**Flight Ticket Price**

This project is about predicting the flight ticket price. There are number of factors which influence the flight ticket price such as:-

1. No. of airlines (competitors) in the market.

2. The duration of travel

3. The day of travel (whether a weekday or weekend or holiday)

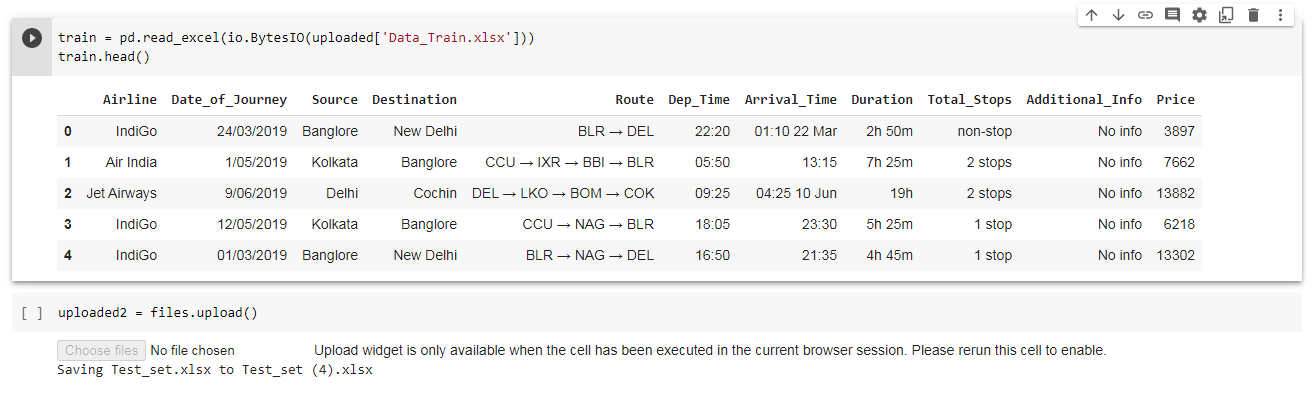
4. The time of travel (whether during normal timings or odd time)

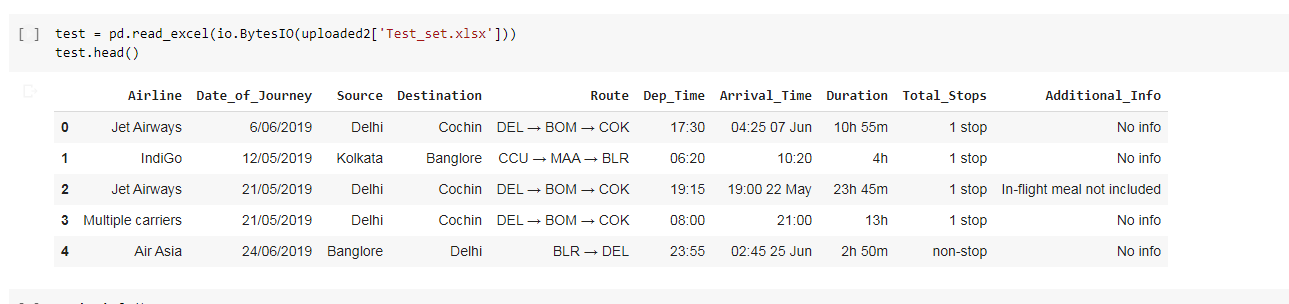
5. The no. of stops (whether a connecting flight or direct flight)

6. The category of seat (whether a first class or business class or economy class)

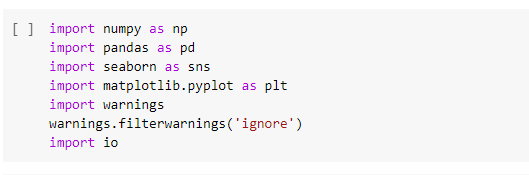
Here we are given a dataset which has information about flight and its travel details. Both training and test data are provided. We are required to analyze the factors influencing the price and build a model to predict it.

The training dataset is as shown below:-

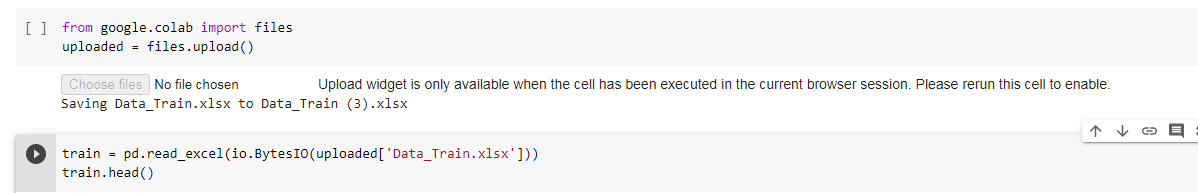


The testing dataset is as below:- 

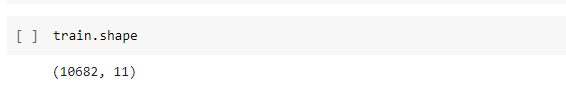
1. Firstly, we need to import necessary libraries like pandas, numpy for reading the data and performing various operations in dataframe, matplotlib.pyplot and seaborn for pictorial representations and warnings library is imported for not displaying warnings signs in the notebook.



2. After importing libraries, we need to import both the datasets training and test dataset



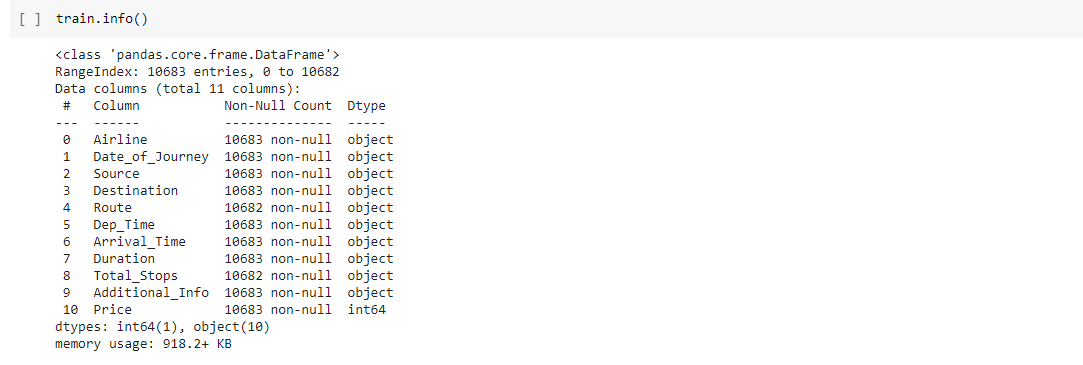
*3. Checking the shape of the datasets*



There are 10683 rows and 11 columns in the training datasets

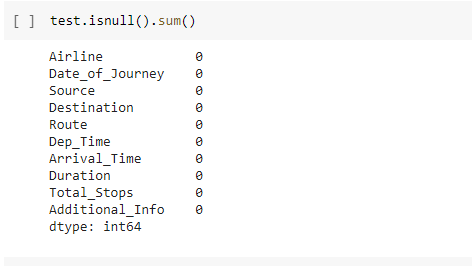
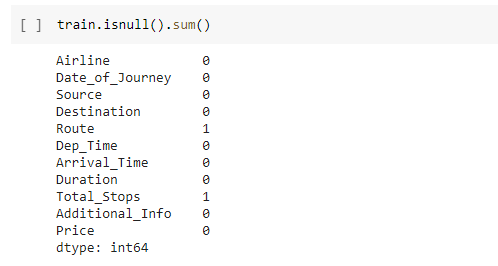
There are 2671 rows and 10 columns in the test datasets. Test dataset has one less column when compared to training dataset since it does not have target (price) attribute.

*4. Checking the datatypes of the datasets*



Apart from Date of journey and price all are in object datatype.

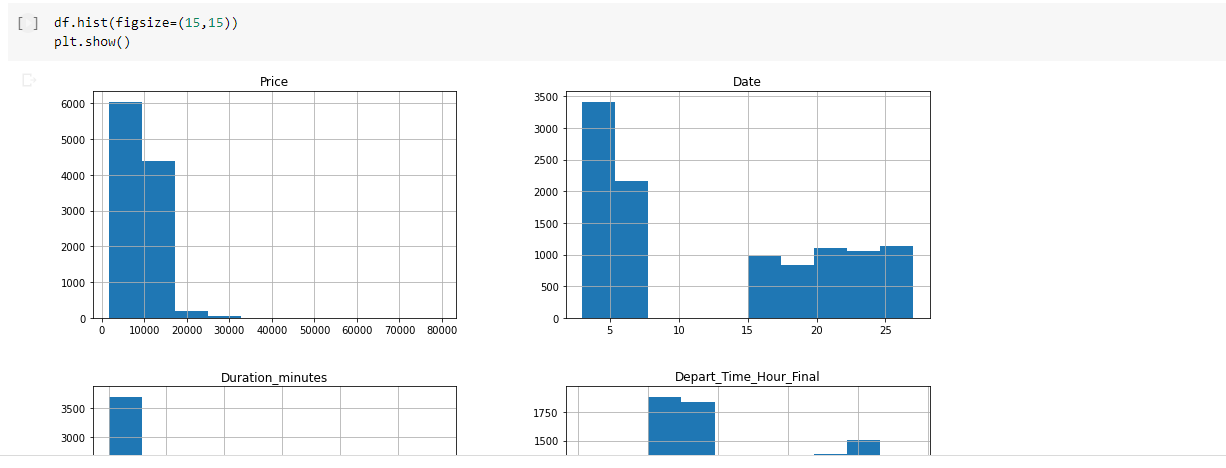
5. Missing values



There are no missing values in the test dataset.

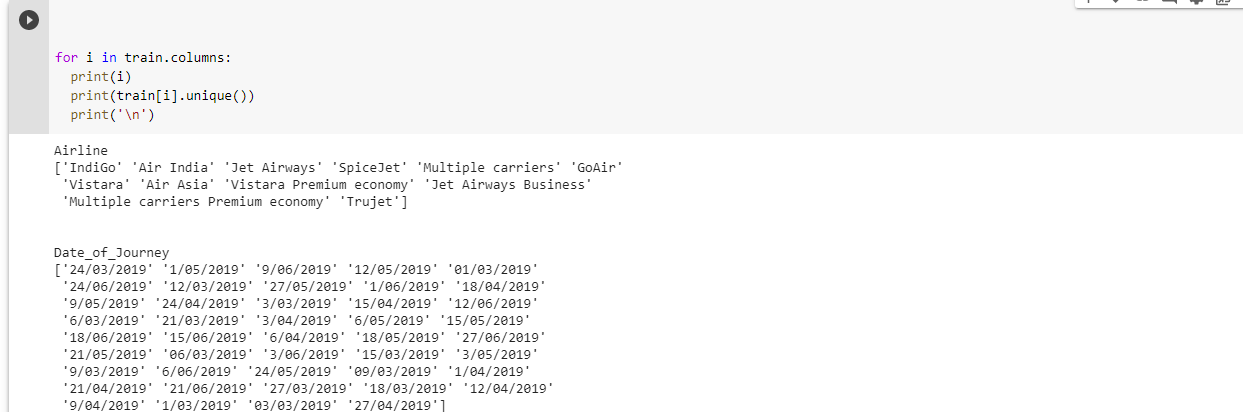
There is one value missing in the route and total\_stops columns

6. Univariate Analysis

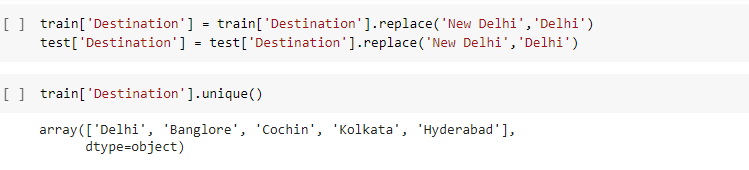


a. We check the unique value and their counts in the datasets.

Training dataset. Testing dataset too has same unique names but with different counts.

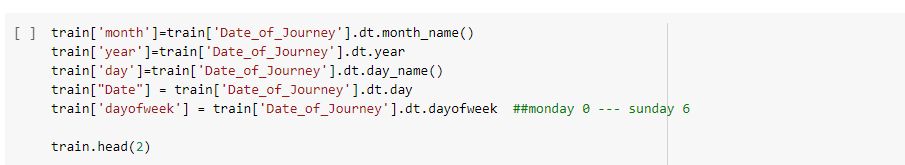


New Delhi and Delhi are shown as separate air destinations. But as of now, only one airport is fully operational, hence we change New Delhi to Delhi to retain one value.

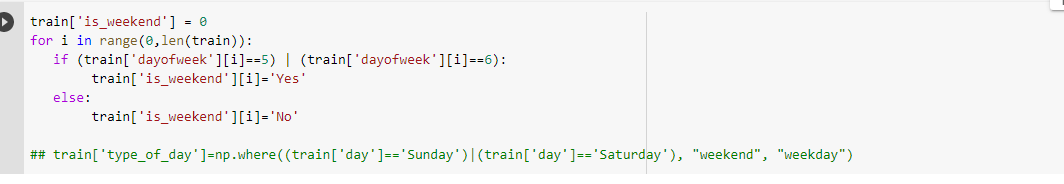


*b. Deriving month, year and day of journey*

We derive month, year and day of journey out of date for further analysis



*c. Further classifying day into weekend or weekday*



We repeat the same - deriving month, year and day and classification into week end and weekday for training dataset.

*d. Deleting the year as it has only one value - 2019*

*e. Cleaning the ‘Duration’ column*

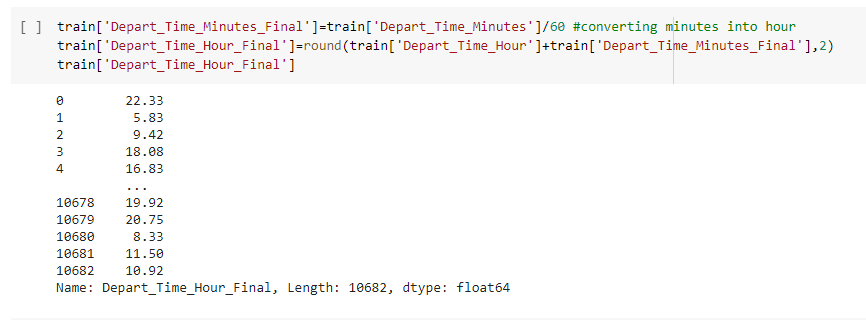
The duration column is object datatype and has values like 10h 55m, 4 h, 13h etc. All these must be brought to a standard value.

*f) Cleaning the departure and arrival times*



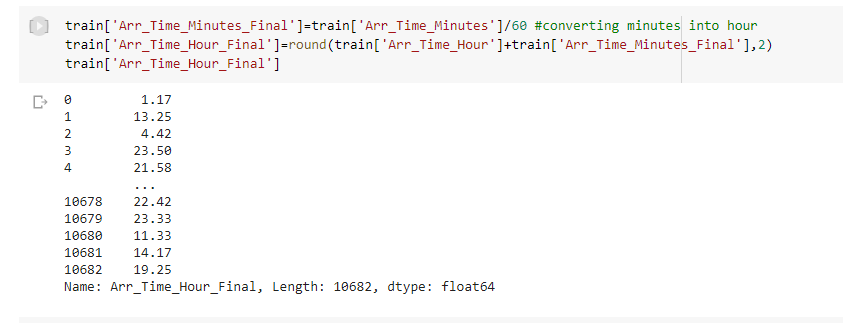
*h) Combining the departure minutes with departure hours*

Departure hours and minutes are separate which needs to be combined together.



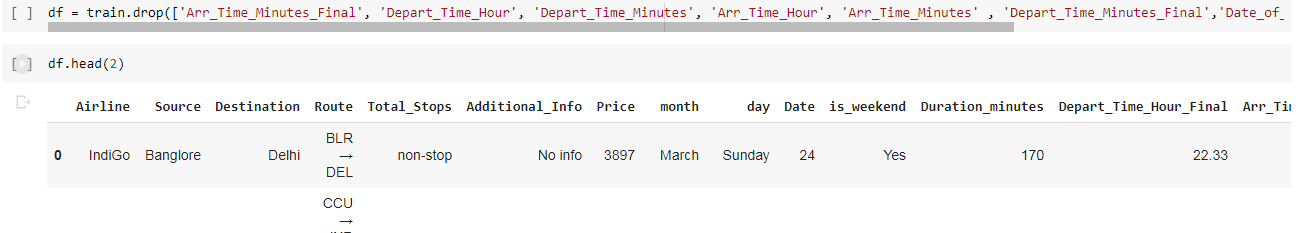
*i) Combining the arrival minutes with arrival hours*

Similarly Arrival hours and minutes which are separate needs to be combined together.



Note: We repeat the same process for testing dataset also. For the sake of simplicity of document, since the codes are repetitive, they are not shown here.

*j) After arriving at final departure time and arrival time, we can remove columns like departure time hour, minutes, arrival time hour, minutes etc from both training and test data set*



**k) Handling missing values**

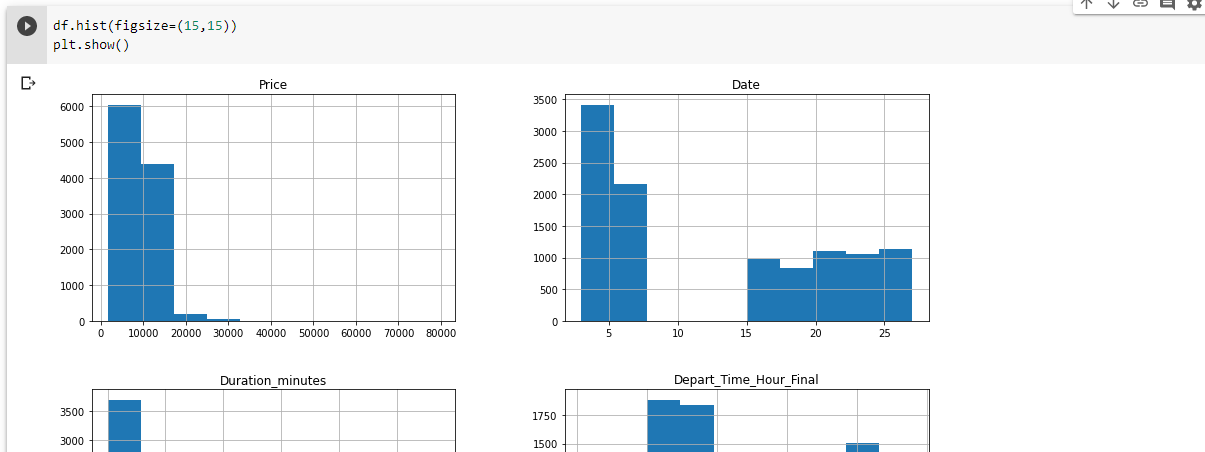
As investigated earlier, there are two missing data in training data and no missing values in test data.

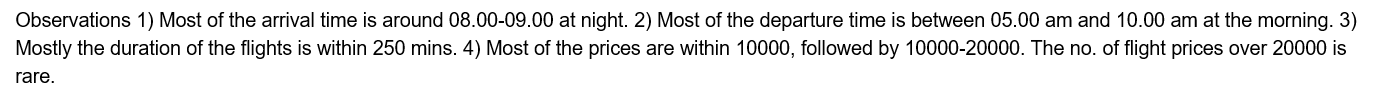


We check the similar flights in the same route and based on that information, we fill in the missing data

Using the similar data, we fill in the route and total stops in the dataset and finally check if the data has any missing values.

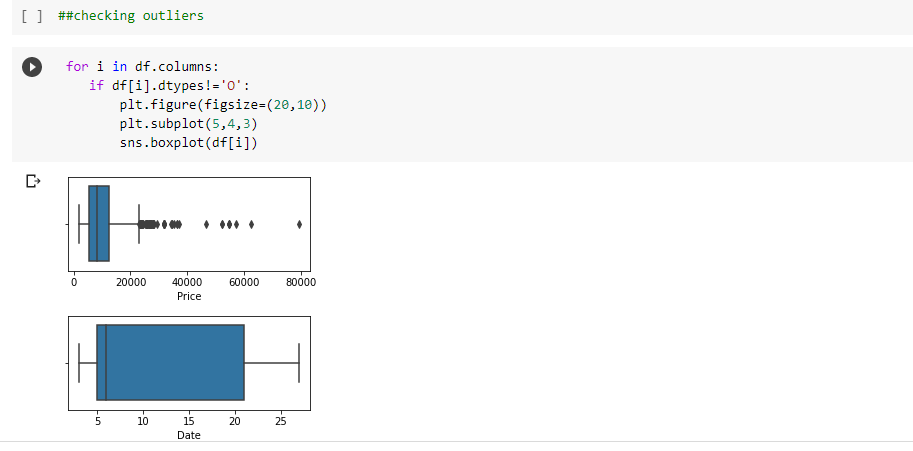
**l) Checking distribution of the data**





Testing data also has almost similar distribution and observations.

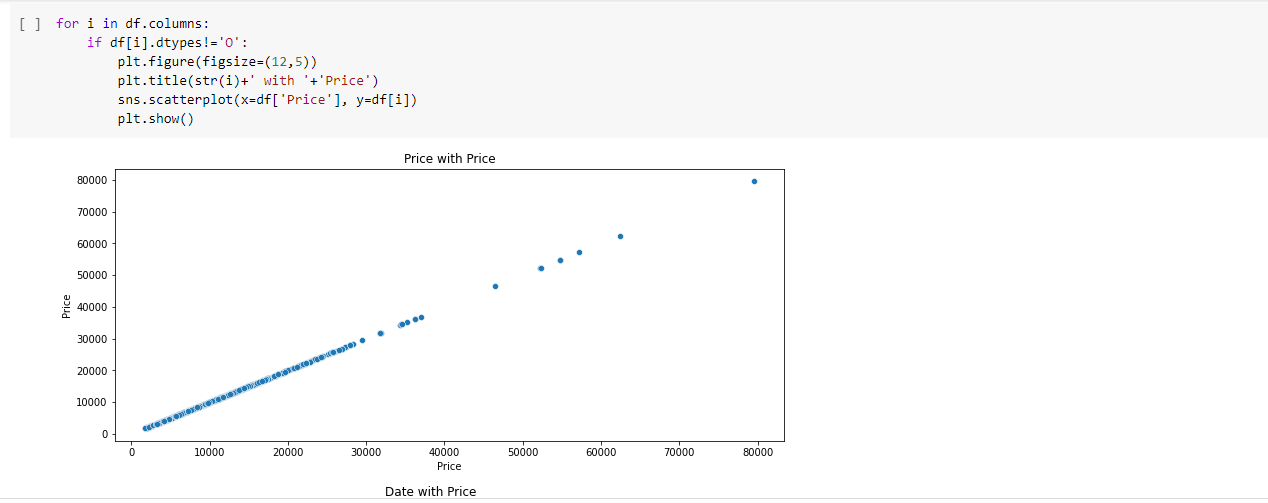
**j) Outliers in the data**



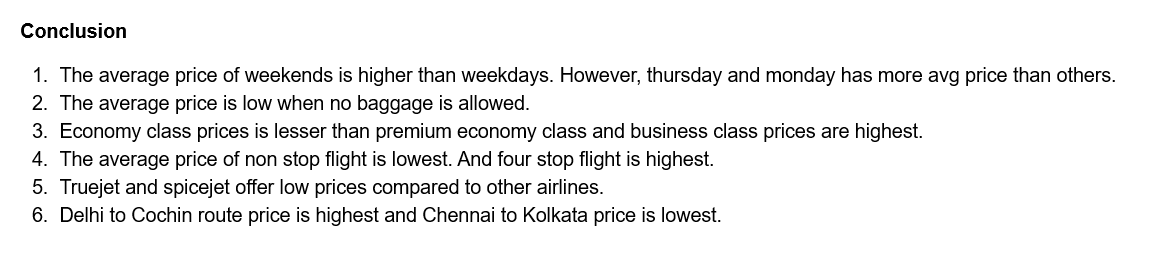
There are outliers in the price and duration minutes in the datasets.

**7. Bivariate Analysis**

**a. Using scatter plots**

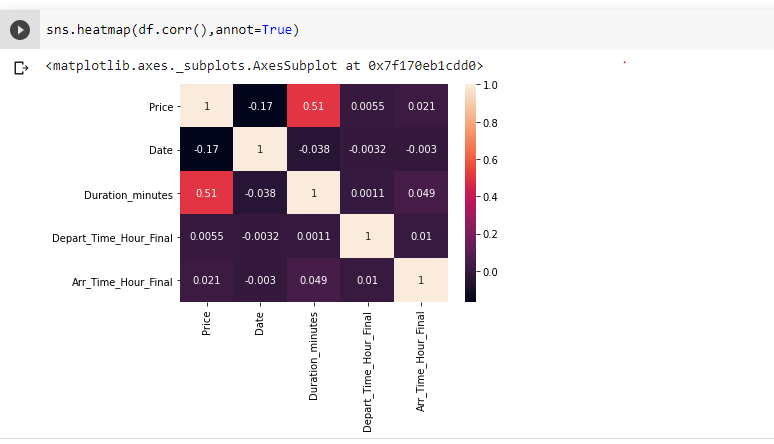


None of the variables show positive relationship with price.



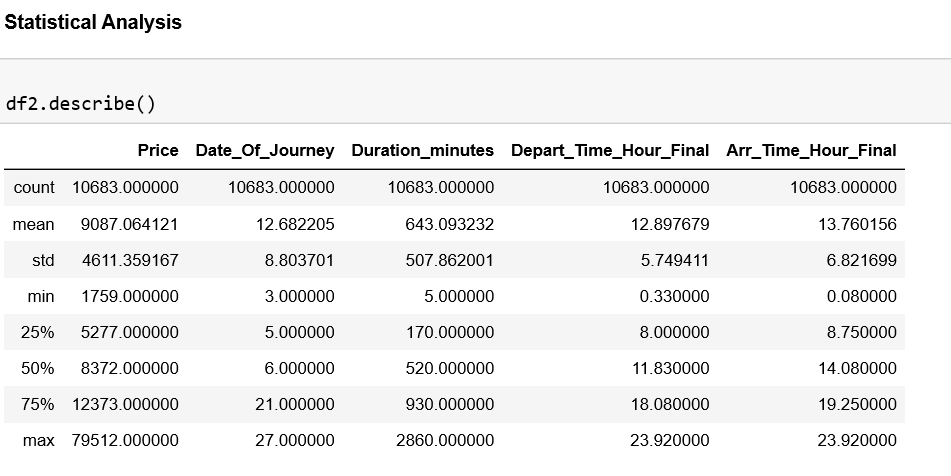
**8. Checking correlation**

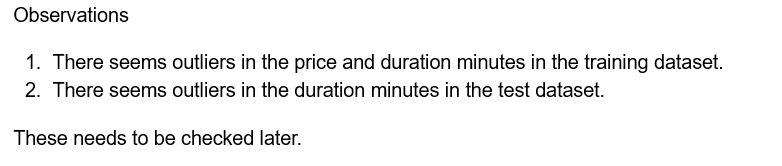
Only duration in minutes and price show little positive correlation.



**9. Statistical analysis**

This provides information about minimum, maximum, quartiles, mean, count and standard deviation information for the attributes.





**10. Pre processing data**

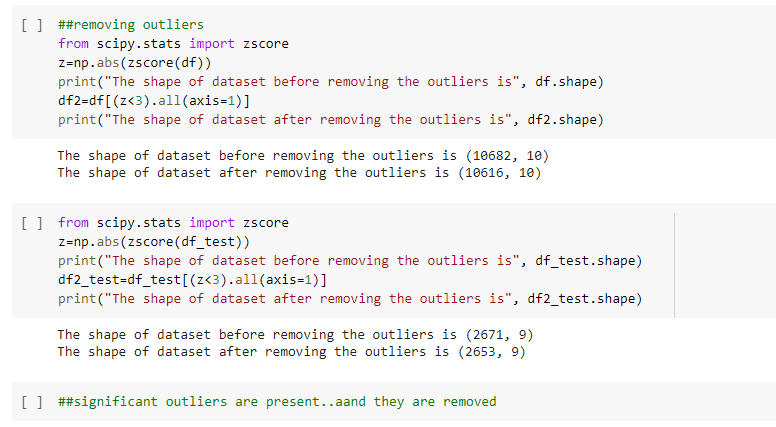
**i) Encoding data using Labelencoder**

All the categorical data needs to be converted to numerical using the Label encoder.



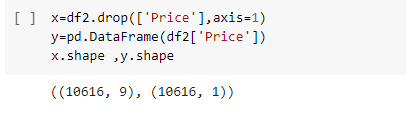
**ii) removing outliers**

We need to remove outliers for both training and test dataset using zscore.

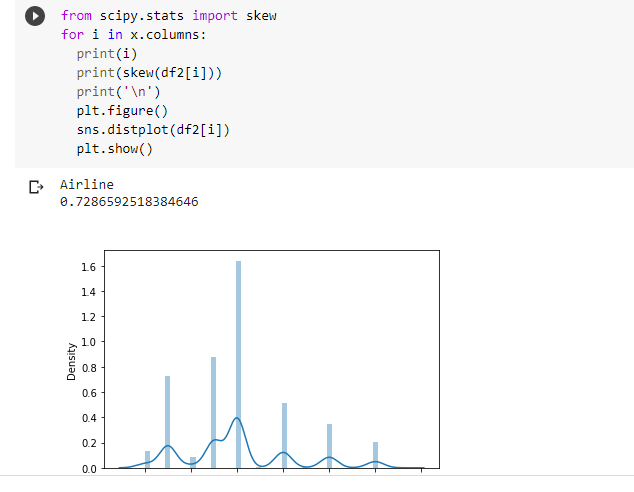


**iii) Splitting data into input and output variables**

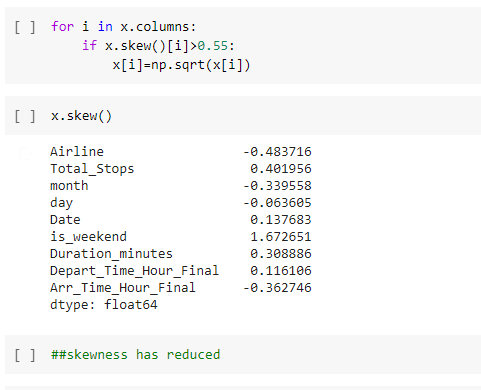
Here only the training dataset will be split for training and test purpose.



**iv) checking the skewness**



We need to remove skewness if the values are above +/- 0.55.

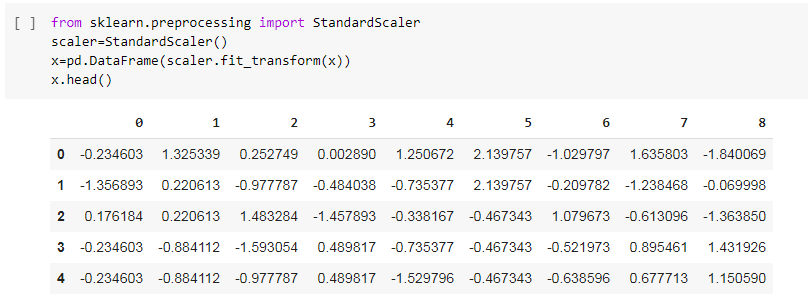


Similarly skewness is removed from test data

**v) scaling the data**

Before training the model with data, we need to scale the data and bring down the values within a normal range. There are two types of scalers:- Standard Scaler and Minmax scaler.

Here we are using standard scaler. This brings the values within the range -1 to 1.



Similarly the test data will also be scaled using standard scaler

**vi) Finalizing the best random state**

We select the random state which gives us best R2 score for model



After selecting the best random state, we will finalize the train and test data.



After finalizing the data, we need to tests the other models as well. We need to use GridSearchCV to select the best parameters for the models.

**viii) Using Gridsearchcv to select best parameters for the models**

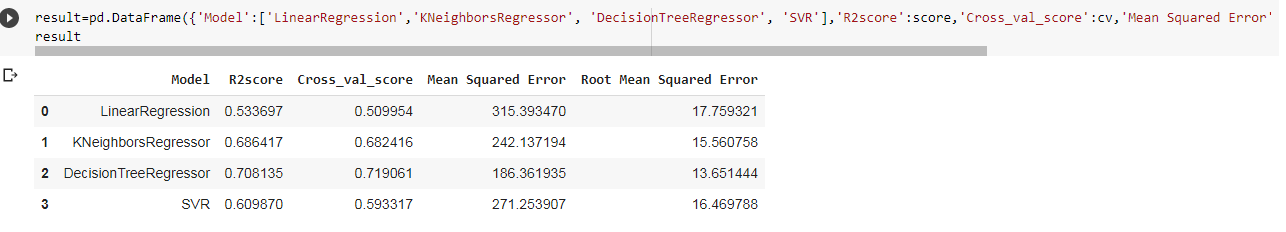


**ix) Checking the model performance**

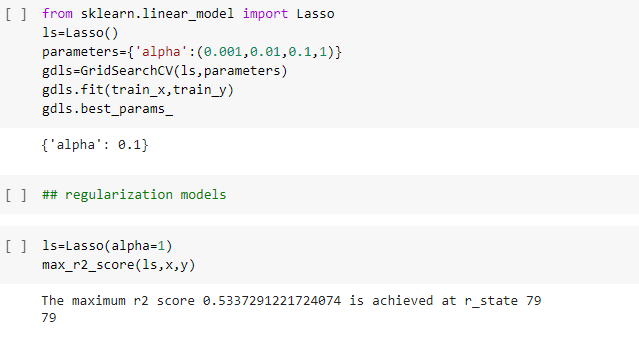
After selecting the best parameters, we can use them to check the r2 score and cross validation score and other metrics for all the models.

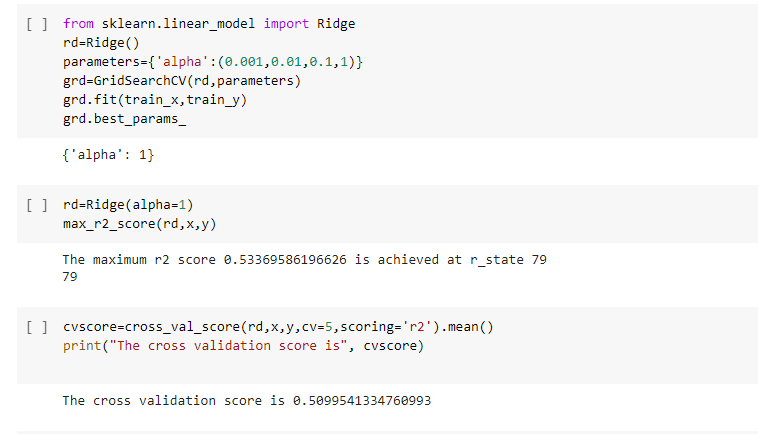


We can observe that except Decision Tree Regressor none of the other models are giving good scores.



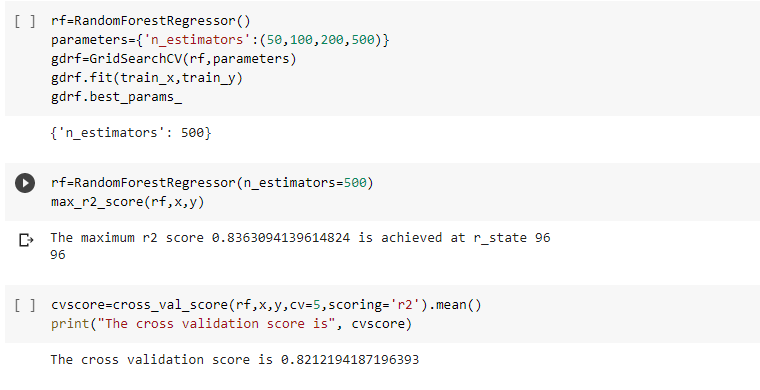
We can check the regularization algorithms and boosting algorithms. We can select best parameters using GridSearchCV. After obtaining best parameters we can calculate r2 score and cross validation score for the same.

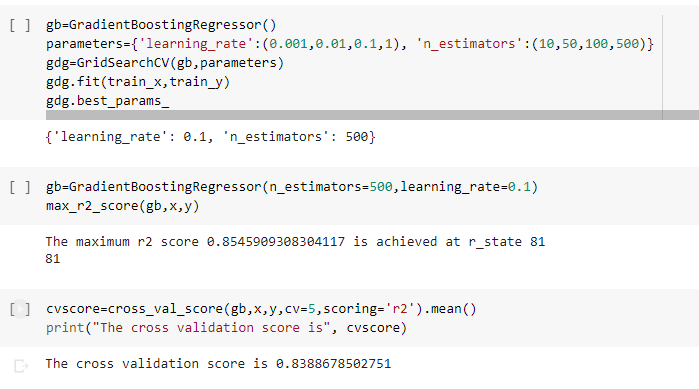


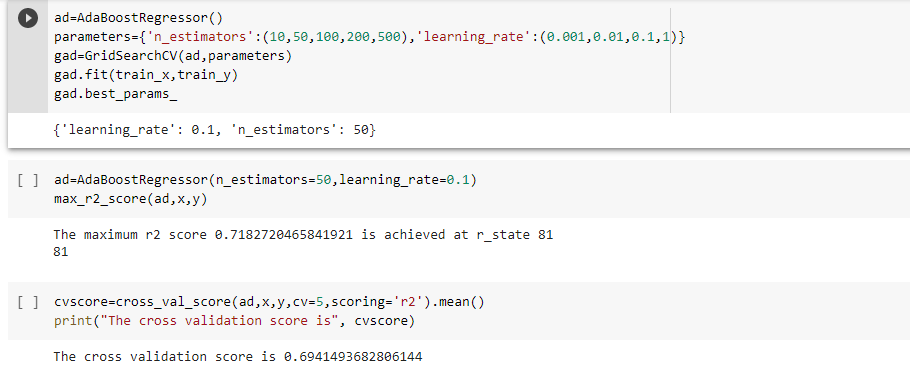


Lasso and Ridge are not showing good r2 score so we need to try the boosting algorithms.

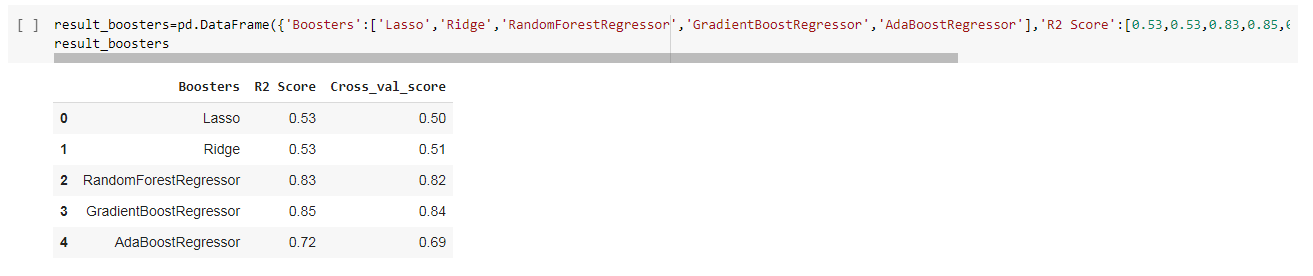
Similarly for the boosting algorithms we use Gridsearch cv for selecting best parameters. And we calculate r2 score and cross validation score for the same.







The results obtained from regularization and boosting algorithms are saved in the dataframe ‘result2’. The earlier results of models are saved in ‘result’. We now compare them both.



By comparing the results of boosting algorithms and the regression models, we can finalize the Random Forest Regressor as it is providing best r2 score and cross validation score.

Later we make predictions on the test data(dftest3) and save it separately in a csv file.

Finally we can save the model.

