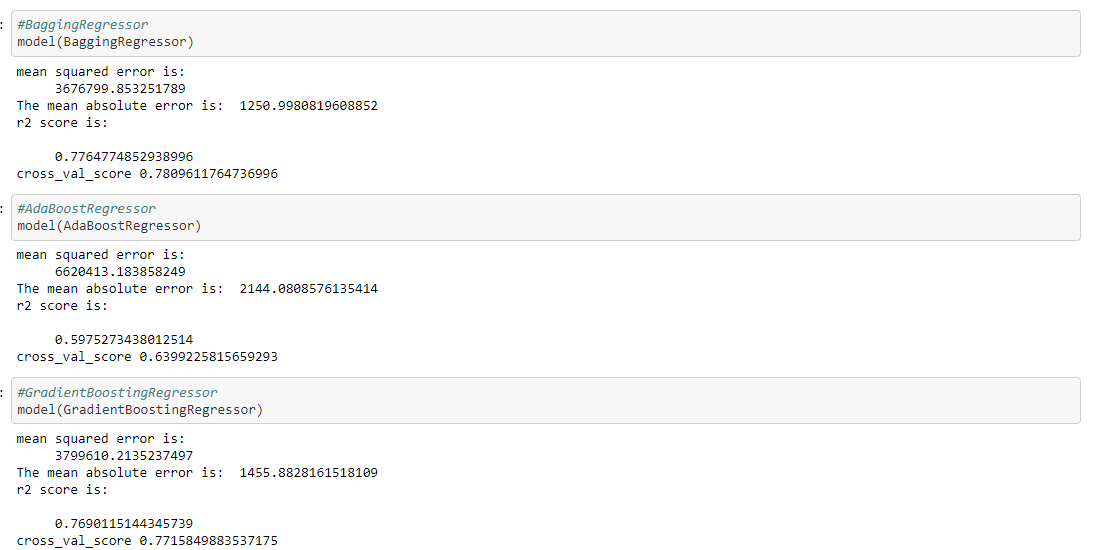
KNeighborsRegressor model is giving 71 percent accuracy.



BaggingRegressor is giving 78 percent of the accuracy.

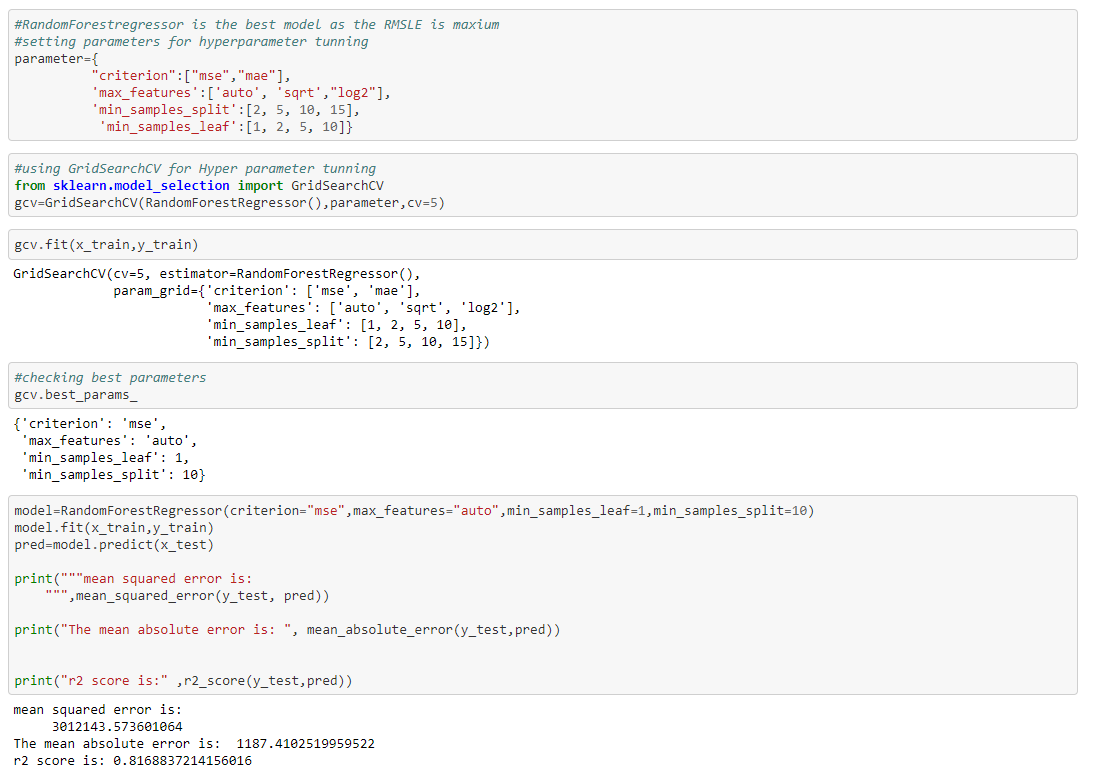


AdaBoostRegressor is giving 59 percent of accuracy.

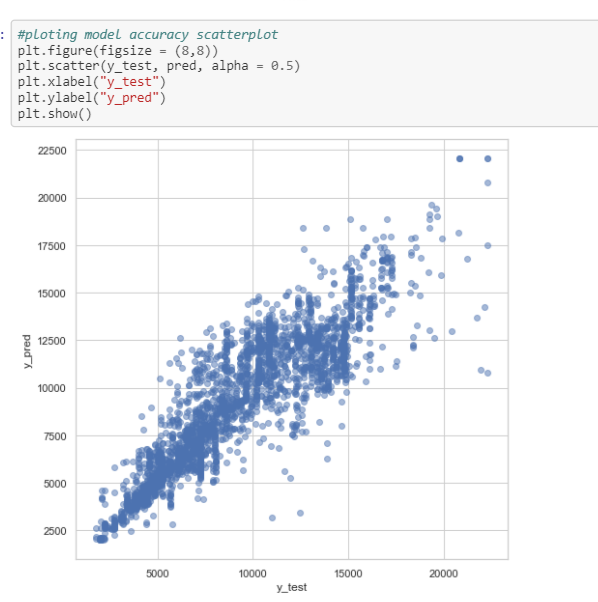


GradientBoostingRegressor is giving 76 percent accuracy.

From all the model above RandomForestRegressor is our best model as the r2 score and cross-validation score is almost identical. It shows that the model does not have an overfitting and underfitting problem. After that, we will do hyperparameter tunning to improve the accuracy of our model.



As shown in the image, we have defined the parameters. We have used GridSearchCv for tunning. Then we trained the model with the best parameters given. It has increased the efficiency of our model by three percent.



As we can see, our test and predicted vale are showing linear relations, and the error is significantly less.

**Conclusion:**

In this paper, we have gone through the process of prediction model building. The paper showed how to clean data with various techniques and what will happen if we will not clean the data. We have also analyzed the data graphically to find out insights from the given data. It was one of the main objectives. This paper showed how to create a different types of models and how to evaluate them. Also, it showed how to approach with hyperparameter tunning of the selected model.

**References:**

* Connect to Amadeus travel apis: Amadeus for developers. (n.d.). Retrieved June 06, 2021, from https://developers.amadeus.com/blog/flight-price-analysis-model-machine-learning
* Josua, N. (2020, September 28). Predicting airfare price using machine learning techniques. Retrieved June 06, 2021, from <https://medium.com/swlh/predicting-airfare-price-using-machine-learning-techniques-bf3a13ad07d1>