**Flight Price Prediction**

**Author:**

**PRATAP LAVATE**

**Problem Definition:**

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 traveller’s saying that flight ticket prices are so unpredictable. Here you will be provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

Size of training set: 10683 records

Size of test set: 2671 records

**Train Dataset Description:**

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

**Test Dataset Description:**

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.

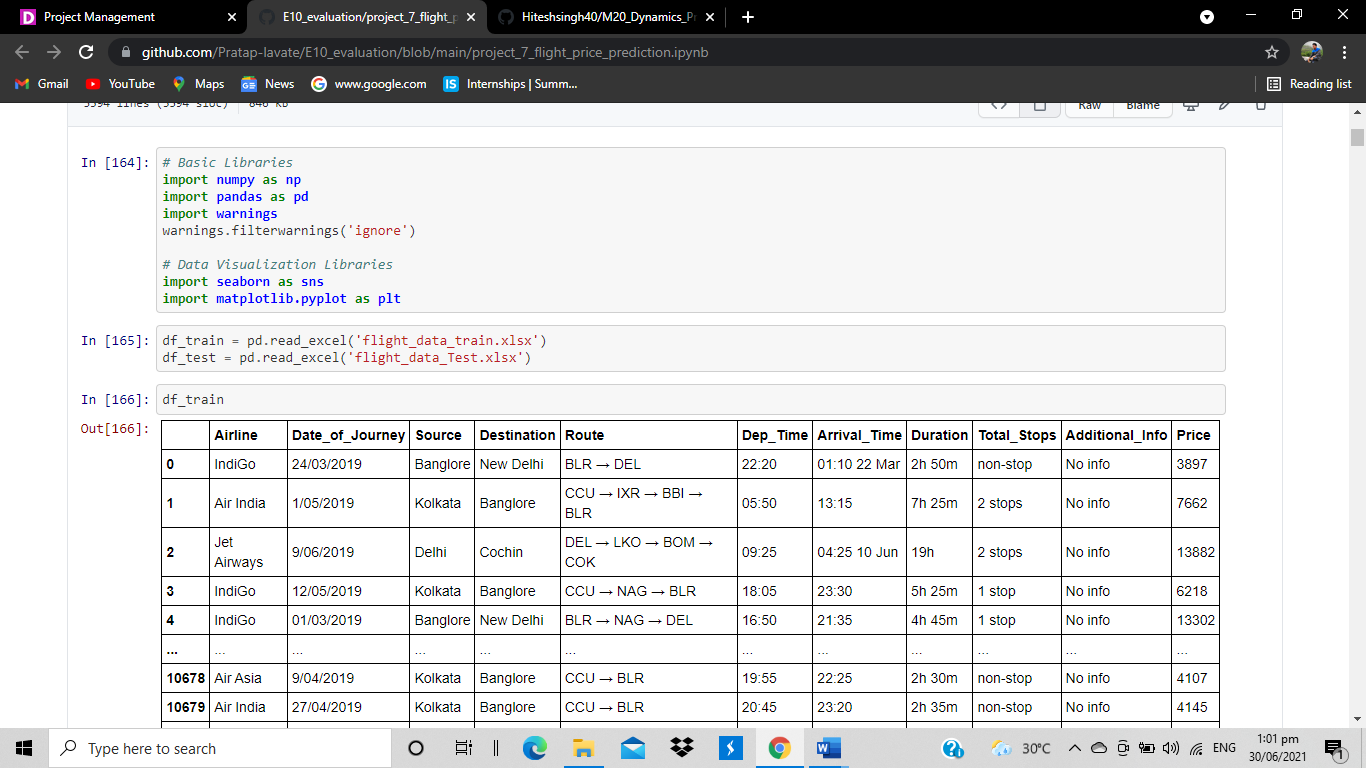
Total\_Stops : Total stops between the source and destination.

Additional\_Info : Additional information about the flight

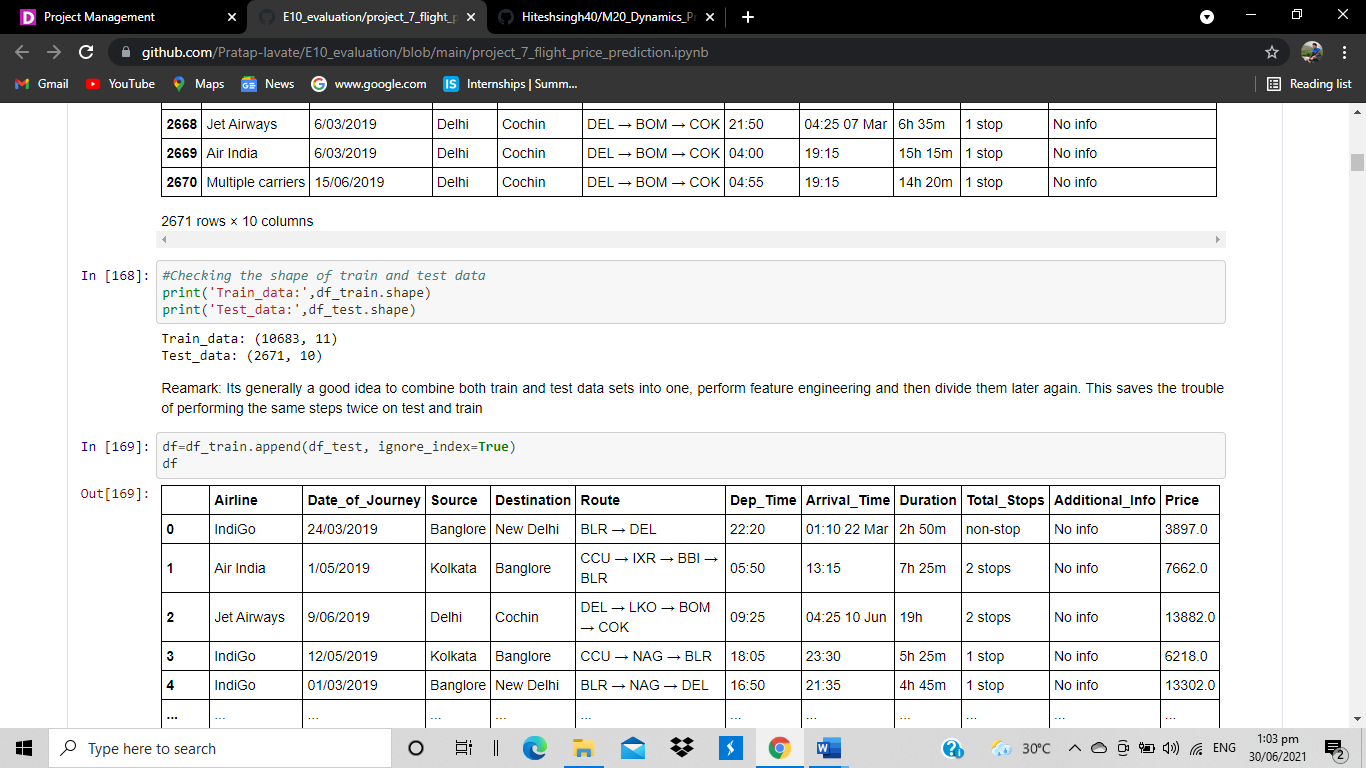
The principle objective is to build a model that can predict prices of flight tickets on the basis of the details provided in the data-set.

In this dataset two different datasets are given: train and test. Training dataset has dependent variable i.e. Price.

* **Data Analysis:**

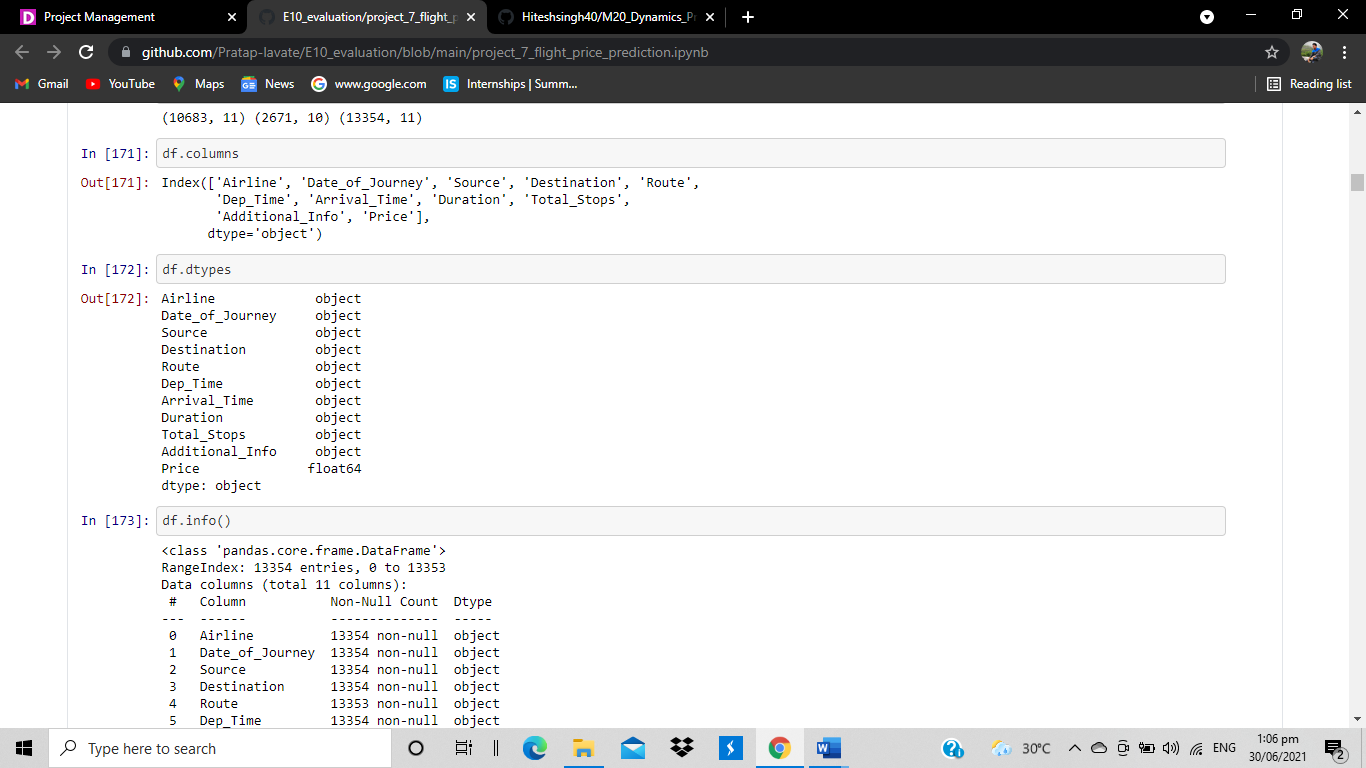
Firstly, import all libraries and load both train and test data-set into jupyter notebook using pandas.read\_csv function.

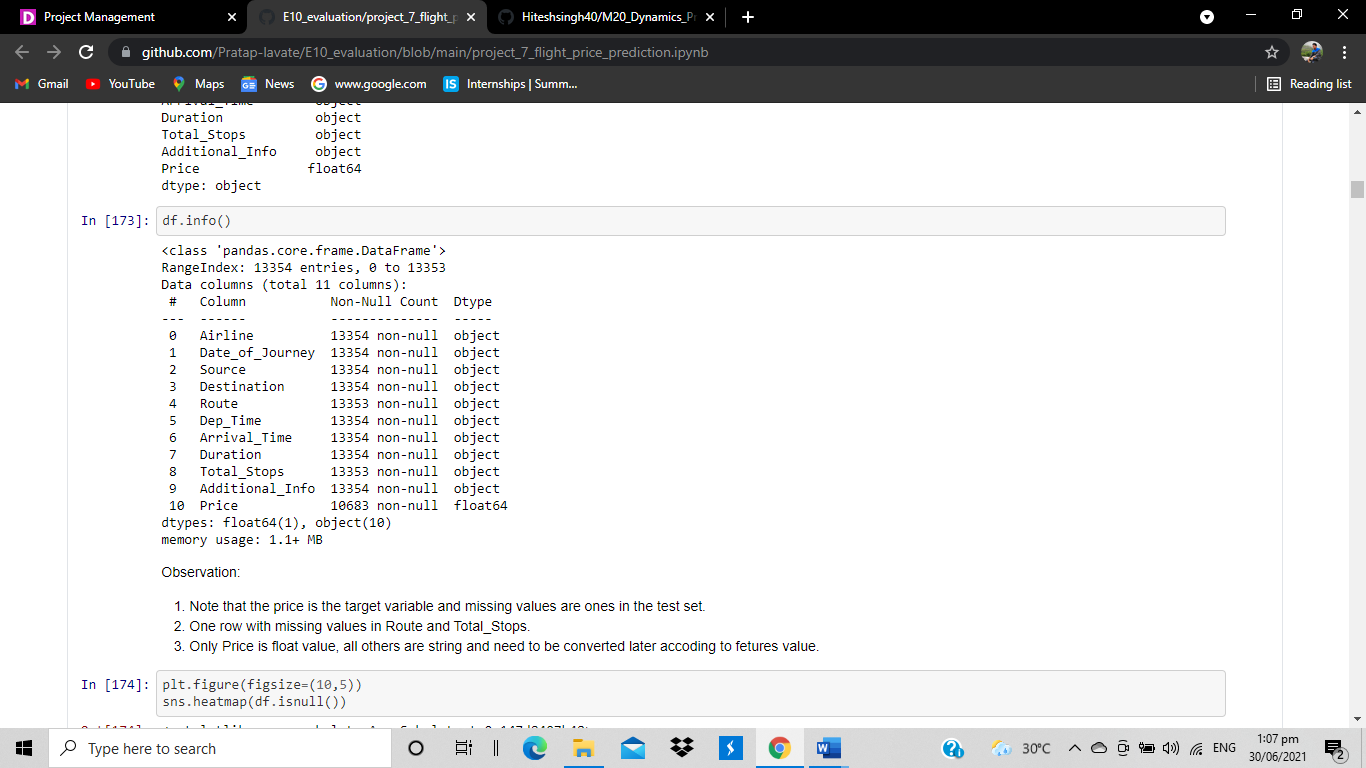
* Generally it is a good idea to add both train and test data, perform feature engineering and then divide them later again. This saves time and complexity of performing same step twice once on train data and same on test data. So, Let’s combine them into a data frame ‘data’ with a ‘root’ column specifying where each observation belongs.

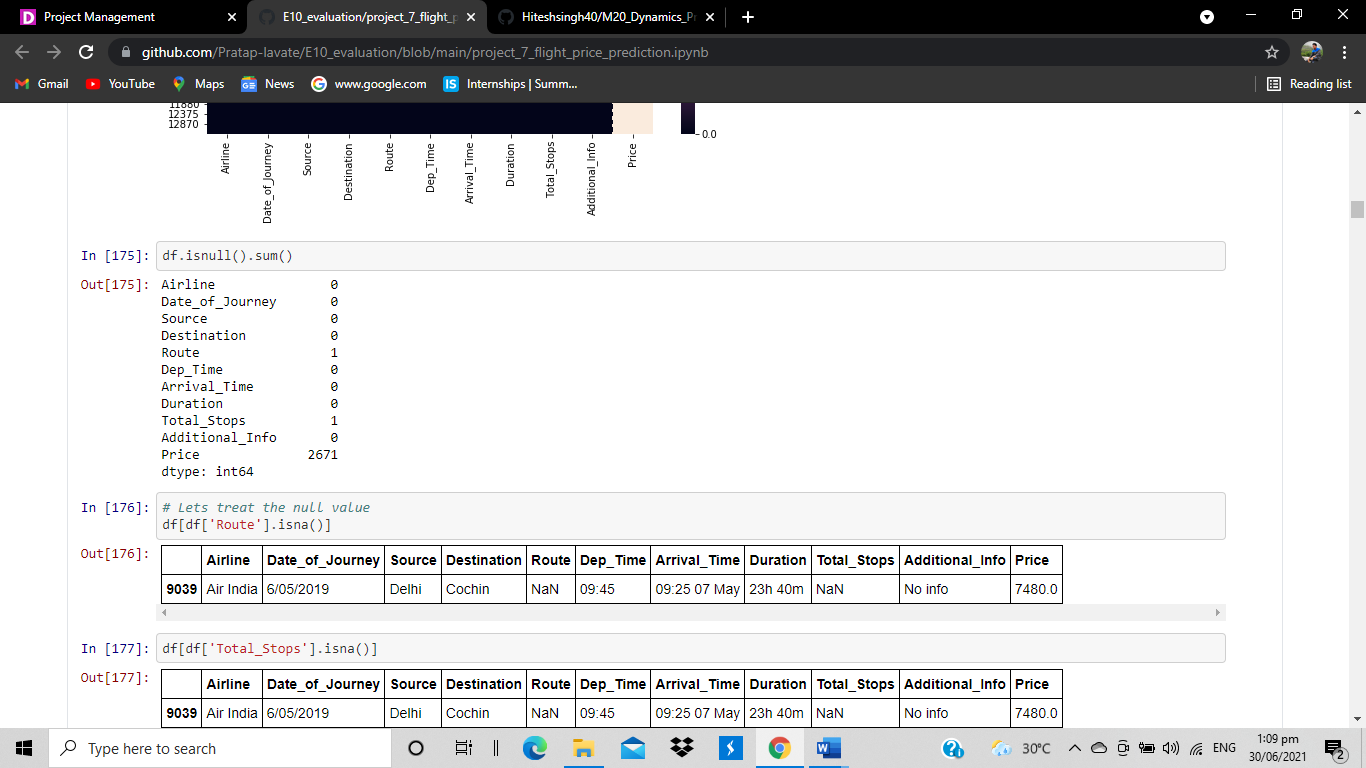


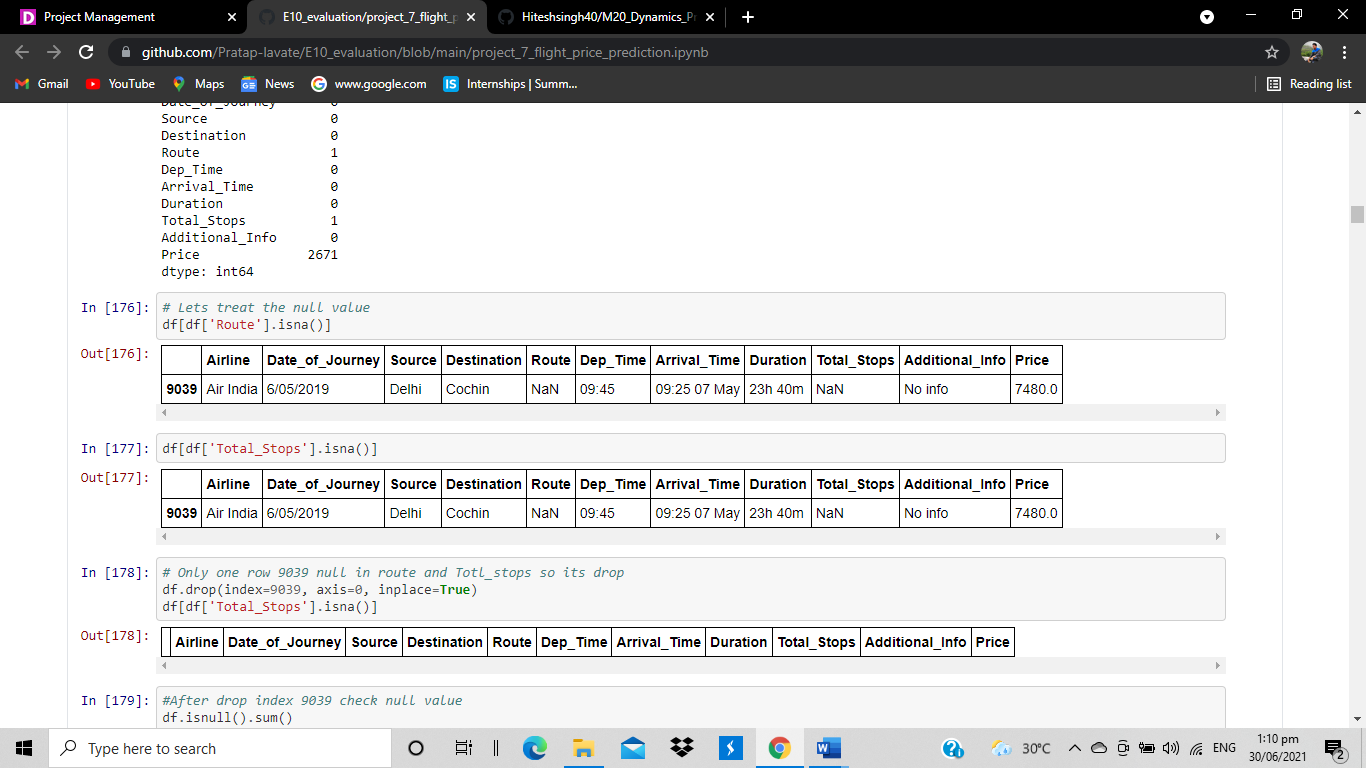
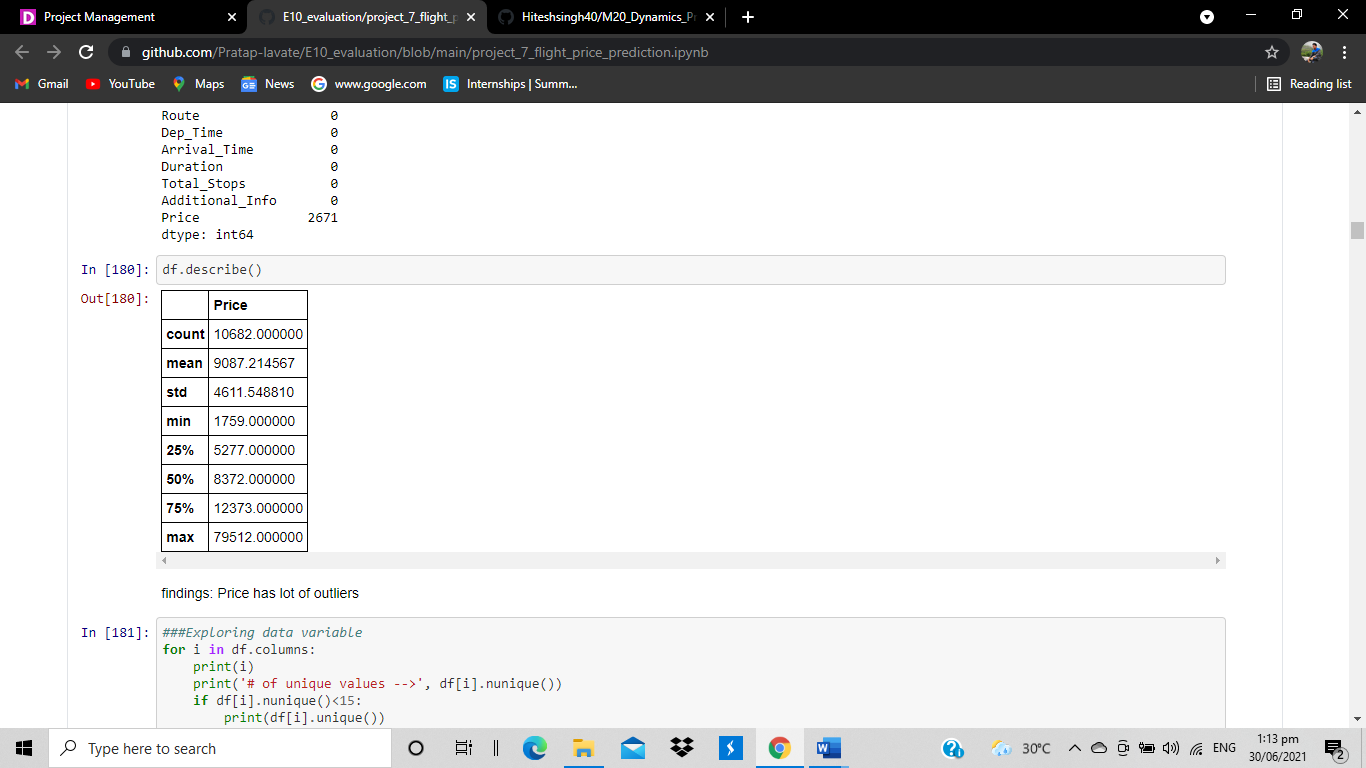
* We can see that, training set data have 10683 rows and 11 columns and testing data have 2671 rows and 10 columns. Whereas adding data with both training and testing set has 13354 rows and 11 columns. Test data has 10 columns as it doesn't contain output variable 'Price'.

On checking for data type of adding dataset, we observed that All the attributes has data types as object except for Price. However, Date cannot be object type. Attribute 'Date\_of\_Journey','Dep\_Time','Arrival\_Time' are object type which is a problem over here, as date can not be object type.



* On Checking for null values/ missing values in the data-set it was observed that data set has many null values .

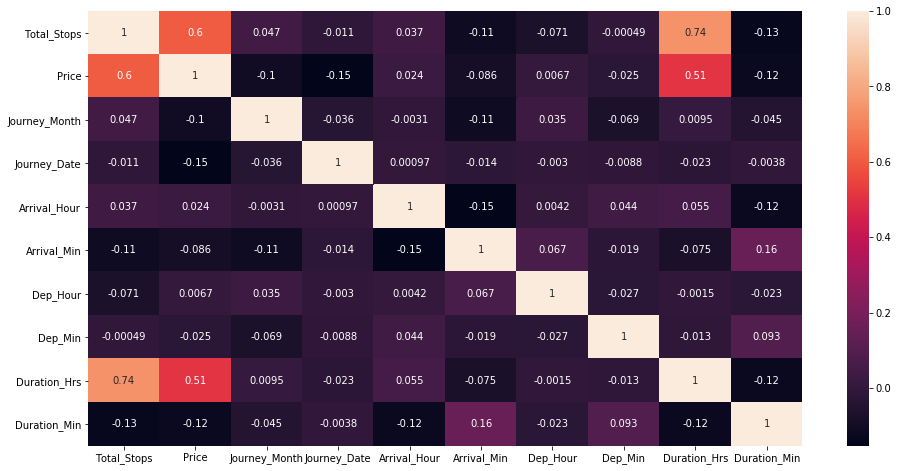
1. Note that the price is the target variable and missing values are ones in the test set.
2. One row with missing values in Route and Total\_Stops.
3. Only Price is float value, all others are string and need to be converted later accoding to fetures value.

* Then we checked the null values of given features and it’s a only column in that feature so we drop that column using drop function.
* We use describe function to see insights from the feature.

We see that total 10682 counts with mean 9087.21 (average price of flight ticket is 9087). Variation is 4611 is huge, minimum price of ticket is 1759 and maximum price of ticket is 79512.

Price has lot of outliers

* Correlation heatmap



Observation:

1. Total\_Stops and Price are postive coorelated.
2. Total\_Stops higly positive correlated with Duration\_Hrs and negative correlated with Duration\_Min.
3. Price is postive correlated with Duration\_Hrs

* **Exploratory data analysis:**
* Counts and countplot

1) **Airline**

Jet Airways 4746

IndiGo 2564

Air India 2191

Multiple carriers 1543

SpiceJet 1026

Vistara 608

Air Asia 405

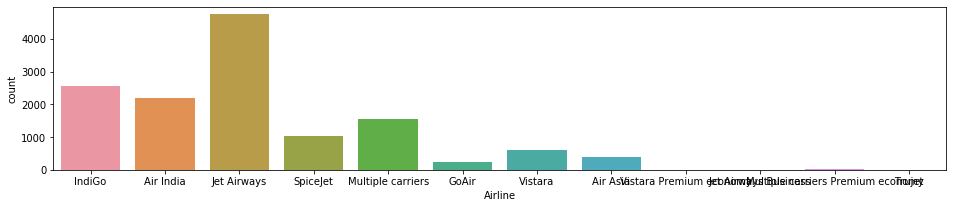
GoAir 240

Multiple carriers Premium economy 16

Jet Airways Business 8

Vistara Premium economy 5

Trujet 1



2**)Destination**

Cochin 5681

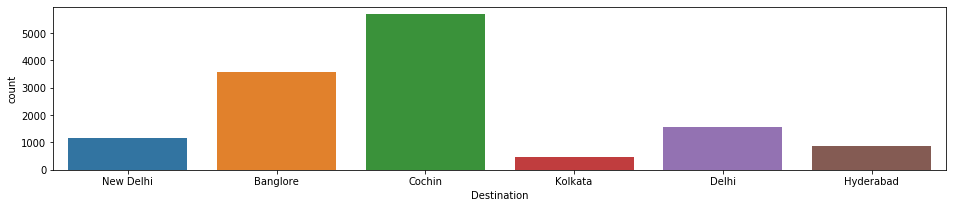
Banglore 3581

Delhi 1582

New Delhi 1170

Hyderabad 883

Kolkata 456



3**)source**

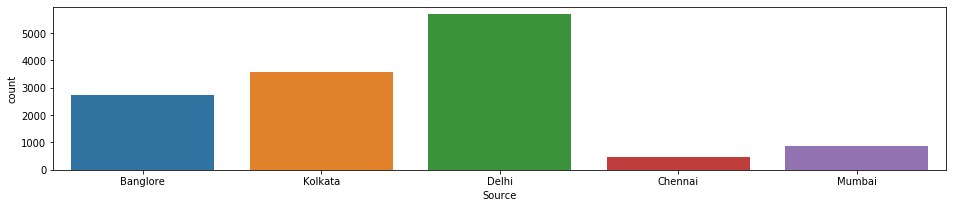
Delhi 5681

Kolkata 3581

Banglore 2752

Mumbai 883

Chennai 456



Interpretation:

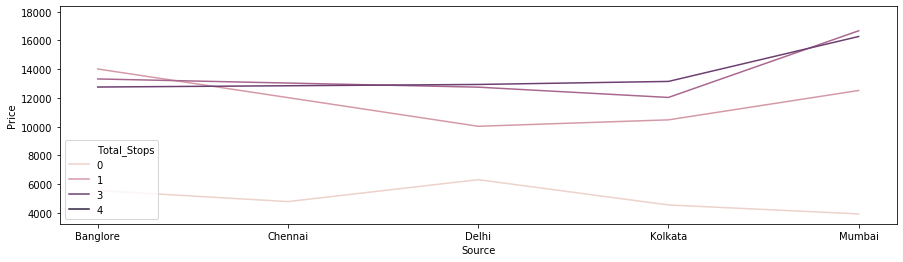
1) Jet airways has maximum count with 4746

2) maximum count in destination is cochin with 5638 counts

3)maximum count in source is delhi with 5681 counts

* **Multivariate analysis:**

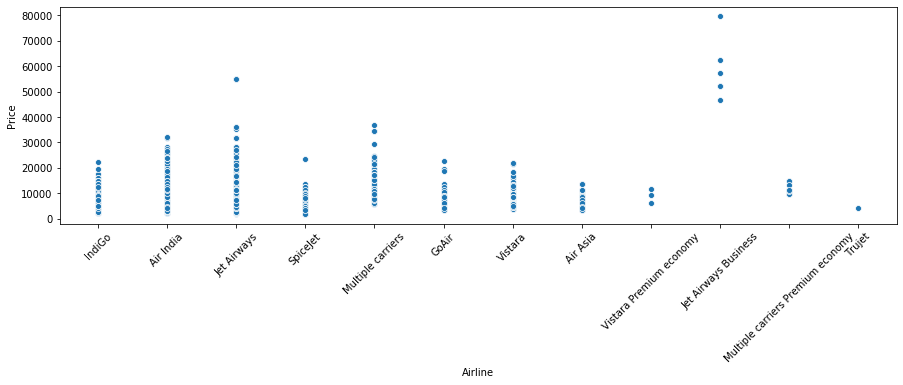
Line plot of source vs price with total step as a line



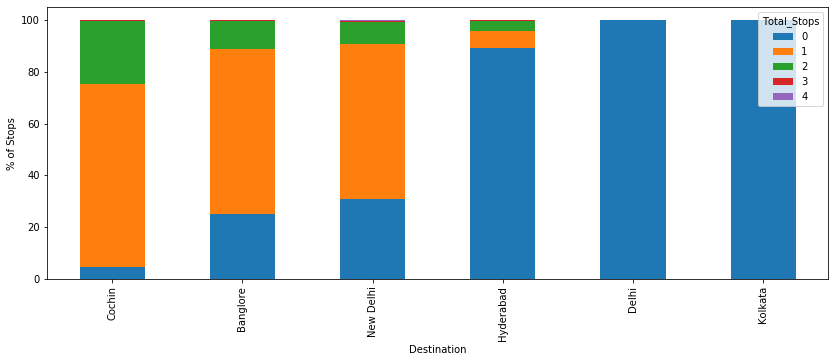
Observation:

1. If Total stop is 0 than journey start from Delhi than ticket price is more than other sourses.
2. If total stops more than 1 than ticket price is more than other source.

* Scatter plot of airlines vs price, on x axis airline and on y axis price is there,

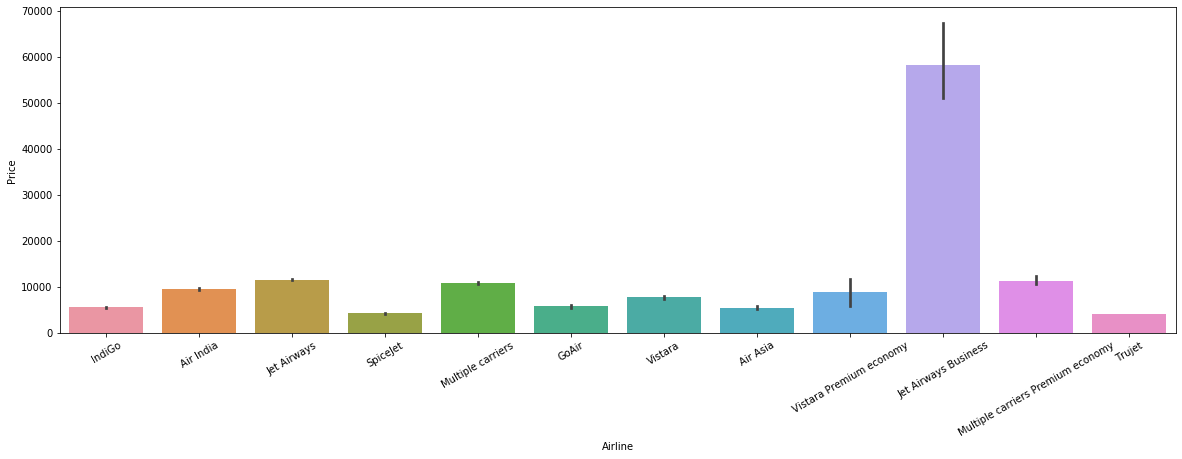


* 1. jet airways bussiness is minimum count with higher price 2) spicejet has minimum price
* Cross plot of destination and total stops



1. For travelling to Cochin without stops not any flight from source.
2. Fo Delhi and kolkata more flight is available from sourse station without stop.

* Bar plot of airline and price

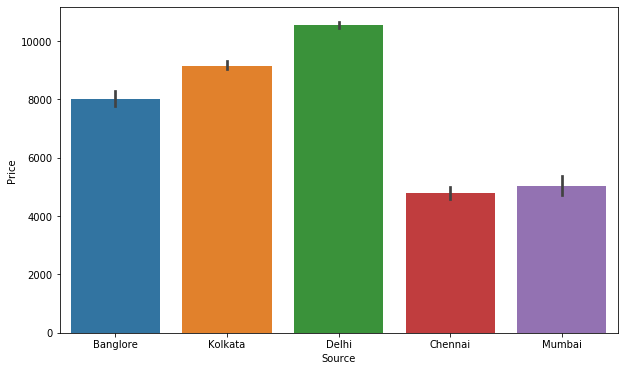


Conclusion: 1)jet airways business has maximum price

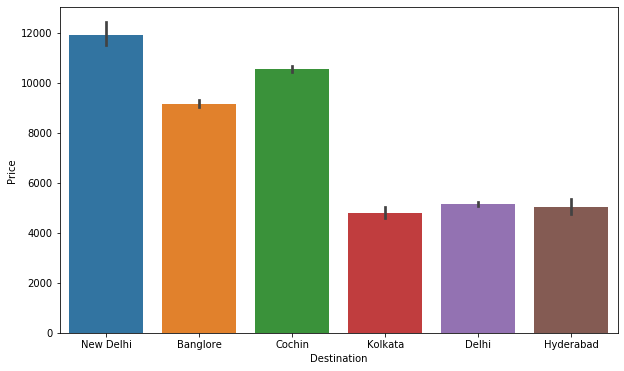
2)indigo,spicejet, trujet has minimum price

* Barplots (vs price)

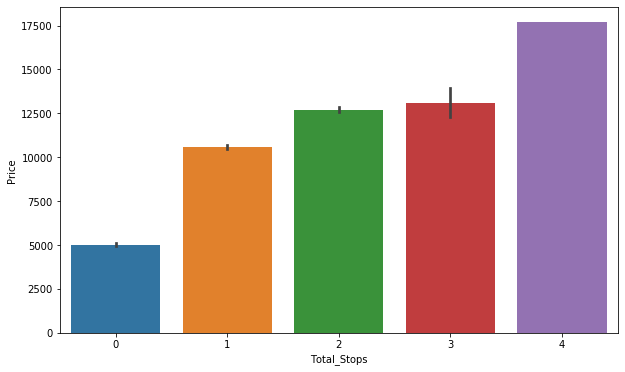
1)Bar plot of source and price



* 1. Bar plot of destination and price



* 1. Bar plot of total stop and price



Conclusion of barplot vs price

1) In source delhi with maximum price and Chennai has minimum price

2) For destination delhi has maximum price and cochin has minimum price

* 1. As stops increases price also increases
* **EDA Concluding Remark:**

Jet Airways Business is the costliest of all airlines. But it can also be seen that, price for Jet airways Business is fixed in respective of parameters like weekends, holidays etc. Trujet has least price. But frequency of Trujet is only 1 for the year 2019. Rest all airlines have variable prices. Prices increase with decreasing days, weekdays, holidays etc.

Most of the flight take off from Delhi has the highest price. Mumbai & Chennai has least of all sources. Prices in Chennai are fixed. Prices vary very drastically from source Delhi and Bangalore. Cochin has destination for many flights with highest of all price. Kolkata and Hyderabad has least price of all destination cities. Prices in kolkata are fixed. Prices vary very drastically from source Delhi and Cochin.

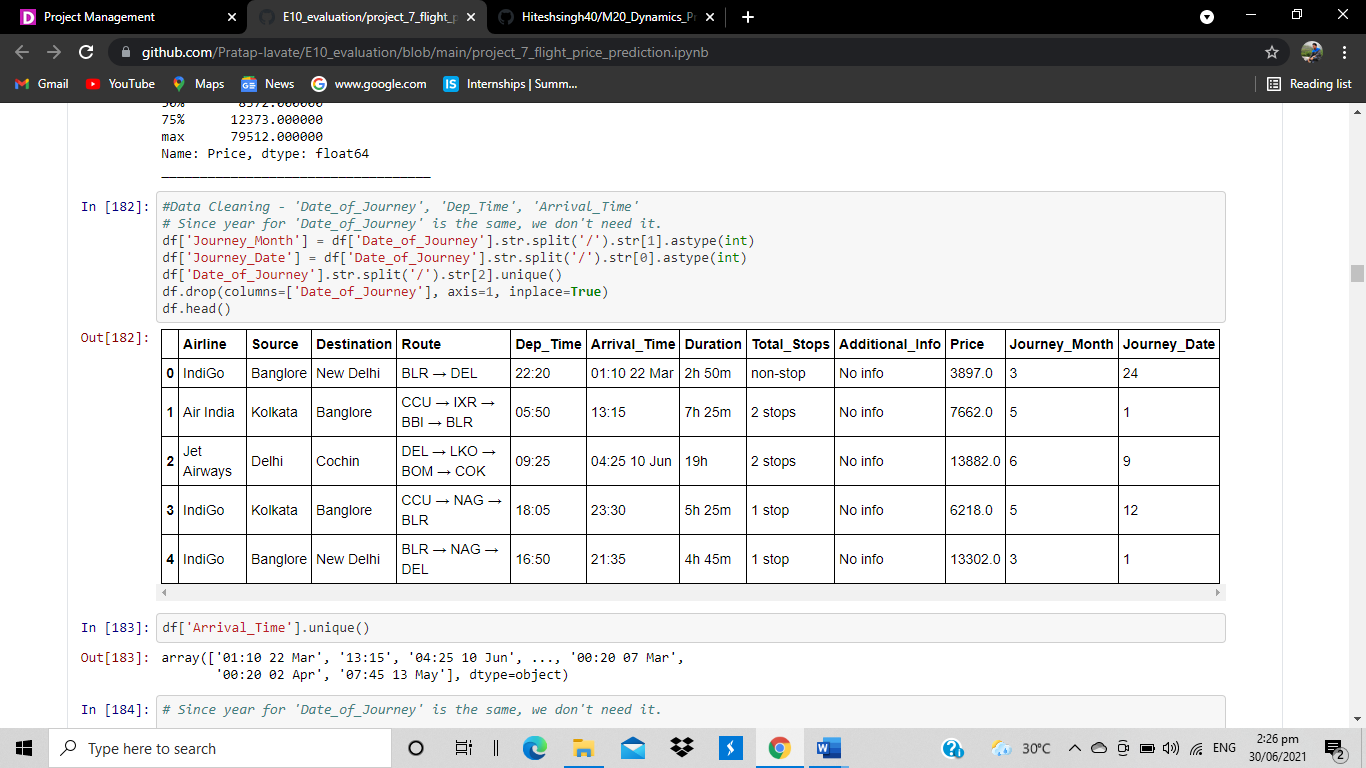
Higher the no. of stops, higher is the flight price. 4 stop flight has higher ticket price of all. Non-Stop flights have least price of all. However 4 stop flights have fixed flight tickets. Whereas for others prices vary drastically.

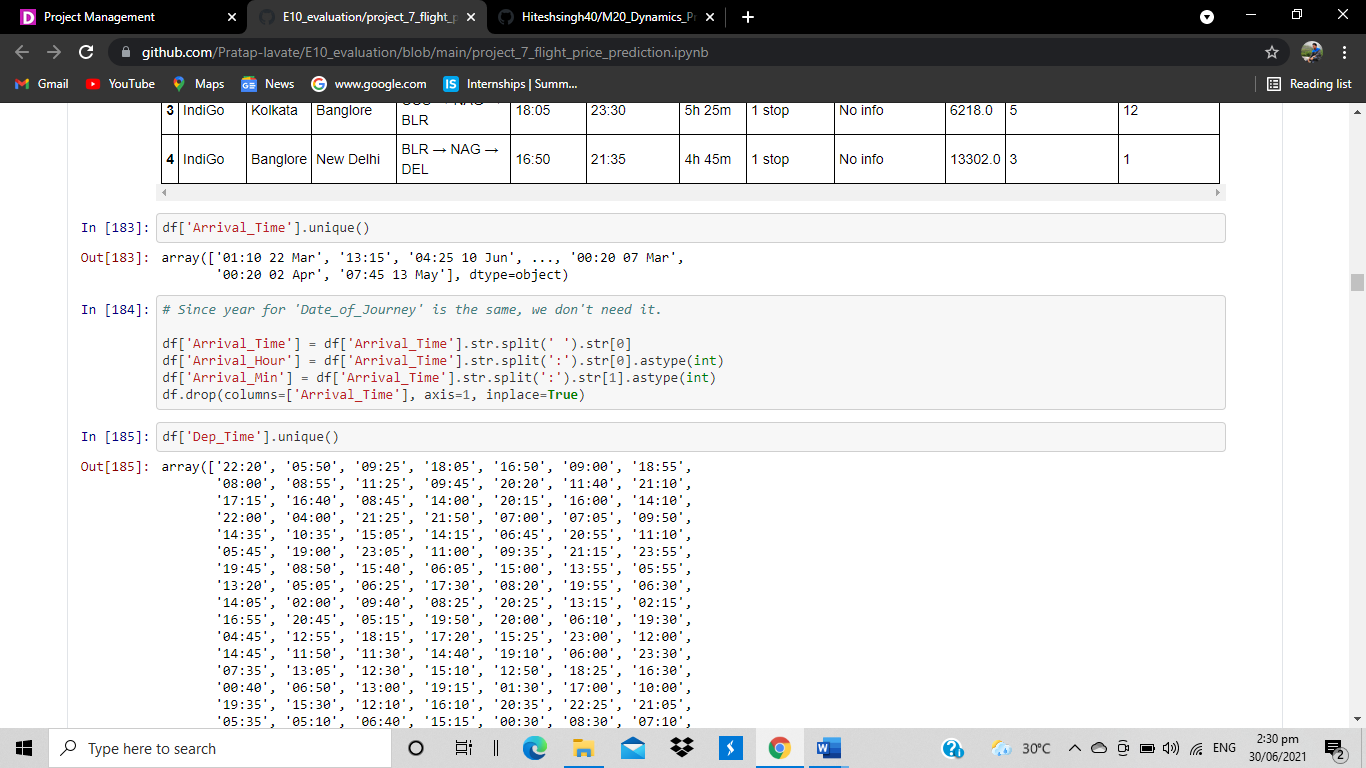
Flight running in January month has highest price. Month of May, June, September, December have nearly equal price. March month has comparatively lower price. 3rd day of month has the highest price for year 2019. Rest all other days have more or less same amount of flight price.

* **Pre-Processing Pipeline:**

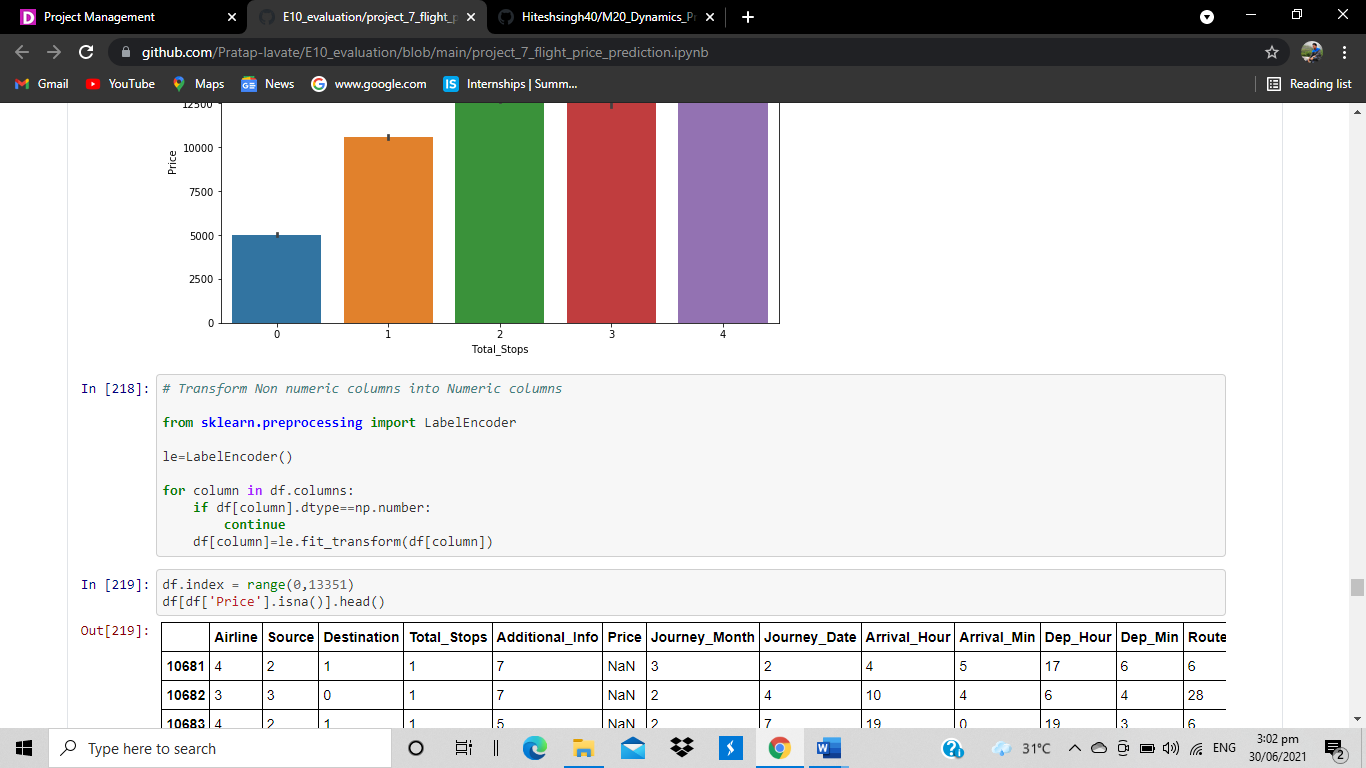
On checking for data type of dataset, All the attributes has data types as object except for Price. However, Date cannot be object type. Attribute 'Date\_of\_Journey','Dep\_Time','Arrival\_Time' are object type which is a problem over here and we need to convert it to date-time stamp.

So, firstly converting 'Date\_of\_Journey' from object type to datetime. And performing feature engineering on the same by creating new attributes of month, day and year. And finally deleting base attribute date\_of\_journey

Repeating the same procedure for Arrival\_time and dep\_time. Creating two new attributes from arrival\_time named Arrival\_hour and Arrival\_min and deleting the former one. Performing feature engineering on attribute Dep\_Time by creating two new attributes from the same named Dep\_Hour and Dep\_Min. And finally deleting the former one.

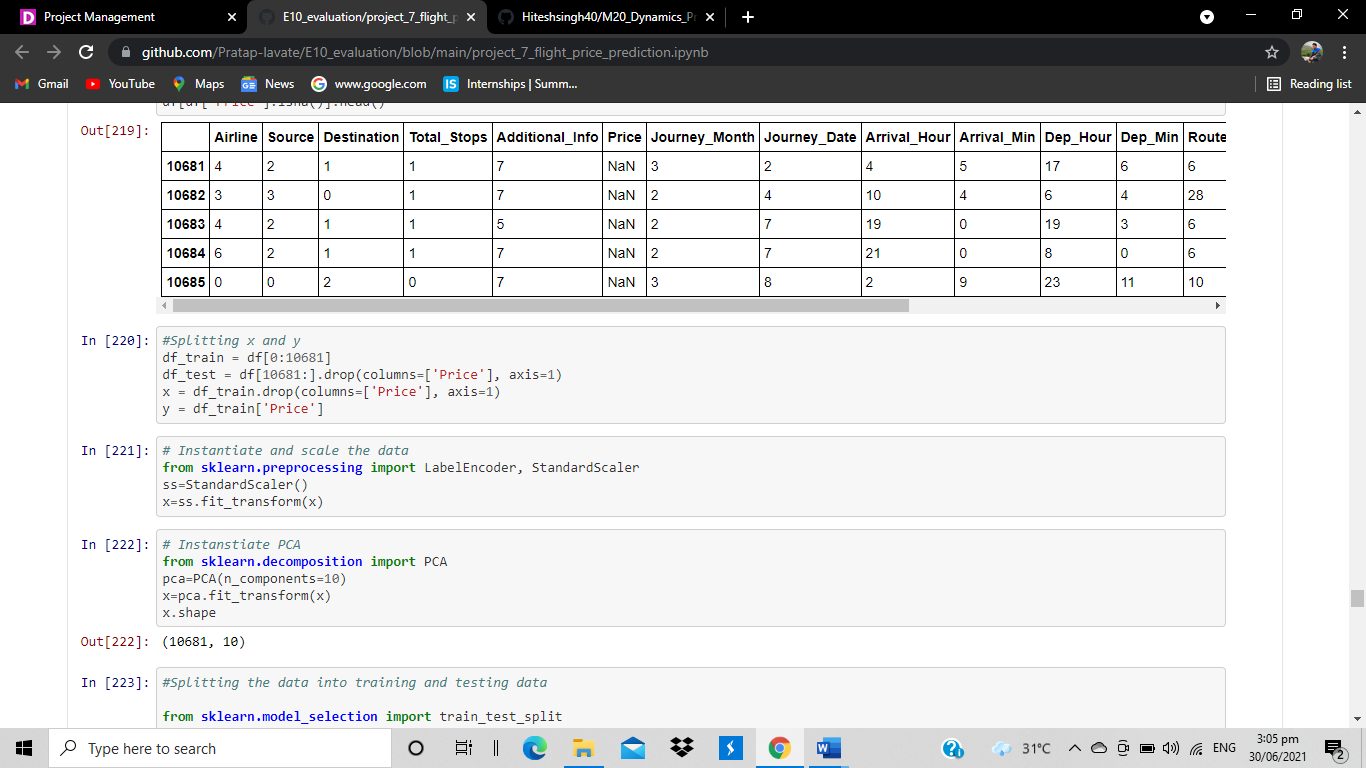


Next step is encoding categorical (object type data type) to numeric form before fitting and evaluating the model. Encoding is a required Pre-Processing step when working with categorical data for machine learning algorithms. Machine learning models require all input and output variables to be numeric. Using label encoding, each unique categorical value will be assigned a integer value. Every object type data is converted into int type value.

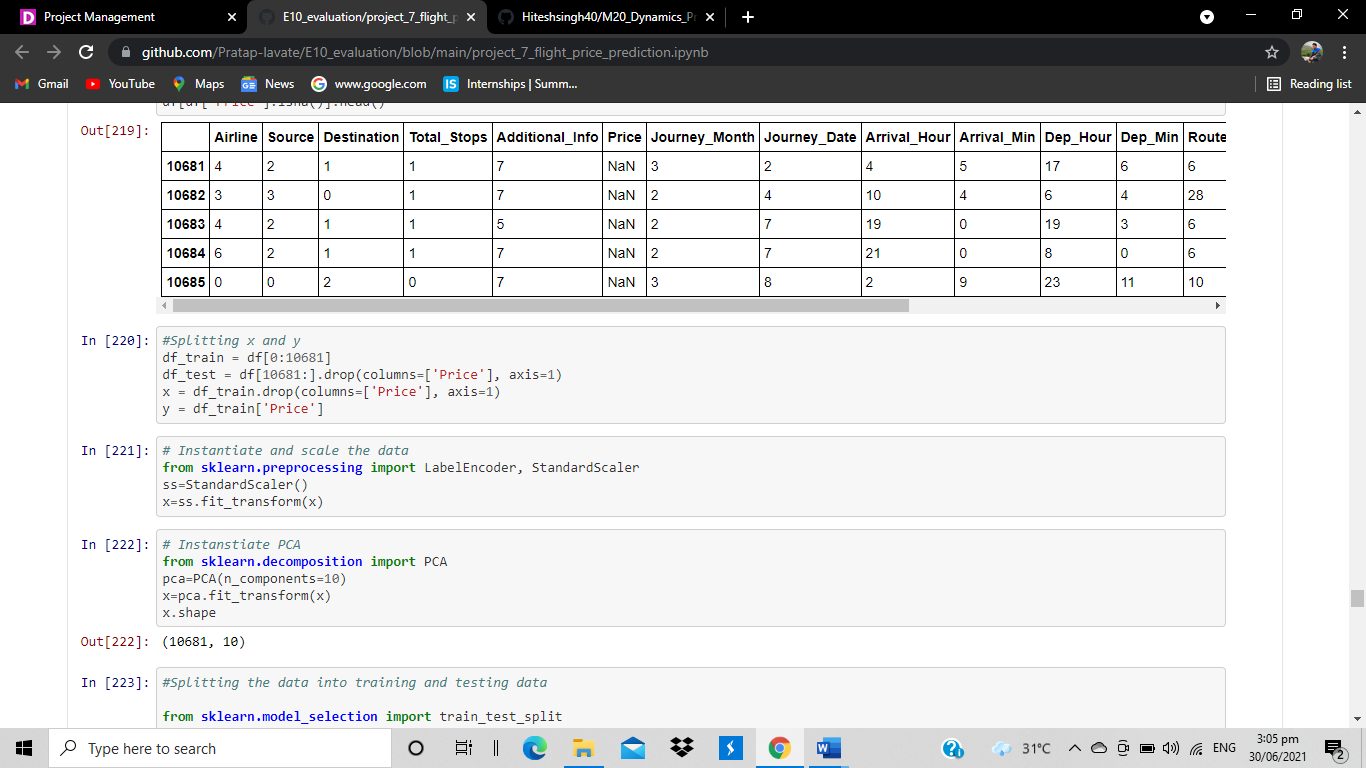


* **Building Machine Learning Models:**

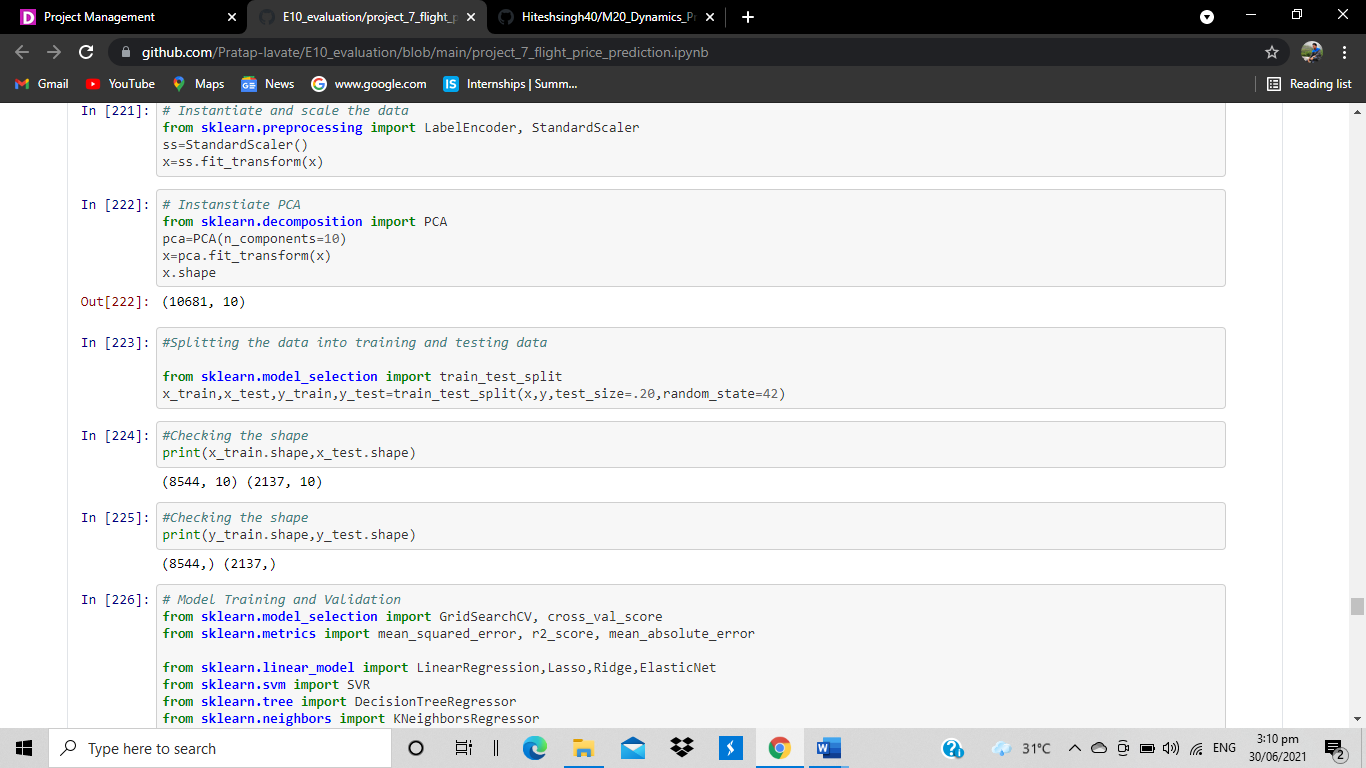
Formerly we have concated train and test set into one dataset ‘data’ using an extra attribute ‘root’ to identify where each observations belong. Now dividing dataset back into train and test dataset and dropping uncessary columns Price and root form test set and root from train set.



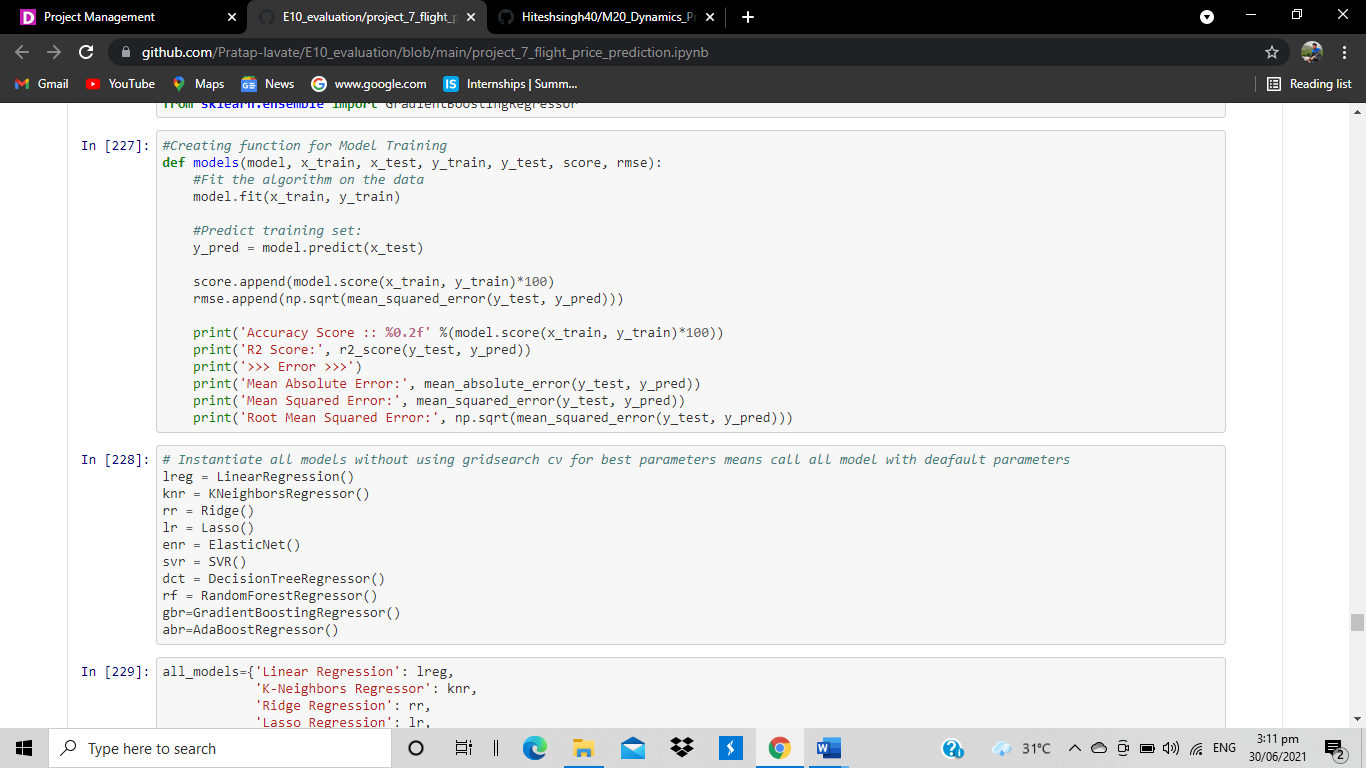
Using standard scaler to Transform features by scaling each feature to a given range



Splitting data into training and testing



Then we create a function for model training in that we compute accuracy score, R2 score, mean absolute error, mean squared error, root mean squared error



------------- Linear Regression ------------

Accuracy Score :: 43.48

R2 Score: 0.4673987970433001

>>> Error >>>

Mean Absolute Error: 2374.586532721971

Mean Squared Error: 11244149.300025102

Root Mean Squared Error: 3353.2296819670887

------------- K-Neighbors Regressor ------------

Accuracy Score :: 83.66

R2 Score: 0.8149846755766669

>>> Error >>>

Mean Absolute Error: 1159.734581188582

Mean Squared Error: 3905999.3087880206

Root Mean Squared Error: 1976.3601161701326

------------- Ridge Regression ------------

Accuracy Score :: 43.48

R2 Score: 0.4673965645208019

>>> Error >>>

Mean Absolute Error: 2374.59512823957

Mean Squared Error: 11244196.432506489

Root Mean Squared Error: 3353.2367098829286

------------- Lasso Regression ------------

Accuracy Score :: 43.48

R2 Score: 0.46737499851548125

>>> Error >>>

Mean Absolute Error: 2374.589843503917

Mean Squared Error: 11244651.728856336

Root Mean Squared Error: 3353.304598281572

------------- Elastic Net ------------

Accuracy Score :: 41.63

R2 Score: 0.4417584197378458

>>> Error >>>

Mean Absolute Error: 2466.3215239043716

Mean Squared Error: 11785462.817401707

Root Mean Squared Error: 3432.9961866279004

------------- Support Vector Regression ------------

Accuracy Score :: 6.57

R2 Score: 0.0736123557705829

>>> Error >>>

Mean Absolute Error: 3394.920807681104

Mean Squared Error: 19557674.529437654

Root Mean Squared Error: 4422.405966149835

------------- Decision Tree Regression ------------

Accuracy Score :: 99.60

R2 Score: 0.565747956863625

>>> Error >>>

Mean Absolute Error: 1405.265091249415

Mean Squared Error: 9167825.344291063

Root Mean Squared Error: 3027.841697363167

------------- Random Forest Regressor ------------

Accuracy Score :: 96.40

R2 Score: 0.784231145060016

>>> Error >>>

Mean Absolute Error: 1178.8352936526505

Mean Squared Error: 4555260.494666742

Root Mean Squared Error: 2134.30562353819

------------- Gradient Boosting Regression ------------

Accuracy Score :: 66.15

R2 Score: 0.6240711552355718

>>> Error >>>

Mean Absolute Error: 1988.8958483087183

Mean Squared Error: 7936519.9200664265

Root Mean Squared Error: 2817.182975964896

------------- AdaBoost Regression ------------

Accuracy Score :: 8.95

R2 Score: 0.03723577855408411

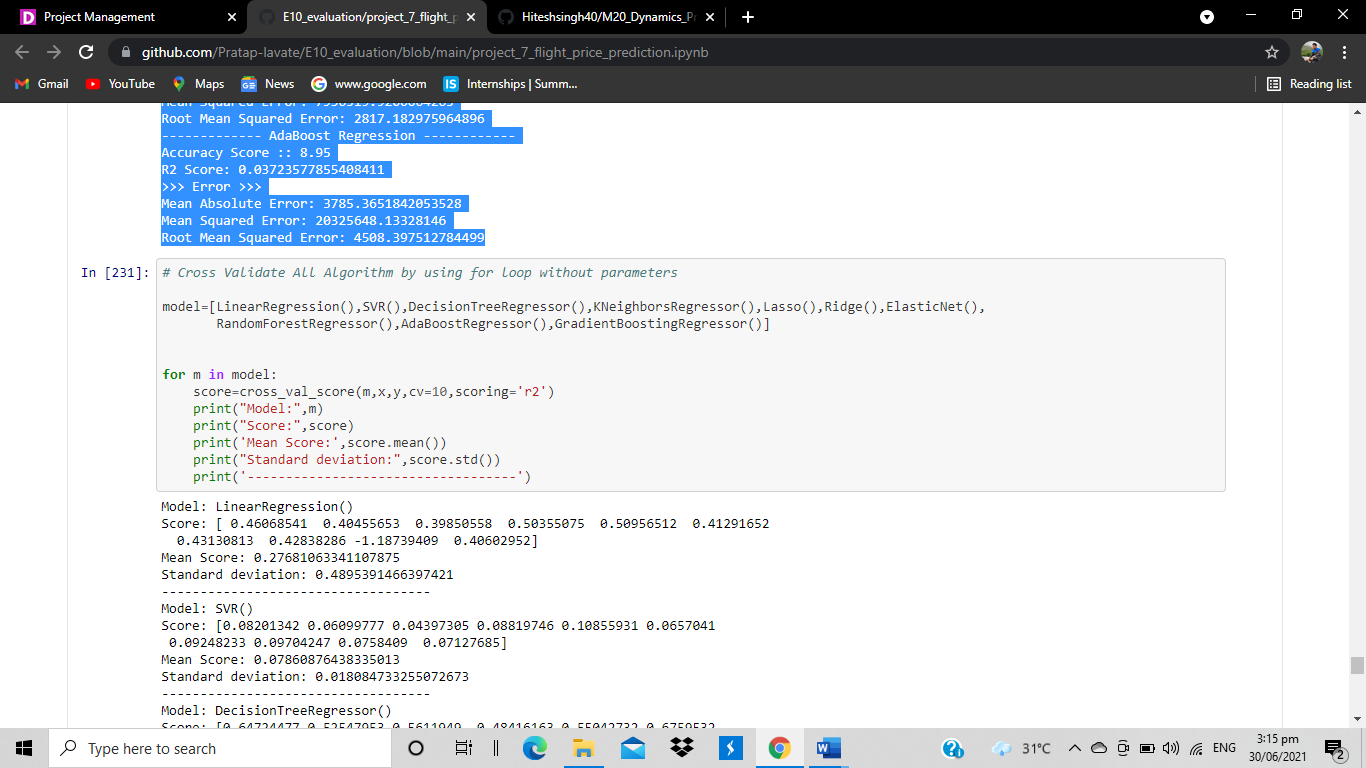
>>> Error >>>

Mean Absolute Error: 3785.3651842053528

Mean Squared Error: 20325648.13328146

Root Mean Squared Error: 4508.397512784499

* Then we create a cross validation function to check its biasness in that we compute score, mean score, standard deviation.



The output is,

Model: LinearRegression()

Score: [ 0.46068541 0.40455653 0.39850558 0.50355075 0.50956512 0.41291652

0.43130813 0.42838286 -1.18739409 0.40602952]

Mean Score: 0.27681063341107875

Standard deviation: 0.4895391466397421

-----------------------------------

Model: SVR()

Score: [0.08201342 0.06099777 0.04397305 0.08819746 0.10855931 0.0657041

0.09248233 0.09704247 0.0758409 0.07127685]

Mean Score: 0.07860876438335013

Standard deviation: 0.018084733255072673

-----------------------------------

Model: DecisionTreeRegressor()

Score: [0.64724477 0.52547953 0.5611949 0.48416163 0.55042732 0.6759532

0.58267383 0.64495672 0.53468467 0.57363482]

Mean Score: 0.5780411388906112

Standard deviation: 0.0577898558070417

-----------------------------------

Model: KNeighborsRegressor()

Score: [0.76690686 0.75653693 0.73185901 0.79732757 0.78191357 0.76300662

0.77638971 0.81857999 0.7964854 0.7394818 ]

Mean Score: 0.772848745648001

Standard deviation: 0.025565055013531277

-----------------------------------

Model: Lasso()

Score: [ 0.46065621 0.40459108 0.39844475 0.50354724 0.50959573 0.41284968

0.43134572 0.42842261 -1.14022257 0.40605377]

Mean Score: 0.28152842340757317

Standard deviation: 0.4754322771666128

-----------------------------------

Model: Ridge()

Score: [ 0.46068282 0.40455737 0.39850264 0.50355301 0.50956628 0.41291214

0.43131112 0.42838647 -1.17945167 0.40603103]

Mean Score: 0.2776051224594237

Standard deviation: 0.4871637513616513

-----------------------------------

Model: ElasticNet()

Score: [0.43365545 0.3890215 0.37233994 0.48853915 0.49145157 0.38255007

0.42075631 0.42133389 0.457288 0.3927612 ]

Mean Score: 0.4249697088188695

Standard deviation: 0.04063345552554408

-----------------------------------

Model: RandomForestRegressor()

Score: [0.76923581 0.75005642 0.71693772 0.8199606 0.81759169 0.79783202

0.78291088 0.82111197 0.79218226 0.72000764]

Mean Score: 0.7787826989310028

Standard deviation: 0.03707916290872579

-----------------------------------

Model: AdaBoostRegressor()

Score: [ 0.12868004 0.04262971 0.0586476 0.12019224 0.05642399 0.25171482

0.206172 0.1036103 -0.05520153 0.30474553]

Mean Score: 0.12176147134801059

Standard deviation: 0.1018361695638587

-----------------------------------

Model: GradientBoostingRegressor()

Score: [0.57483658 0.55994419 0.6243801 0.62558816 0.63845603 0.62019078

0.5778008 0.58895762 0.58936454 0.55295463]

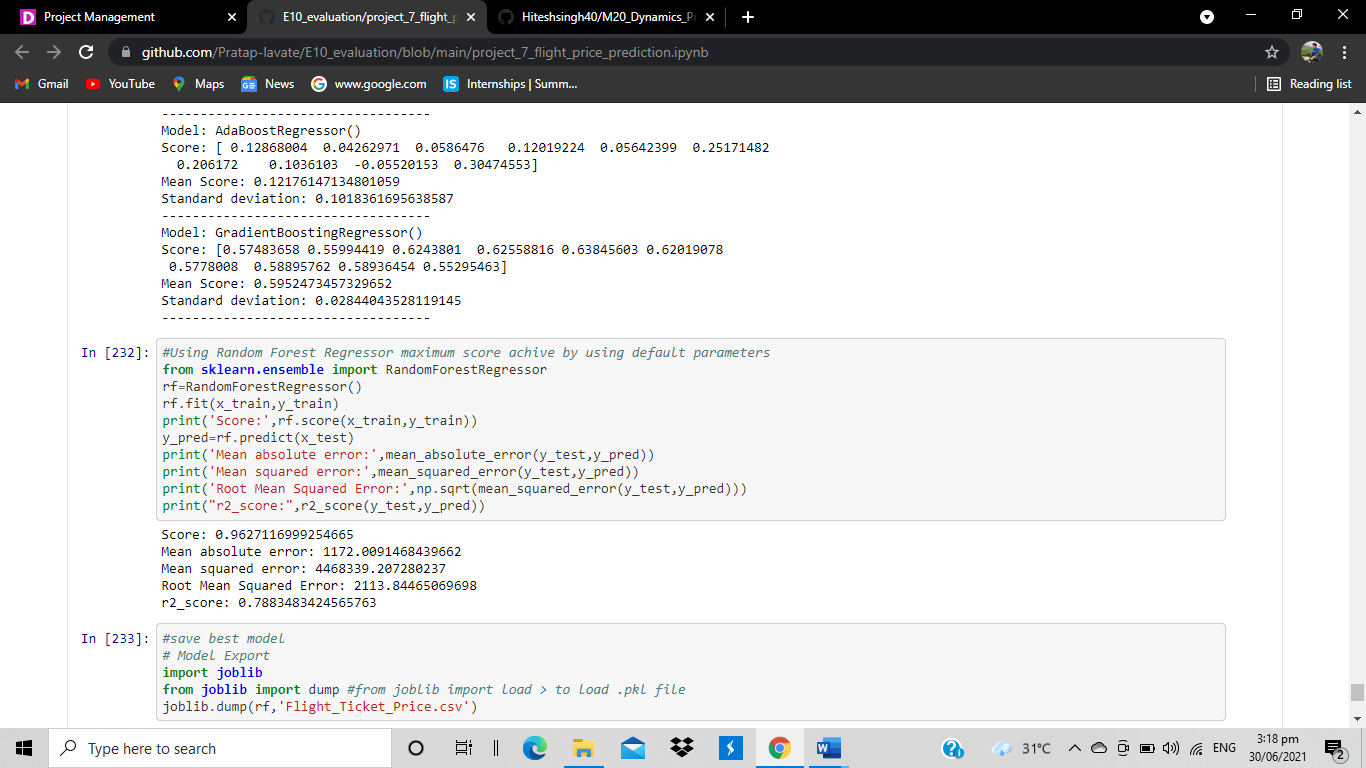
Mean Score: 0.5952473457329652

Standard deviation: 0.02844043528119145

-----------------------------------

Atlast we use random forest with default parameter and we got maximum score with mse 1172.0

And R2 score is 0.78 means 78% of variability of price is explained with the help of independent variable.



* **Concluding Remarks:**

We started with the data exploration where we checked on information about the data set, its data types, shape of data, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data pre processing part, we computed missing values with mean/median/mode of the data by checking distribution plot, converted features into numeric ones using encoding, grouped values into categories.

Next we trained 10 different machine learning models which included Decision Tree regressor, random Forest regressor, support vector machine, Ada Boost regressor, Gradient Boost regressor and , Lasso, Ridge, ElasticNet checked for its accuracy score/r2\_score and mean square error and picked one of them (random forest) and applied cross validation on it to fix problem of under fitting/ over-fitting with high performance is 96.27%.

Of course there is still room for improvement, like doing a more extensive feature engineering, by comparing and plotting the features against each other and identifying and removing the noisy features. Another thing that can improve the overall result would be a more extensive hyper-parameter tuning on several machine learning models.