

**NAME OF THE PROJECT**

“Avocado Project”

**Submitted by:**

**Leena chatterjee**

**ACKNOWLEDGMENT:**

* I have taken efforts in this project(creating this blog). However, it would not have been possible without the kind support and help of each individual of DATA TRAINED organizations. I would like to extend my sincere thanks to all of them.
* I am highly indebted to all team of Data trained for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

**Bibliography**:

* <https://www.grammarly.com/>
* <https://www.google.com/>
* <https://in.zapmetasearch.com/>
* https://projects.datatrained.com/

**Avacado Project**

Problem Statement:

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

This data was downloaded from the Hass Avocado Board website in May of 2018 & compiled into a single CSV.

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

**Label**

My task is to make a mode that can consider the data provided and predict the Average Price.

**Conceptual Background of the Domain Problem**

Data science is the field where we can predict the probability. Here basically we need to analyse a predictive model and find out the Average Price

In this dataset those variables are ====🡺

Attributes inside the Avacado dataset Date - The date of the observation

Average Price - 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

Programme:

import numpy as np

import pandas as pd

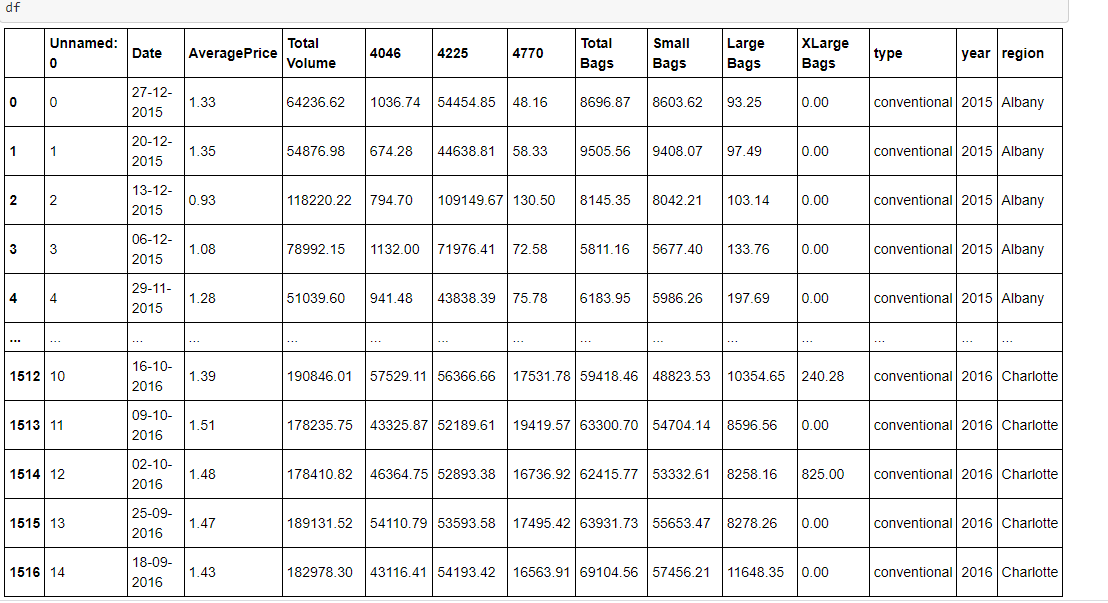
import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

df=pd.read\_csv("Avogadofruit.csv")



Observation – df is the dataset , it is containing 1517 rows × 14 columns.

**Data types detection**:

df.dtypes provided data type information of each column

Unnamed: 0 int64

Date object

AveragePrice float64

Total Volume float64

4046 float64

4225 float64

4770 float64

Total Bags float64

Small Bags float64

Large Bags float64

XLarge Bags float64

type object

year int64

region object

dtype: object

Observation – we found that this dataset contains int , obj and float type of data

Null value detection :

df.isnull().sum() ---- Output of this provided number of null value present on dataset

Unnamed: 0 0

Date 0

AveragePrice 0

Total Volume 0

4046 0

4225 0

4770 0

Total Bags 0

Small Bags 0

Large Bags 0

XLarge Bags 0

type 0

year 0

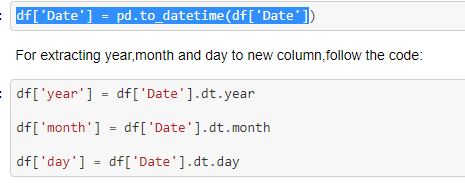
region 0

Observation – We found no null value of this dataset

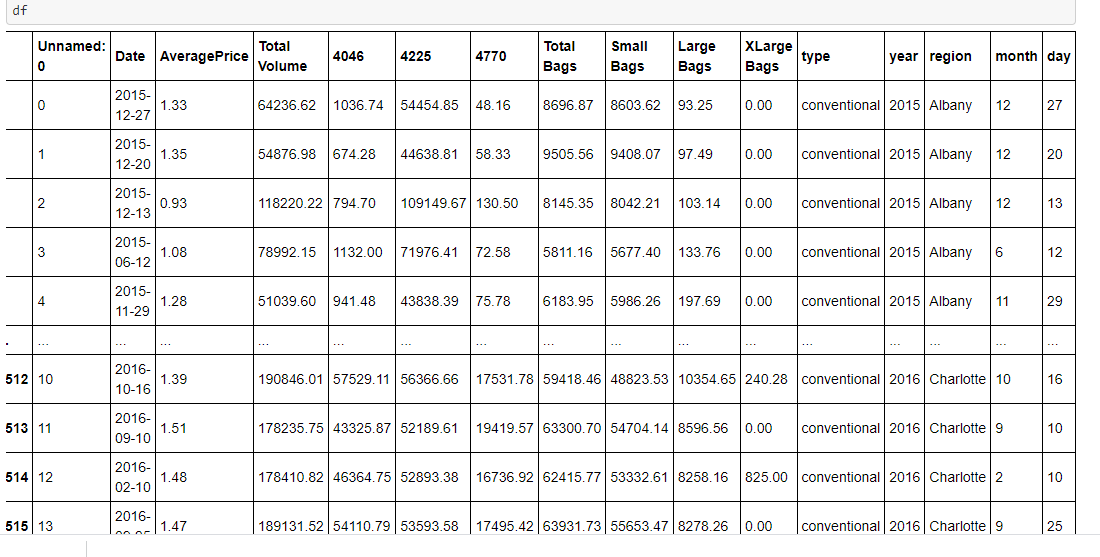
In this data set we have datetime data column:

df = pd.read\_csv("Avogadofruit.csv", parse\_dates=['Date'])

df['Date'].dtype= datetime64[ns] so we need extract year , date , month



**Now the new data frame --🡪 df is**



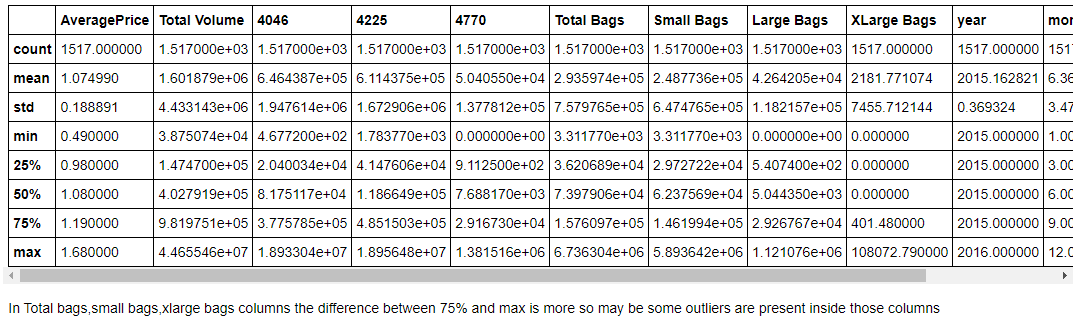
Observation—we found that unnamed0, and date we need to drop because from date we already extracted month , year and day

Unnamed0 we have dropped because this column because it generated by deault as a index number , it won’t provide much info to dataset

df.drop(['Unnamed: 0' ,'Date'] , axis=1,inplace=True)

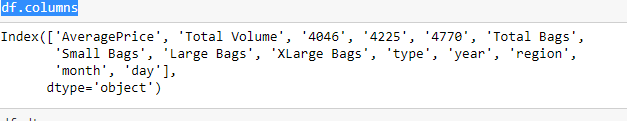
describe the dataset

df.describe()



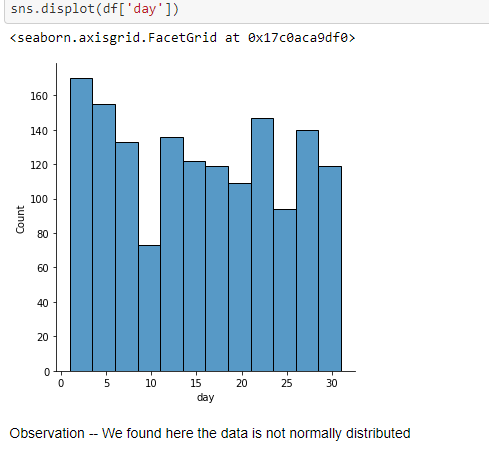
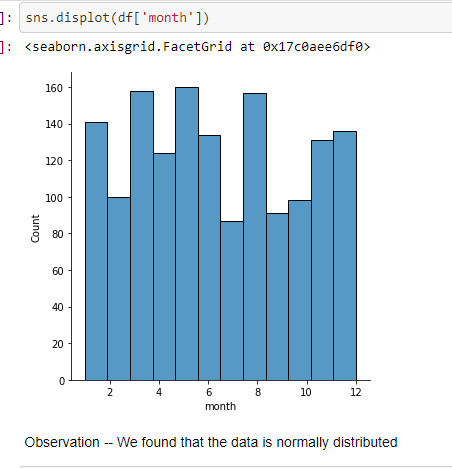
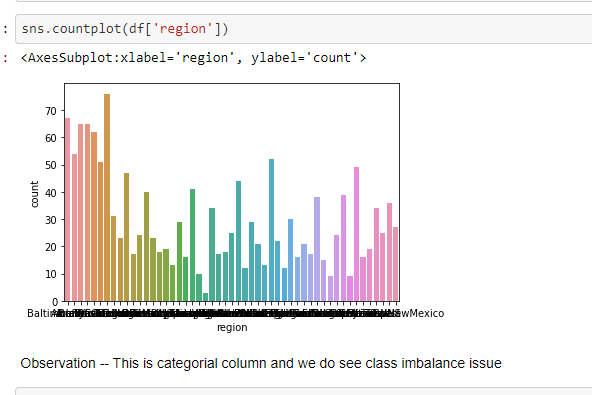
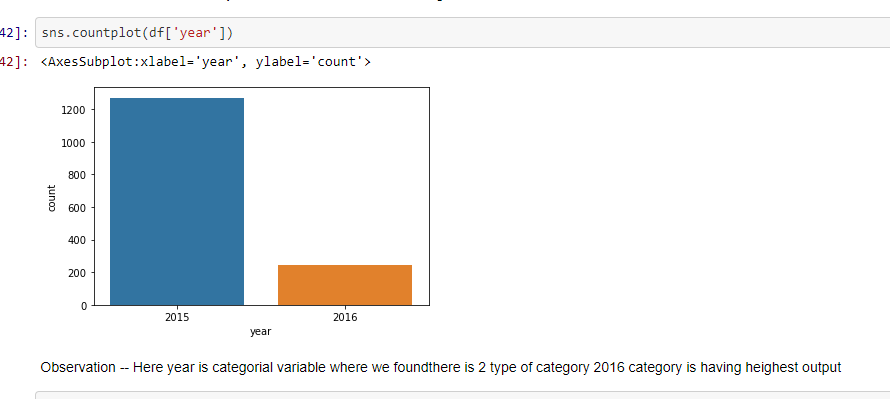
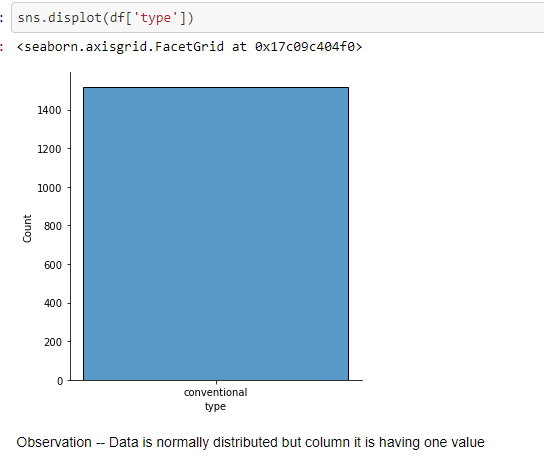
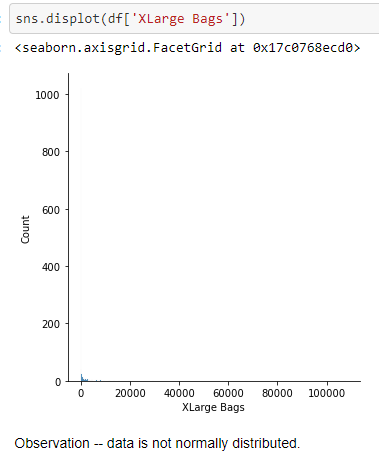
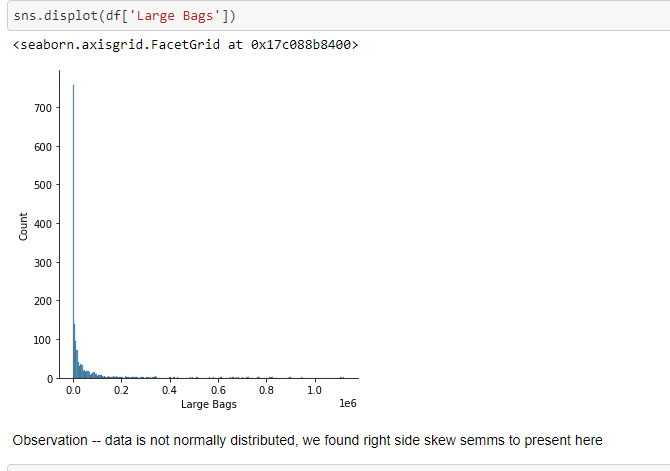
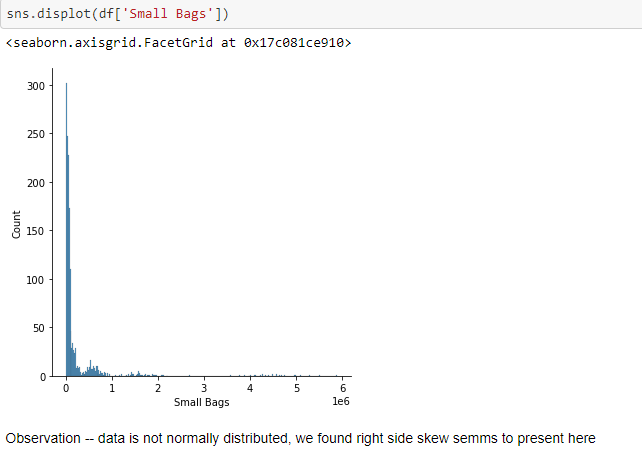
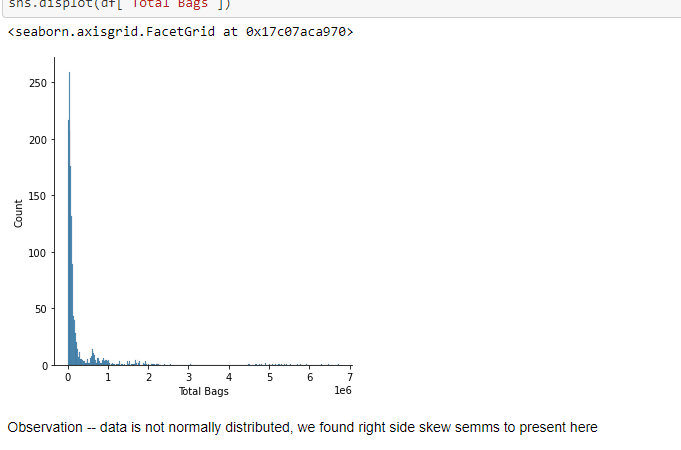
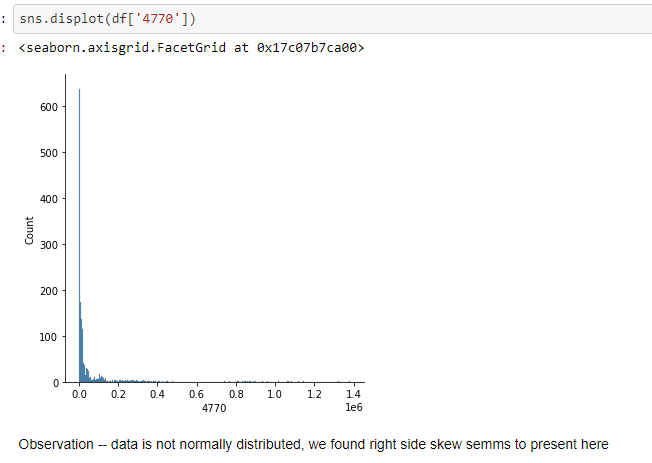
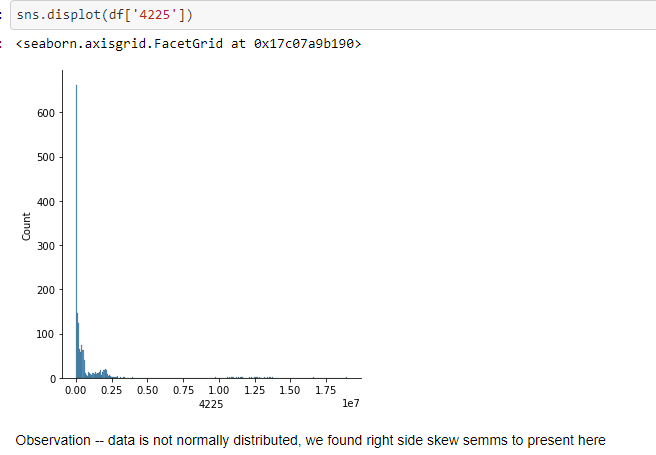
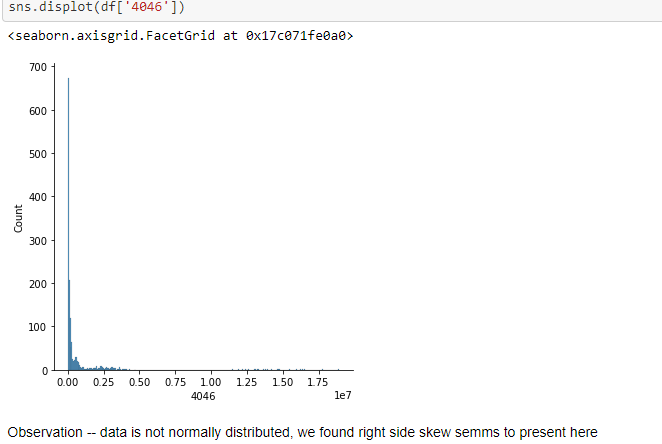
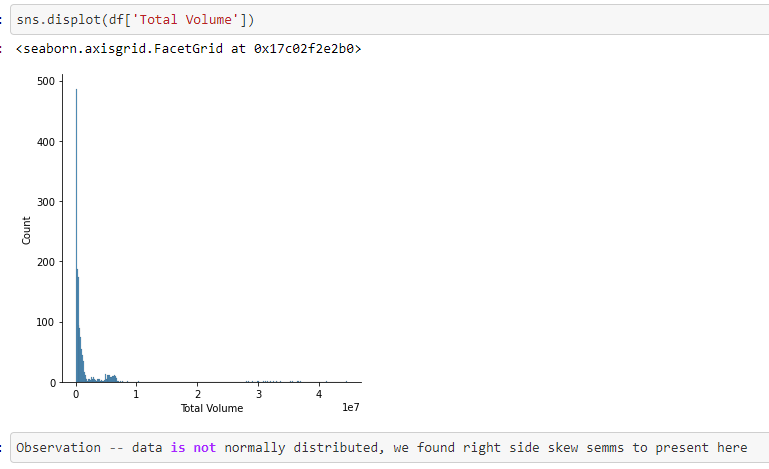
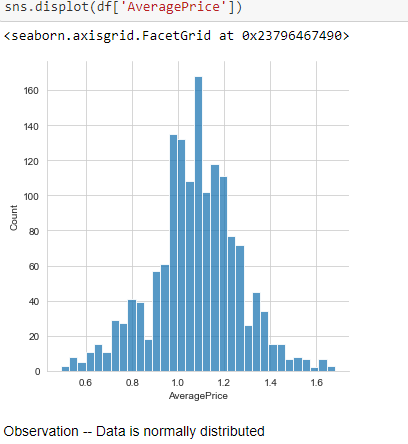
Checking the columns

df.column



# Data visualization

Univariate analysis:



**Now we will convert categorical data to numeric using label encoder**

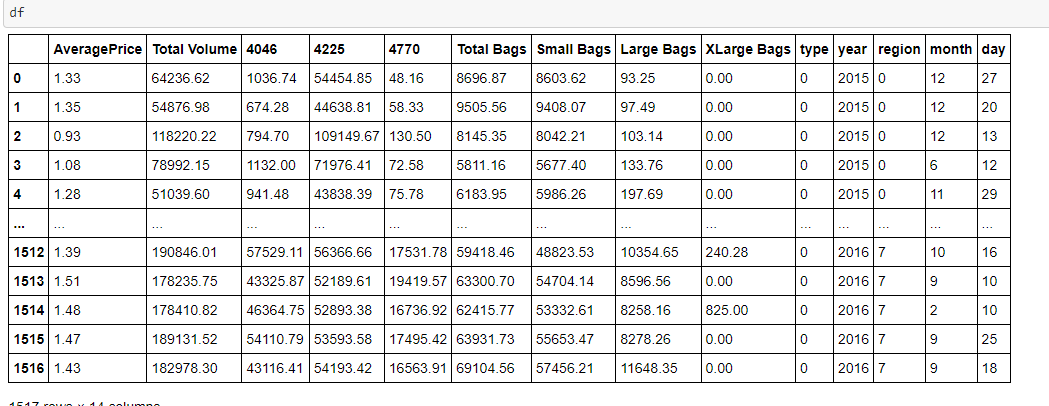
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df['type']=le.fit\_transform(df['type'])

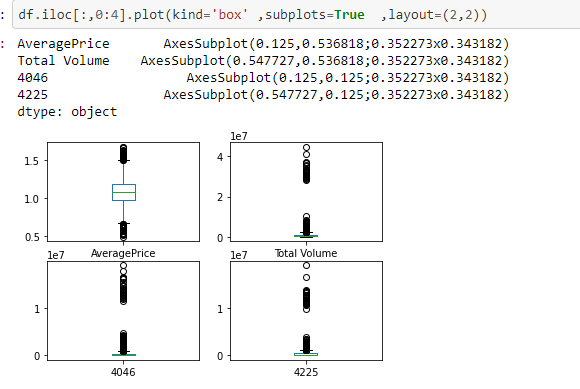
df['region']=le.fit\_transform(df['region'])

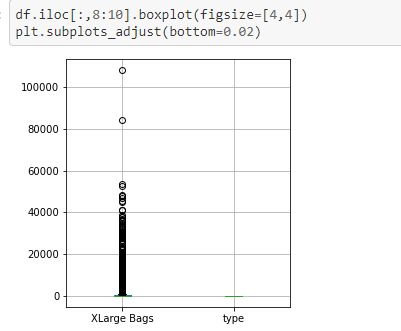
now df is

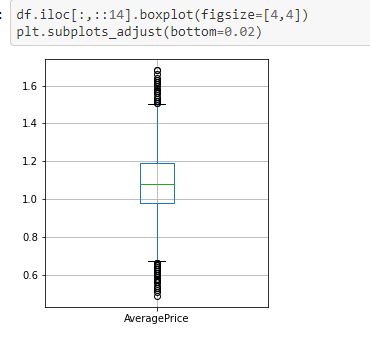
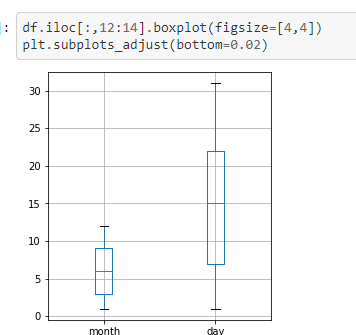
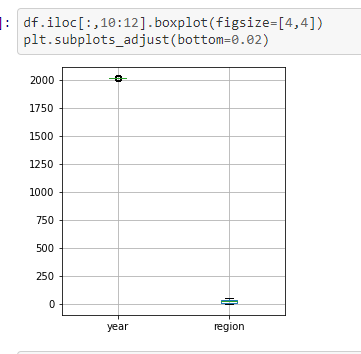


Outlier detection through boxplot:

A **boxplot** is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). ... It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed

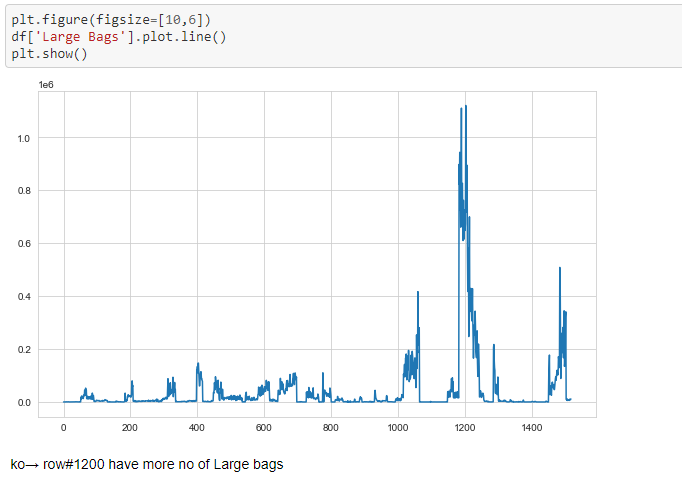
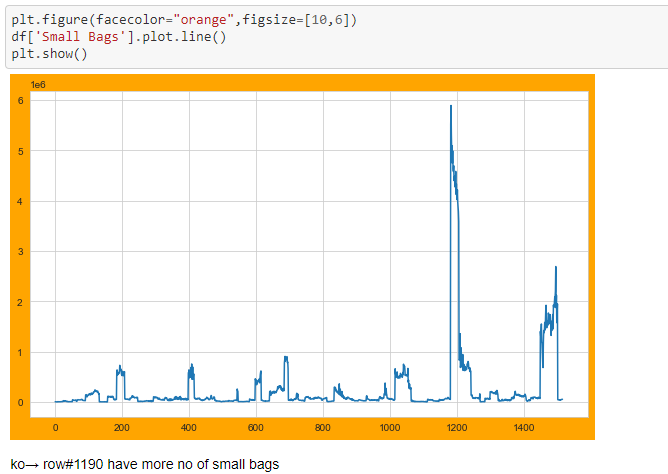
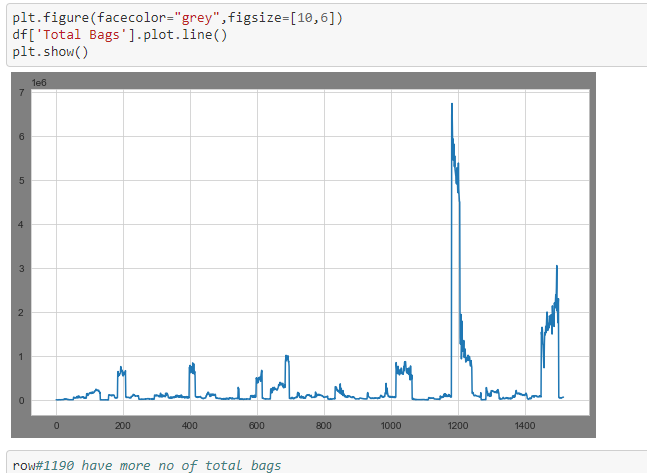
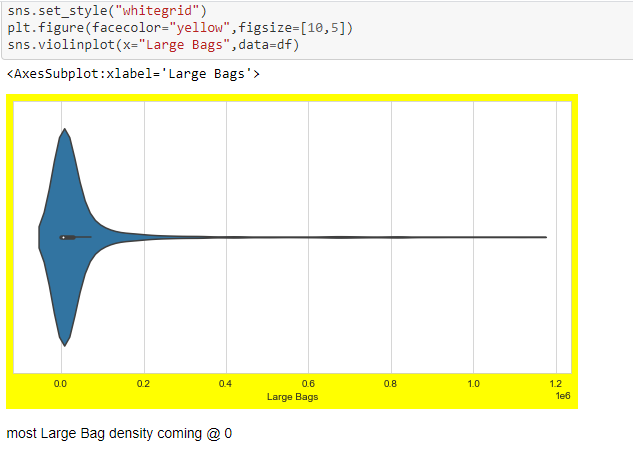
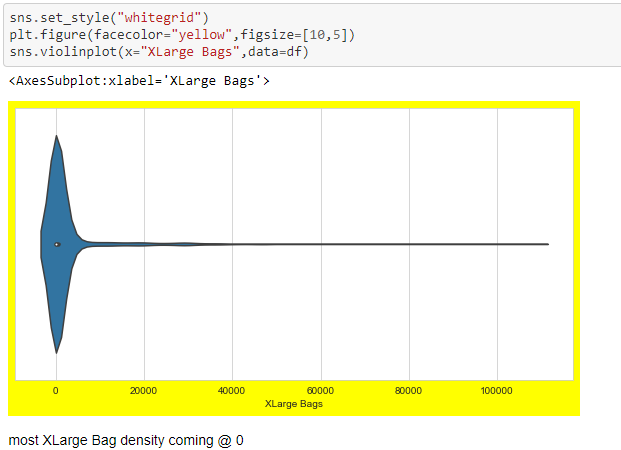
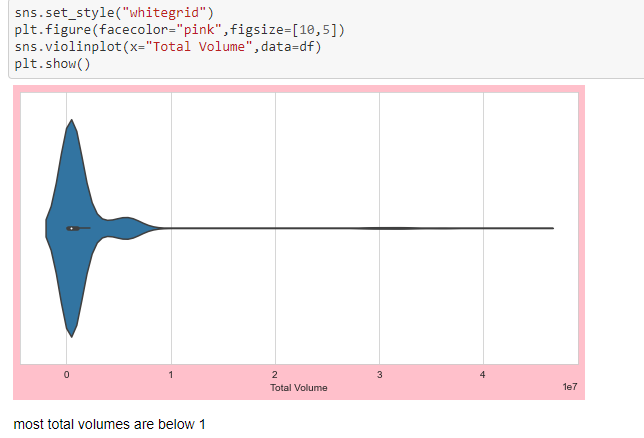
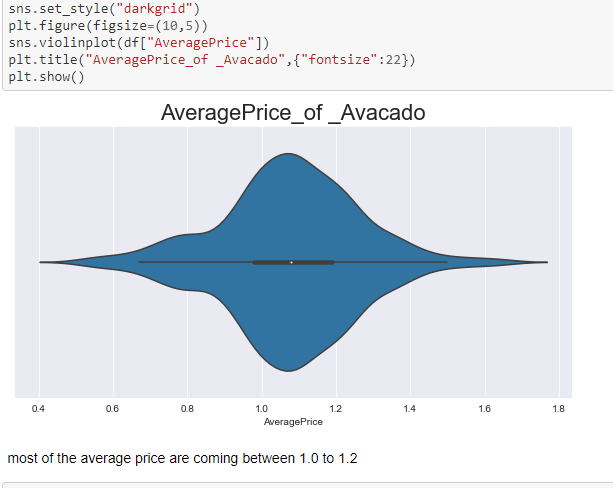






Observation –Where the value presents above and below vertices showing as dots those are outliers

**Some more plots to understand dataset :**

****

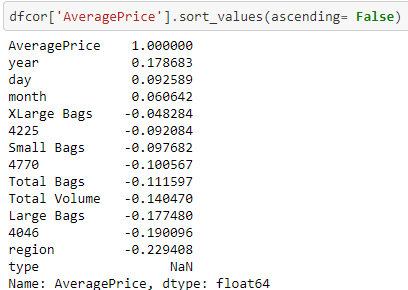
**Correlation:**

For checking correlationship of output label with other input features we will calculate cormatrix and sort the value ascending order

corr() is used to find the pair wise correlation of all columns in the data frame. Any nan values are automatically excluded. For any non-numeric data type columns in the data frame it is ignored

Input---

dfcor=df.corr()

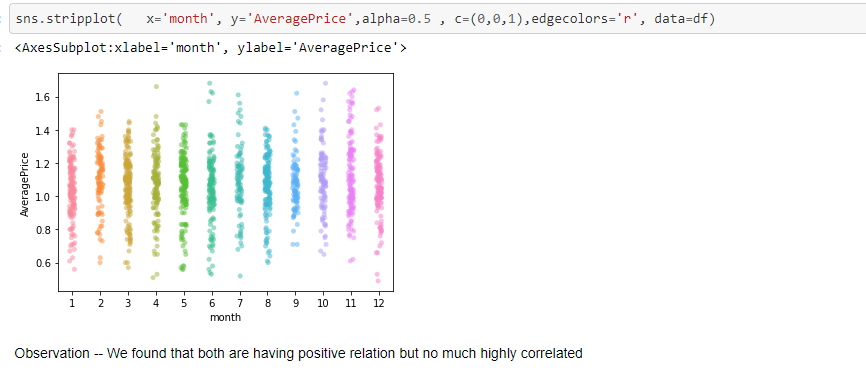
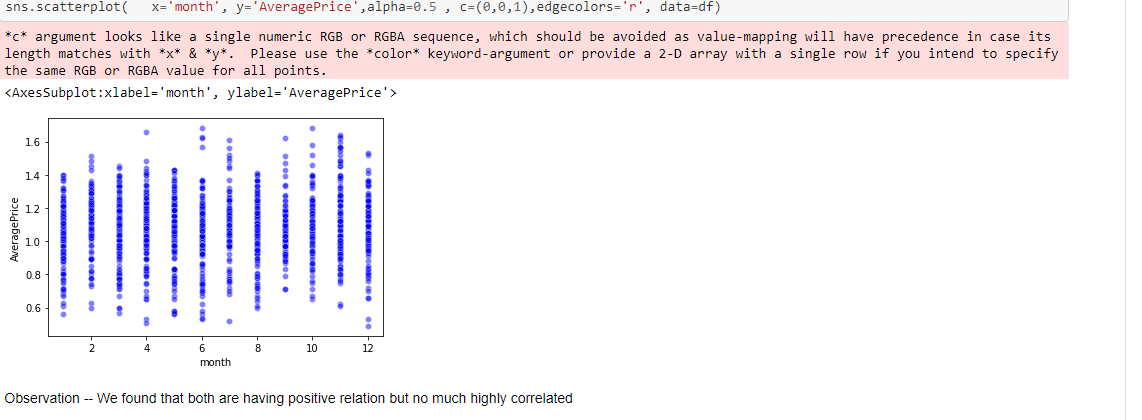
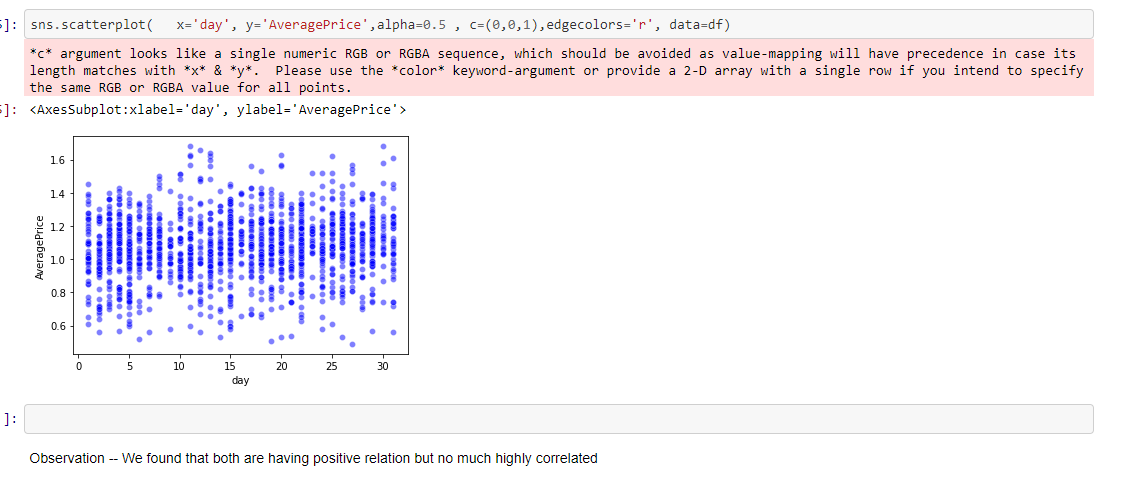
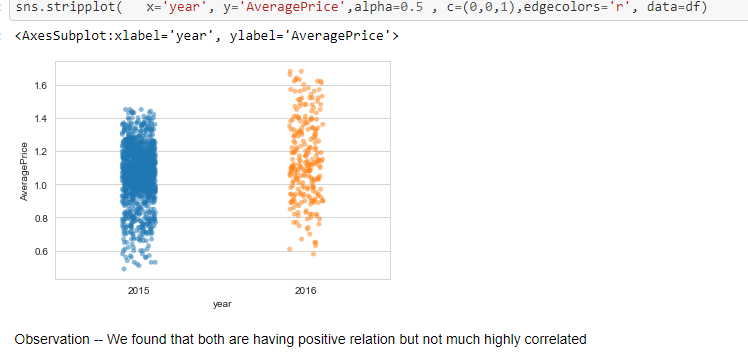
****

**Bi-variate analysis:**

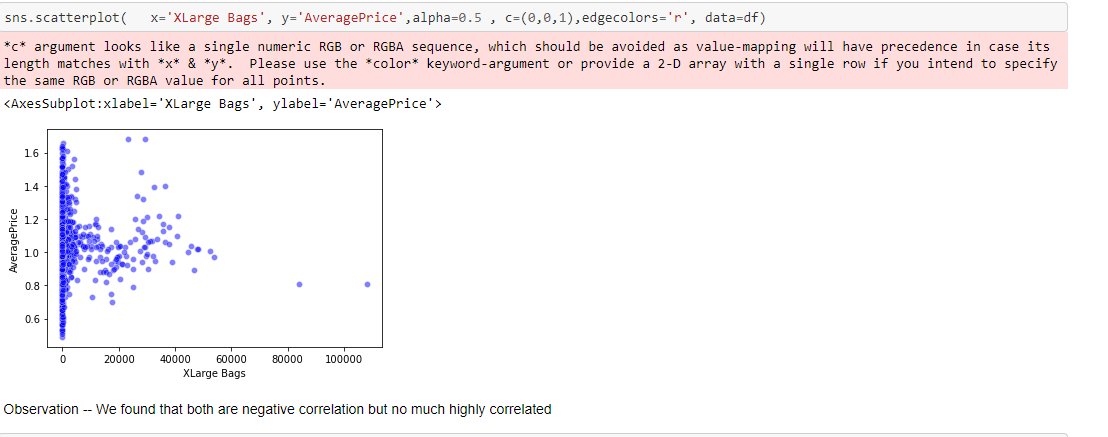
Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. Bivariate analysis can be helpful in testing simple hypotheses of association.

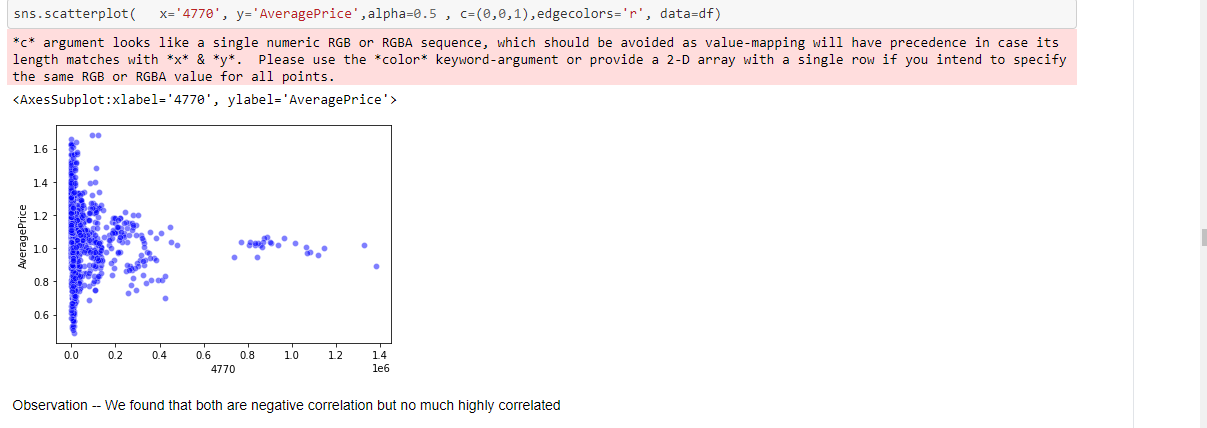
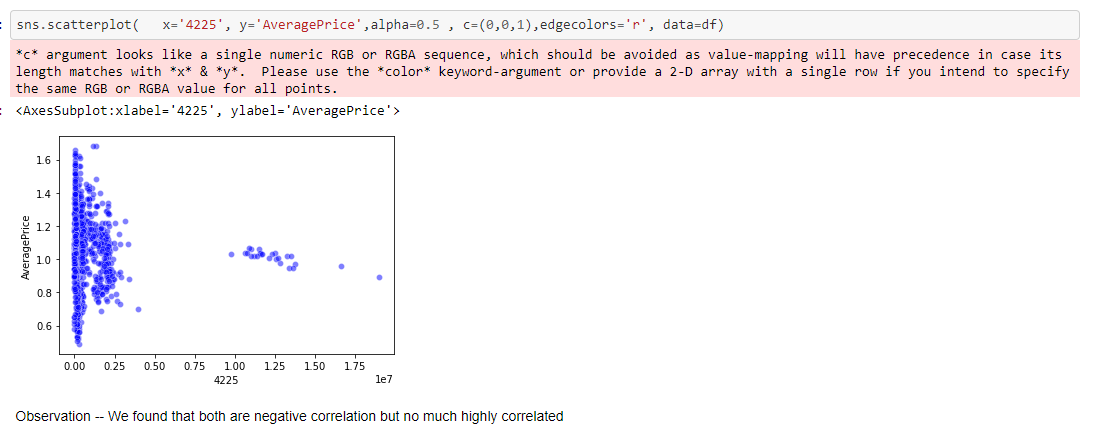
Bivariate analysis is one of the simplest forms of quantitative analysis. It involves the analysis of two variables, for the purpose of determining the empirical relationship between them. Bivariate analysis can be helpful in testing simple hypotheses of association.

If both two variable are continuous we check data distribution through scatter plot and if one is categorical, other one is continuous we will check through stripplot .below we have shown some bivariate analysis

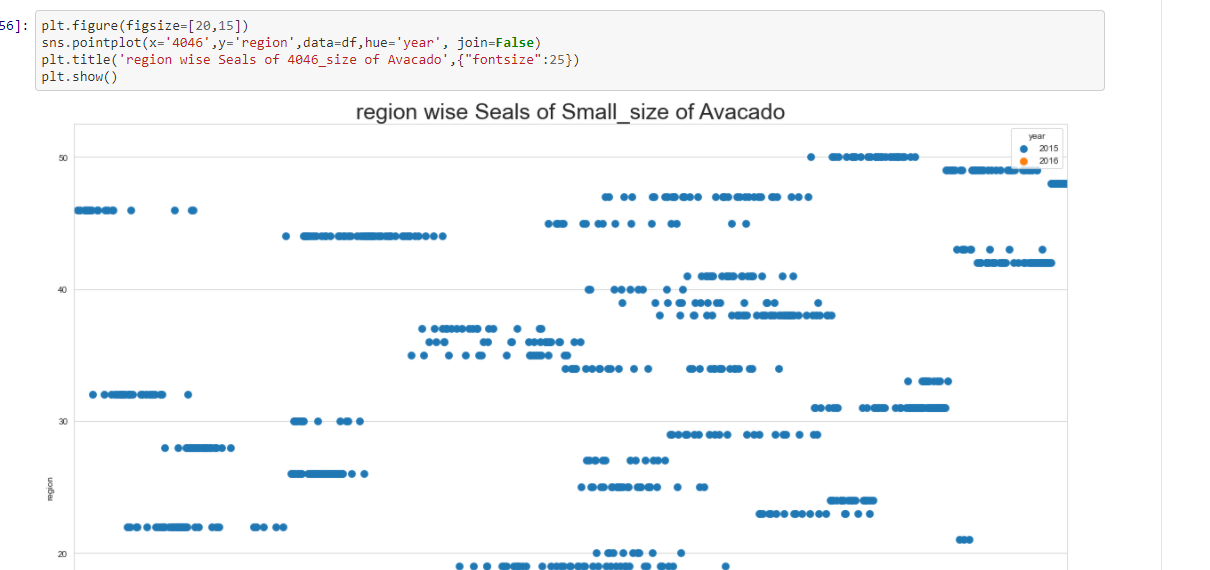


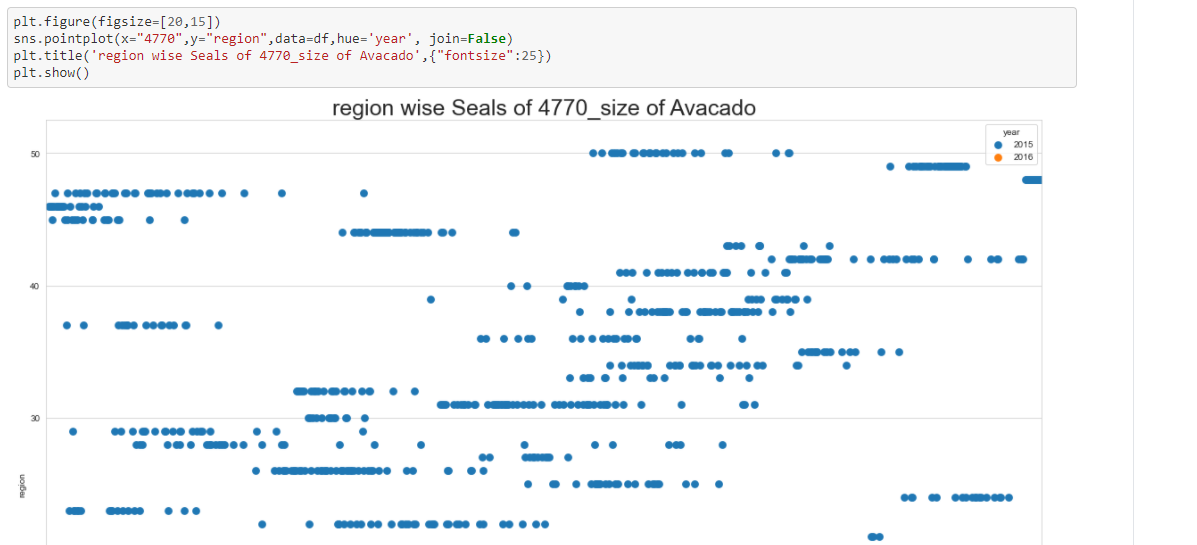
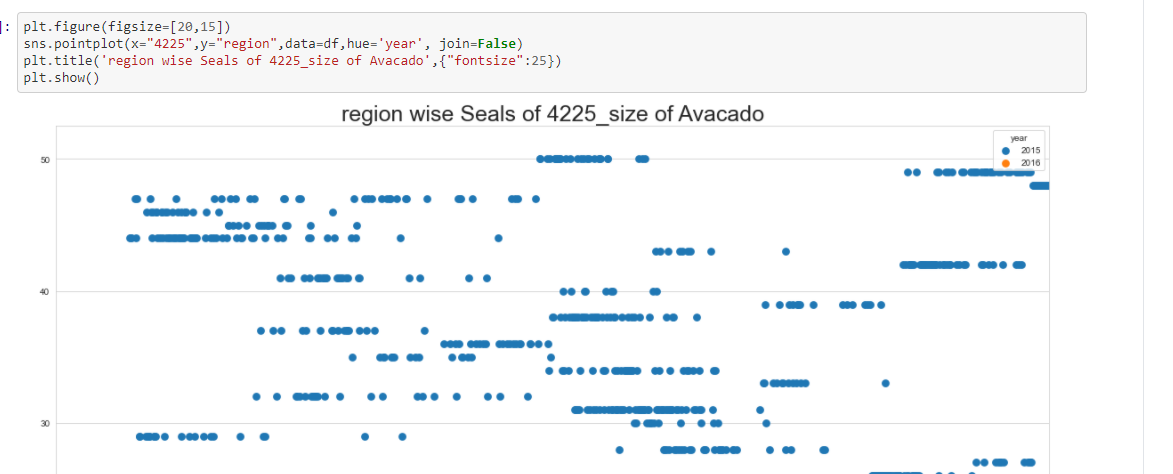
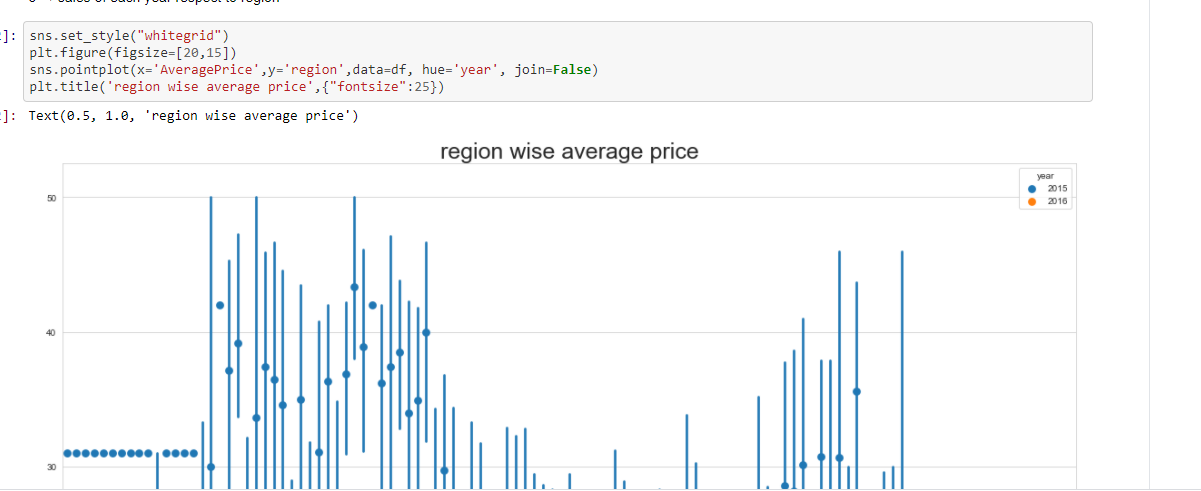
# Now we will see negative correlation plotting that which features having negative correlation with label(output)





**Some more Bi-variate analysis**

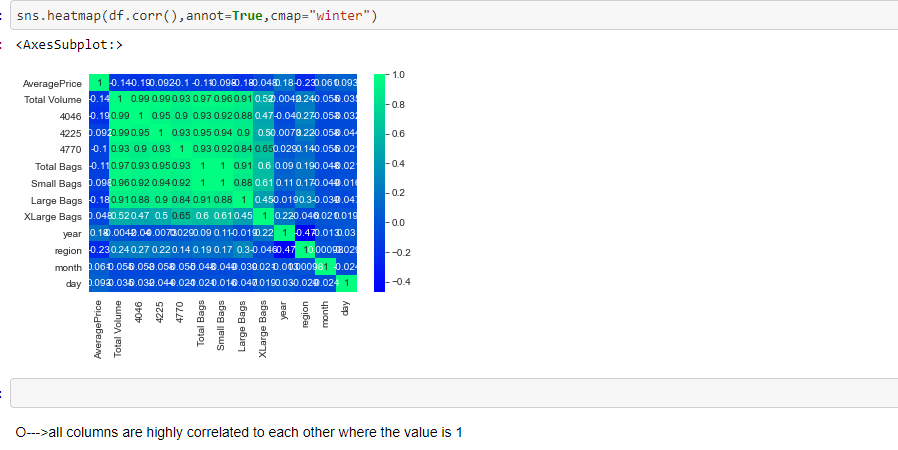
****



**Multivariate Analysis**

Multivariate statistics is a subdivision of statistics encompassing the simultaneous observation and analysis of more than one outcome variable. Multivariate statistics concerns understanding the different aims and background of each of the different forms of multivariate analysis, and how they relate to each other

Pair plot and heat map we can use to analysis Multivariate analysis





**Data pre-processing**

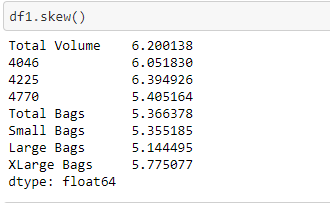
df1 will be my new dataset where we are dropping target column and categorial column because we will work with df1 dataset from where we will be removing outlier and skew , hence we should not remove skew and outlier

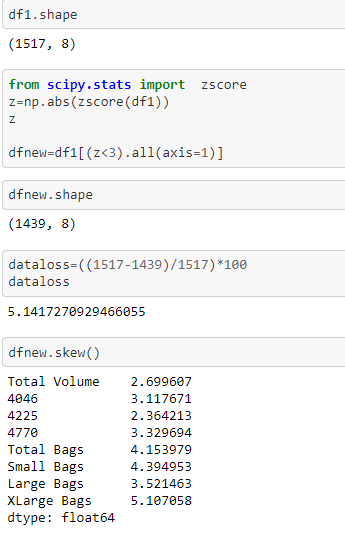
df1=df.drop(['AveragePrice' ,'year','month','day','region'] , axis=1)

Skew---A data is called as skewed when curve appears distorted or skewed either to the left or to the right, in a statistical distribution. In a normal distribution, the graph appears symmetry meaning that there are about as many data values on the left side of the median as on the right side.

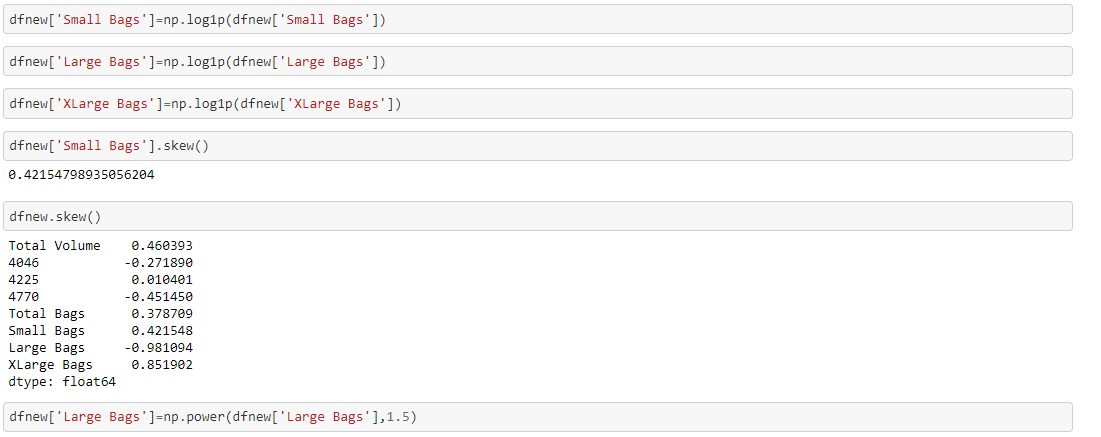
Outlier- In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses

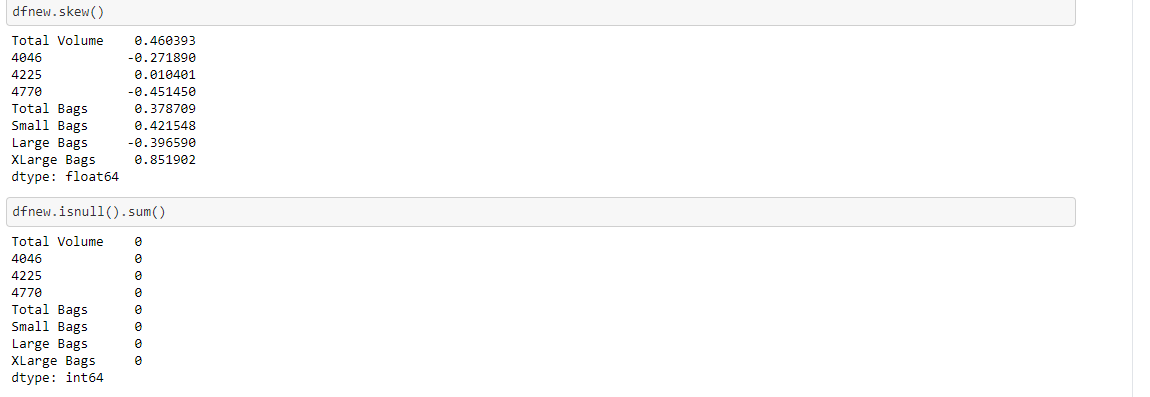
Skew detection:







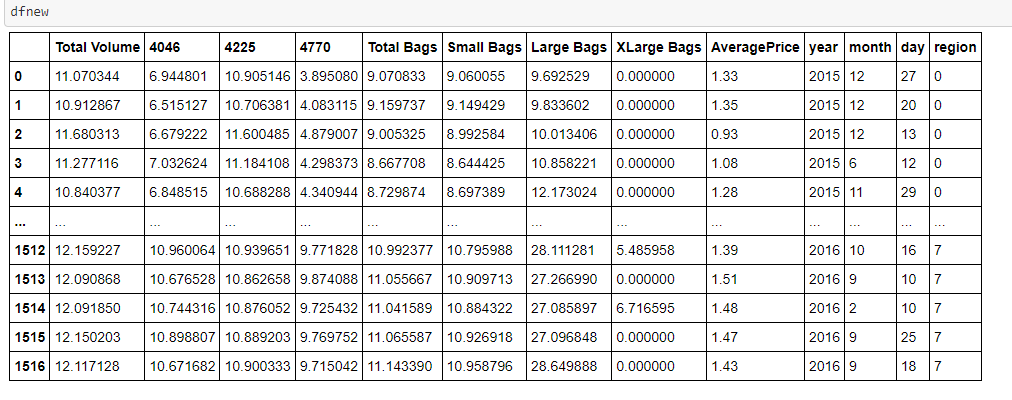




**Now appending other column**

dfnew[['AveragePrice,'year','month','day','region']]=df[['AveragePrice' ,'year','month','day','region']]

New dfnew is—



**Data pre-processing**

import statistics as st

from sklearn.model\_selection import train\_test\_split

y=dfnew['AveragePrice']

x=dfnew.drop(('AveragePrice'), axis=1)

StandardScaler. StandardScaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation. ... StandardScaler makes the mean of the distribution 0. About 68% of the values will lie be between -1 and 1.

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_std = sc.fit\_transform(x)

scaling the input

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_std, y,test\_size=.22 , random\_state=11)

**Linear regression:**

**lg=LinearRegression()**

**lg.fit(x\_train,y\_train)**

**lg.score(x\_train,y\_train)**

**lg.fit(x\_train,y\_train)**

**lg.score(x\_train,y\_train)**

**Regularization**

In mathematics, statistics, finance, computer science, particularly in machine learning and inverse problems, **regularization** is the process of adding information in order to solve an ill-posed problem

ls=Lasso(0.001)

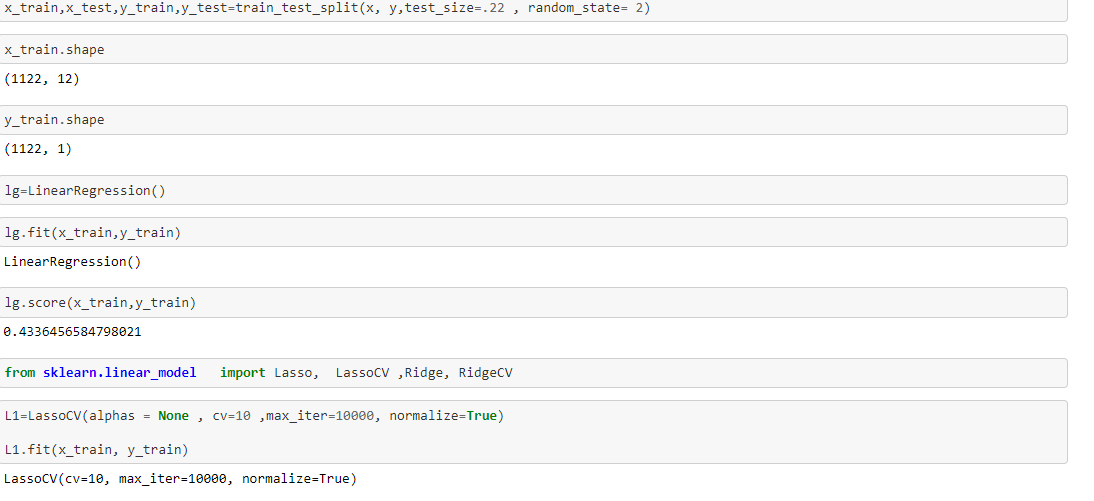
ls.fit(x\_train,y\_train)

ls.score(x\_train,y\_train)

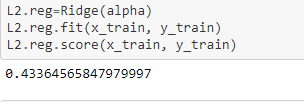
rd=Ridge(0.001)

rd.fit(x\_train,y\_train)

rd.score(x\_train,y\_train)



****

****

We have performed train test where we have send data to model ( some data for training and some for testing ). We have used 5 model to

* DecisionTreeRegressor
* KNeighborsRegressor()
* RandomForestRegressor()
* XGBRegressor

**ALGORITHIM**

**DecisionTreeRegressor**

dtr=DecisionTreeRegressor()

dtr.fit(x\_train,y\_train)

preddtc=dtr.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,preddtc))

print("R2SCORE" ,r2\_score(y\_test,preddtc))

mse== 0.013279179810725553

R2SCORE 0.607163788208468

**KNeighborsRegressor**

kmn=KNeighborsRegressor()

kmn.fit(x\_train,y\_train)

predkmn=kmn.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,predkmn))

print("R2SCORE" ,r2\_score(y\_test,predkmn))

mse== 0.01163988643533123

R2SCORE 0.6556587863020041

**RandomForestRegressor**

rvr=RandomForestRegressor()

rvr.fit(x\_train,y\_train)

predrvr=rvr.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,predrvr))

print("R2SCORE" ,r2\_score(y\_test,predrvr))

mse== 0.007987251369185646

R2SCORE 0.607163788208468

**AdaBoostRegressor**

from sklearn.ensemble import AdaBoostRegressor

ad=AdaBoostRegressor()

ad.fit(x\_train,y\_train)

predrvr=ad.predict(x\_test)

print("mse==",mean\_squared\_error(y\_test,predrvr))

print("R2SCORE" ,r2\_score(y\_test,predrvr))

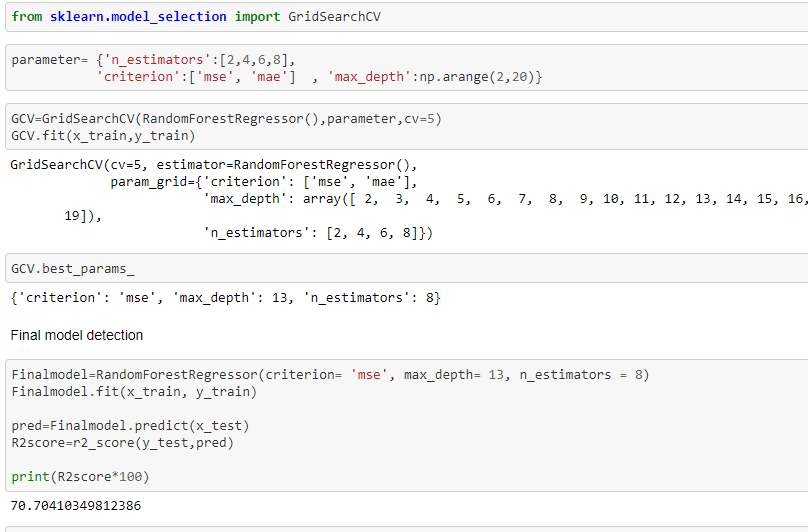
mse== 0.01478407064313581

R2SCORE 0.5626448026845006

# I found RandomForestRegresso r is the best model which having minimum difference between cross\_val\_score and R2 score

**Hyper tuning the model :**

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate “hyperparameters.” Hyperparameters can be thought of as the “dials” or “knobs” of a machine learning model



**Model saving:**

We saved the model in binary format



**Best model selection**

We have calculated cross validation score of each model. Cross-validation is a statistical method used to estimate the skill of machine learning models and we found RandomForestRegressor has having less difference between R2 and cross validation score .So as per logic RandomForestRegressor is our best model

**Conclusion**

* we have transform categorical data to numeric using Label Encoder
* We have plotted graphical view of each column to understand data distribution , kernel density, as well as for finding outlier concept
* we divided data x and y as a data and target
* We optimize model using hyper tuning parameter (hyper parameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem)
* RandomForestRegresso r is the best model which having minimum difference between cross\_val\_score and R2 score
* We got our final model
* We saved out final model in as .pkl file . It is basically Binary format of output

**Hardware and Software Requirements and Tools Used**

Hardware

* Good performance PC [Minimum – 8gb RAM +SSD]
* Enough space in hard disk drive

Software requirements

* jupyter note book
* Sometimes you may need Google colab to cross check the output

Package

* Numpy ---import numpy as np ( For calculation )
* **P**anda-import pandas as pd (read data frame )
* Imblearn----- For class sampling

Here the list of some other function

* For plotting- 1>import seaborn as sns 2> import matplotlib.pyplot as plt
* For ignore new version warning--- import warnings

warnings.filterwarnings('ignore')’

import statistics as st

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error ,r2\_score import statistics as st

from sklearn.tree import DecisionTreeRegressor

from sklearn.svm import SVR

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

**Limitation:**-

* The data could be incomplete. even the lack of a section or a substantial part of the data, could limit its usability.
* We don’t get always accurate information as data might be not totally completed .

Thank you