**Problem Definition :**

**Avocado is a fruit consumed by people heavily in the United States.**

**Content**

This data was downloaded from the Hass Avocado Board website in May of 2018 & compiled into a single CSV. Here's how the [Hass Avocado Board describes the data on their website](http://www.hassavocadoboard.com/retail/volume-and-price-data):

The table below represents weekly 2018 retail scan data for National retail volume (units) and price. Retail scan data comes directly from retailers’ cash registers based on actual retail sales of Hass avocados. Starting in 2013, the table below reflects an expanded, multi-outlet retail data set. Multi-outlet reporting includes an aggregation of the following channels: grocery, mass, club, drug, dollar and military. The Average Price (of avocados) in the table reflects a per unit (per avocado) cost, even when multiple units (avocados) are sold in bags. The Product Lookup codes (PLU’s) in the table are only for Hass avocados. Other varieties of avocados (e.g. greenskins) are not included in this table.

Some relevant columns in the dataset:

* Date - The date of the observation
* AveragePrice - the average price of a single avocado
* type - conventional or organic
* year - the year
* Region - the city or region of the observation
* Total Volume - Total number of avocados sold
* 4046 - Total number of avocados with PLU 4046 sold
* 4225 - Total number of avocados with PLU 4225 sold
* 4770 - Total number of avocados with PLU 4770 sold

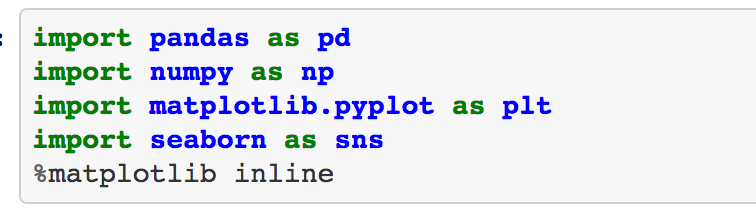
**Inspiration /Label**

The dataset can be seen in one angle to find the average price .

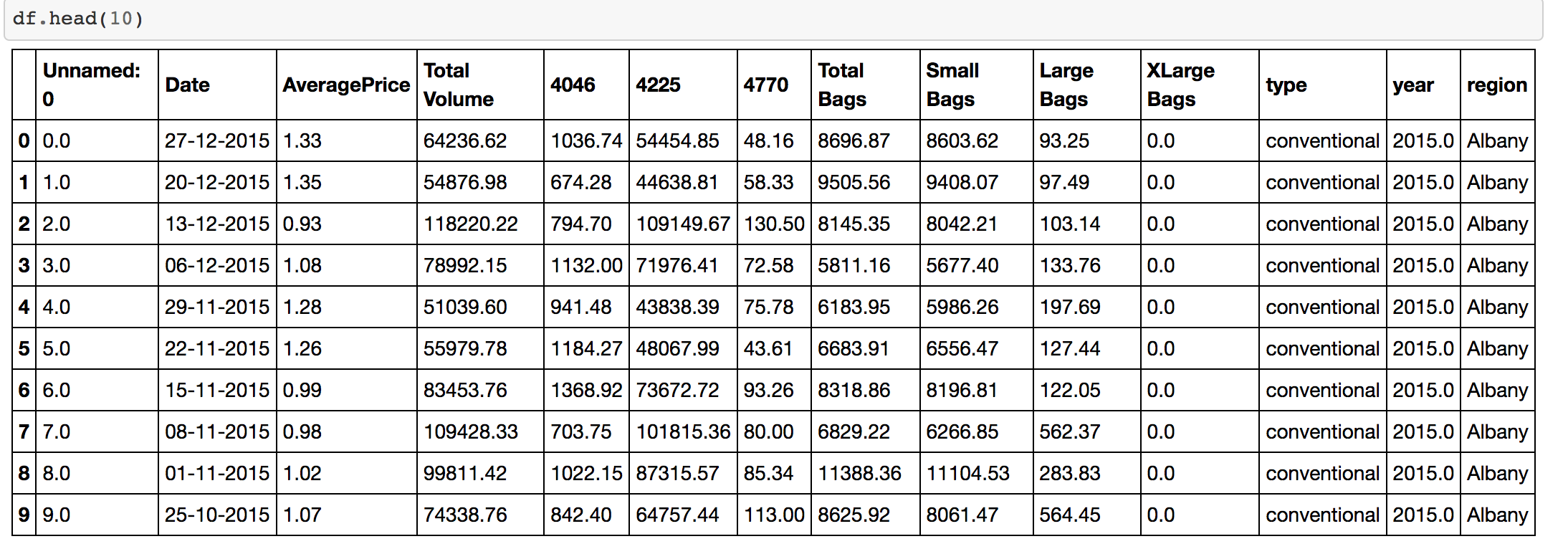
Task: Regression

Exploratory Analysis :

# Importing All necessary Libraries



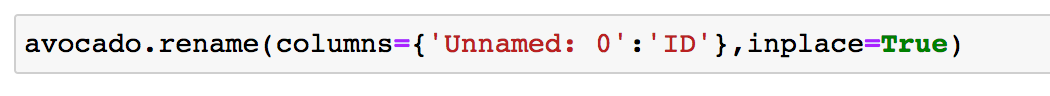
**Loading the data :**

****

“read\_csv” is an important function of pandas which allows to read csv files and we can make various operations on the dataset. As my file is a CSV file that’s why I have used “read\_csv” function to load the data from the specific directory.The name of the dataset id df.

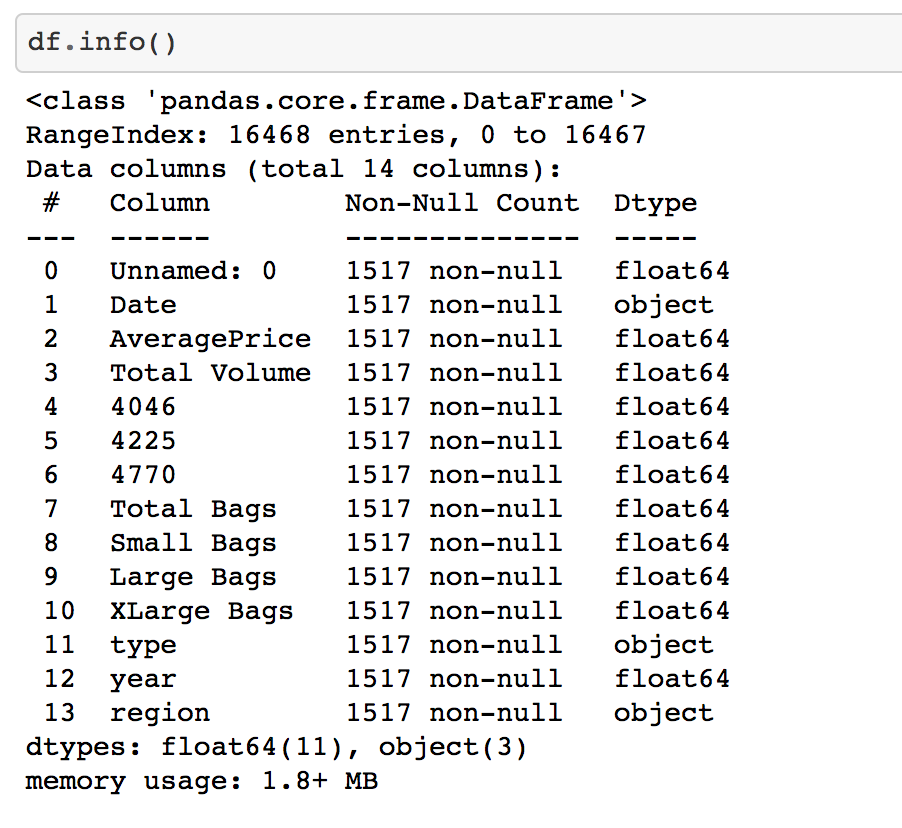
## **Description of Data :**

**Rename Unnamed column to another name :**

****

In the dataset, there is one column names “Unnamed: 0” . I have changed the column name to “ID”.

**df.info()**



Normaly to explore the data we can use various functions such as shape, columns, dtypes, info(), head(), tail(), describe(). Here, I have used df.info()

By using info() we can get a concise summary of a DataFrame. It includes the index dtype and column dtypes, non-null values and memory usage.

In our dataset we can see that there are 11 Numeric columns and three object type column.

**Observations:**

Numeric features

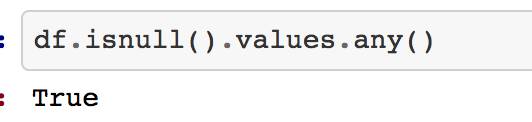
Numeric features = [' Unnamed: 0', 'AveragePrice', 'Total Volume', '4046', '4225', '4770','Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'year']

Catagorical features

Catagorical features =[‘type’, ‘region’, ‘Date’]

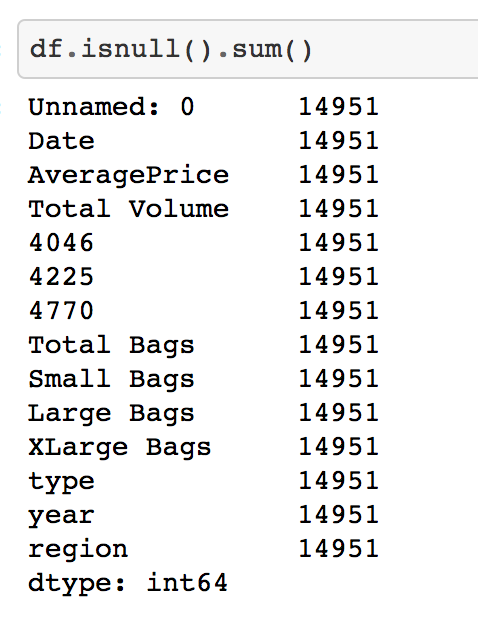
# **Missing Values**

There are different ways to check the missing values in our dataset.



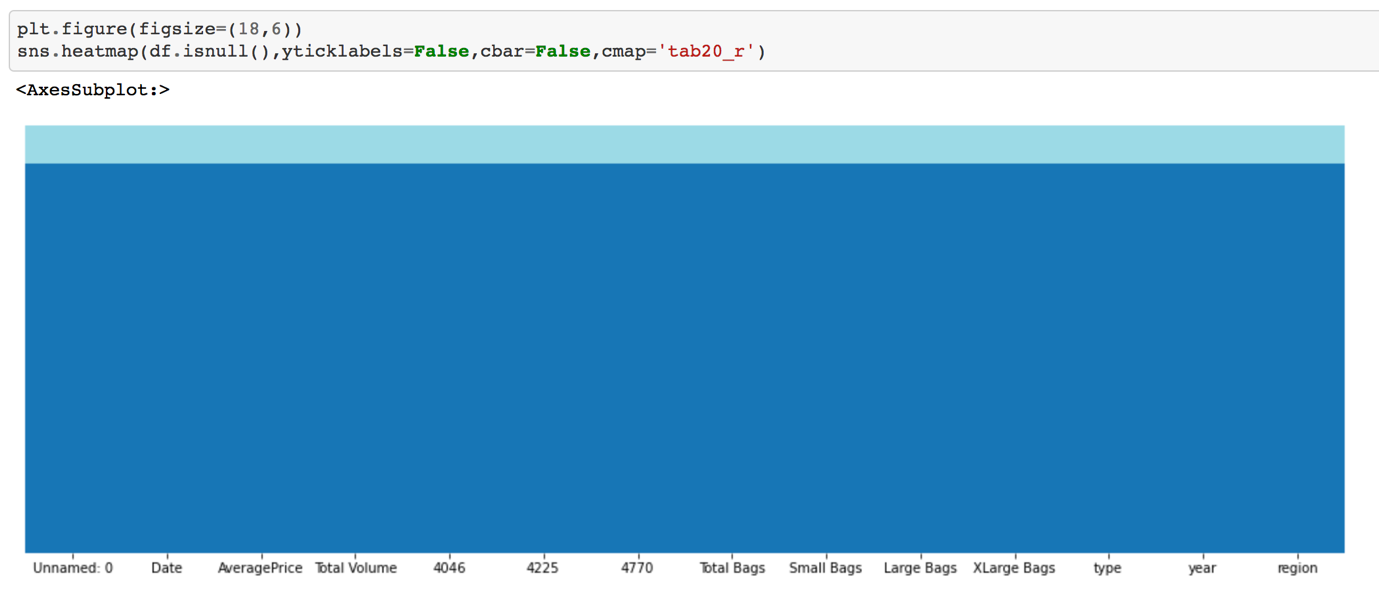
Here , we can see that there are null values present in our dataset.

But, If we want to see column wise we can use another method.



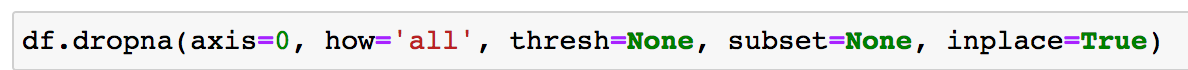
In every colulm there are Null values in the datset.

If we want to visualizise the null values then there are another method. Here, we can see that in every column there are same number of Null Values. That means our dataset have some rows where all the elements are missing. So, we will fix the issue.



By any method we will get the same results.

# Imputing Missing values:



By using the above code I have imputed the missing values. If all the values in one row are Null then I have deleted those rows.

# Exploratory the catagorical columns :



region:['Albany' 'Atlanta' 'BaltimoreWashington' 'Boise' 'Boston'

'BuffaloRochester' 'California' 'Charlotte' 'Chicago' 'Columbus'

'DallasFtWorth' 'Denver' 'Detroit' 'GrandRapids' 'GreatLakes'

'HarrisburgScranton' 'HartfordSpringfield' 'Houston' 'Indianapolis'

'Jacksonville' 'LasVegas' 'LosAngeles' 'Louisville' 'MiamiFtLauderdale'

'Midsouth' 'Nashville' 'NewYork' 'Northeast' 'NorthernNewEngland'

'Orlando' 'Philadelphia' 'PhoenixTucson' 'Pittsburgh' 'Plains' 'Portland'

'RaleighGreensboro' 'RichmondNorfolk' 'Roanoke' 'SanDiego' 'SanFrancisco'

'Seattle' 'SouthCarolina' 'SouthCentral' 'Southeast' 'Spokane' 'StLouis'

'Syracuse' 'Tampa' 'TotalUS' 'West' 'WestTexNewMexico']

California 76

Albany 67

BaltimoreWashington 65

Boise 65

Boston 62

Atlanta 54

PhoenixTucson 52

BuffaloRochester 51

Spokane 49

Columbus 47

NewYork 44

Jacksonville 41

Detroit 40

SouthCentral 39

SanDiego 38

West 36

Tampa 34

Louisville 34

Charlotte 31

Portland 30

Houston 29

NorthernNewEngland 29

WestTexNewMexico 27

Nashville 25

TotalUS 25

Denver 24

SouthCarolina 24

GrandRapids 23

Chicago 23

Pittsburgh 22

RichmondNorfolk 21

Orlando 21

Syracuse 19

HarrisburgScranton 19

Midsouth 18

GreatLakes 18

MiamiFtLauderdale 17

DallasFtWorth 17

Roanoke 17

StLouis 16

Indianapolis 16

RaleighGreensboro 16

SanFrancisco 15

Philadelphia 13

HartfordSpringfield 13

Northeast 12

Plains 12

LasVegas 10

Southeast 9

Seattle 9

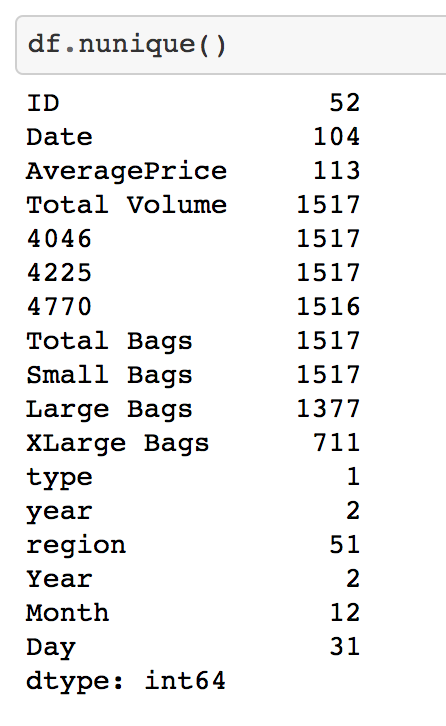
LosAngeles 3

Name: region, dtype: int64

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To explore the catagorical columns and to count the number of values we can use the following code. We can get the following observations after using this code :

# **Checking Unique values :**



To explore the dataset it’s also necessary to explore the unique values. If we see our dataset we can see that in some columns there are more unique values and some columns contains less unique values. Columns with less unique values normally effect more to predict the outcome. But if there are constant value in this case there will have no use of that column. From the dataset we can see that ‘type’has only one unique value. So, this column should be deleted. It will not effect our dataset.

Other observations :

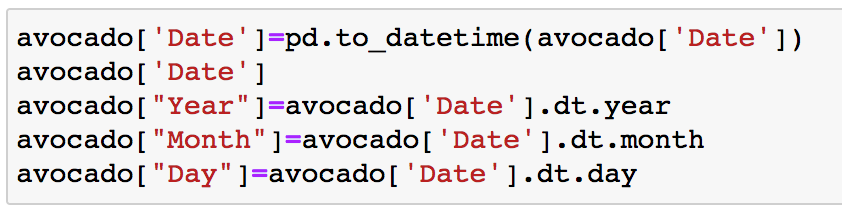
1. All the values in "Total Volume","4046","4225","Total Bags" these 4 columns are unique.So, these columns may not effect much to predict the outcome.

2.'4770' columns has 1516 unique values which is huge. So,this column may not effect the target variable as well.

3.The other columns have unique values. But not all the values. Some values are same. So these columns will effect much the final prediction.

4.I have split the Date column.

**Split Date Column :**



To split the ‘Date’ column into year, month and day in different columns firstly I have changed the datatype of Date column to datetime64. Then I have splited into different columns by using the previous code.

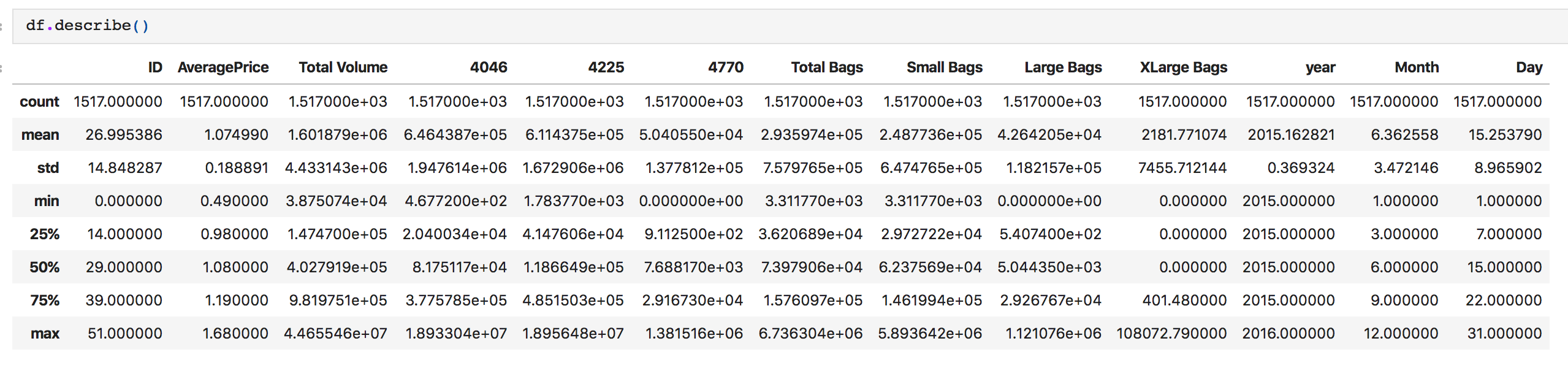
### Change class into numeric type:

### 

### 

### For analyzing the data with target all the columns should be numeric type. Also, if we want build our model we need to use the numeric column. So, it’ s necessary to convert all the object type column into numeric type. As multiple columns are object type and region column has many categorical values ; thets why I have used Label Encoder to convert all the columns by using some lines of codes.

# **Summary Statistics :**



By using describe() function we can explore the count, mean , median, standard deviation, minimum value, 25th, 50th and 75th percentile , maximum value.

We can find the following observations from the dataset :

1.Maximum values of ID,AveragePrice, Total Volume, 4046 ,4225 ,4770 ,Total Bags ,Small Bags ,Large Bags ,XLarge Bags ,year ,Month ,Day are : 51.000000 ,1.680000 ,4.465546e+07 ,1.893304e+07 ,1.895648e+07 ,1.381516e+06 ,6.736304e+06 ,5.893642e+06 ,1.121076e+06 ,108072.790000 ,2016.000000 ,12.000000 ,31.000000

2. Minimum values of ID,AveragePrice, Total Volume, 4046 ,4225 ,4770 ,Total Bags ,Small Bags ,Large Bags ,XLarge Bags ,year ,Month ,Day are 0.000000 ,0.490000 ,3.875074e+04 ,4.677200e+02 ,1.783770e+03 ,0.000000e+00 ,3.311770e+03 ,3.311770e+03 ,0.000000e+00 ,0.000000 ,2015.000000 ,1.000000 ,1.000000

75 percentile and max value

1. In the 'Total Volume' 75 percentile and max value has huge diffenece. Most probably there are outliers.

2. In the "4046","4225","4770","Total Bags","Small Bags","Large Bags","XLarge Bags" colums 75 percentile and max value has huge diffenece. Most probably there are outliers. So, there are outliers too.

3. In the "ID", "AveragePrice", "year", "Month", "Day" columns 75 percentile and max value has no huge diffenece. It looks normal.

50 percentile(median) and mean value

1. In the 'Total Volume',"4046","4225","4770","Total Bags","Small Bags","Large Bags","XLarge Bags" 50 percentile and mean value has huge diffenece. Most probably there are outliers.

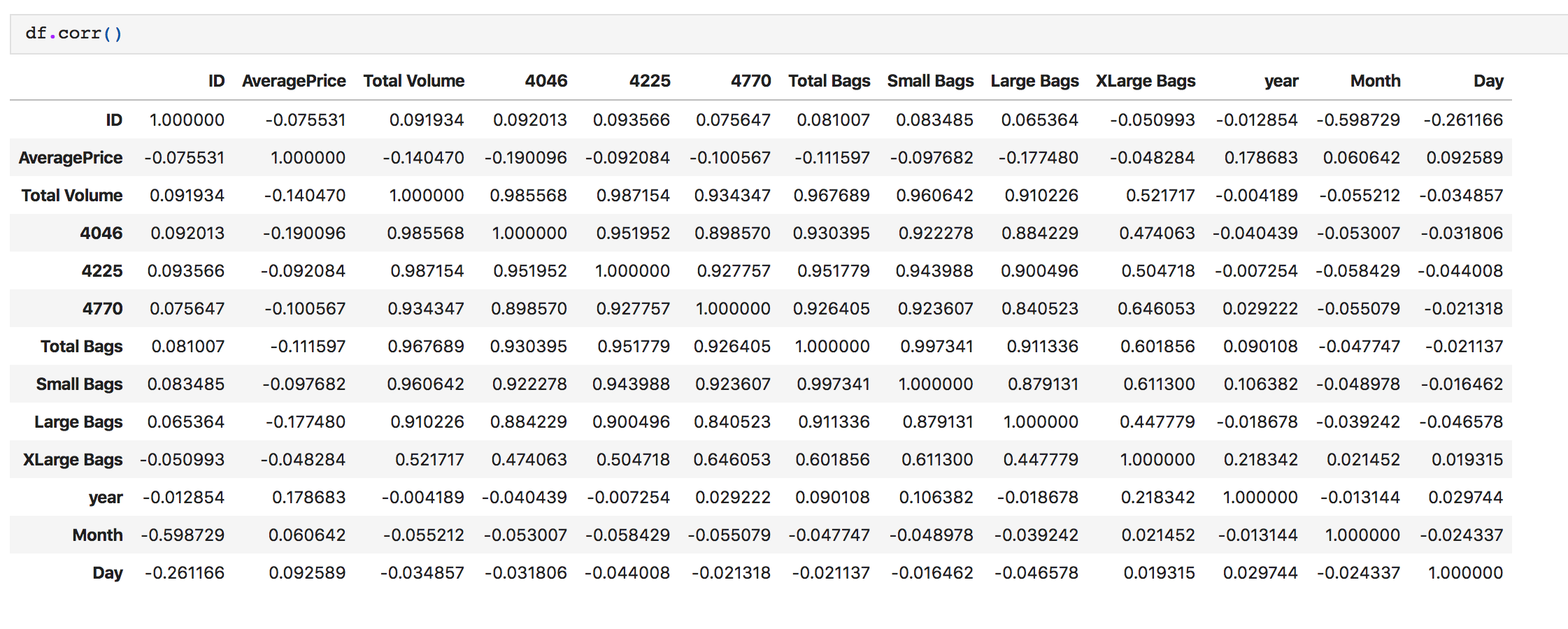
2.In the "ID", "AveragePrice", "year", "Month", "Day" columns 50 percentile and mean value has no huge difference. It looks normal.

# **Correlation:**

### Correlation only with target variable :

### 

We can see the correlation of every column with the target variable by using barplot. The peak points which are above 0 are positively correlated with the target variable. And the peak points which are under 0 are negatively correlated with the target variable. But the problem is we can no know the exact value of correlation by using barplot. To know the exact correlation we have another technique.



From the above figure we we be able to know the exact correlation of each column with the target variable.

The findings are mentioned below:

1. Negative correlation with Average price : 'ID','Total Volume', '4046', '4225', '4770',

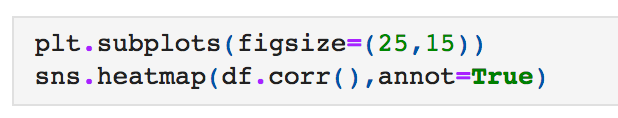
'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags'

2. Positive correlation : 'year','Month', 'Day'

3. Strong correlation : 'Total Volume', '4046', 'Large Bags', 'year'

4. Weak correlation: 'ID','4225', '4770','Total Bags', 'Small Bags','XLarge Bags' ,'Month', 'Day'

**Correlation among all the variables:**





From the previous steps we have only known about the correlation of each column with the target variable. But by using heatmap we will be able to explore the correlation among all the numeric variables present in the dataset.

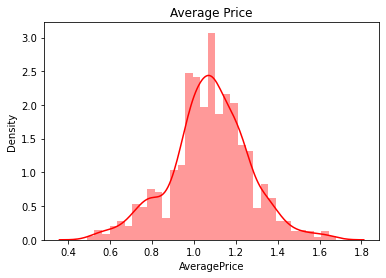
We can also get the same result using df.corr() method.

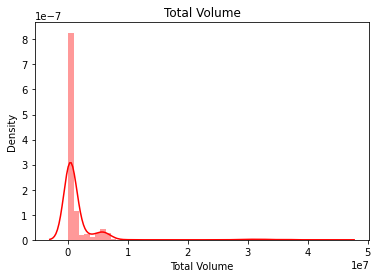
**Data Visualization :**

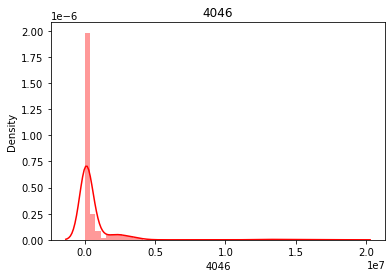
Our feature variable of interest is AveragePrice.

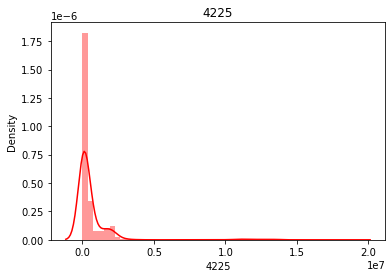
This column shows that the target variable is regression type. So, finally we need to use regression type algorithm.

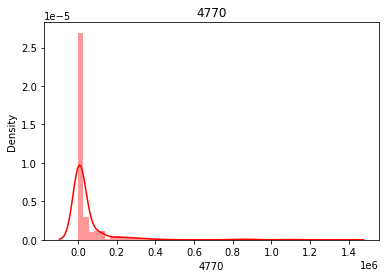
**Univariate anslysis :**

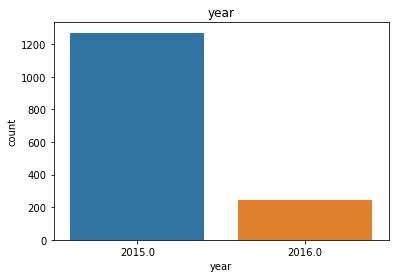
****

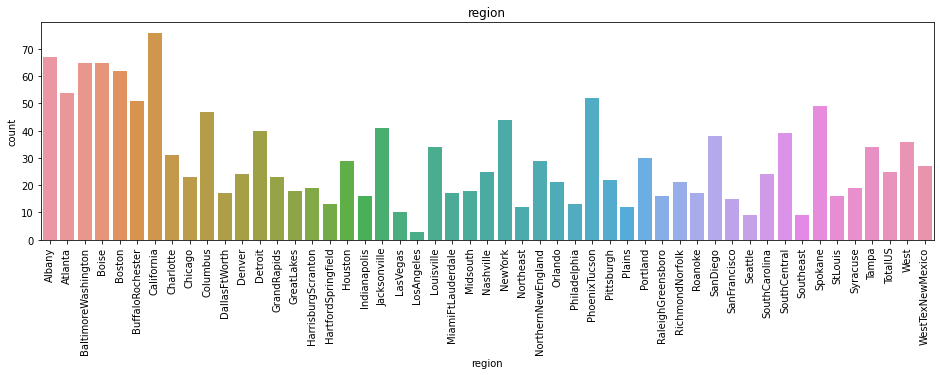
****

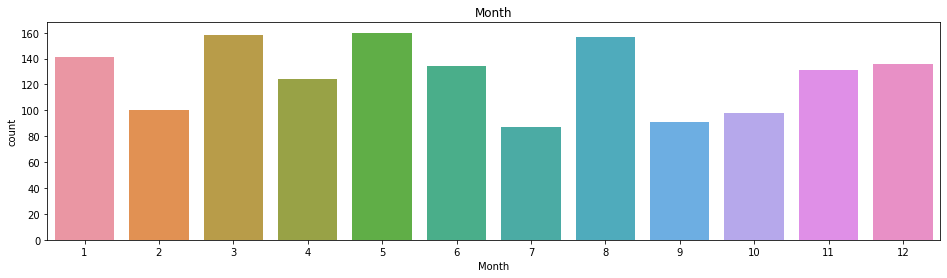
****

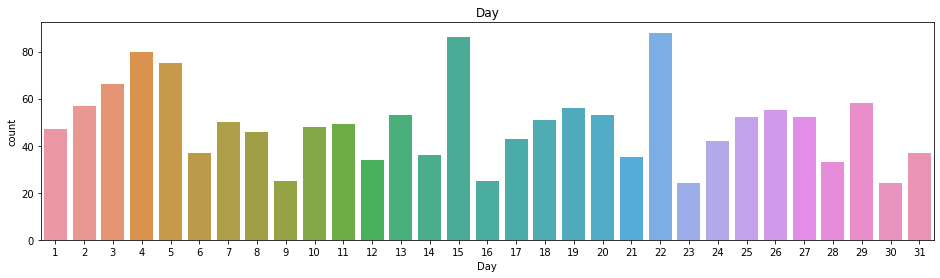
****

****

****

****

****

****

In some columns the values are continuous type. For continuous type of data I have used distplot with the red coloured mean value to visualize every column perfectly. In some columns such as 4770,4225,4046 the peak point is too high. That means there are outliers. Some of the columns are catagorical and there are few specific values. In these cases I have used countplot.

Sample code for histogram :

sns.distplot(df['column\_name'],color='r')

plt.title('column\_name’)

plt.show()

Sample code for countplot :

plt.subplots(figsize=(20,6))

sns.countplot(x='Column Name',data=df)

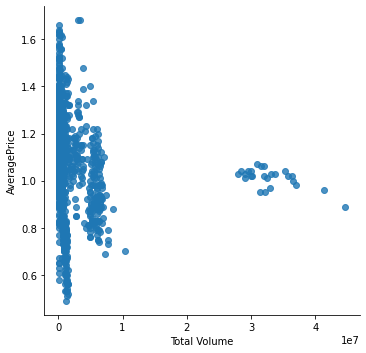
plt.title('Column Name')

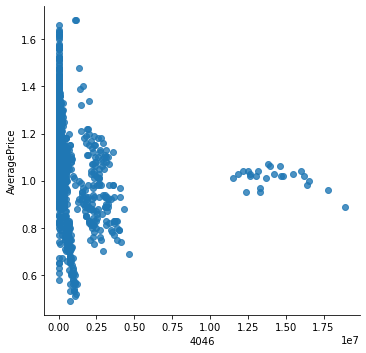
plt.xticks(rotation=20)

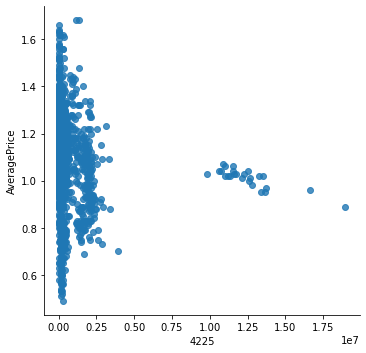
plt.show()

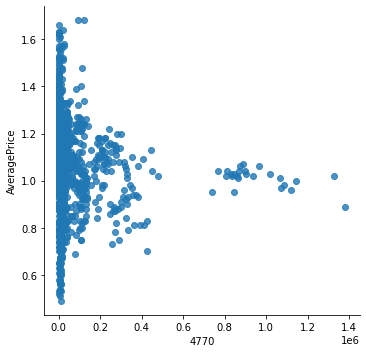
print(df.Column Name.value\_counts())

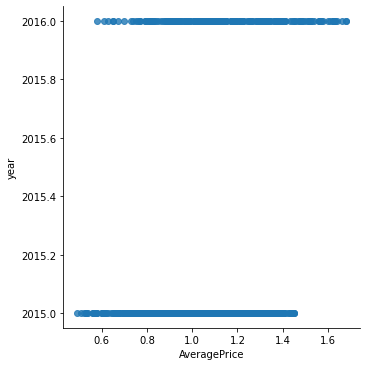
**Bivariate Analysis :**

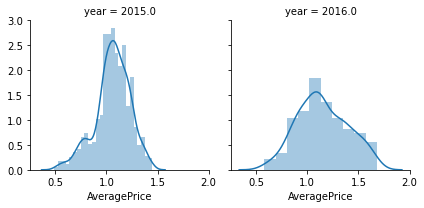
****

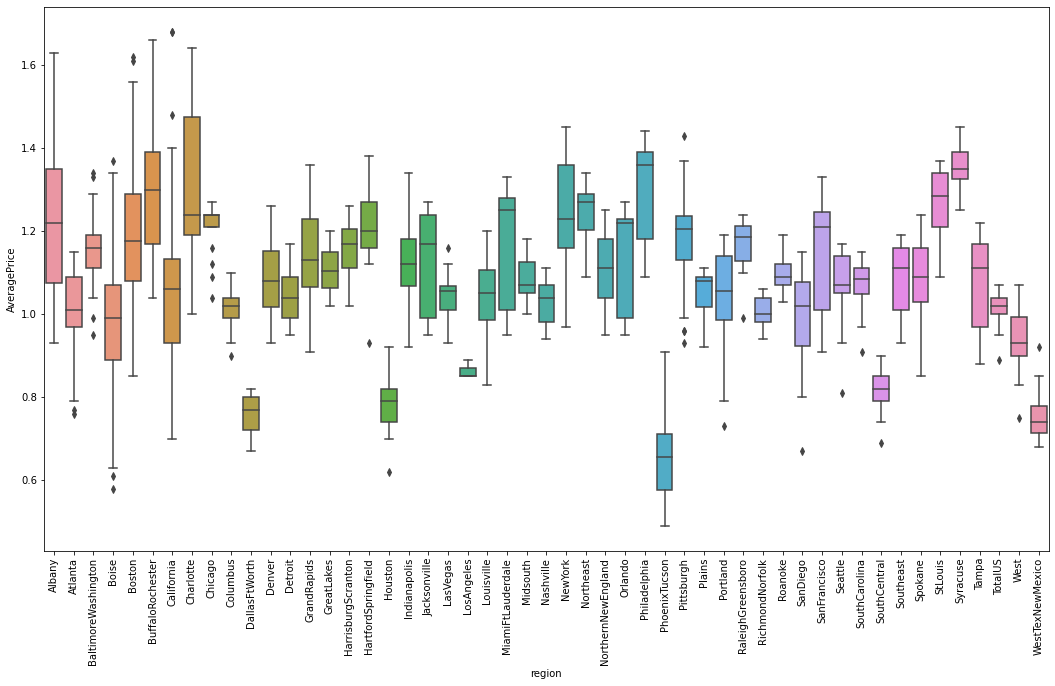
****

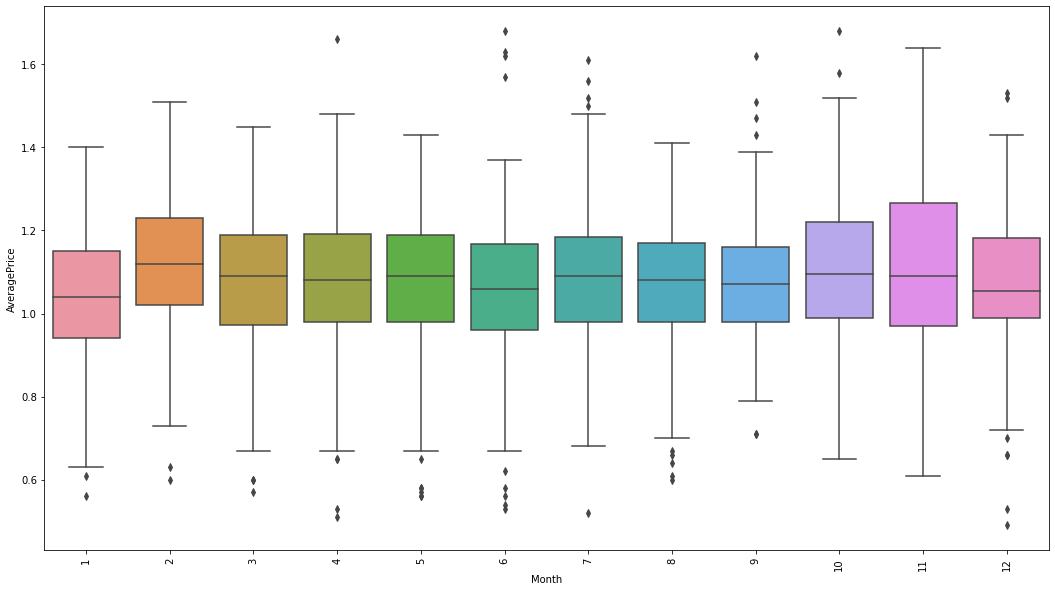
****

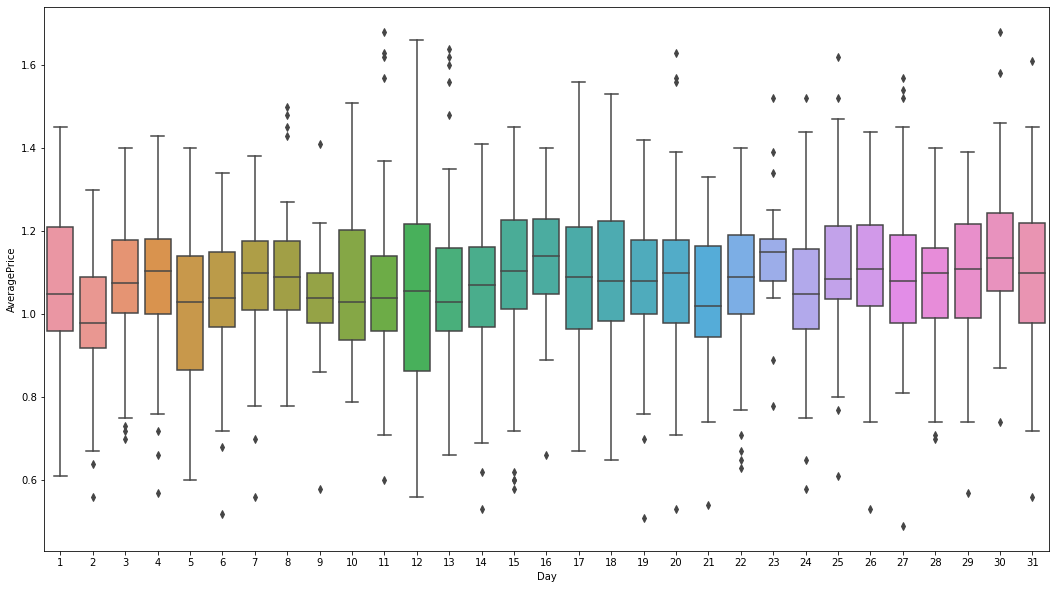
****

****

****

****

****

****

In bivariate analysis most of the times I have used scatterplot. Because most of the values are continuous types. We have used x as the column name and y for target variable named ‘AveragePrice’. In some cases such as ‘region’ , ‘month’, ‘day’ vs ‘AveragePrice’ I have used boxplot. Because ‘region’ , ‘month’, and ‘day’ are categorical type variable and there are huge data as well. The sample codes are given below :

**For boxplot:**

plt.figure(figsize=(18,10))

plt.xticks(rotation='vertical')

sns.boxplot(x='column\_name',y='AveragePrice', data=df)

**For Scatterplot :**

plt.figure(figsize=(8,4))

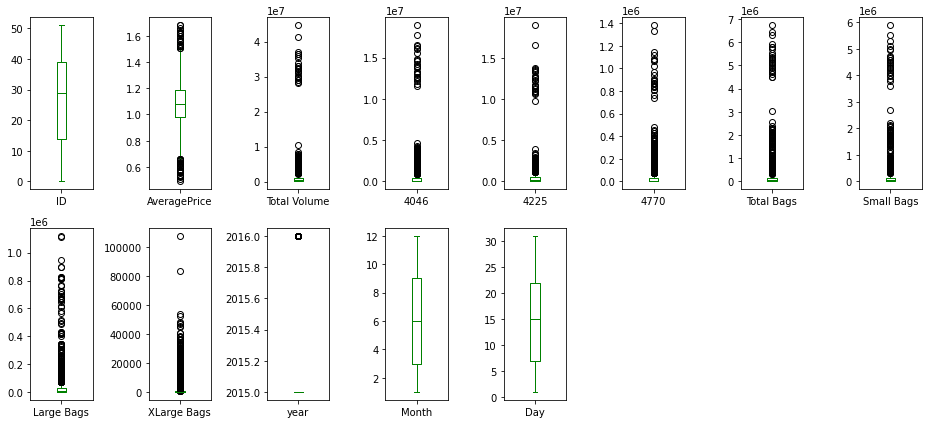
sns.lmplot(x='column\_name',y='Target\_variable',fit\_reg=**False**,data=df)

plt.show()

# **Check Outliers :**

df.plot(kind='box',subplots=**True**,layout=(4,5),color='green',figsize=(13,13))

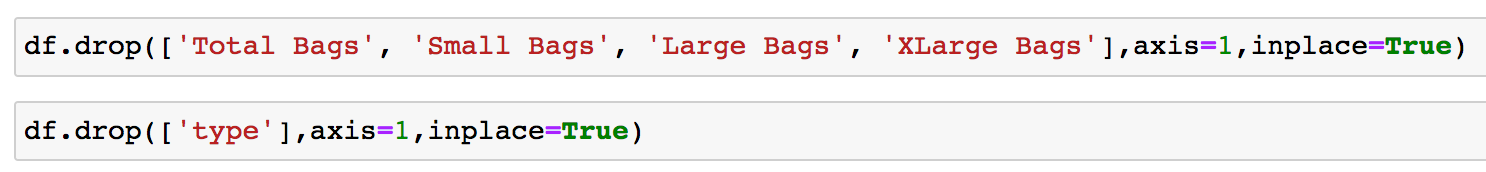
plt.tight\_layout()



After illustrating the above figure we can explore that AveragePrice,Total Volume, 4046, 4225, 4770, Total Bags, Small Bags, Large Bags, Xlarge Bags having outliers. The other columns don’t have outliers. So, we need to treat the outliers.

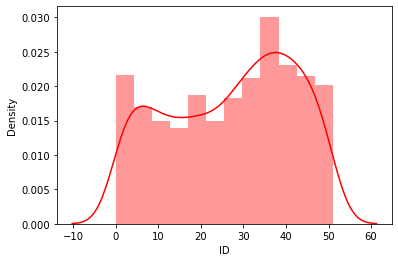
We can also check the outliers individually.

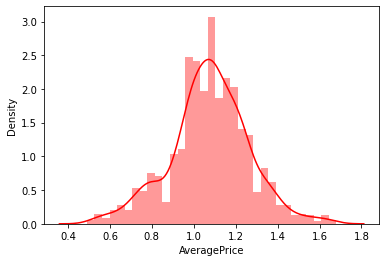
# Dropping columns :

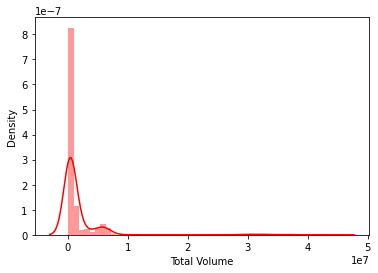


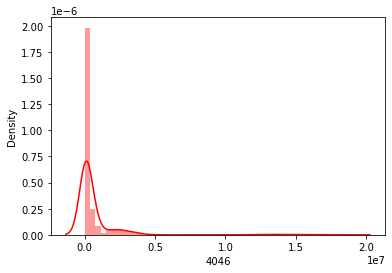
1. These ‘type’ colum will be deleted. Because there are either one single value.
2. Total Bags, Small Bags, Large Bags, Xlarge Bags having a lot of outliers.And most of the values are unique. That’s why I have deleted these colums as well.

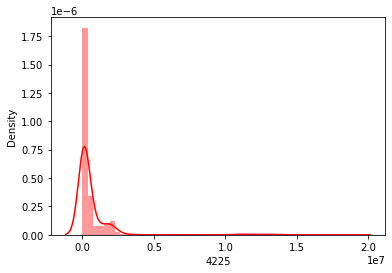
# Skewness:

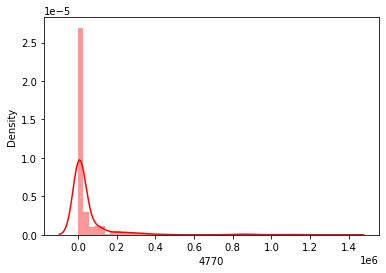


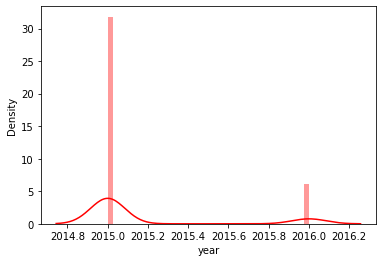


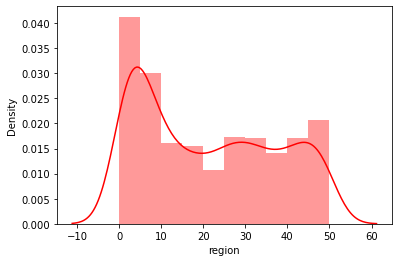


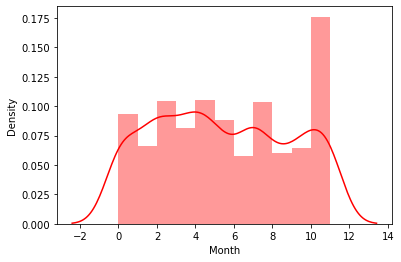


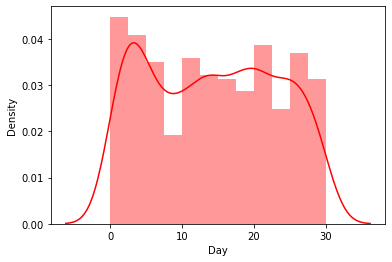






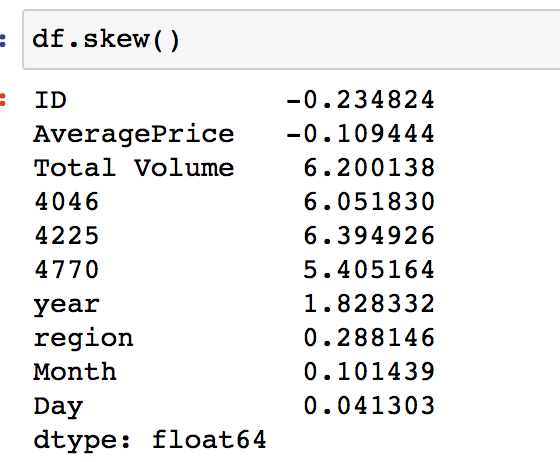






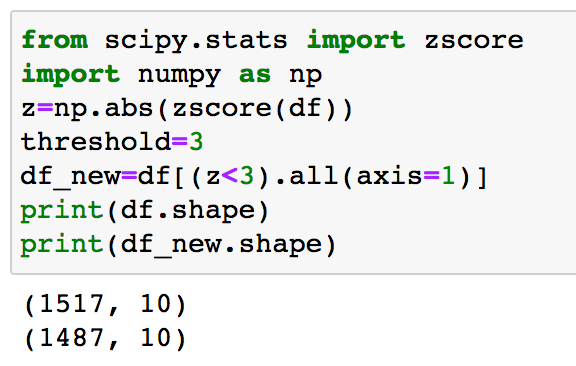
In most of the columns skewness is present. In the Total Volume, 4046, 4225, 4770, year columns right skewness is present.

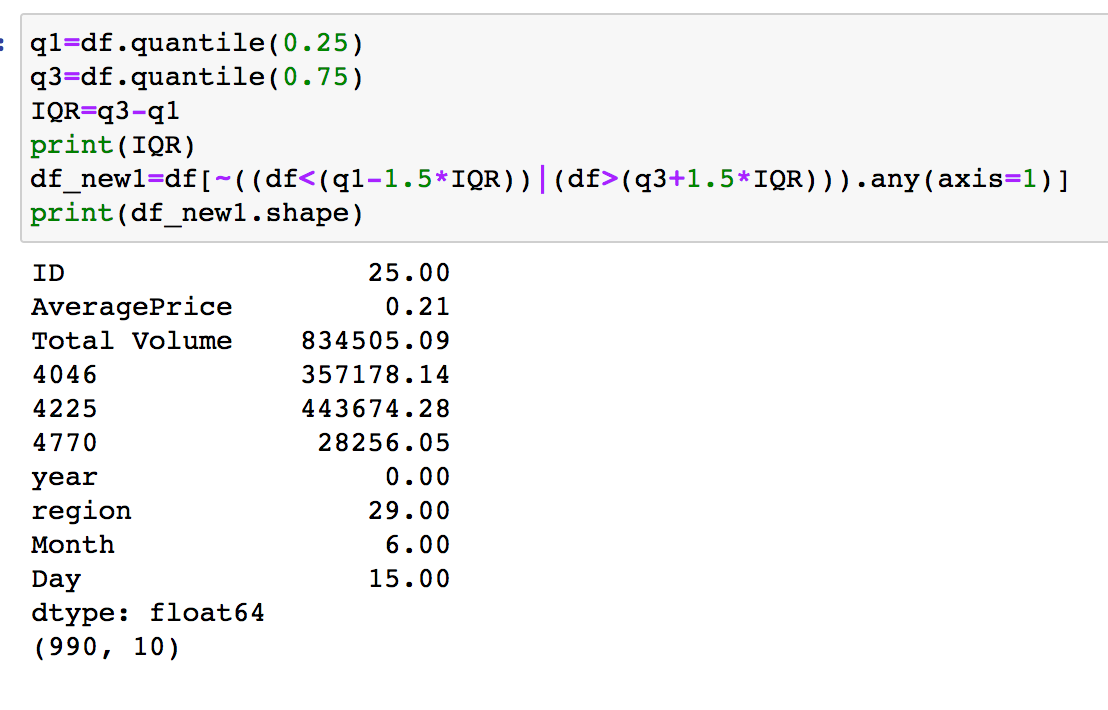
**df.skew()**

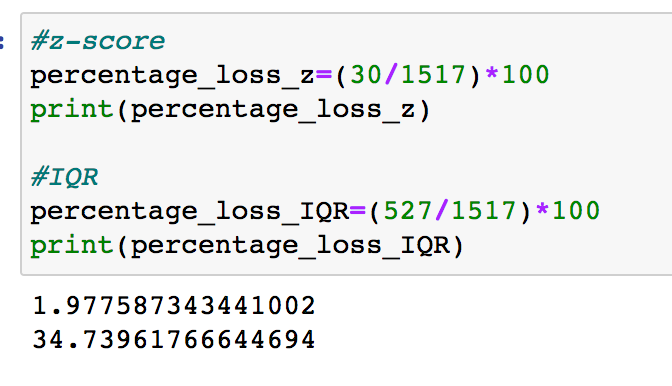


If we want to know the exact value then skew() function is the best way to know the skewness of the variavles. Here, The standard value I have used is 0.56. If the value is not in between -0.56 and 0.56 that means skewness is present in those columns.

# **Removing Outliers :**

****

****



I have used both z-score and IQR method to remove the outliers. But using both of the methods we can see that the percentage of data losing of z-score is almost 2 % and IQR is almost 35%. By using IQR I have loosen a lot of data. That’s why I have used z\_score.

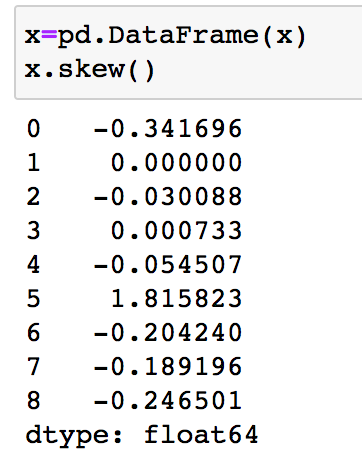
**Split the data into x and y**



I have splitted the dataset into x and y where x represents all the columns except the target variable AveragePrice and y represents the target variable.

### Treating Skewness via yeo-johnson method:

### 



After treating the skewness we can explore that the skewness of almost all the columns has been removed. Now, only one column having skewness which is 1.82.

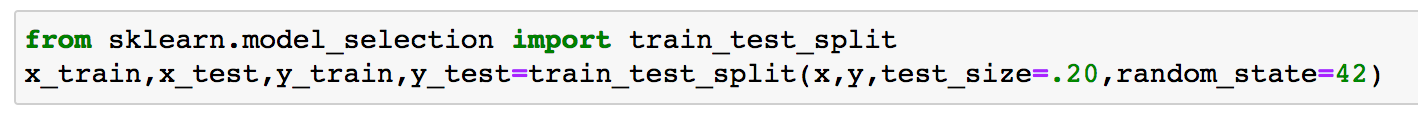
# **Model Training :**

### Spilitting the data into input and output variable :

### 

We can split the data into x and y. x is having all the columns except the target variable. Y is having only the target column.

### Spliting the data into training and testing set :

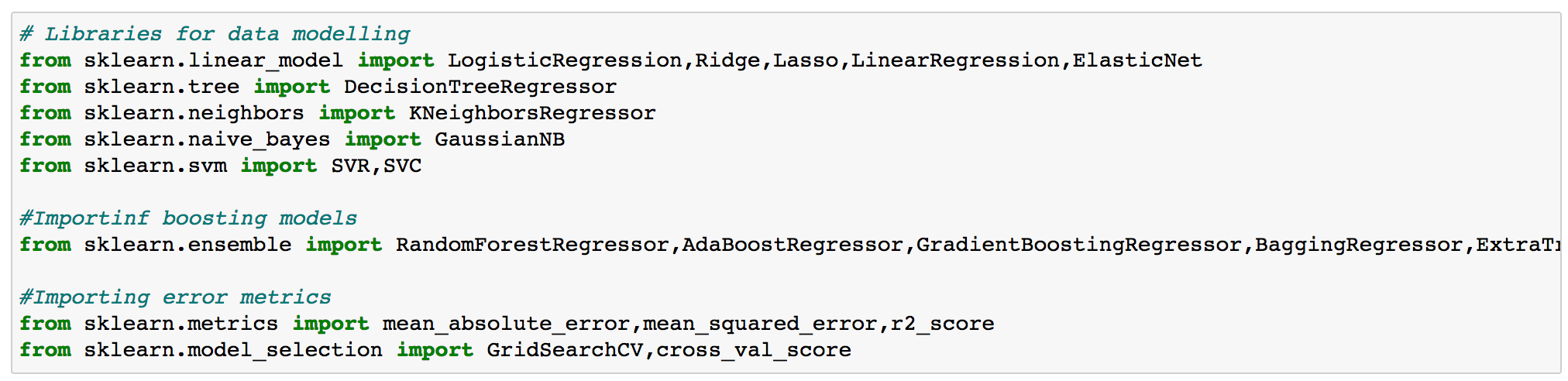


**Build Model :**

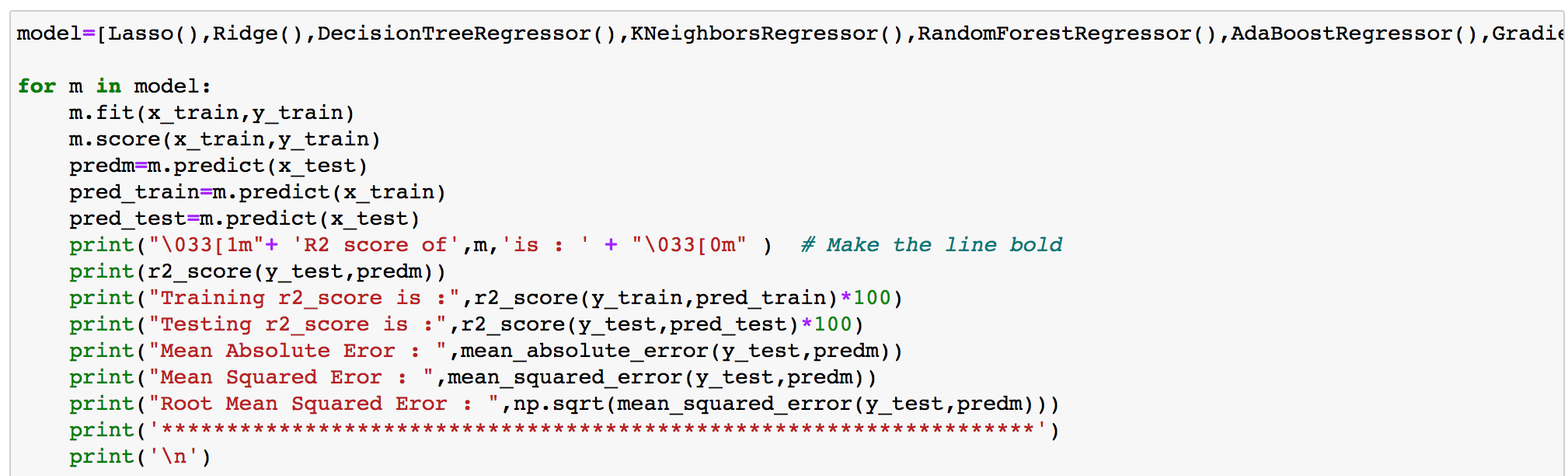
**Scaling :**

As I have used yeo-johnson method to remove the skewness that’s why there is no need to scale the dataset. It automatically scale the data as well.

# **Importing all the model Library :**



# **All algorithms are in one code :**



By using all the algoriths one by one we can use one function to implement all the algorithms. If we summarize the result we get the following r2 score :

Lasso(),Ridge(),DecisionTreeRegressor(),KNeighborsRegressor(),RandomForestRegressor(),AdaBoostRegressor(),GradientBoostingRegressor(),BaggingRegressor(),ExtraTreesRegressor(),LinearRegression(),SVR(),ElasticNet(),RandomForestRegressor()

1. R2 Score of Lasso()is : -0.002700101752157291

2. R2 Score of Ridge() is : 0.314036312443166

3. R2 Score of DecisionTreeRegressor() is : 0.7004478541441683

4. R2 Score of KNeighborsRegressor() is : 0.7012701159105127

5. R2 Score of RandomForestRegressor() is : 0.8323141228135249

6. R2 Score of GradientBoostingRegressor() is : 0.7386288417238536

7. R2 Score of BaggingRegressor() is : 0.8066834502482474

8. R2 Score of ExtraTreesRegressor() is : 0.8780266306177331

9. R2 Score of LinearRegression() is : 0.31405425973319234

10. R2 Score of SVR() is : 0.6902969949351738

11. R2 Score of ElasticNet() is : -0.002700101752157291

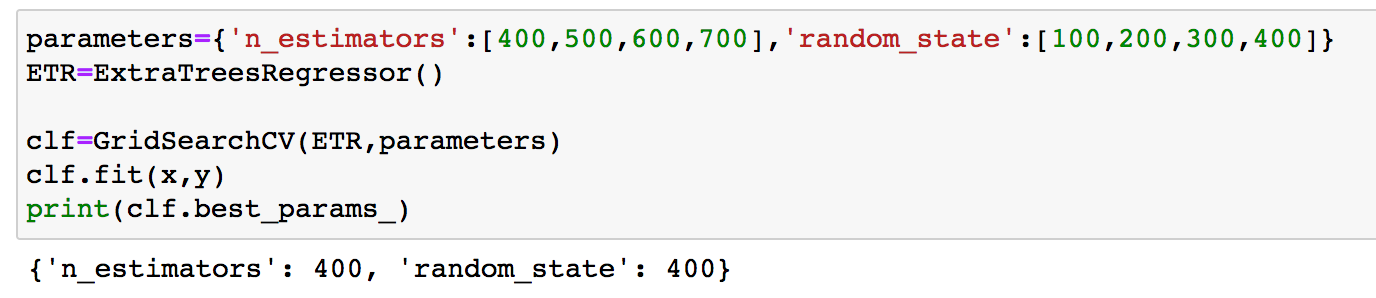
12. R2 Score of AdaBoostRegressor() is : 0.6352453884260243

We have got good R2 Score by using the following algorithms :

ExtraTreesRegressor(), BaggingRegressor(), RandomForestRegressor()

But out of these algorithms ExtraTreesRegressor () is giving the best result. We will do the hyperparameter tuning to reduce the overfitting.

# **Using Best Parameter :**



Here the best parametes for ExtraTreesRegressor () are 'n\_estimator': '400', 'random\_state': 400.

# **Using Best Parameter :**

After using the best parametes we have got the accuracy for ExtraTreesRegressor (). The r2\_score is 0.9890167364016736.

# **Cross Validation score :**



Result :

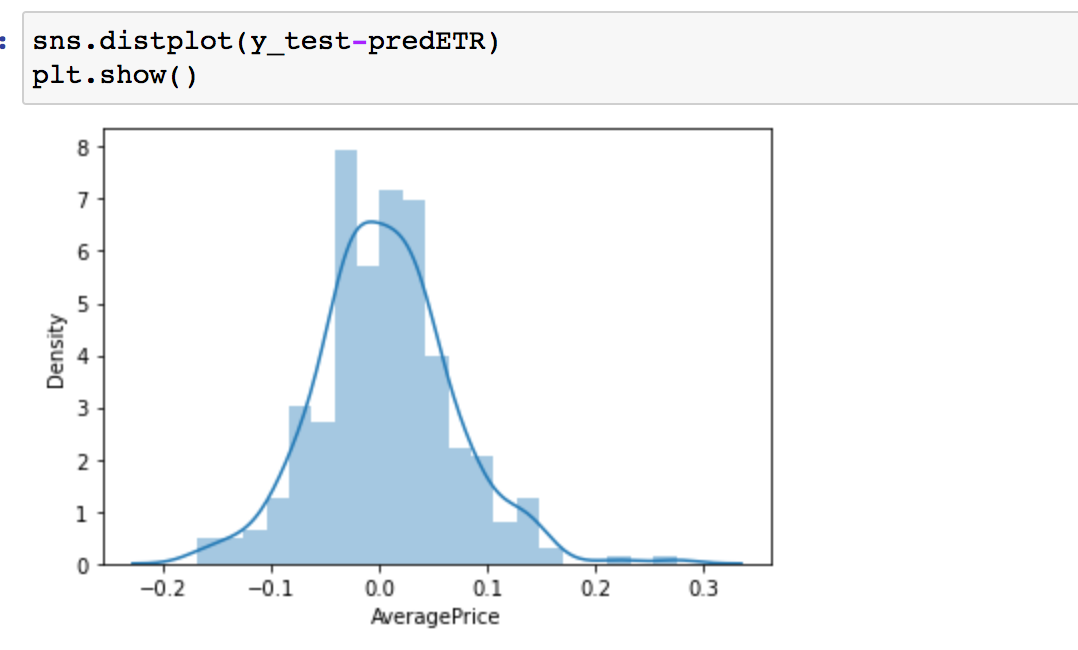
odel : ExtraTreesRegressor()

Mean Score : 0.17814803556094547

Standard deviation : 0.30906649700725297

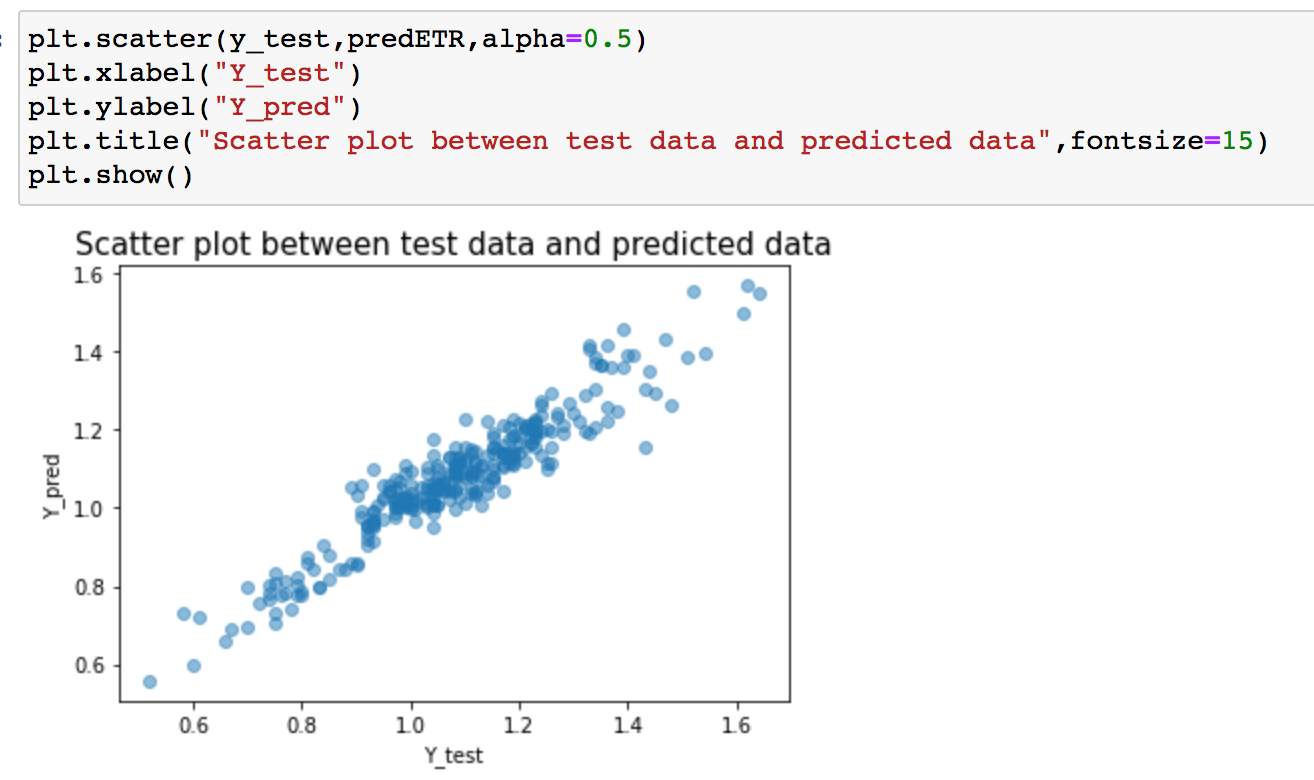
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Plotting the distribution plot and we find the Gaussian plot



We can explore that the after using the ExtraTreesRegressor() algorithm and hyperparameter tuning the distribution looks normal.

# Scatter plot between test data and prediction

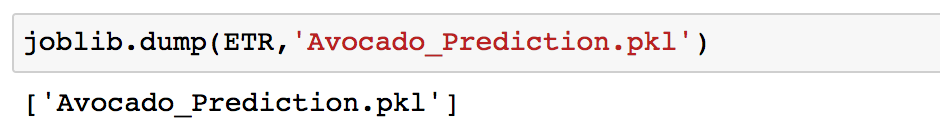


# The predicted data and the original are almost on the same line. So, this model will be accepted.

# Saving the model

**import** **joblib**

# Save the model as a pickle in a file



To save our model first we need to import joblib. Then we can save our model as pkl file.

Name : Md Rashidunnabi

Email : [rashidunnabi11@gmail.com](mailto:rashidunnabi11@gmail.com)

Phone : +8801319527349

Whatsapp : +8801319527349