Flight Price Prediction

In this blog post, I’ll be going through the process of making a machine learning project to build a model to predict flight prices based on the information provided. The dataset comprises prices of flight tickets from various flights between various cities between the months of March and July in 2019.

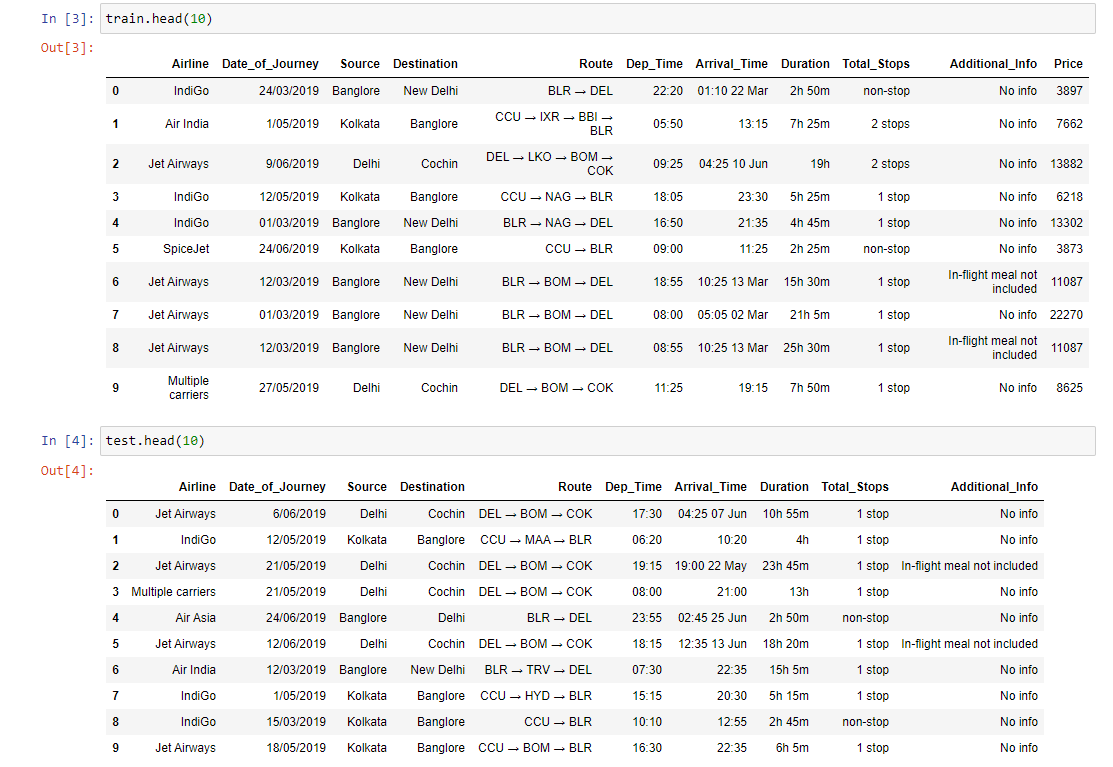
After loading the desired libraries and dataset, let us take a look at the columns or features of the datasets.

* **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

As we can see there are 11 features with ‘Price’ being the target variable and the remaining 10 columns being the independent variables. Since ‘Price’ is continuous and not categorical, it will be a regression problem and not a classification problem.

# Data Analysis

We are provided with two separate datasets for training and testing, the shape of these datasets are (10683, 11) and (2671, 10) respectively.



Initial observations from looking at the first ten rows of both the training and testing datasets are that even though the problem statement specifically mentioned the months of the flight, there are no specific columns for the month, we can extract the values of the month from the date column and make a separate column for it to study the varying prices based on the month.

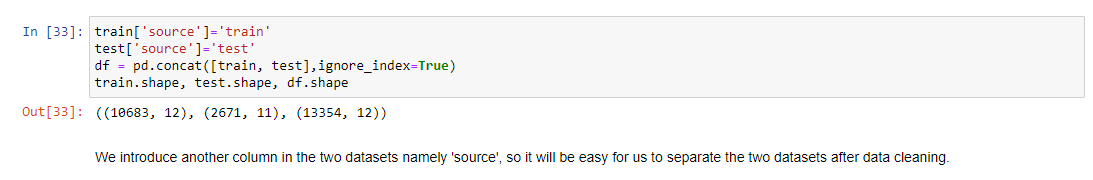
The arrival time column has date mentioned with time, we need to either make a separate column for that date or drop it, same need to be studied for the departure time column as both the arrival time and departure time columns are very similar so we may not see it in the first ten rows but there may be some row with the date mentioned with time in departure time.

The total stops column has the value ‘non-stop’ which would need to be changed and then we can convert the entire column into float or int so we can fit it in a model.

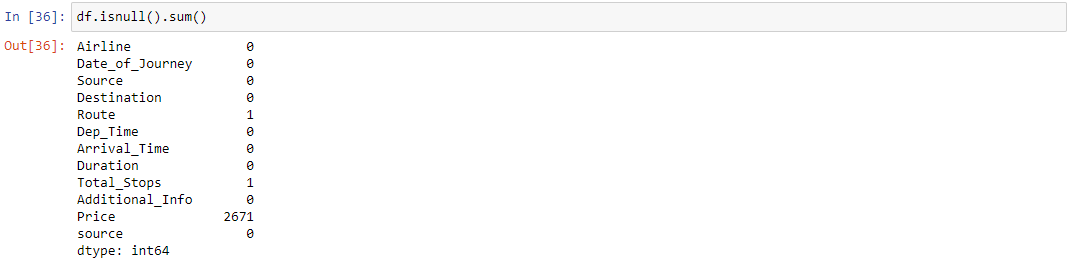
Most of the rows in the additional info column have ‘No info’, so we might consider dropping the column if we may not find any interesting observation after studying it a bit thoroughly.

Now whenever we are provided with two separate training and testing datasets, there are two ways to process the data, first is processing the training set and then repeating all the steps on the testing set. The second way is, merging both the training and testing datasets into one single set and then processing it and studying it, then splitting the datasets back to training and testing and fitting them into the model to get desired results. I usually prefer to go with the second way, no specific reason behind it, just individual choice.

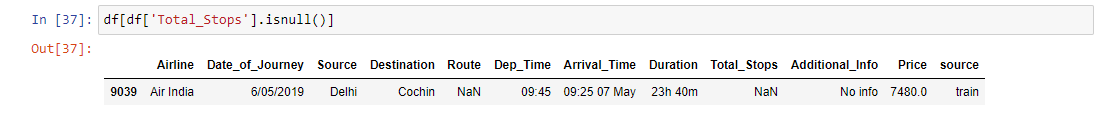
Before merging the two datasets, make a new column in them that refers to their source, so it will be easier for us to separate these data sets in the later stages of our project.



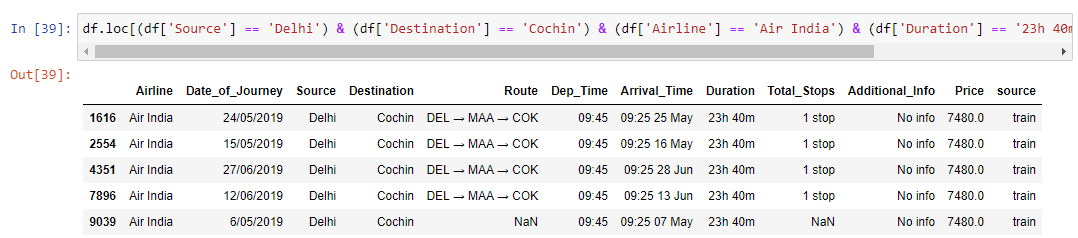
Now let us check the dataset for any missing values if there are any and treat them accordingly.



The 2671 missing values in Price are the values from the testing dataset, we are building this model to predict these values only so obviously there is no need to treat them, apart from that there is only one missing value in the Route and Total Stops column each, they both must be from the same row since we get the values of total stops from the route column only.



As I suspected both the missing values are in the same row. Since there is only one row with a missing value we can choose to either drop this value or manually fix this value. I preferred to fix the value based on the values from other rows of the dataset with similar values in other columns.

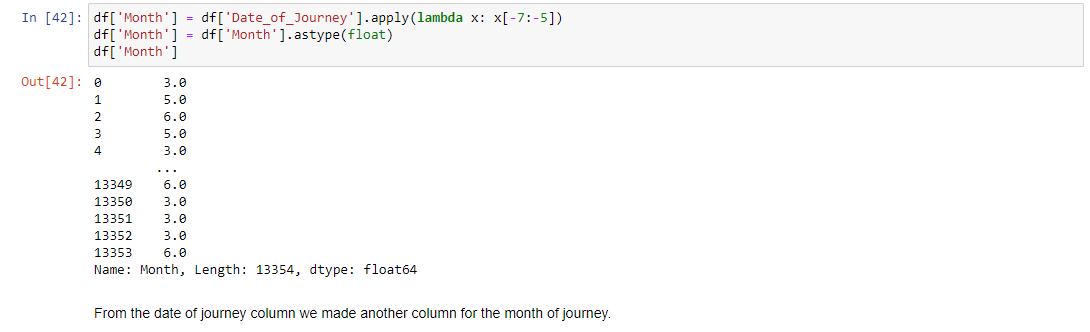


As we can see there are four more rows with similar values from which we can extract the route and total stops value for our missing value row.

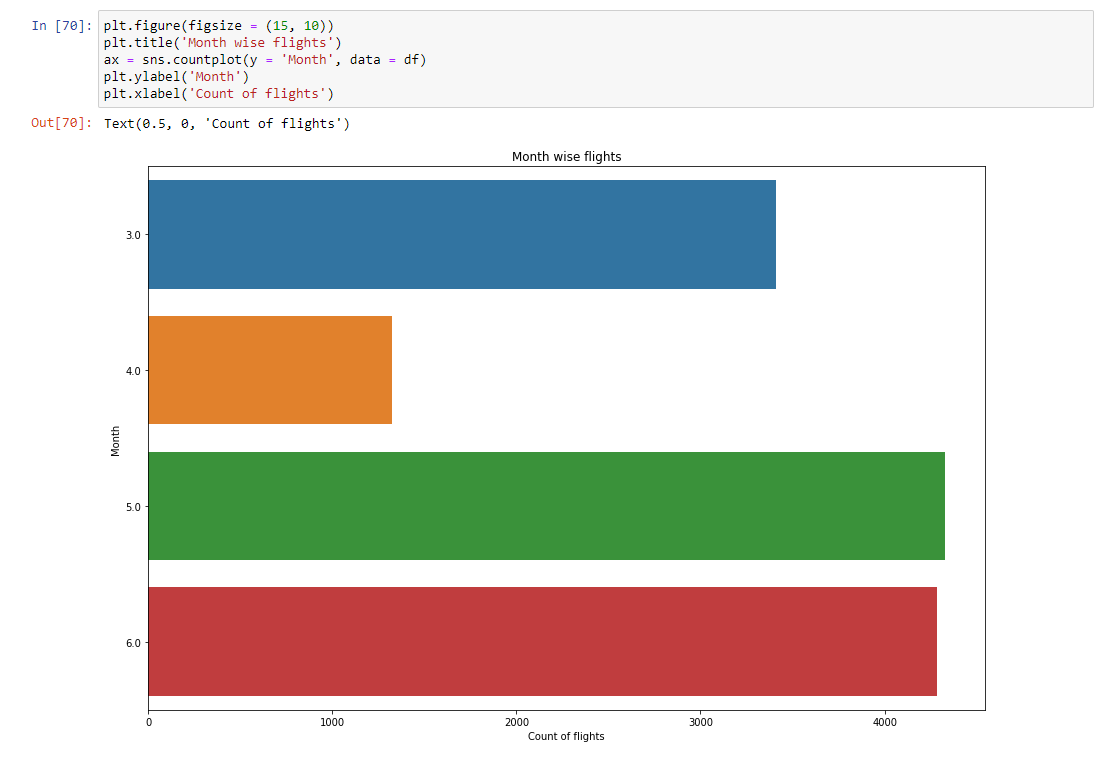
Now that we have dealt with the missing value problem of the dataset, let us now move to feature engineering since we have a lot of small changes to make in the dataset to make it easier to work with.

# Feature Engineering & EDA

First thing first, we will make a separate ‘Month’ column, values of which we will extract from the date column as I have mentioned above.

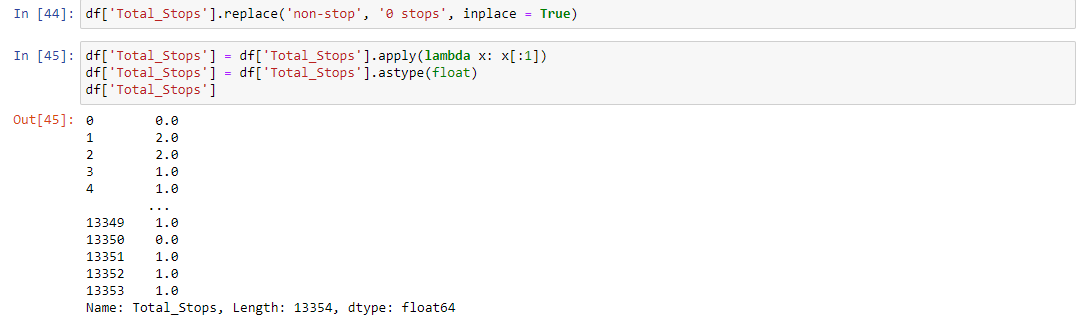


We can also use dt.month, it doesn’t make much difference. Let us take a look at the graphical representation of the month column.

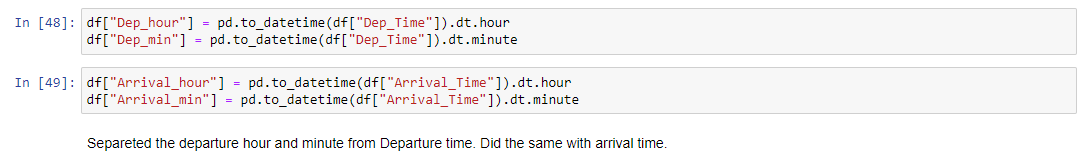


As we already knew from the problem statement, there were only 4 unique values for the month column since the data was acquired between March and July of 2019. We notice that the number of flights in April is the lowest and almost a quarter of the values from May and June. The high number of flights in May and June may be due to the fact that kids have summer vacations and families love to travel in that period.

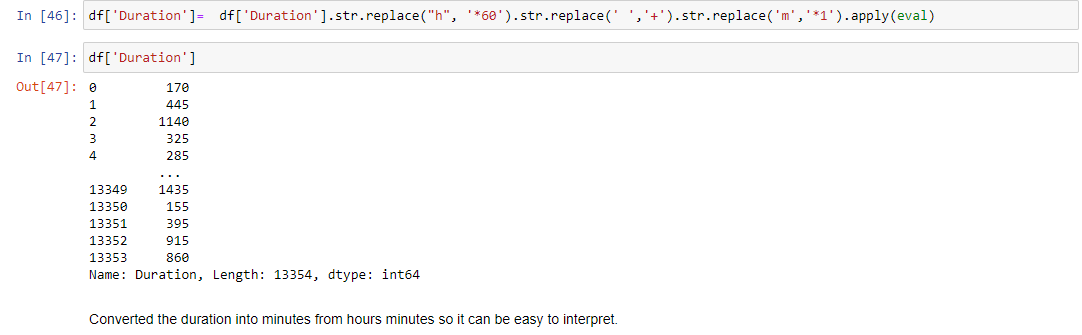
The next thing we did was fixed the values of Total stops. First, we changed the value ‘non-stop’ to 0 stops and then extracted the first character from the data and turned it into float to get our desired values for the column.



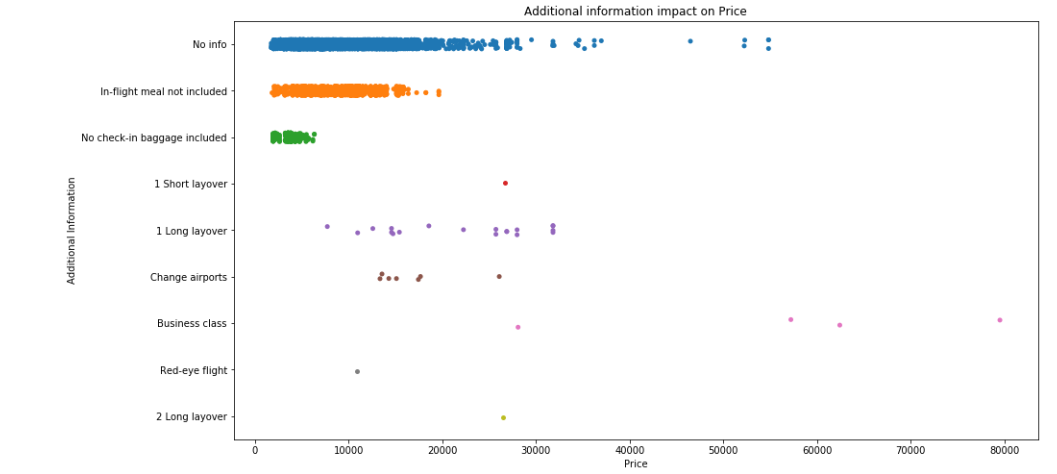
Arrival time was another column that was bugging us when we made our initial observations, so we used the similar method that we used in total stops and month columns to drop the values of date from the arrival time column. Now we used dt.hour and dt.minute to make two separate columns for arrival hour and arrival minute and repeated the same process for departure time, then we dropped both the arrival time and departure time column since we have already extracted their values into two separate columns each.



Another change that I implemented in the dataset was converting the values of the ‘Duration’ column into minutes from hours minutes and changing it from object data type into int data type. It is easier to process minutes alone instead of hours + minutes.

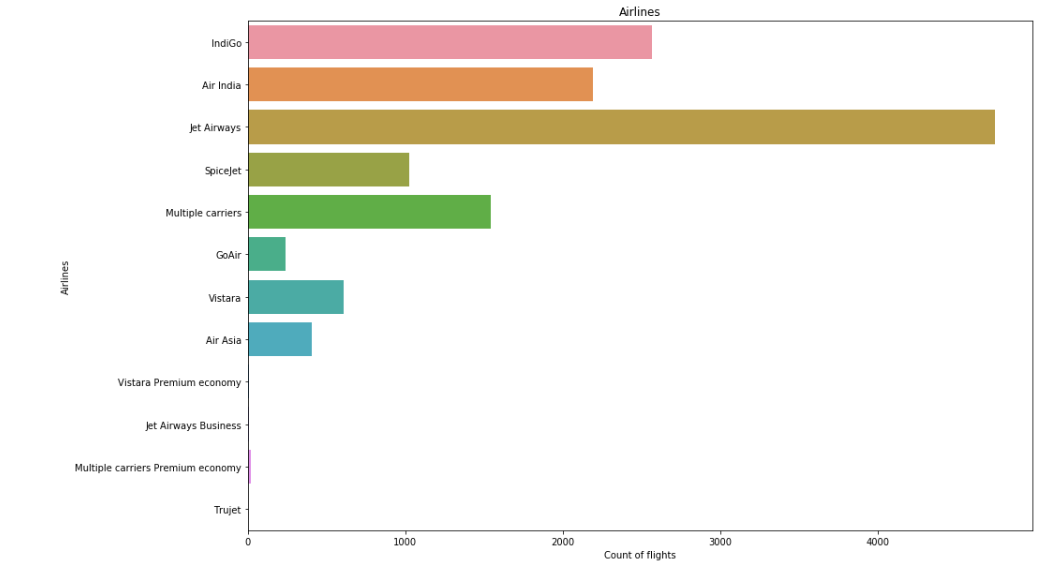


On taking a look at the ‘Additional Info’ column I realized that there were some unique values, which impacted the prices so there was no need to drop the column, there were some problems with the column like there were two values for no information which were ‘No Info’ and ‘No info’. I fixed this problem and then proceeded to study this column graphically.



As expected the value for No info is highest, apart from that we can notice that flights that do not have In-flight meals included and No check-in baggage included have low ticket prices. On the other hand, even though we do not have enough data to support this hypothesis, from the data available we can make this conclusion that the Business class tends to have high ticket prices, which is true.

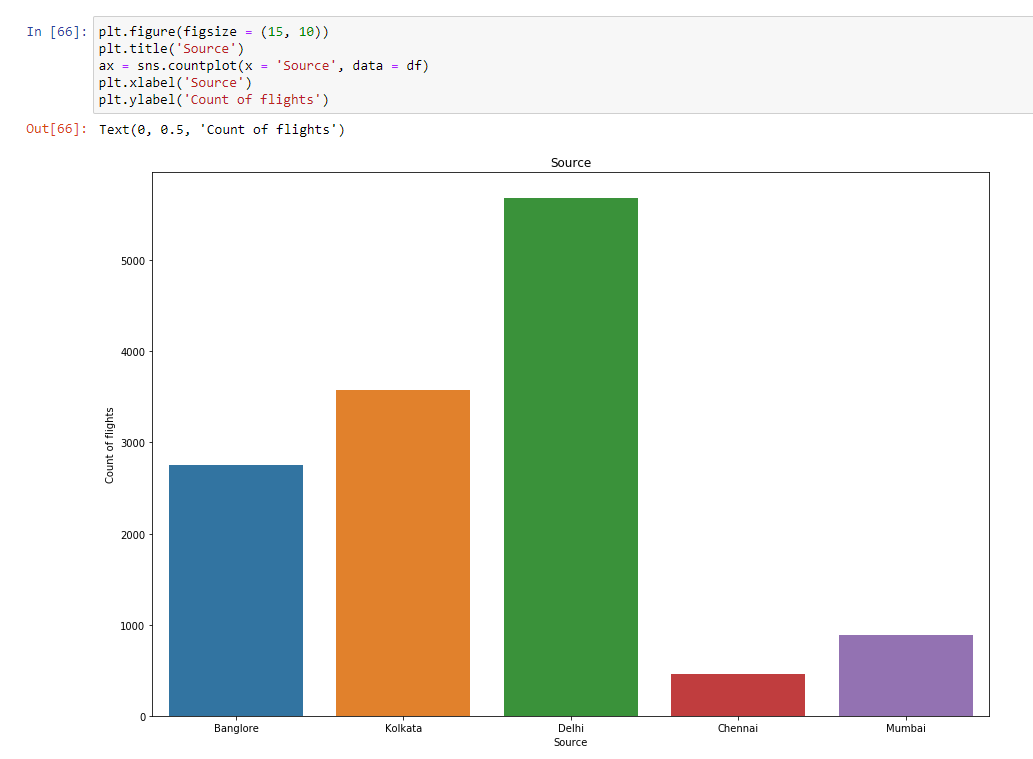
On studying the graph of Airline columns we can see that Jet Airways has the highest number of flights, almost double of IndiGo which is in second place. In comparison, the number of flights of Vistara Premium economy, Jet Airways Business and Trujet are almost negligible.



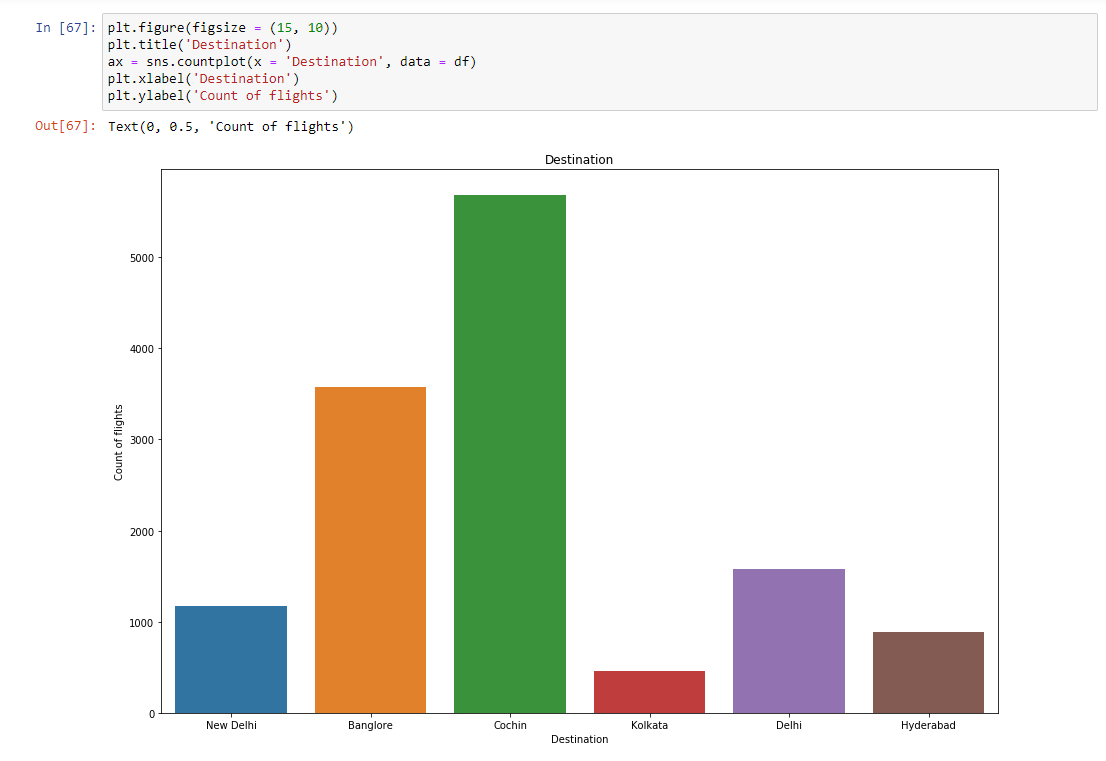


In fact Trujet only has 1 flight that flew from Mumbai to Hyderabad.

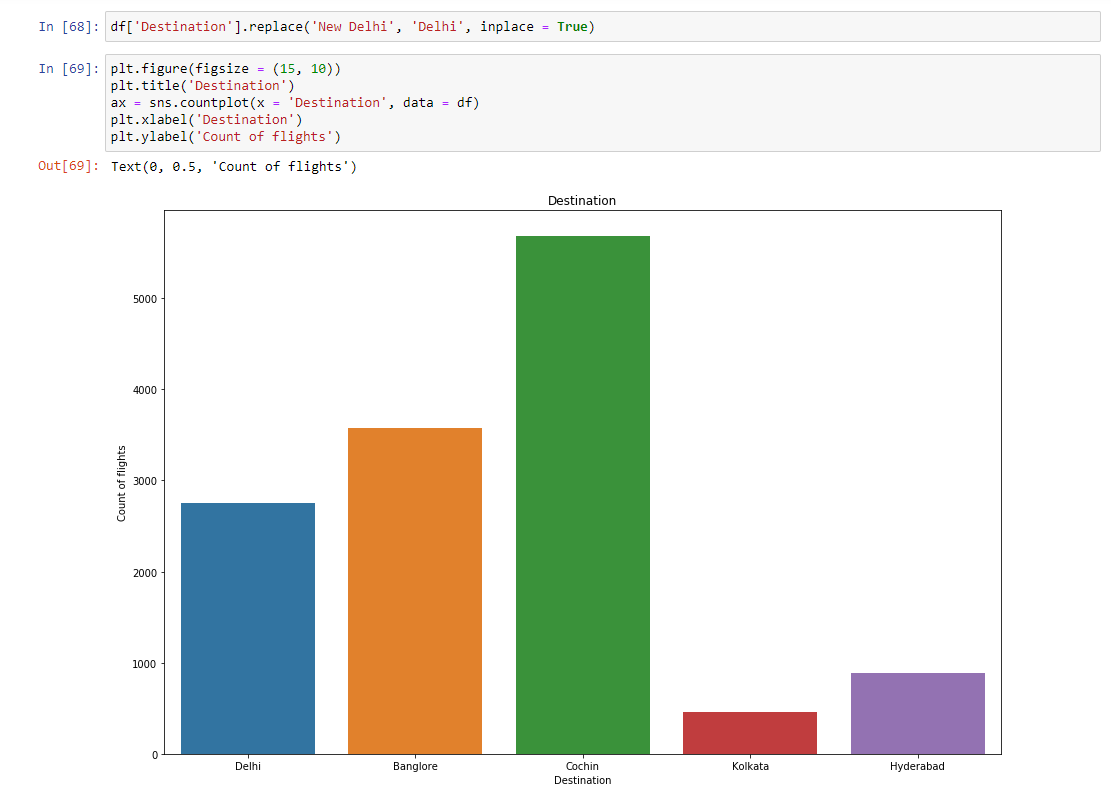
Now let us study the graphs of source and destination columns.



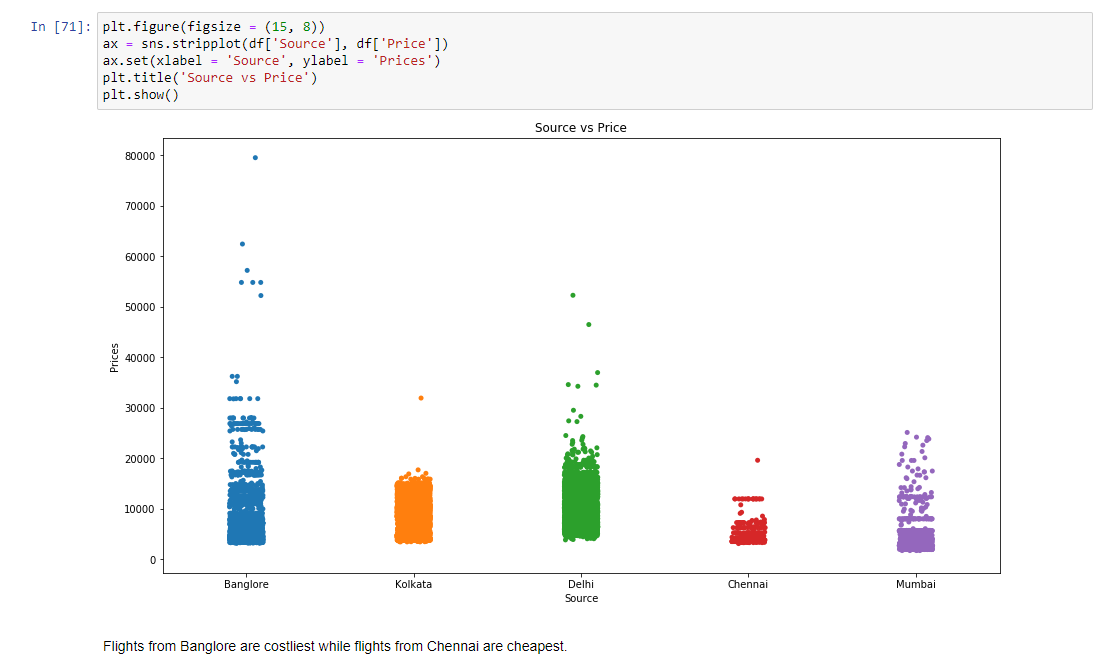
Very simple and easy to interpret graphs, most of the flights seem to take off from Delhi, while Mumbai and Chennai seem to have very low amounts of flights taking off from there.



The first thing to notice in the destination graph is that there are two different values for Delhi, that is, Delhi and New Delhi, so we need to fix this first and then proceed.



Delhi may have the highest count in Source but in Destination, it has a very low count in comparison. Surprisingly Cochin which is not even present in the Source column has the highest count in the Destination column, while Kolkata and Hyderabad have the lowest count. Mumbai and Chennai which had low values in the Source columns are not even present in the Destination column.



Taking a look at the Source vs Prices graph we can notice that Prices for flights from Delhi and Bangalore vary the most, they have prices ranging from lowest to highest, especially in Bangalore. Chennai with the lowest count of flights also seems to have the cheapest flight tickets.

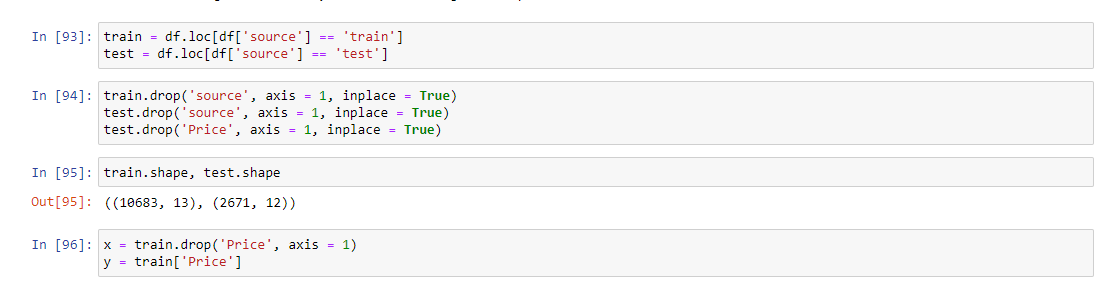
# EDA Concluding Remark

In conclusion, we can say that the number of flights is highest in the 2 summer vacation months while the tickets are cheaper in flights which do not provide certain services like check-in baggage and in-flight meals. We also noticed how the number of flights varied from city to city and Delhi being the source of most flights while Cochin seems to be the destination of most flights.

Since we have made the desired changes in the dataset by introducing new columns, changing and dropping some of the old columns, it is almost time for us to build the model to predict the prices. But first, let us change the object data types into float or int with the help of label encoder.

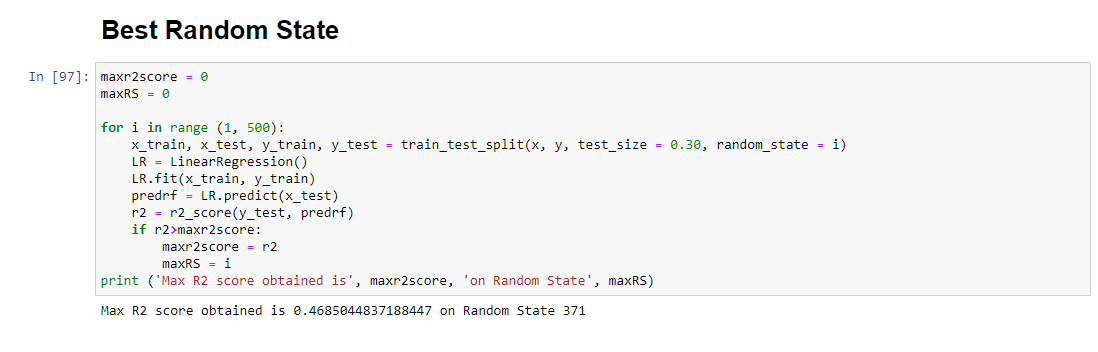


Now we will separate the datasets back to train and test like they were originally presented to us, we’ll make the separation based on the ‘source’ dataset that we introduced at the start while we were merging the two datasets. (Mind the difference between ‘source’ and ‘Source’)

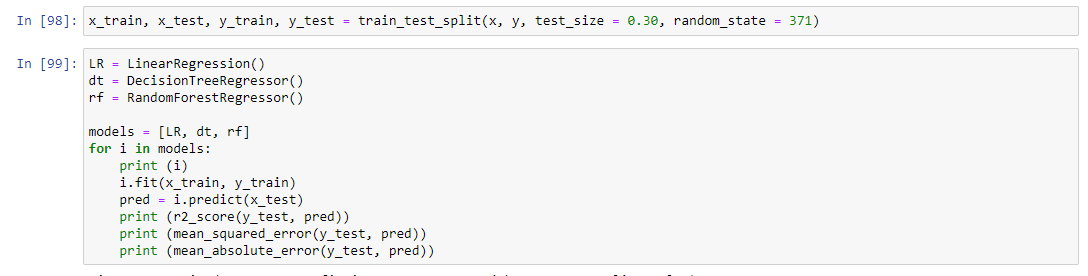


# Building Machine Learning Models

Now it is time for us to further split the train data set into train and test so we can build our model and train and test it. Before that I usually use a for loop to find the best random state for the train test split function, it is totally optional but it sort of gives the sense of the accuracy of the data.

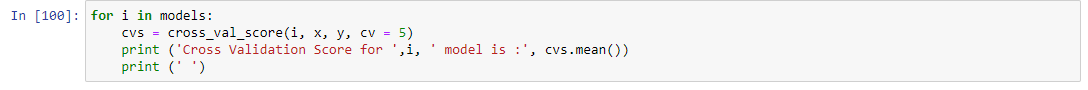


After we have found the value for the best random state, we will proceed with the train\_test\_split function to create training and testing datasets and fit them into models like Linear Regression, Decision Tree Regressor, and Random Forest Regressor to find our ideal model. For evaluation of our models, we will use R2 score, mean absolute error and mean squared error, all of which are available in sklearn.metrics. We will use a for loop to loop over all these models and use only 1 block of code instead of separate codes for each model.



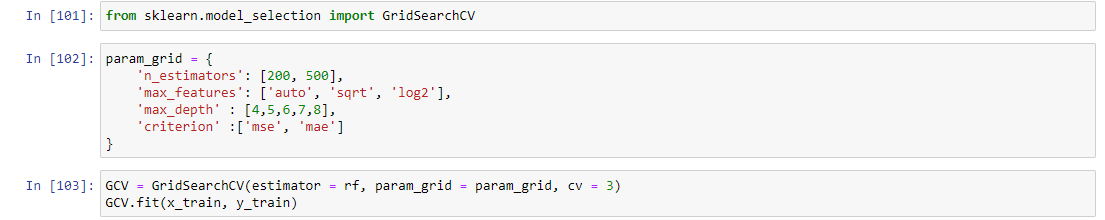
As observed above the Linear Regression model has the R2 score of 0.47 with mean squared error and mean absolute error 10305642.00 and 2440.93 respectively. As for Decision Tree Regressor, the R2 score was 0.69, and the values for mean squared error and mean absolute error was 6007927.87 and 1112.85 respectively. Now moving to the Random Forest Regressor model, the R2 score was 0.77, and the values for the mean squared error and mean absolute error were 4459968.56 and 1051.05.

As we can see, the Random Forest model has the highest R2 score and lowest error values, let us take a look at the cross val score of these models to avoid any overfitting in the models.



The cross val score in this case will evaluate the models based on the R2 score only, even though we have not mentioned it. The cross val score for Linear Regression, Decision Tree Regressor and Random Forest Regressor was 0.41, 0.65, and 0.73 respectively. Since the difference between the R2 score and cross val score is the least in Random Forest, we will work with it only.

Now we will hypertune the parameters of our Random Forest model with the help of GridSearchCV. After fitting the model and getting the best value for the parameters, we will fit the values of these parameters into our model.



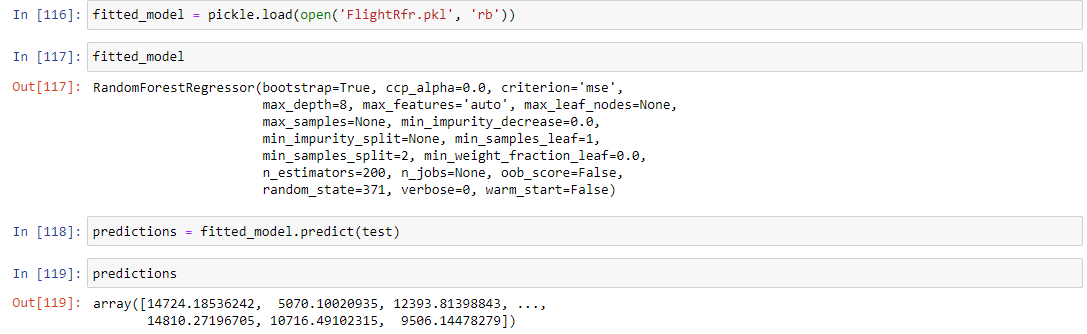


As we can see the R2 score of our model increased from 0.77 to 0.78 while the value for mean squared error also got reduced even though the value of mean absolute error increased slightly.

We are done with building our model and we can now save our model using pickle.



Now we will use this saved model to predict the price values for the test dataset and with this, we are done with this project.



# Conclusion

In this project we worked on the dataset comprising of flight data from four different months of summer, we noticed how the months of May and June being the 2 months of summer vacation have the highest count of flights while Delhi being the state which was the source of most flights while Cochin seems to be the destination of most flights. We also noticed how the prices vary from city to city and how the prices are impacted by the quality of flights for example flights with no check-in baggage services and no in-flight meals seem to be cheaper, while business class flights are costlier.