|  |
| --- |
|  |
| Flight Price Prediction |
|  |

|  |
| --- |
| **Submitted by: Rohit Kattewar** |

Flight Price Prediction



**Introduction –**

We might have often heard travellers saying that flight ticket prices are so unpredictable. So, this Article is generally on how can we predict the prices of the flight ticket by using the data science skills. Data scientists develop these flight price trend forecasts using Machine learning algorithms and mathematical models.

Machine learning is a method of data analysis that automates analytical model building. It is a branch of [artificial intelligence](https://www.sas.com/en_in/insights/analytics/what-is-artificial-intelligence.html) based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

**Problem statement –**

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.

The dataset is provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

**Content of the article:**

This article explains the complete process to build a machine learning model. Below mentioned are the various phases that we will go through, throughout the project -

1. Exploratory data analysis and Data modelling - details

2. Outlier detection and skewness treatment

3. Encoding the data — Label Encoder

4. Scaling the data — Standard scaler

5. Fitting the machine learning models

6. Cross-validation of the selected model

7. Model hyper-tuning

8. Saving the final model and prediction using saved model.

**Dataset Description:**



Dataset Source: https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects

### **FEATURES:**

**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.

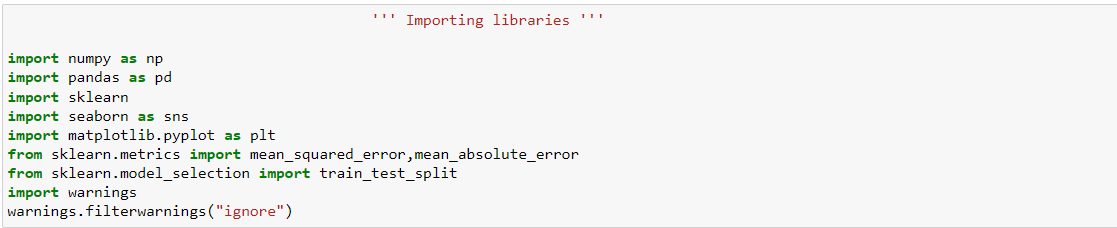
**Additional\_Info:** Additional information about the flight

**Price:** The price of the ticket

**Data Analysis:**

“**Data analysis** is a process of inspecting, [cleansing](https://en.wikipedia.org/wiki/Data_cleansing), [transforming](https://en.wikipedia.org/wiki/Data_transformation), and [modelling](https://en.wikipedia.org/wiki/Data_modelling) [data](https://en.wikipedia.org/wiki/Data) with the goal of discovering useful information, informing conclusions, and supporting decision-making.” – Wikipedia.

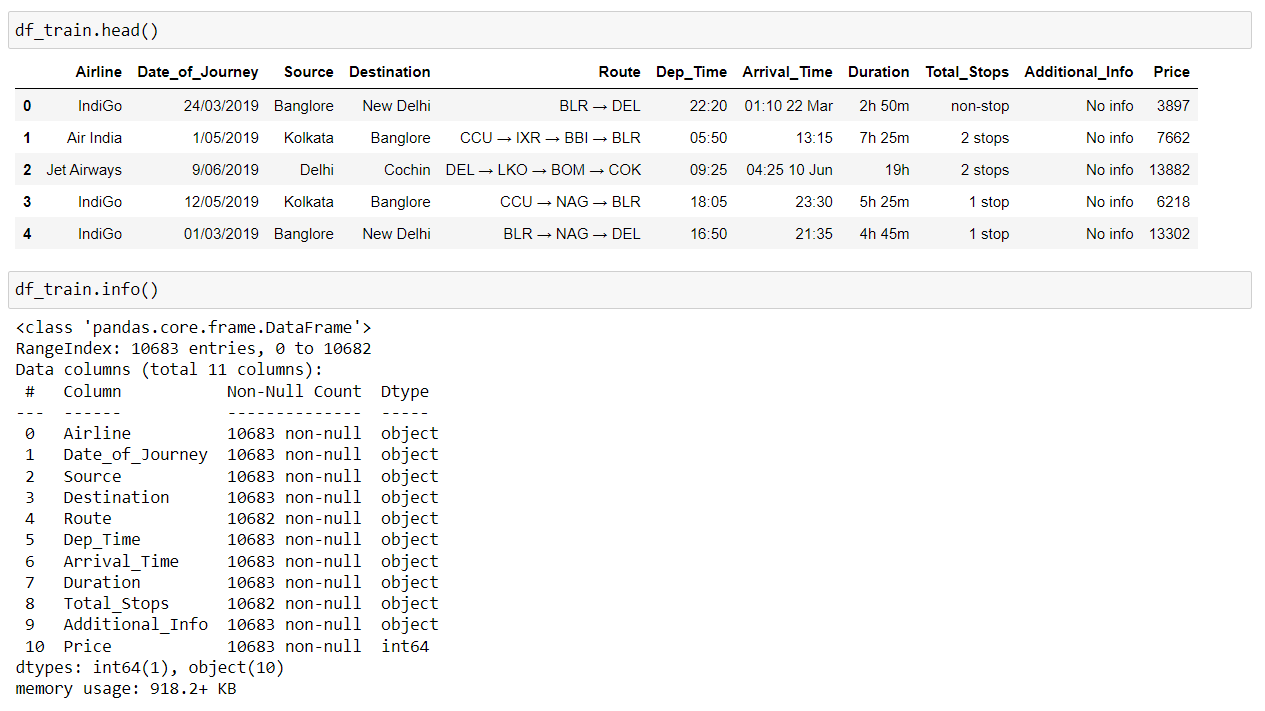
**Here we will start with importing some necessary libraries and further we will do the Exploratory Data Analysis.**



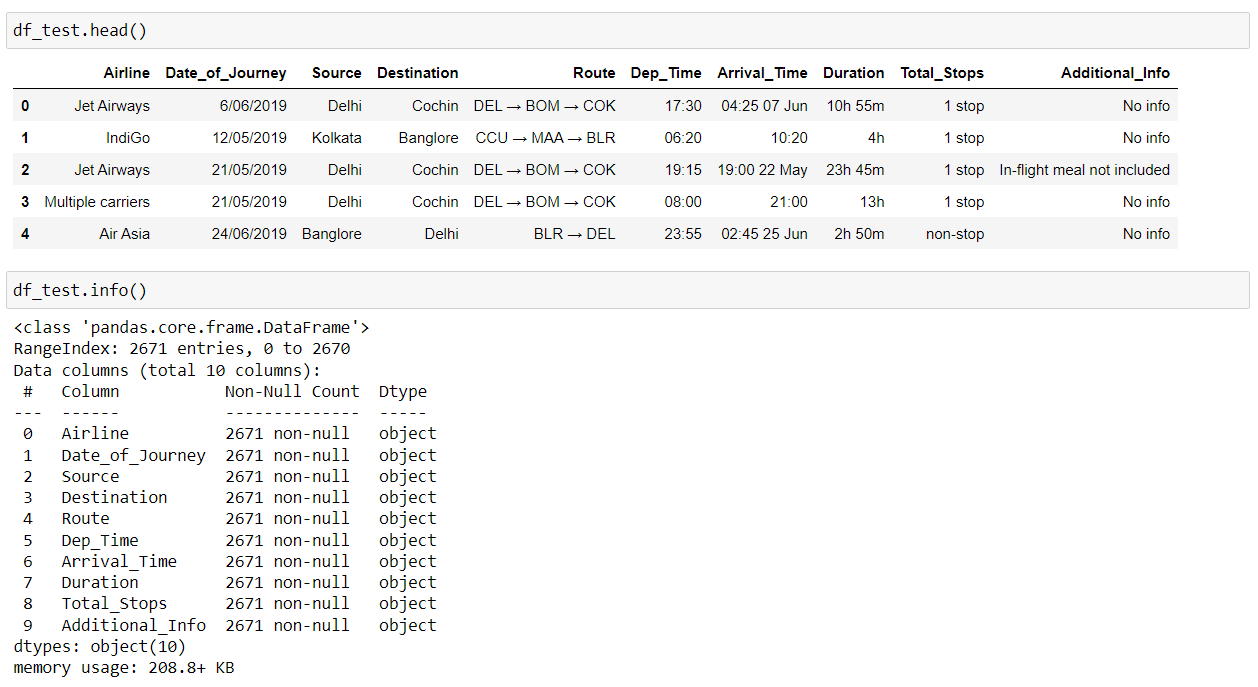
We will be using two datasets — Train data and Test data.

**Train dataset** - Training datasets is consisting of both categorical and numerical values.

**Test dataset** - The test dataset is similar to the training data set, but we won’t find the “Price” column in it. Because, it is to be predicted by using the model we will be building.



(Above image contains the top 5 rows of train dataset and the info of the same respectively.)



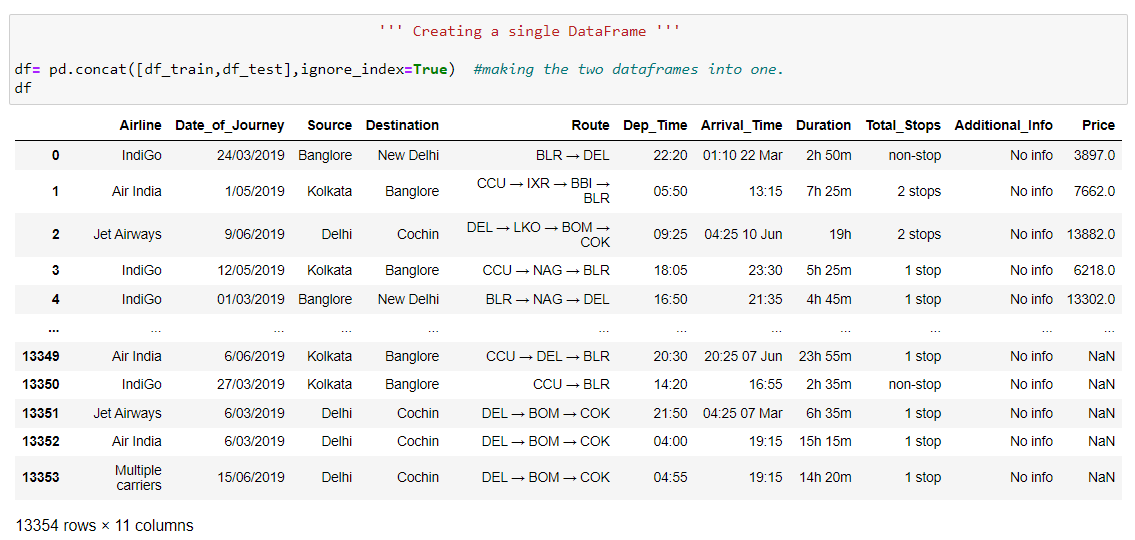
(Above image contains the top 5 rows of test dataset and the info of the same respectively.)

**df.info()** - gives us the information about number of values present in each column, and data types of each column.

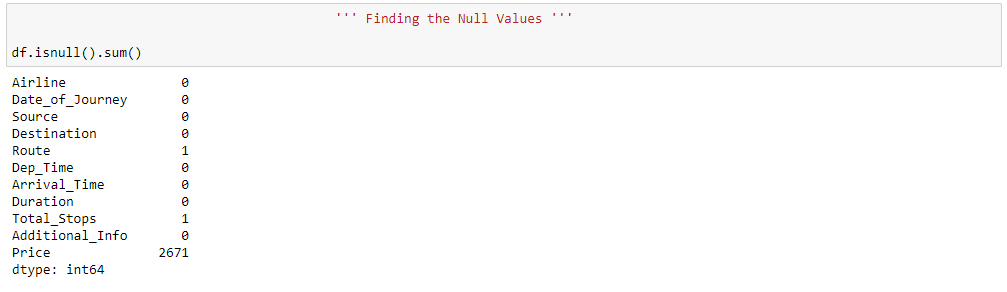
**Observations:**

1. All the columns from both the datasets has the same data type as “Object”, except the ‘Price’ column (test dataset) which has “int64” data type.
2. Date\_Of\_Journey, Dep\_Time, Arrival\_Time, Duration These columns has “object” data type which we will need to convert into date-time data type.
3. The Total\_Stops column has text ‘stops’ and ‘non-stops’, which we need to convert into integer datatypes.

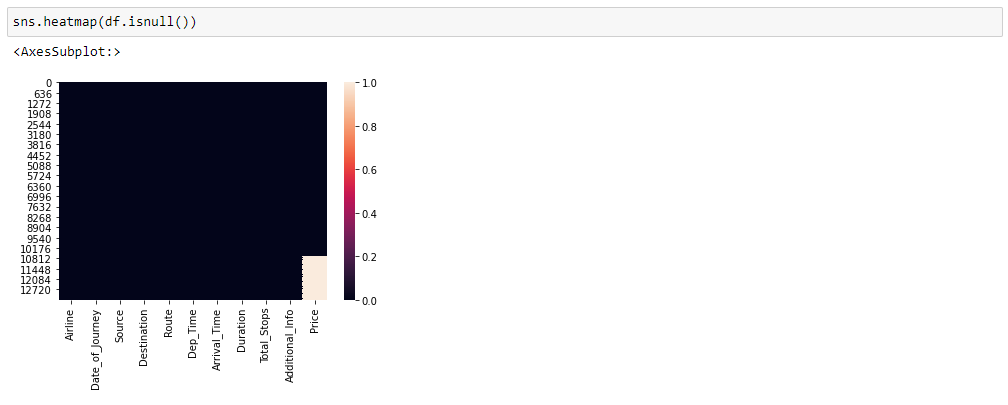
**From here we are combining both the datasets and make them as a single dataset:**



**We now check if there’s any NaN values present in the dataset:**

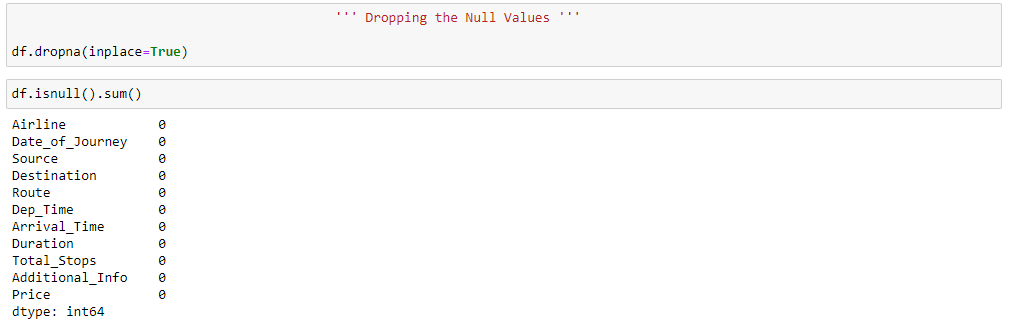


Below, the heatmap is representing are there any NaN values present:



The white portion at the bottom-right side of the image shows there are plenty NaN values available and those are present in the ‘Price’ columns.

**Treating NaN values:**

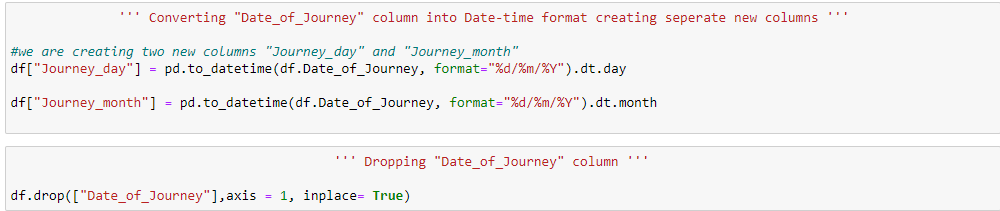




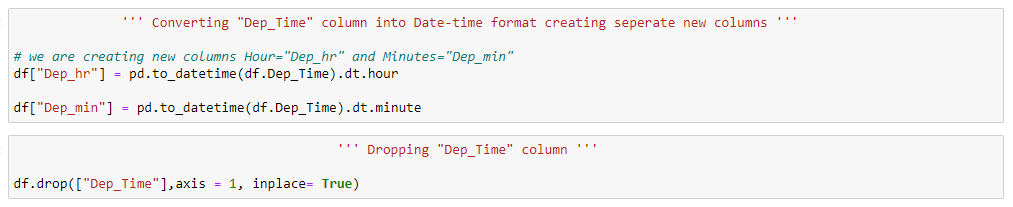
Hence, we successfully dropped the null values present in the dataset.

**We are converting the datatypes of the following columns, and spliting the values of these columns to create new seperate columns :**

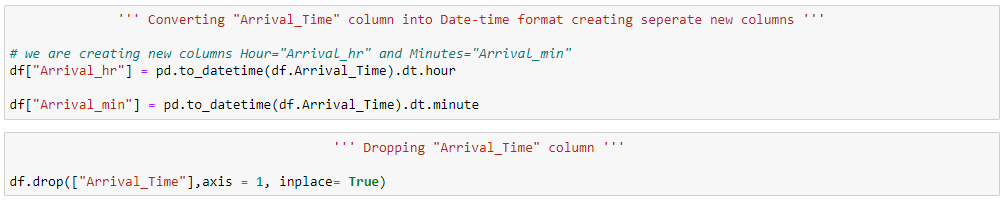
1. Date\_of\_Journey
2. Dep\_Time
3. Arrival\_Time
4. Duration



We converted the data type of the **Date\_of\_Journey** column into date time format and created new columns naming “**Journey\_day**” and “**Journey\_month**”.



We converted the data type of the **Dep\_Time** column into date time format and created new columns naming “**Dep\_hr**” and “**Dep\_min**” representing departure hour and departure minutes respectively.



We converted the data type of the **Arrival\_Time** column into date time format and created new columns naming “**Arrival\_hr**” and “**Arrival\_min**” representing Arrival hour and Arrival minutes respectively.

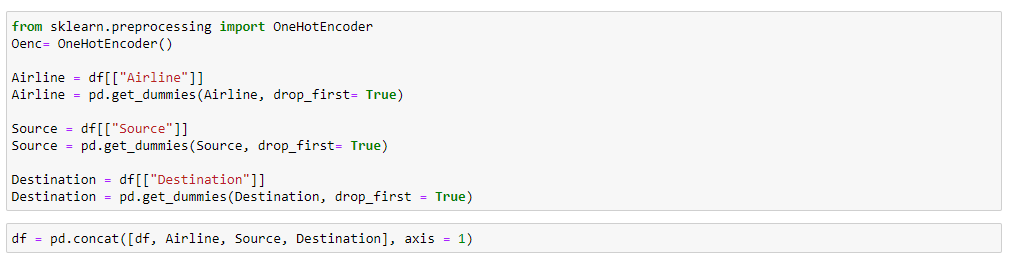
We now convert the data type of “**Duration**” columns and split the column between two new columns as “**Duration\_hr**” and “**Duration\_min**” representing duration hour and duration minutes.



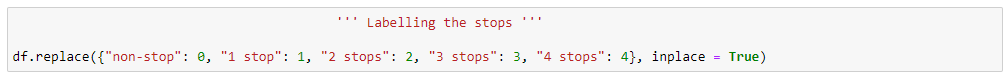
**Encoding:**

We are going to use the One Hot Encoding technique to convert the data into numeric.

**One Hot Encoding –**

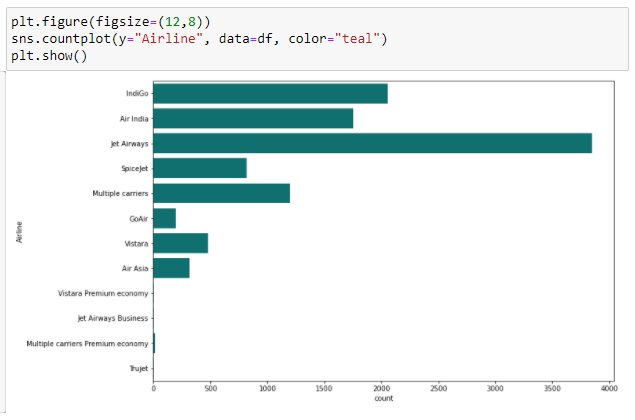


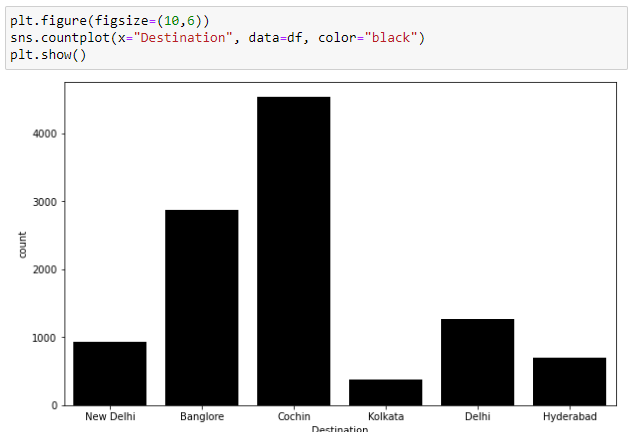
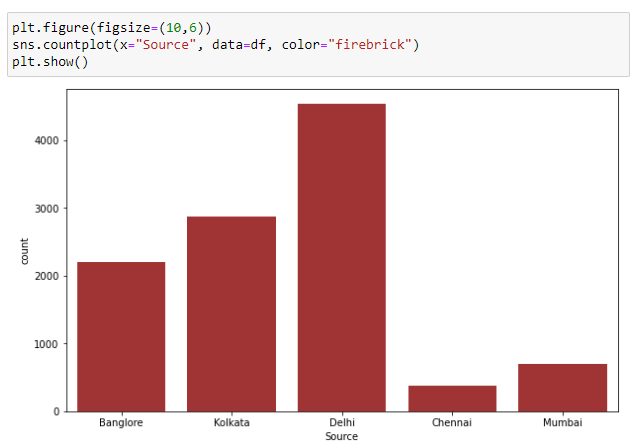
We are also going to label the values from the “**Stops**” column –

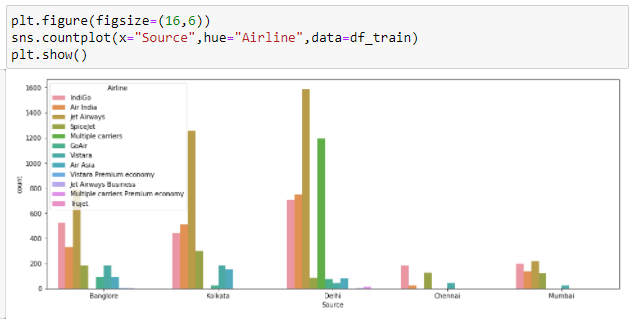


**Data Visualization:**

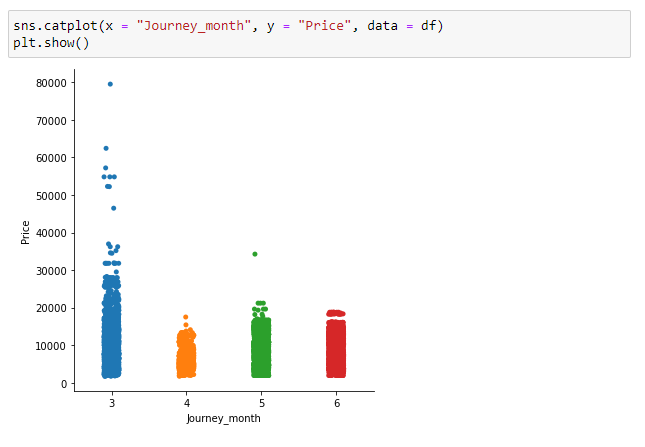
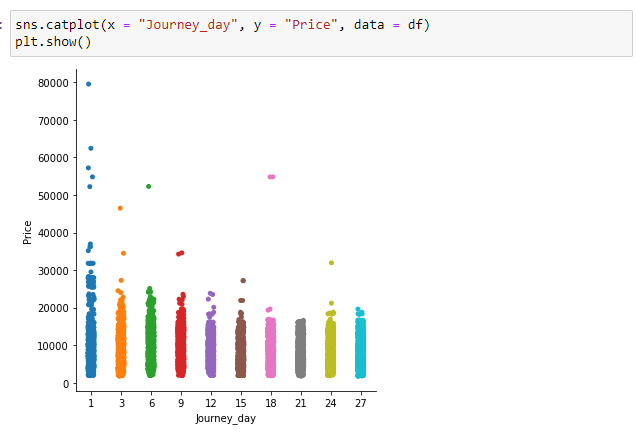
Plotting the Countplots for the representation of categorical data –





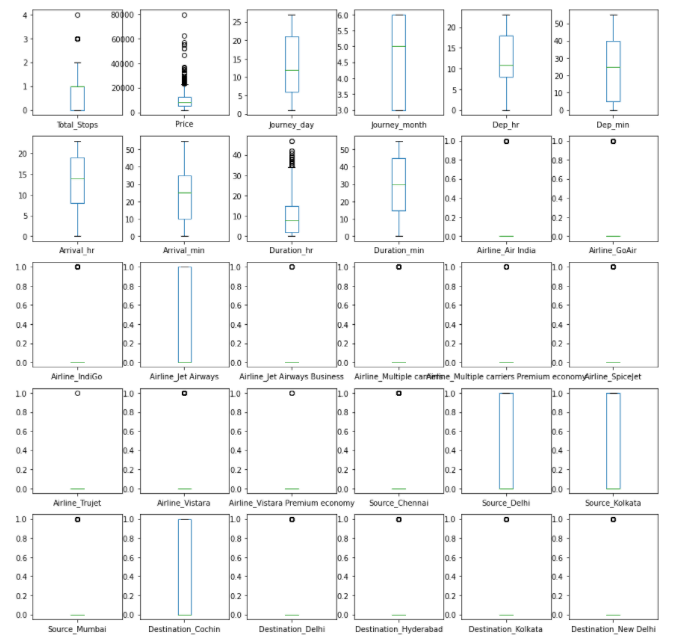


Catplot shows the relationship between categorical and numerical data –



boxplot –

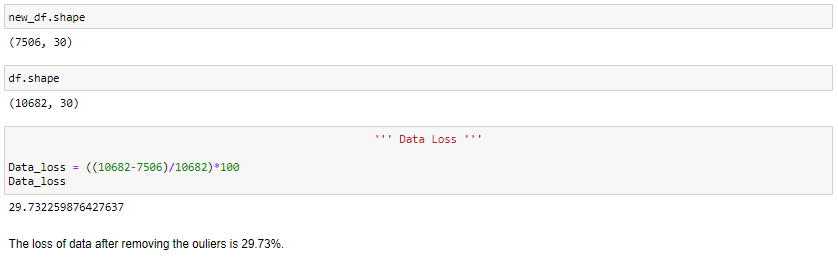




**Outlier Removal –**

We are removing the Outliers present in the dataset –



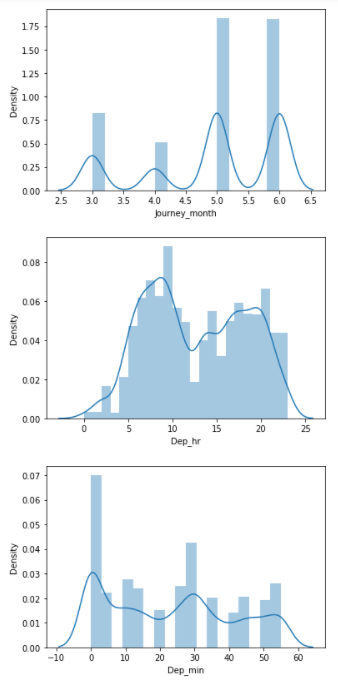
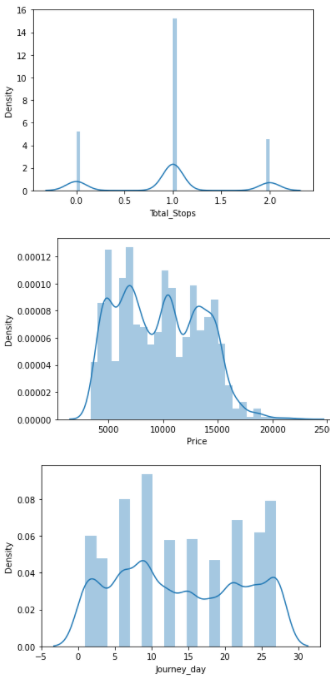
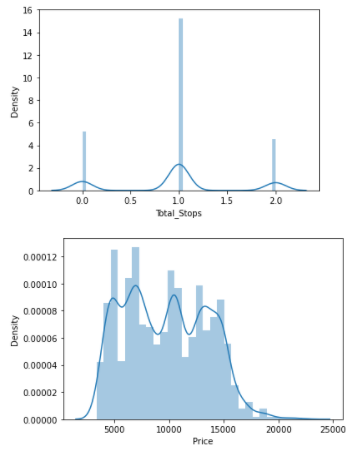


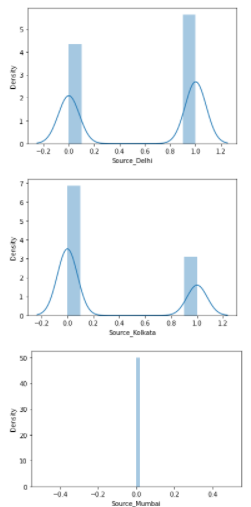
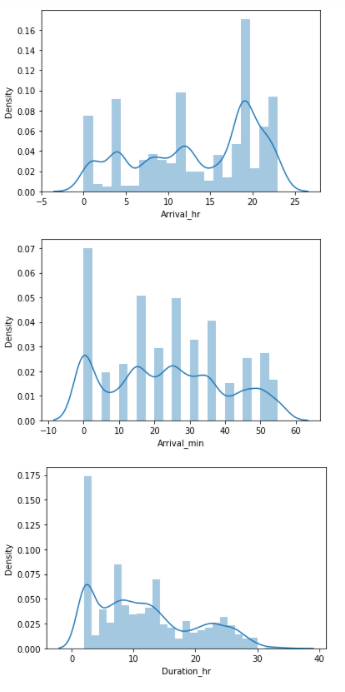
After removal of the we lost 29.73% data.

**Skewness:**

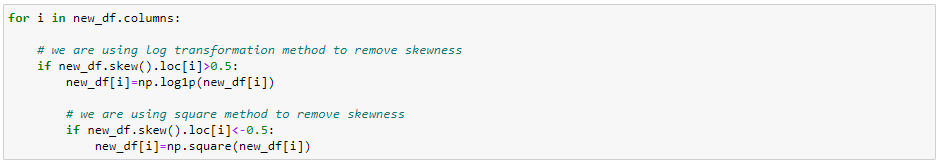
We now proceed further with the skewness present in our data which will allow our model to learn better.

Plotting the distribution plot to see the skewness present in the data-





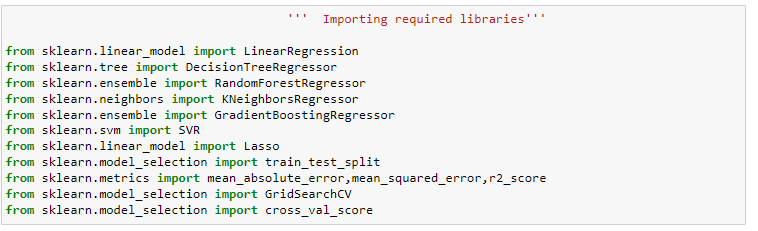
**Removing Skewness:**

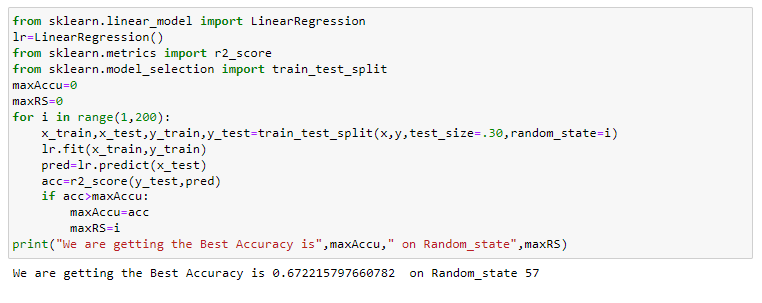


We removed the skewness by using the log transformation method as well as square method.

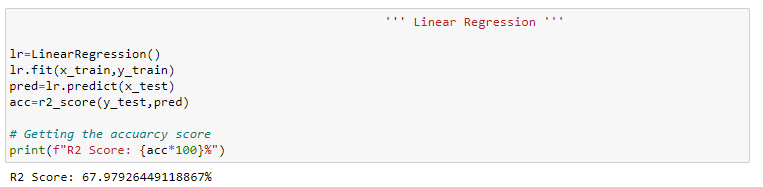
**Fitting the model :-**

We now proceed to the main step of our machine learning, fitting the model and predicting the outputs. We fit the data into multiple regression models to compare the performance of all models and select the best model –



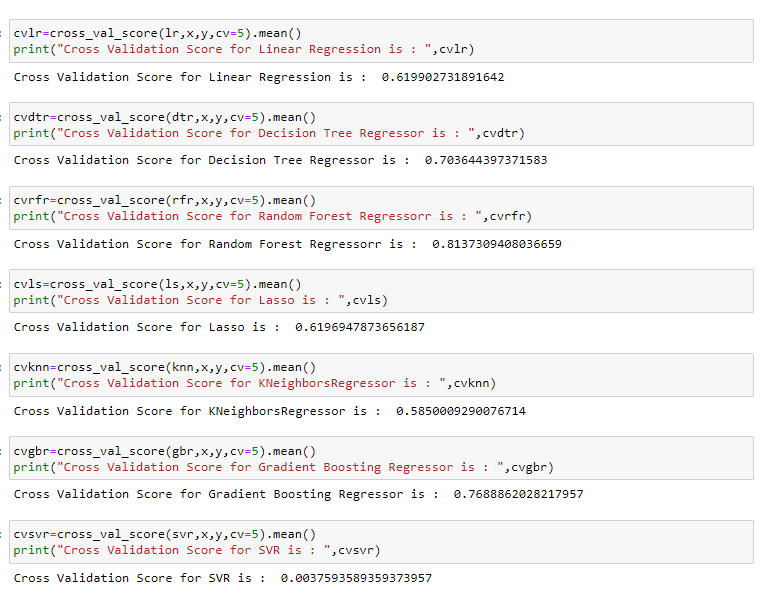


We will train our model with these values and choose the best models amongst them.





**Cross Validation Score –**

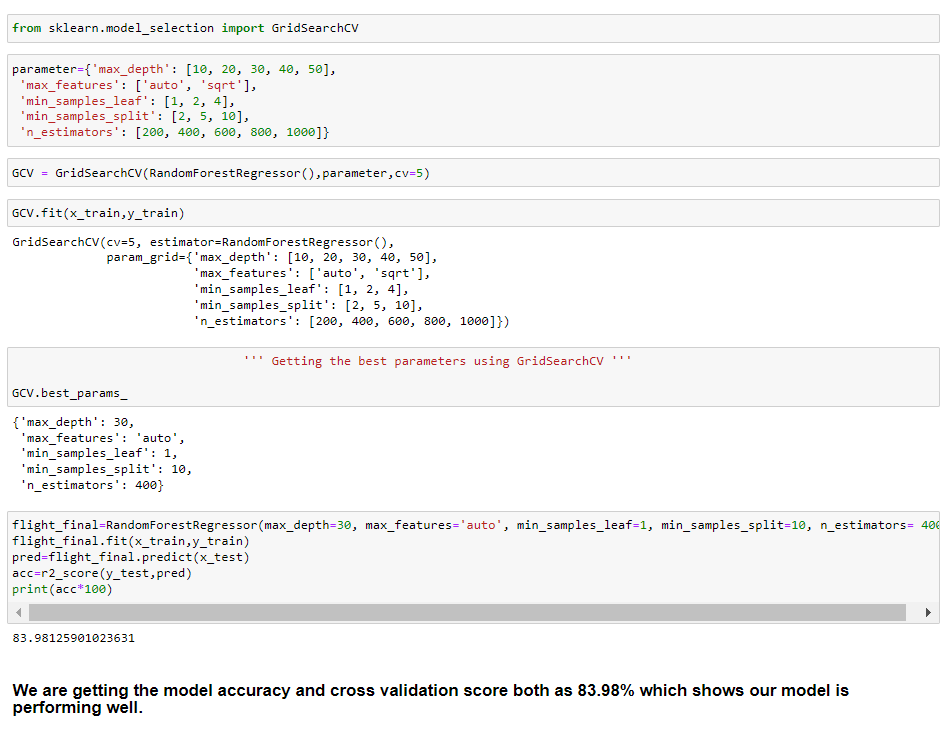


Now, to choose the best model to we need to find the difference between each model R2 Score and Cross Validation score. That model will be selected as a best model which has the least difference between their r2 score and CV score.

We are selecting **Random Forest Regressor** as our best model as it has least difference between its Accuracy score and CV score.

**Hyper Parameter Tuning:**

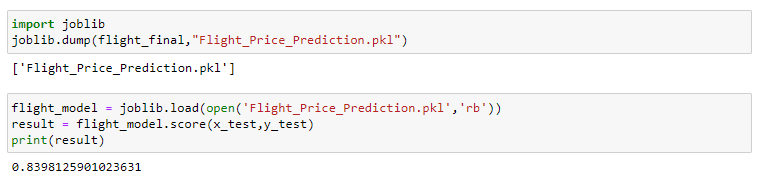
Grid Search CV is a technique used to validate the model with different parameter combinations, by creating a grid of parameters and trying all the combinations to compare which combination gave the best results. We apply grid search on our model –



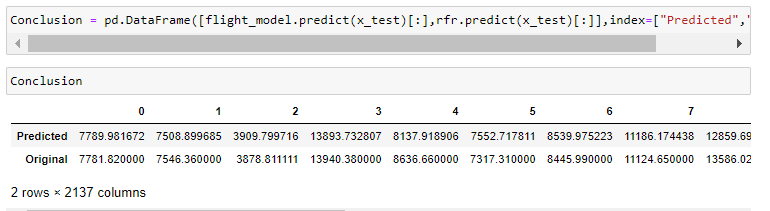
Hence, we can see that our model is performing well. The Random Forest Regressor giving the accuracy as **83.98**%.

**Saving the Model:**

We are saving the model using best parameters, and create model object using joblib.



**Conclusion –**



**---- --:-- ---- ---- --:-- ----**

**---- --:-- ----**