Blog on Avocado Data Set Implementing Machine Learning

The above picture shows a different kind of Avocado species available.

Persea americana, commonly known as avocado, is becoming increasingly important in global agriculture. There are dozens of avocado varieties, but more than 85% of the avocados harvested and sold in the world are of the Hass one. Furthermore, information on the market of agricultural products is valuable for decision-making; this has made researchers try to determine the behavior of the avocado market, based on data that might affect it one way or another. In this paper, a machine learning approach for estimating the number of units sold monthly and the total sales of Hass avocados in several cities in the United States, using weather data and historical sales records, is presented. For that purpose, four algorithms were evaluated: Linear Regression, Multilayer Perceptron, Support Vector Machine for Regression and Multivariate Regression Prediction Model. Using the Multivariate Regression Prediction Model, an application that allows avocado producers and sellers to plan sales through the estimation of the profits in dollars and the number of avocados that could be sold in the United States was created.

Introduction

Currently, 85% of the avocados produced and sold in the world are of the Hass variety. This variety grows almost the whole year round and in different regions. Some of the leading producing countries are Mexico, United States, Chile, Australia, South Africa and Israel, with Mexico being the largest producer in the world, representing about a third of the worldwide production. The United States is the leading country concerning imports, and has evolved towards a market of almost a million tons of avocados. The avocado market has grown 16% every year since 2008 in the United States, and this trend is expected to continue, at least in the medium term. States like Florida, California and Hawaii are producers of avocado in this country, but the production does not meet the market demands, so avocados are imported from Mexico, Chile, Peru, New Zealand and the Dominican Republic, among other countries. However, avocado consumption is not uniform across the country. For example, about 90% of the families in California consume avocados, in a proportion of more than three units per month. However, in some states of the Great Plains, only a little more than half of the families consume this fruit, in a proportion of no more than two units per month.

Collecting information on the market and on better practices concerning avocado cultivation would be of great help to producers, vendors, associations, and companies. This could be used to choose the right places to sell avocados, to carry out successful marketing campaigns or to develop innovations for the production and sales of such product. An example of this fact is that about fifty million dollars are spent annually on advertising and promotional activities on healthy avocado consumption.

Several authors have found that weather is one of the factors affecting economic and commercial behaviour. In order to predict future behaviors or relationships, there is a subset of artificial intelligence called machine learning, which includes building classification and regression models .

Data Sources

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:

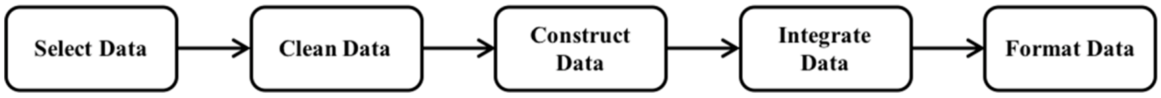
The table below represents weekly 2018 retail scan data for National retail volume (units) and price. Retail scan data comes directly from retailers' cash registers based on actual retail sales of Hass avocados. Starting in 2013, the table below reflects an expanded, multi-outlet retail data set. Multi-outlet 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. green skins) are not included in this table.

The avocado, a tree likely originating from south-central Mexico, is classified as a member of the flowering plant family Lauraceae. The fruit of the plant also called an avocado, is botanically a large berry containing a single large seed.

You can download the data set and code from this Link:-
https://raw.githubusercontent.com/chainhaus/pythoncourse/master/avocado.csv

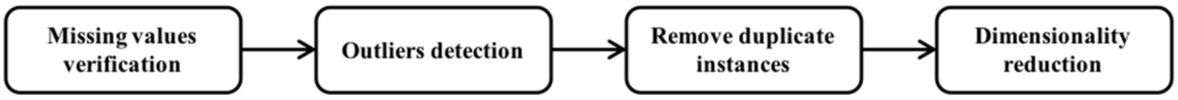
Data Selection and Cleaning

Data acquired was prepared and cleaned in order to obtain an appropriate dataset for understanding the fluctuation of the avocado market in the United States, based on different weather conditions. For this purpose, we followed the Cross Industry Standard Process for Data Mining (CRISP-DM) methodology and the data cleaning process proposed exposes the data preparation tasks that were considered.



As a first step, data was selected. The number of avocados sold over a four-week period in two different years (the current year and the immediately previous one) were labeled as units-cy and units-py, respectively. The total sales in dollars for the same years, labeled as sales-cy, and sales-py, were chosen from the data on the market in different cities in the United States that had been retrieved. Concerning the weather, we selected the variables of maximum temperature (◦C), minimum temperature (◦C), average temperature (◦C), maximum humidity (%), minimum humidity (%), average humidity (%) and precipitation (mm).

Afterwards, we cleaned the two datasets following the data cleaning tasks presented



- Searching for missing values: blank spaces, words such as “NaN” or “null” and special characters such as “*” or “?” were searched for, as to verify if there were any missing values in the datasets. There was missing data only on the weather; this may be due to failures in the operation of the meteorological stations that capture such data. Since there was little data missing (a total of five records), we decided to delete those instances in both the weather and the market data. Finally, no data imputation process was implemented.
- Outliers detection: to detect if there were values that significantly deviated from the others, we used Clustering (DBSCAN, Density-based spatial clustering of applications with noise) and Distance (LOF, Local Outlier Factor) algorithms. No outliers or extreme values were found for the two datasets.
- Searching for duplicated instances: We used a filter in order to detect if there were repeated instances. The filter found no instances with this problem, since we constructed the datasets carefully and imputation algorithms were not implemented in the previous steps.

Dimensionality reduction: an algorithm for this task was applied. Typically, the algorithm searches for a subset of most relevant features to represent the dataset. The objective is to contribute to learning accuracy. Considering the limited data dimensionality of the datasets used, the dimensionality was not reduced.

Results

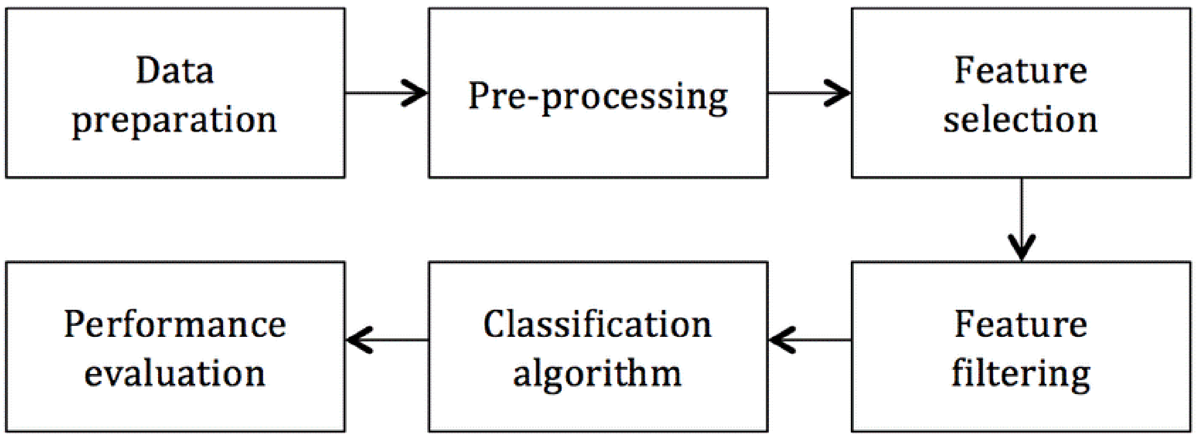
Modeling for the Forecasting of the Avocado Market

Aiming at observing the fluctuation of the avocado market in the United States based on weather conditions, several machine learning techniques were evaluated to estimate the number of avocados sold and the total sales (in dollars) of this agricultural product.

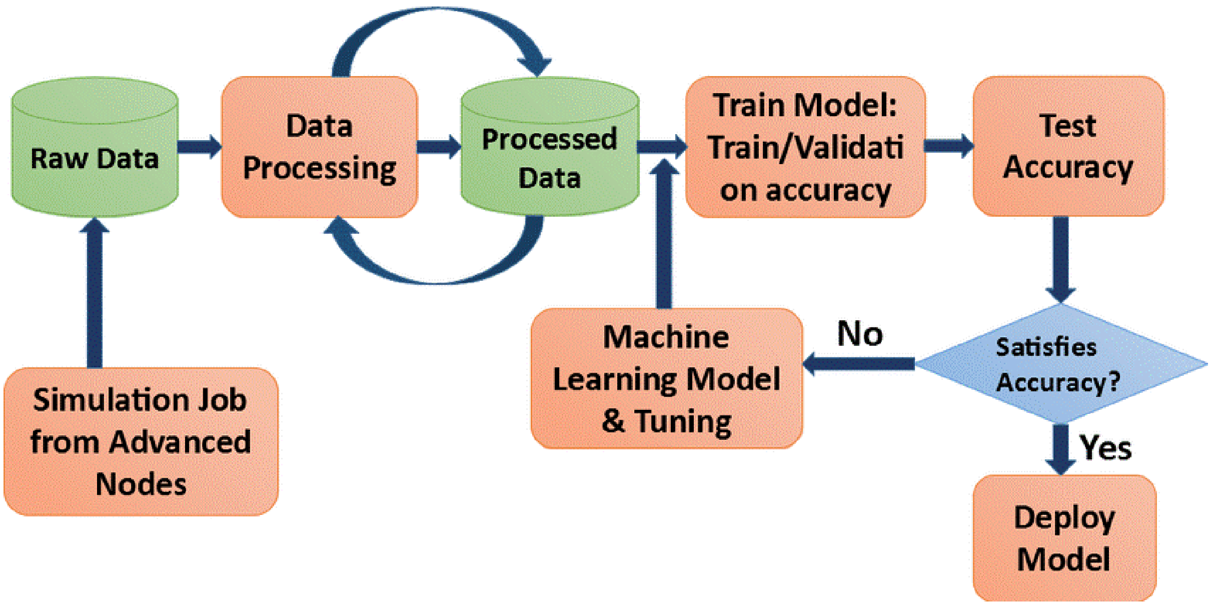
- Linear Regression: a technique used to determine the relationship of a y variable with one or many other x1, . . . , xk variables. In a machine learning approach, it searches for several functions that model the relationship between the variables and selects the one that most closely approximates to or fits the data given.
- Multilayer Perceptron: consists of units, called neurons, interconnected and organized in layers. Each neuron processes information, converting the input it receives into processed output. Through the links of neurons, knowledge is generated.
- Support Vector Machine for Regression: these algorithms seek to estimate a function from a training data. For this purpose, an initial set of points is required, which also contains two other subsets of points and which belongs to one of two possible classes. Based on these, the support vector machine creates a hyper-plane in order to find the largest distance separating the classes and thus builds a model that is capable of predicting which class a new point.
- Multivariate Regression Prediction Model: an algorithm that associates traditional decision trees with linear regression functions. To some special nodes, commonly known as leaves, the algorithm assigns a probability vector that indicates the chances that a class will take a certain value. The instances are classified following a path from the root of the tree to a leaf, according to the results of the tests performed in each of the test nodes.

To evaluate the predictive models, a 10-fold cross-validation method was used. In addition, we compared the models generated using the Correlation Coefficient (CC), Mean Absolute Error (MAE) and the Relative Absolute Error (RAE).

Work Flow:



Process how to build the model:-


Raw Data
Data Processing
Processed Data
Train Model: Train/Validati on accuracy
Test Accuracy
Satisfies Accuracy?
No
Machine Learning Model & Tuning
Yes
Deploy Model
Simulation Job from Advanced Nodes

Inspiration /Label

The data set can be seen in two angles to find the city or region and find the average price. So, I will predict this data set in both ways.

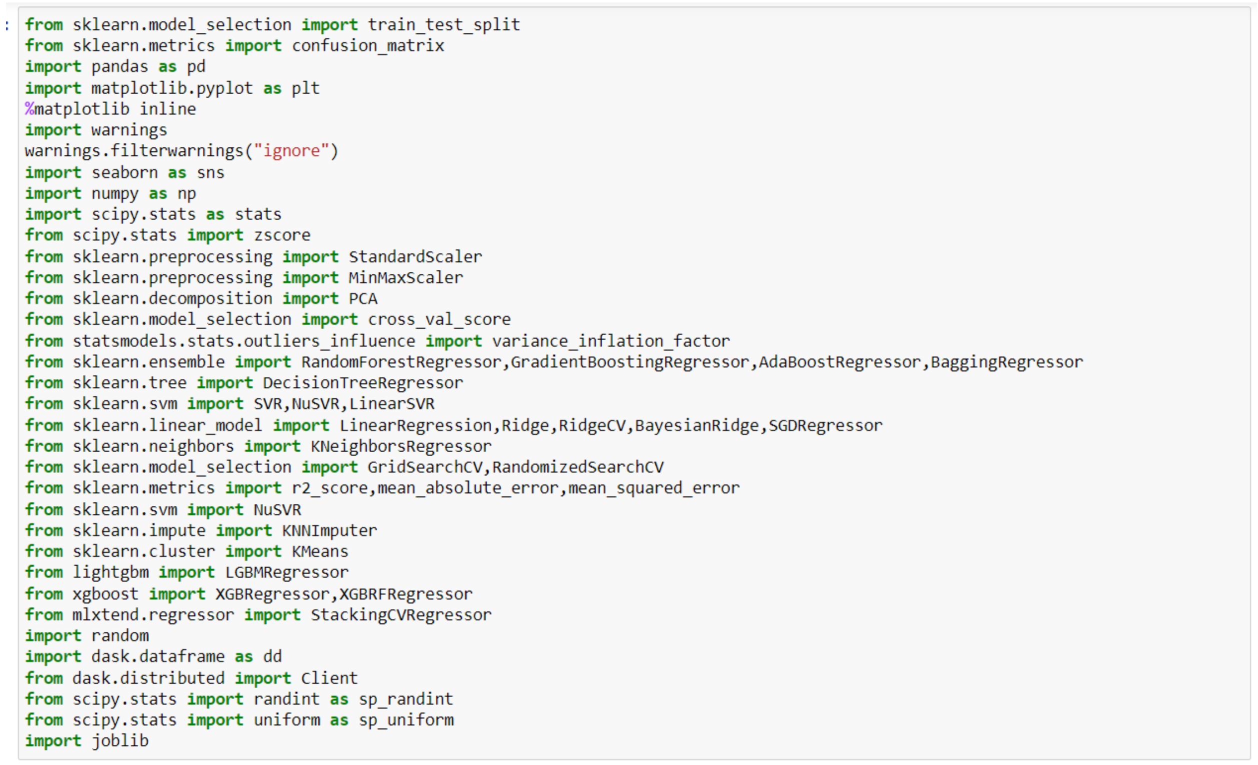
Fields/Columns:

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

Problem description:

The goal is to predict the average price which is continuous in nature of the different types of avocados and use the region in which region they are lying.

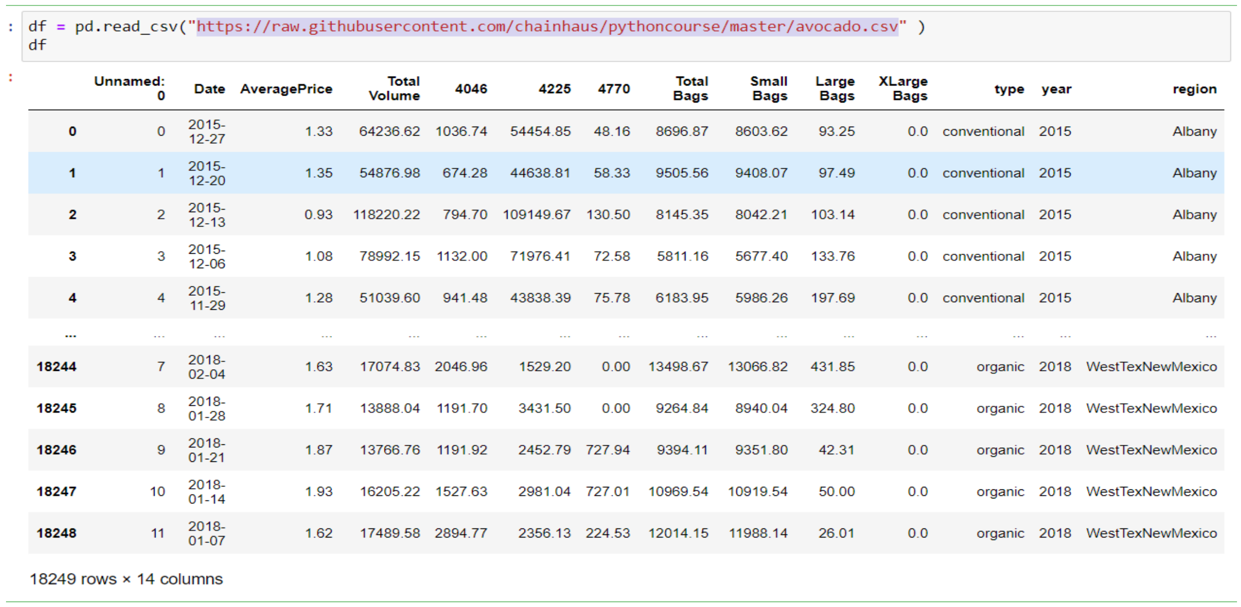
Importing Library:



```
: from sklearn.model_selection import train_test_split
  from sklearn.metrics import confusion_matrix
  import pandas as pd
  import matplotlib.pyplot as plt
  %matplotlib inline
  import warnings
  warnings.filterwarnings("ignore")
  import seaborn as sns
  import numpy as np
  import scipy.stats as stats
  from scipy.stats import zscore
  from sklearn.preprocessing import StandardScaler
  from sklearn.preprocessing import MinMaxScaler
  from sklearn.decomposition import PCA
  from sklearn.model_selection import cross_val_score
  from statsmodels.stats.outliers_influence import variance_inflation_factor
  from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor,AdaBoostRegressor,BaggingRegressor
  from sklearn.tree import DecisionTreeRegressor
  from sklearn.svm import SVR,NuSVR,LinearSVR
  from sklearn.linear_model import LinearRegression,Ridge,RidgeCV,BayesianRidge,SGDRegressor
  from sklearn.neighbors import KNeighborsRegressor
  from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
  from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
  from sklearn.svm import NuSVR
  from sklearn.impute import KNNImputer
  from sklearn.cluster import KMeans
  from lightgbm import LGBMRegressor
  from xgboost import XGBRegressor,XGBRFRegressor
  from mlxtend.regressor import StackingCVRegressor
  import random
  import dask.dataframe as dd
  from dask.distributed import Client
  from scipy.stats import randint as sp_randint
  from scipy.stats import uniform as sp_uniform
  import joblib
```

I am importing all libraries which I required for EDA, visualization, prediction and finding all matrices. The reason for doing this is that it becomes easier to use all the import statement at one go and we do not require to import the statement again at each point. We could find all the importing statements at one place without finding it on whole notebook and can update them also.

Loading Data Set into a variable:



```
: df = pd.read_csv("https://raw.githubusercontent.com/chainhaus/pythoncourse/master/avocado.csv" )
  df
```

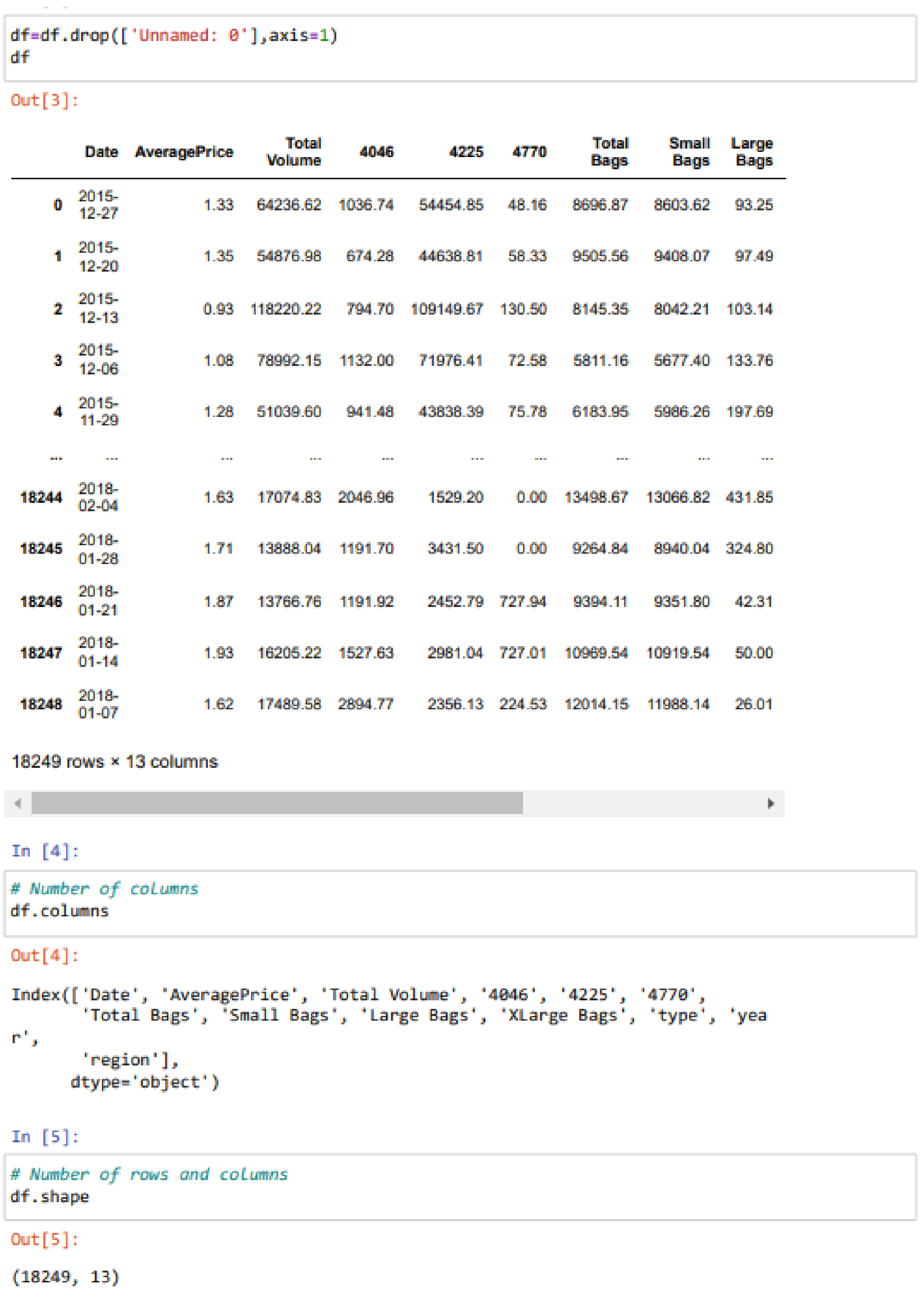
	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	organic	2018	WestTexNewMexico
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	organic	2018	WestTexNewMexico
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	organic	2018	WestTexNewMexico
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	organic	2018	WestTexNewMexico
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	organic	2018	WestTexNewMexico

18249 rows × 14 columns

Here I am loading the data set into a variable i.e. “df” and processing it in dataset 18249 rows and 14 columns. As in this data set, most of the columns are float in nature and type and sex is of categorical value.

Exploratory Data Analysis:

Drop column name unnamed



```
df=df.drop(['Unnamed: 0'],axis=1)
df
```

Out[3]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69
...	...	...	...	...	...	...	...	...	...
18244	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85
18245	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80
18246	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31
18247	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00
18248	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01

18249 rows × 13 columns

In [4]:

```
# Number of columns
df.columns
```

Out[4]:

```
Index(['Date', 'AveragePrice', 'Total Volume', '4046', '4225', '4770',
       'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bags', 'type', 'yea
r',
       'region'],
      dtype='object')
```

In [5]:

```
# Number of rows and columns
df.shape
```

Out[5]:

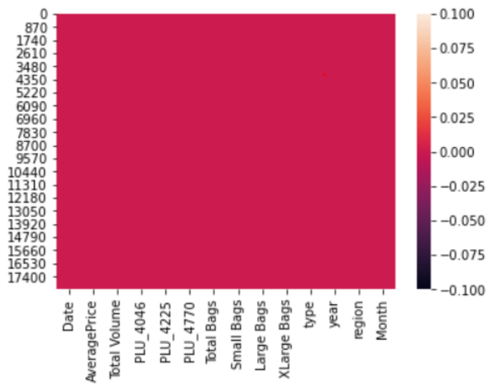
```
(18249, 13)
```

As seen in data set there is one unnamed column which does not play any important role for prediction in the price of avocado, so I am dropping that column.

Also, I am checking the shape of the data set as there are 18249 rows and 13 columns.

Also, most of the column are of same data type that is float and Date, type and region is of object data type.

```
In [19]: #checking null values using heatmap
         sns.heatmap(df.isnull())

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x24cba707f08>
```



Above I am checking the null values, as find there are no null values in the data set because the red colour is distributed equally correspond to each column.

```
In [15]: #checking the mean of price at each year of organic type of avacado
         organic.groupby('year')['AveragePrice'].mean()

Out[15]: year
         2015    1.673324
         2016    1.571684
         2017    1.735521
         2018    1.567176
         Name: AveragePrice, dtype: float64

In [16]: #finding how much type of avacado is sell in last 4 year
         df.groupby('year')['type'].value_counts()

Out[16]: year  type
         2015  conventional    2808
               organic         2807
         2016  conventional    2808
               organic         2808
         2017  conventional    2862
               organic         2860
         2018  conventional     648
               organic          648
         Name: type, dtype: int64
```

In above, I am finding that year 2017 is aggressive year where avocado price is higher as compared to other year and 2015 is at second number.

Also, I am finding that at each year present in the data set, which type of avocado is has total count, so both type of avocado is present almost in same amount in the data set.

Anaylise the dataset with the help of describe funtion

```
: df.describe()
```

	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	
count	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	18249.000000	18249.000000	18249.0
mean	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.396392e+05	1.821947e+05	5.433809e+04	3106.426507	0.499918	2016.1
std	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.862424e+05	7.461785e+05	2.439660e+05	17692.894652	0.500014	0.9
min	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000	2015.0
25%	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.088640e+03	2.849420e+03	1.274700e+02	0.000000	0.000000	2015.0
50%	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.974383e+04	2.636282e+04	2.647710e+03	0.000000	0.000000	2016.0
75%	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.107834e+05	8.333767e+04	2.202925e+04	132.500000	1.000000	2017.0
max	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.937313e+07	1.338459e+07	5.719097e+06	551693.650000	1.000000	2018.0

Above statistics data show that their multiple outliers mostly in XLarge Bags There is also difference between mean and 50% value in some of the columns which used to get fix for better prediction

- Also, number of rows in each column are same, means there are no null values in the data set.

- Also, the mean and 50%value of most of the column are same and the STD and mean are very close to each other.

- Most of the column statistics data are near to 0 values.

- By checking the difference between the 75% and max value there are outliers in some of the column.

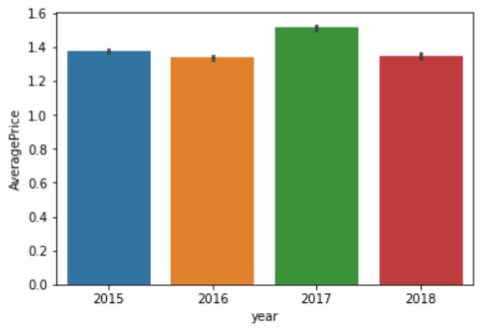
<u>Data Visualization:</u>

In this portion we can plot different graph using different columns and try to visualize the data using matplotlib and seaborn library.

We use different graph include:

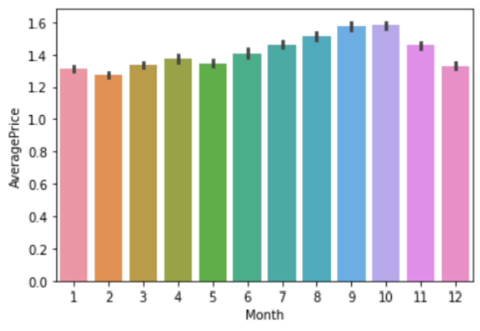
- Bar plot
- Count plot
- Line plot
- Histogram and Pair plot

```
In [21]: #checking ratio of year and price that which year had max average price
         sns.barplot(x = "year" , y = "AveragePrice" ,data=df )

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x24cbc7c7e08>
```



so above graph showing that 2017 year is that yera where max average price is there

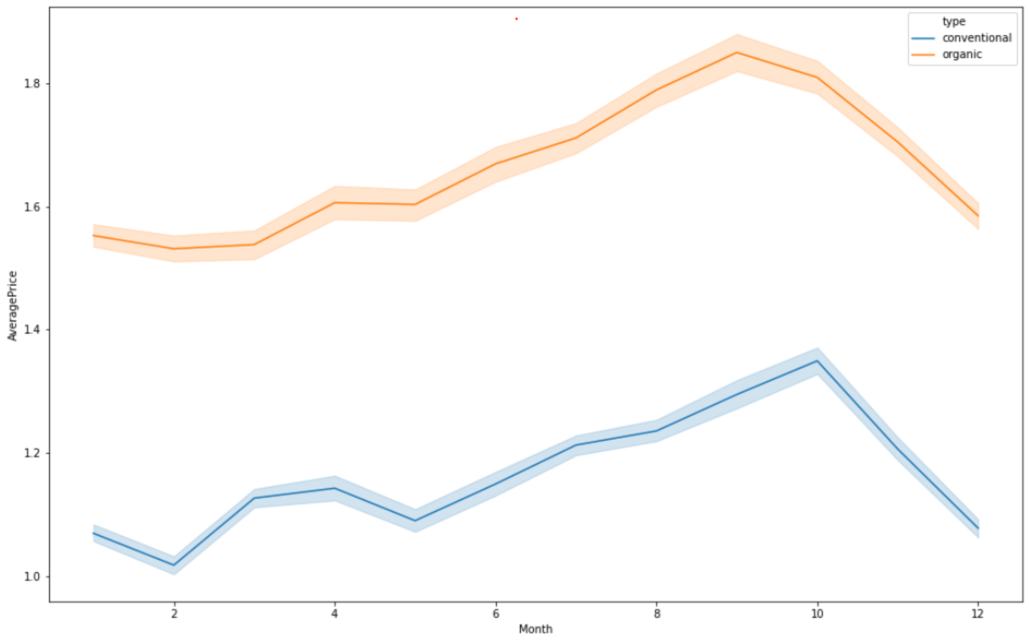
```
In [22]: #checking ratio of month and price that which year had max average price
         sns.barplot(x = "Month" , y = "AveragePrice" ,data=df )

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x24cbc942548>
```



From above we came to know that:

- Year 2017 is that year where the price is maximum as compared to other year, and there is less difference among rest of the year.
- September and October are the month where max no of average price is there, but the thing is almost for whole year the price is almost same for the avocado this prove that there is so much craze of avocado rather than India.

```
In [25]: plt.figure(figsize=(16,10))
         sns.lineplot(x="Month", y="AveragePrice", hue='type', data=df)
         plt.show()
```



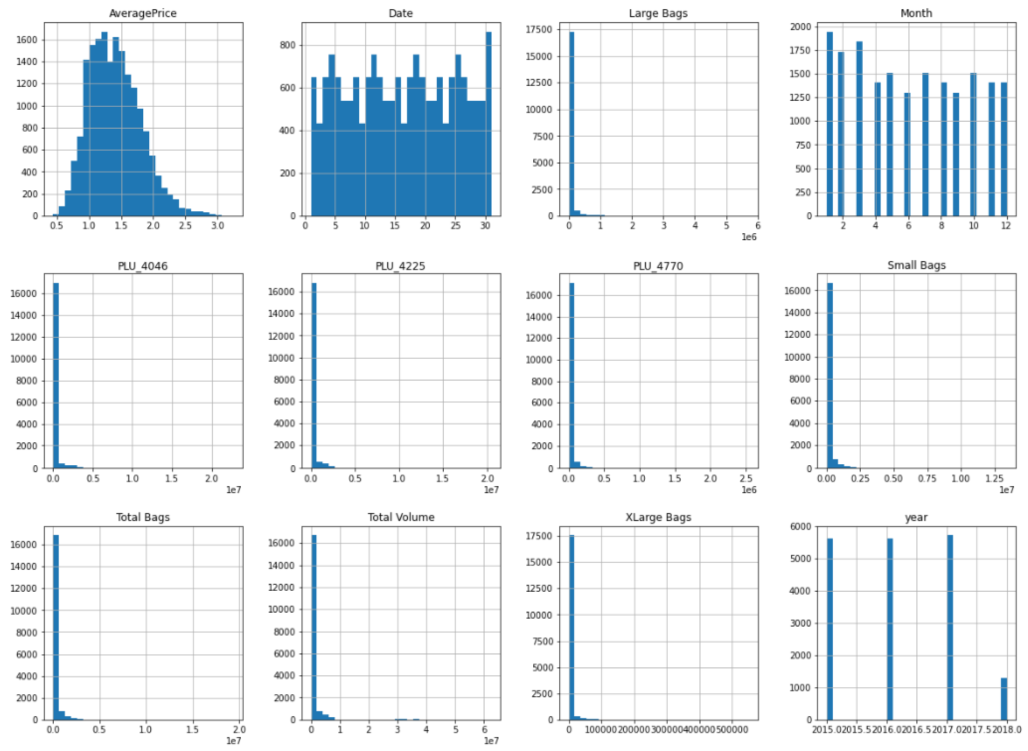
From above graph:

- There is hike between month 8–10 of both type of avocado both for conventional and organic type of avocado.
- Also, the conventional type of avocado is varying in term of price as seen in line plot because in starting the price is high but then it get decrease and so on.

Plotting Histogram:

- A **histogram** shows the frequency on the vertical axis and the horizontal axis is another dimension. Usually it has bins, where every bin has a minimum and maximum value. Each bin also has a frequency between x and infinite
- So, in this we can also check whether the graph is right skewed, left skew or the graph is normally distributed graph.

```
In [29]: #plotting histogram for univariate analysis and checking the Normal Distribution
         df.hist(figsize=(20,20), grid = True, layout = (4,4), bins = 30)

Out[29]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDAB7248>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDB17C88>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDD811C8>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDDB7F08>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDDF0D88>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDE29C48>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDE64A48>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDE9F788>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDE9F988>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDED7908>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDF4D488>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDF84288>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBDFB8FC8>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBE381F48>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBE3BCF88>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024CBE3F5F08>]],
               dtype=object)
```



From plotting this histogram, I used the bin size as 30, we can take any bin size (suited as per as data).

- Average price column is normally distributing over the histogram.
- Rest of the data are not much varying in term of numbers, so they are almost left skewed data
- To make the column as normal distributed we can use different methods, but I am using numPy log to make the skew values as normal distributed.

```
In [67]: df.skew()

Out[67]: Date            0.012943
         AveragePrice    0.350548
         Total Volume   -0.058010
         PLU_4046       -0.426890
         PLU_4225       -0.419363
         PLU_4770        0.056523
         Total Bags     -0.233833
         Small Bags     -0.315340
         Large Bags     -0.617846
         XLarge Bags     1.205366
         type           -0.023627
         year            0.229681
         region          0.036655
         Month           0.108797
         dtype: float64

In [68]: #making the skew less than or equal to 0.55 for better prediction and plotting Normal distribution
         skew=('Total Volume','PLU_4046','PLU_4225','PLU_4770','Total Bags','Small Bags','Large Bags','XLarge Bag
         s')
         for col in skew :
             if df.skew().loc[col]>0.55:
                 df[col] = np.log1p(df[col])
```

In above image we are first calculating the skew value and some of the column skew values are far from zero.

- The best skew value for normally distributes is very close to zero, so we are using "log1p" method to make the skew value near to zero

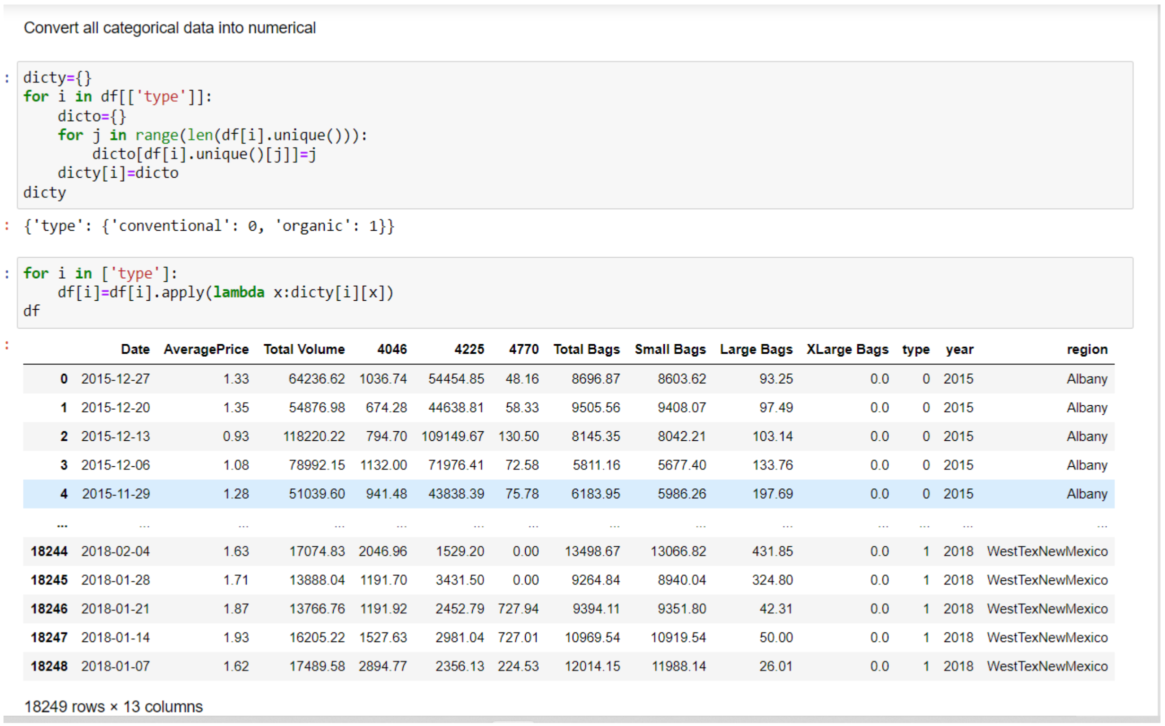
- In the last cell I am again checking the skewness value and there is difference between the first skewness value and second, now the skewness value of each column is near to zero.

Note: Making the skewness value near to zero will help to get better score.

Label Encoding:

Sklearn provides a very efficient tool for **encoding** the levels of categorical features into numeric values. **Label Encoder encode labels** with a value between 0 and n_classes-1 where n is the number of distinct **labels**. If a **label** repeats it assigns the same value to as assigned earlier.

Convert Type into numeric value by using encoder.


Convert all categorical data into numerical

```
dicty={}
for i in df[['type']]:
    dicto={}
    for j in range(len(df[i].unique())):
        dicto[df[i].unique()[j]]=j
    dicty[i]=dicto
dicty
```

```
{'type': {'conventional': 0, 'organic': 1}}
```

```
for i in ['type']:
    df[i]=df[i].apply(lambda x:dicty[i][x])
df
```

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	0	2015	Albany
1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	0	2015	Albany
2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	0	2015	Albany
3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	0	2015	Albany
4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	0	2015	Albany
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	1	2018	WestTexNewMexico
18245	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8940.04	324.80	0.0	1	2018	WestTexNewMexico
18246	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9351.80	42.31	0.0	1	2018	WestTexNewMexico
18247	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10919.54	50.00	0.0	1	2018	WestTexNewMexico
18248	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11988.14	26.01	0.0	1	2018	WestTexNewMexico

18249 rows × 13 columns

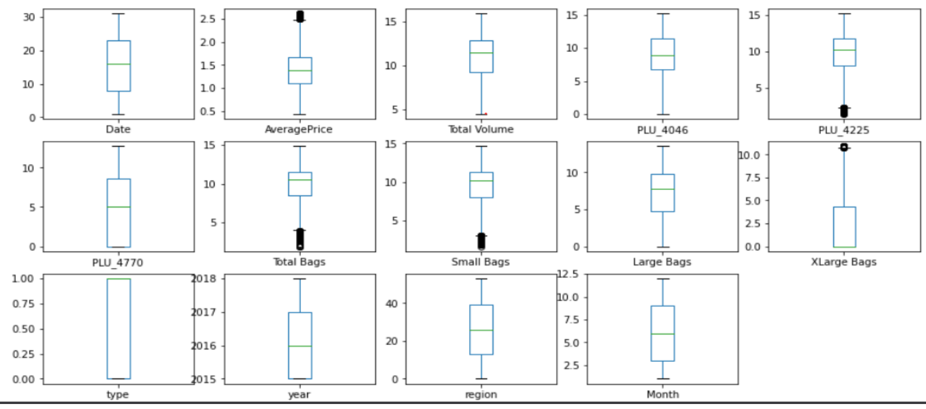
Outliers:

An **outlier** is a data point in a data set that is distant from all other observations. A data point that lies outside the overall distribution of the data set.

Now that we know outliers can either be a mistake or just variance, how would you decide if they are important or not. Well, it is simple if they are the result of a mistake, then we can ignore them, but if it is just a variance in the data, we would need think a bit further.

For avocado problem first check the outliers of each column.

```
In [57]: #plotting the boxplot of each column to check the outliers
         df.plot(kind='box',subplots = True,layout=(4,5),figsize = (15,10))

Out[57]: Date              AxesSubplot(0.125,0.71587;0.133621x0.16413)
         AveragePrice    AxesSubplot(0.285345,0.71587;0.133621x0.16413)
         Total Volume     AxesSubplot(0.44569,0.71587;0.133621x0.16413)
         PLU_4046        AxesSubplot(0.606034,0.71587;0.133621x0.16413)
         PLU_4225        AxesSubplot(0.766379,0.71587;0.133621x0.16413)
         PLU_4770          AxesSubplot(0.125,0.518913;0.133621x0.16413)
         Total Bags     AxesSubplot(0.285345,0.518913;0.133621x0.16413)
         Small Bags      AxesSubplot(0.44569,0.518913;0.133621x0.16413)
         Large Bags     AxesSubplot(0.606034,0.518913;0.133621x0.16413)
         XLarge Bags    AxesSubplot(0.766379,0.518913;0.133621x0.16413)
         type              AxesSubplot(0.125,0.321957;0.133621x0.16413)
         year           AxesSubplot(0.285345,0.321957;0.133621x0.16413)
         region          AxesSubplot(0.44569,0.321957;0.133621x0.16413)
         Month          AxesSubplot(0.606034,0.321957;0.133621x0.16413)
         dtype: object
```



From above image we can clear see that there are number of black dots in most of the column which are referring to the outliers, so it means most of the data are outside the distribution.

So now we detect the outliers now the second step is to remove the outliers, there are different way to remove the outliers that are find the IQR, zscore values.

I am using both zscore value then I again check if there are some of the outliers then I will remove it by replacing the outliers with the mean value of that column.

```
In [58]: #calculate the zscore
         z = np.abs(zscore(df))
         print(z)

         [[1.28930873 0.19535234 0.06196232 ... 1.21001338 1.68655445 1.64461178]
          [0.49165249 0.1435096  0.13430157 ... 1.21001338 1.68655445 1.64461178]
          [0.30600376 1.2322071  0.21823548 ... 1.21001338 1.68655445 1.64461178]
          ...
          [0.60560338 1.20440158 0.76950251 ... 1.98075592 1.47810526 1.46287832]
          [0.19205286 1.35992979 0.69459605 ... 1.98075592 1.47810526 1.46287832]
          [0.98970911 0.55636736 0.65956156 ... 1.98075592 1.47810526 1.46287832]]

In [60]: threshold = 3
         print(np.where(z<3))
         print(df.shape)

         (array([    0,     0,     0, ..., 17650, 17650, 17650], dtype=int64), array([ 0,  1,  2, ..., 11, 12, 13],
         dtype=int64))
         (17651, 14)

In [61]: #Assign the value to df_new which are less the threshold value and removing the outliers
         df_new = df[(z<3).all(axis = 1)]

In [62]: print(df.shape)
         print(df_new.shape)
         df = df_new
         print('Shape after removing outlires',df.shape)

         (17651, 14)
         (17525, 14)
         Shape after removing outlires (17525, 14)
```

So, I first find the zscore value and then I decide to make one threshold value as 3 which is standard of industry recommend value and then I remove all the outliers which zscore value is greater than 3.

After, removing the outlier’s final there are 17525 and 14 column presents in the data set.

Correlation Matrix:

Correlation Matrix is basically a covariance matrix. A summary measure called the **correlation** describes the strength of the linear association. **Correlation** summarizes the strength and direction of the linear (straight-line) association between two quantitative variables. Denoted by r, it takes values between -1 and +1.

Now I am finding the correlation value of each column, this value is categorized into mainly 2 parts that are:

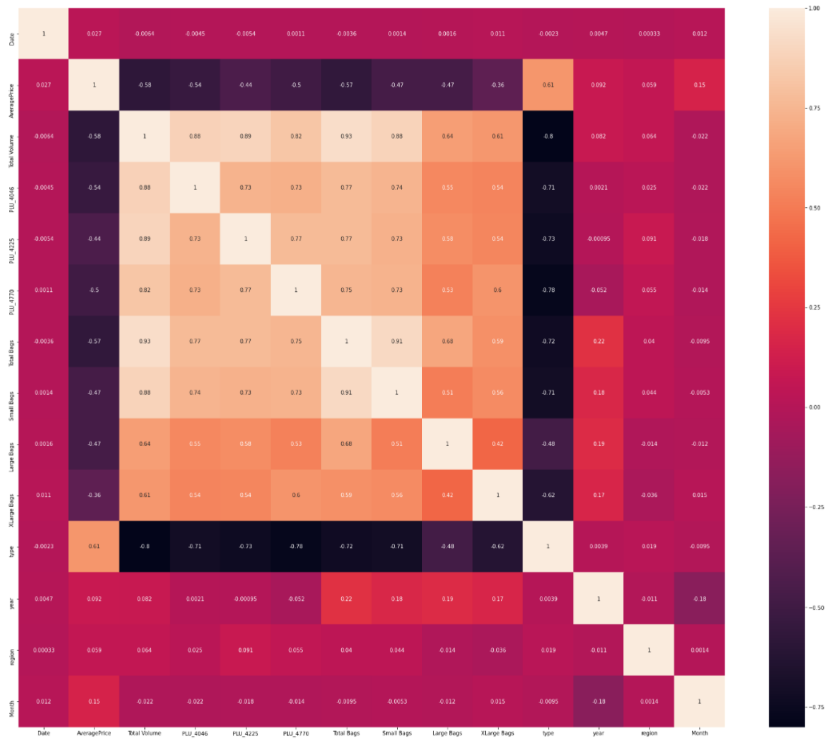
- Positive correlated value

- Negative correlated value

The most the value is positive means that column is much co related and vice versa.

I am using seaborn heatmap to plot the correlated matrix and plot the corr value in the heatmap graph

```
In [59]: #checking the co-relation of all column to each other
         df_cor = df.corr()
         plt.figure(figsize=(30,25))
         sns.heatmap(df_cor,annot=True)
         plt.plot()

Out[59]: []
```

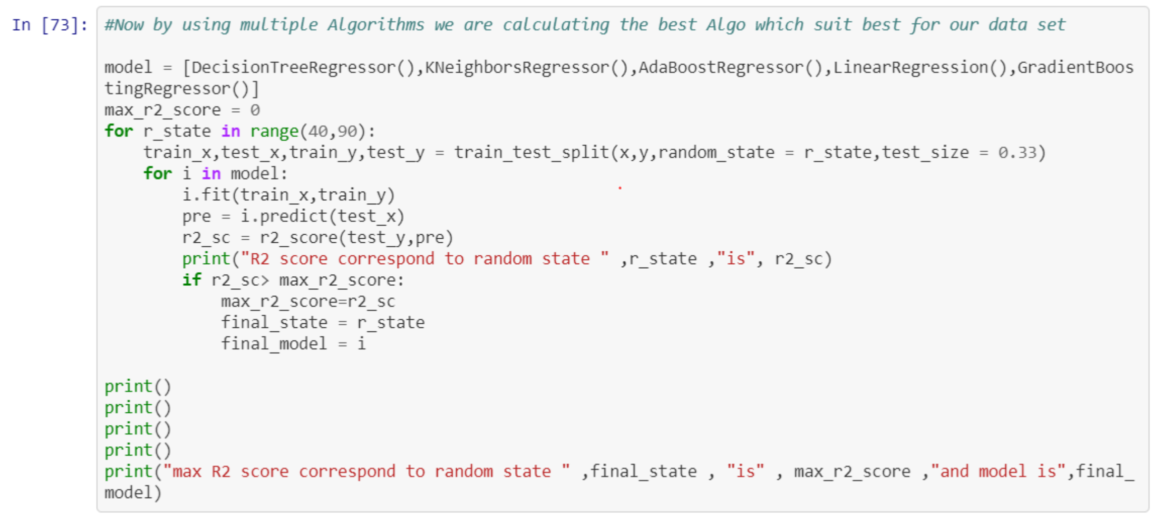


Drop and Standard Scaler:

Here I am making two variable x and y where x is having all column except Average Price and Date, we can also drop the Date column, but I kept for EDA purpose and y is having only Rings column.

Also, I am using the standard scaling method on x variable.

Prediction with Average Price:



```
In [73]: #Now by using multiple Algorithms we are calculating the best Algo which suit best for our data set

         model = [DecisionTreeRegressor(),KNeighborsRegressor(),AdaBoostRegressor(),LinearRegression(),GradientBoos
         tingRegressor()]
         max_r2_score = 0
         for r_state in range(40,90):
             train_x,test_x,train_y,test_y = train_test_split(x,y,random_state = r_state,test_size = 0.33)
             for i in model:
                 i.fit(train_x,train_y)
                 pre = i.predict(test_x)
                 r2_sc = r2_score(test_y,pre)
                 print("R2 score correspond to random state " ,r_state ,"is", r2_sc)
                 if r2_sc> max_r2_score:
                     max_r2_score=r2_sc
                     final_state = r_state
                     final_model = i

         print()
         print()
         print()
         print()
         print("max R2 score correspond to random state " ,final_state , "is" , max_r2_score ,"and model is",final_
         model)
```

Above I am using the for loop which help me to provide the R2 score at each random state and for the best state where R2 score is maximum is come as output value.

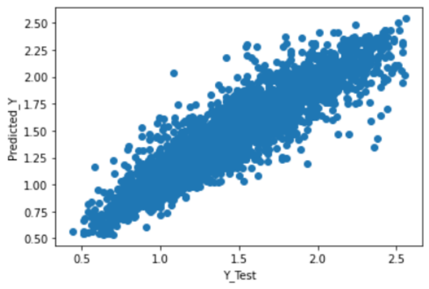
```
max R2 score correspond to random state  80 is 0.8500661139863255 and model is KNeighborsRegressor(algorit
hm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

```
In [79]: #Checking the best parameter for prediction of KNeighborsRegressor Algo using GridSearchCV
         train_x,test_x,train_y,test_y = train_test_split(x,y,random_state = 80,test_size = 0.33)
         KN = KNeighborsRegressor()
         parameters={'n_neighbors' : range(1,30)}
         gridsearch=GridSearchCV(LA,parameters)
         gridsearch.fit(train_x,train_y)
         gridsearch.best_params_

Out[79]: {'n_neighbors': 2}
```

```
In [80]: KNN = KNeighborsRegressor(n_neighbors=2)
         KNN.fit(train_x,train_y)
         pred = KNN.predict(test_x)
         r2_sc = r2_score(test_y,pred)
         print("R2 Score :",r2_sc*100)

         R2 Score : 86.81110301059822
```

```
In [81]: print('Mean Absolute Error: ', mean_absolute_error(test_y,pred))
         print('Mean squared Error: ', mean_squared_error(test_y,pred))
         print('Root Mean Absolute Error: ', np.sqrt(mean_absolute_error(test_y,pred)))

         Mean Absolute Error:  0.09554201244813278
         Mean squared Error:  0.019070146092669434
         Root Mean Absolute Error:  0.30909870987782007
```

```
In [85]: #checking the diff between actual and predicted value using graph
         plt.scatter(x=test_y,y=pred)
         plt.xlabel('Y_Test')
         plt.ylabel('Predicted_Y')

Out[85]: Text(0, 0.5, 'Predicted_Y')
```



In prediction:

- I had done this prediction by taking Average price as an output variable which is continuity in nature so that why I’m using the regression technique

- While calculating the best random state the 80 is best state which providing the highest R2 score value for this model.

- After using the GridSeachCV, I can find the best param and then I used these param for that model.

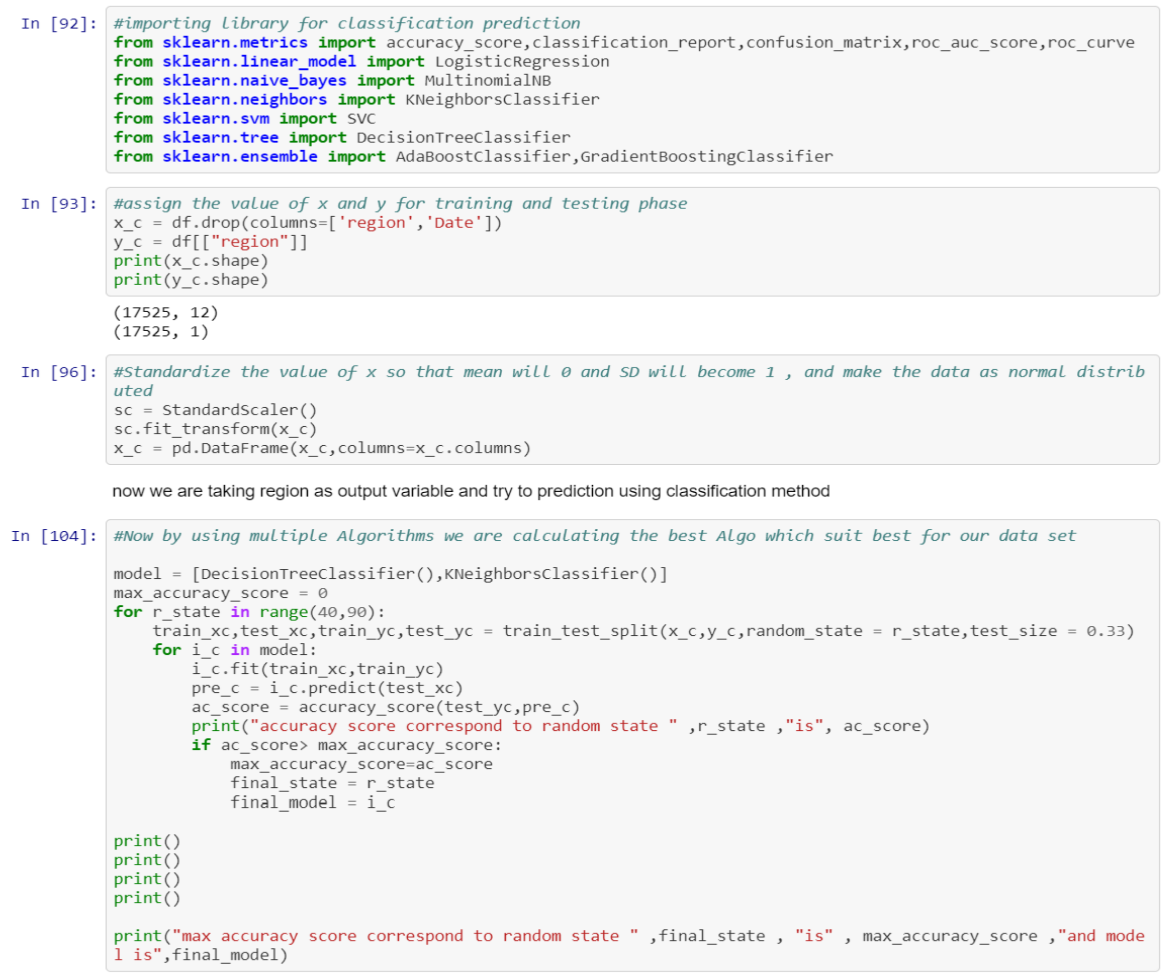
- After using the best param I can get the best R2 score and the model is KNeighboursRegressor.

- There are following matrices which I find, and which are providing the best score.

- I also plot the scatter plot graph and we can see that the actual value and predicted values are very close to each other, so the line is best fit line.

Now I am finding the score by taking region as an y value, I am using classification method because the region data is categorical in nature, so I am importing the classification model and their matrices.

<u>**Prediction with Region:**</u>



```
In [92]: #importing library for classification prediction
         from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_auc_score,roc_curve
         from sklearn.linear_model import LogisticRegression
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.svm import SVC
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
```

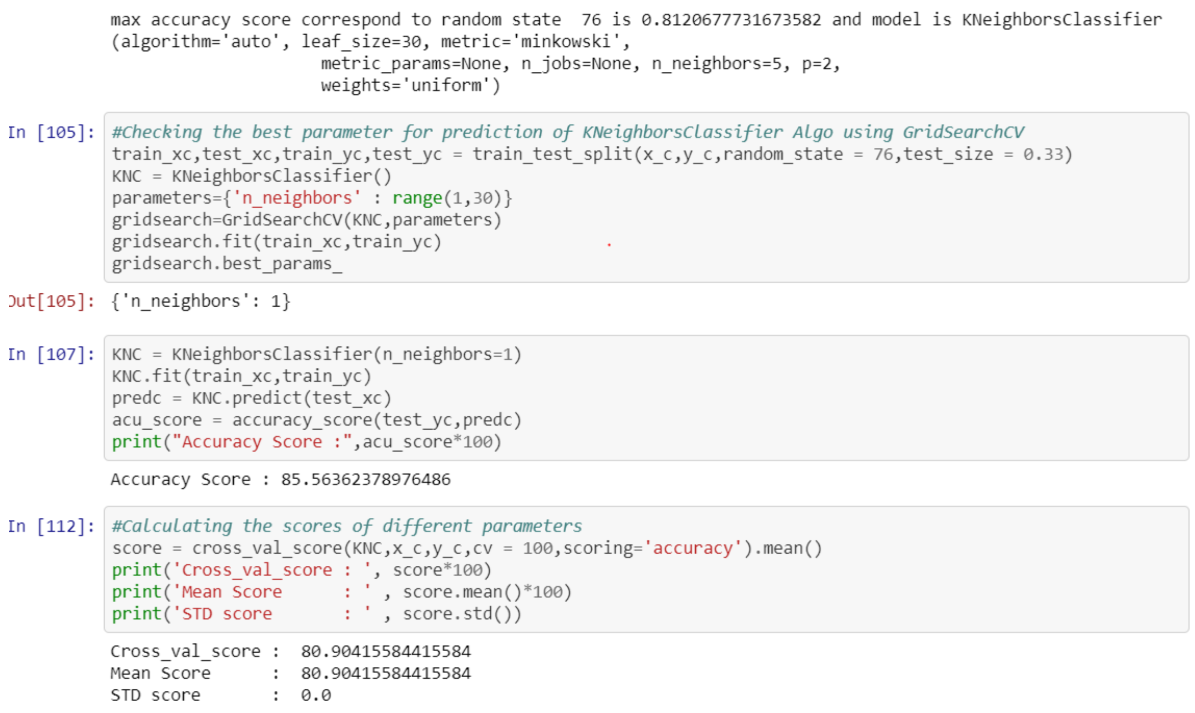
```
In [93]: #assign the value of x and y for training and testing phase
         x_c = df.drop(columns=['region','Date'])
         y_c = df[["region"]]
         print(x_c.shape)
         print(y_c.shape)

         (17525, 12)
         (17525, 1)
```

```
In [96]: #Standardize the value of x so that mean will 0 and SD will become 1 , and make the data as normal distrib
         uted
         sc = StandardScaler()
         sc.fit_transform(x_c)
         x_c = pd.DataFrame(x_c,columns=x_c.columns)
```

now we are taking region as output variable and try to prediction using classification method

```
In [104]: #Now by using multiple Algorithms we are calculating the best Algo which suit best for our data set

          model = [DecisionTreeClassifier(),KNeighborsClassifier()]
          max_accuracy_score = 0
          for r_state in range(40,90):
              train_xc,test_xc,train_yc,test_yc = train_test_split(x_c,y_c,random_state = r_state,test_size = 0.33)
              for i_c in model:
                  i_c.fit(train_xc,train_yc)
                  pre_c = i_c.predict(test_xc)
                  ac_score = accuracy_score(test_yc,pre_c)
                  print("accuracy score correspond to random state " ,r_state ,"is", ac_score)
                  if ac_score> max_accuracy_score:
                      max_accuracy_score=ac_score
                      final_state = r_state
                      final_model = i_c

          print()
          print()
          print()
          print()

          print("max accuracy score correspond to random state " ,final_state , "is" , max_accuracy_score ,"and mode
          l is",final_model)
```

Here I again done the standard scaling and use the same for loop model which provide the best state giving the best accuracy score.



```
          max accuracy score correspond to random state  76 is 0.8120677731673582 and model is KNeighborsClassifier
          (algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                               weights='uniform')
```

```
In [105]: #Checking the best parameter for prediction of KNeighborsClassifier Algo using GridSearchCV
          train_xc,test_xc,train_yc,test_yc = train_test_split(x_c,y_c,random_state = 76,test_size = 0.33)
          KNC = KNeighborsClassifier()
          parameters={'n_neighbors' : range(1,30)}
          gridsearch=GridSearchCV(KNC,parameters)
          gridsearch.fit(train_xc,train_yc)
          gridsearch.best_params_

Out[105]: {'n_neighbors': 1}
```

```
In [107]: KNC = KNeighborsClassifier(n_neighbors=1)
          KNC.fit(train_xc,train_yc)
          predc = KNC.predict(test_xc)
          acu_score = accuracy_score(test_yc,predc)
          print("Accuracy Score :",acu_score*100)

          Accuracy Score : 85.56362378976486
```

```
In [112]: #Calculating the scores of different parameters
          score = cross_val_score(KNC,x_c,y_c,cv = 100,scoring='accuracy').mean()
          print('Cross_val_score : ', score*100)
          print('Mean Score      : ' , score.mean()*100)
          print('STD score       : ' , score.std())

          Cross_val_score :  80.90415584415584
          Mean Score      :  80.90415584415584
          STD score       :  0.0
```

In prediction:

- Here the random state that is occurring is 76 which provide the best accuracy score for the model which is 81%.
- Also, by using the GridSeachCV I am able to find the best param and then find the best accuracy score that is 85%

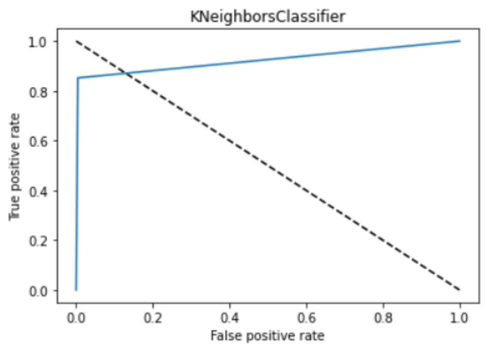
Roc Curve:

A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve, or ROC curve.

It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. Put another way, it plots the false alarm rate versus the hit rate.

The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive.

```
In [114]: pred_prob = KNC.predict_proba(test_xc)[:,1]
          pred_prob

Out[114]: array([0., 0., 0., ..., 0., 0., 0.])

In [134]: fpr,tpr,thresholds = roc_curve(test_yc,pred_prob,pos_label=True)

In [135]: print(fpr)
          print("\n")
          print(tpr)
          print("\n")
          print(thresholds)

          [0.         0.00423355 1.        ]


          [0.         0.85217391 1.        ]


          [2. 1. 0.]

In [136]: #Plotting the graph which tells us about the area under curve , more the area under curve more will be the
          better prediction

          plt.plot([0,1],[1,0],'k--')
          plt.plot(fpr,tpr,label = 'KNeighborsClassifier')
          plt.xlabel('False positive rate')
          plt.ylabel('True positive rate')
          plt.title('KNeighborsClassifier')
          plt.show()
```



Observation:

- Taking price as y variable is predicting well for this model as compare to region.
- Also, I used the Label Encoder to make the categorical data into numeric data i.e., Region and Sex.
- Also, R2 score value is also greater than accuracy score.
- Average price, total bags and total volume is well normally distributed data among all other column.
- There are no outliers in the data set after replacing it through mean value.
- As year is most negative co related column among all columns.
- In between August to October the price of avocado is much higher as compared to other months.
- Date 28,29 and 30 the price of avocado is high.
- Hartford Springfield, San Francisco and New York are having more average price as compared to another region.
- There is hike between month 8-10 of both type of avocado.
- As organic type of avocado is having the more price per unit then conventional.
- I had done prediction using region and price but using price the prediction score is high as compared to region.
- So in this data set I am using both regression and classification technique for making this model.