Avocado Project

Problem Statement:

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.

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

Based on the given data we need to predict the average price of avocado based on the different features mentioned in the dataset.

The response variable in dataset is continuous. So, this is a regression problem.

Data Analysis:

The python libraries are loaded, as these libraries help when we are building with different projects.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Reading the csv file:

The data was present in the zip file format in excel-sheet. The link is given below:

"https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects/blob/master/avocado.csv.zip"

I have converted this excel file to the csv format and read the file as DataFrame (df).

Read the csv file as DataFrame (df):

In [2]:
```
df=pd.read_csv("Avacado_project2310")
df
```

Out[2]:

	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

The dataset contains 18249 rows and 14 columns, that I have confirmed with the "df.shape" code.

To confirm whether the displayed data is right or wrong, I have cross checked with the df.tail() and df.head() code which helped me by displaying the first and last 5 data respectively.

So, by observing the data there was a need to drop one column as it was showing only serial numbers so I have dropped that column.

After dropping the column, the df is now converted to df1.

From the given code got complete information about the datatype, as it is having 9 float64, 1 int64 and 3 object datatype.

There is a data related to the Date column which was showing object type data, I have converted this data into the int32 type.

In [8]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Date          18249 non-null  object 
 1   AveragePrice  18249 non-null  float64
 2   Total Volume  18249 non-null  float64
 3   4046          18249 non-null  float64
 4   4225          18249 non-null  float64
 5   4770          18249 non-null  float64
 6   Total Bags    18249 non-null  float64
 7   Small Bags    18249 non-null  float64
 8   Large Bags    18249 non-null  float64
 9   XLarge Bags   18249 non-null  float64
 10  type          18249 non-null  object 
 11  year          18249 non-null  int64  
 12  region        18249 non-null  object 
dtypes: float64(9), int64(1), object(3)
memory usage: 1.8+ MB
```

In [9]:
```
df1[['Date','Month','Year']] = df1['Date'].str.split("-",expand = True)
# converting objects into integer datatype

df1[['Date','Month','Year']] = df1[['Date','Month','Year']].astype(int)
```

In [10]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Date          18249 non-null  int32  
 1   AveragePrice  18249 non-null  float64
 2   Total Volume  18249 non-null  float64
 3   4046          18249 non-null  float64
 4   4225          18249 non-null  float64
 5   4770          18249 non-null  float64
 6   Total Bags    18249 non-null  float64
 7   Small Bags    18249 non-null  float64
 8   Large Bags    18249 non-null  float64
 9   XLarge Bags   18249 non-null  float64
 10  type          18249 non-null  object 
 11  year          18249 non-null  int64  
 12  region        18249 non-null  object 
 13  Month         18249 non-null  int32  
 14  Year          18249 non-null  int32  
dtypes: float64(9), int32(3), int64(1), object(2)
memory usage: 1.9+ MB
```

Visualization:

The univariate visualization is done by the count plots which gives the perfect count of the variable and sub-variables.

So, using this I got to know that the ‘Type’ columns contain two sub-variables i.e.

i) conventional and ii) organic

Though the dataset contains data from 2013, here we got the highest count for the year 2017,2016,2015 and 2018 respectively.

There are 54 regions mentioned in the dataset, for all of them the count is almost same there is not much difference.

Month column is also present which help up to find out, in which month the count was high.

Multi-variate Analysis:

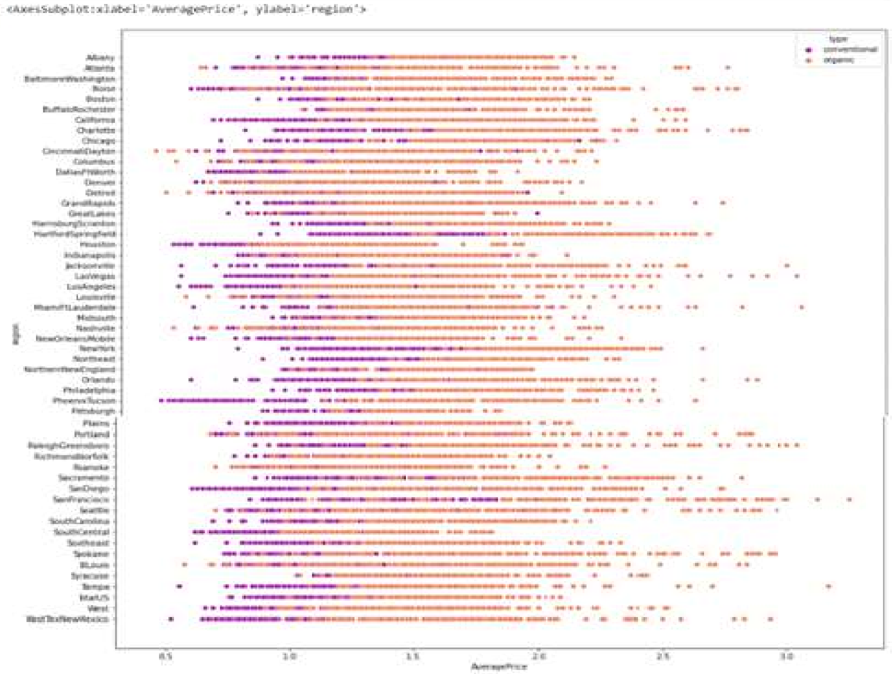
As Average price is our target variable,

1. In multivariate analysis, I have plotted the scatter plot which shows the average price of the conventional and organic type of avocado in different regions.

Scatter Plot:

```
In [24]:
plt.figure(figsize=(16,16))
sns.scatterplot(x='AveragePrice', y='region', hue='type', data=df, palette="plasma")
```

```
Out[24]: <AxesSubplot:xlabel='AveragePrice', ylabel='region'>
```

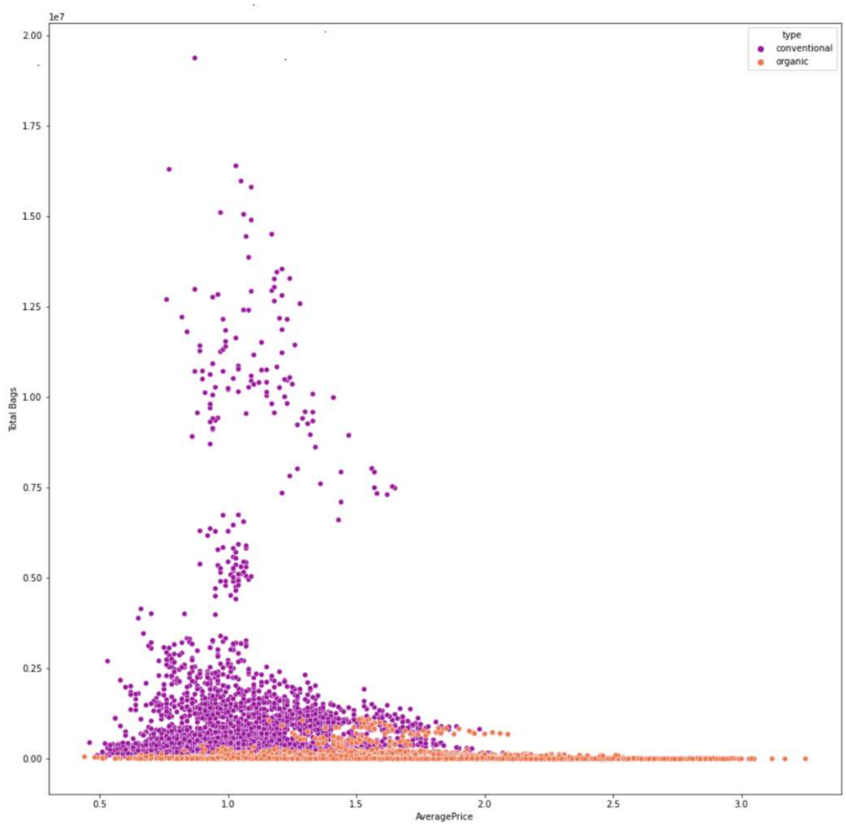


From the above multivariate scatter plot, we can say that the average price for organic avocado is high than the conventional avocado in almost all the region.

2. I have plotted the scatter plot which shows the average price of the conventional and organic avocado based on the bag size.

```
In [25]:
plt.figure(figsize=(16,16))
sns.scatterplot(x='AveragePrice', y='Total Bags', hue='type', data=df, palette="plasma")
```

Out[25]: <AxesSubplot:xlabel='AveragePrice', ylabel='Total Bags'>

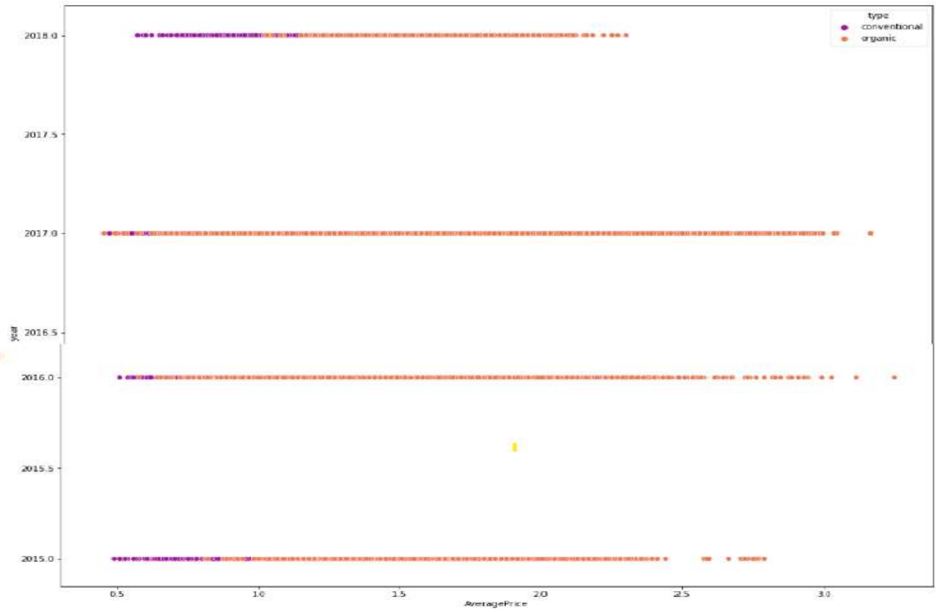


From the above plot we can say that average price for total bags of organic avocado is higher than the conventional avocado bags.

3. The below scatter plot shows the year wise average price of the avocado, From this plot we can say that the average price for the organic avocado is higher than the conventional avocado in almost every year.

```
In [25]: 1  plt.figure(figsize=(16,16))
         2  sns.scatterplot(x='AveragePrice', y='year', hue='type', data=df, palette="plasma")
         3
```

Out[25]: <AxesSubplot:xlabel='AveragePrice', ylabel='year'>



Checking for the correlation:

```
In [28]: 1  df1.corr()
```

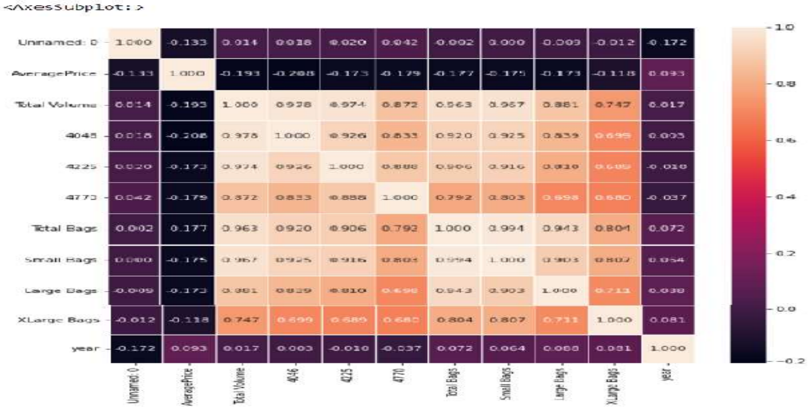
Out[28]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags	year	Month	Year
Date	1.000000	0.093197	0.017193	0.003353	-0.009559	-0.036531	0.071552	0.063915	0.087891	0.081033	1.000000	-0.177050	0.004475
AveragePrice	0.093197	1.000000	-0.192752	-0.208317	-0.172928	-0.179446	-0.177088	-0.174730	-0.172940	-0.117592	0.093197	0.162409	0.027386
Total Volume	0.017193	-0.192752	1.000000	0.977863	0.974181	0.872202	0.963047	0.967238	0.880640	0.747157	0.017193	-0.024689	-0.009747
4046	0.003353	-0.208317	0.977863	1.000000	0.926110	0.833389	0.920057	0.925280	0.838645	0.699377	0.003353	-0.025803	-0.010159
4225	-0.009559	-0.172928	0.974181	0.926110	1.000000	0.887855	0.905787	0.916031	0.810015	0.688809	-0.009559	-0.022108	-0.012393
4770	-0.036531	-0.179446	0.872202	0.833389	0.887855	1.000000	0.792314	0.802733	0.698471	0.679861	-0.036531	-0.033424	-0.009009
Total Bags	0.071552	-0.177088	0.963047	0.920057	0.905787	0.792314	1.000000	0.994335	0.943009	0.804233	0.071552	-0.022724	-0.004988
Small Bags	0.063915	-0.174730	0.967238	0.925280	0.916031	0.802733	0.994335	1.000000	0.902589	0.806845	0.063915	-0.023126	-0.003870
Large Bags	0.087891	-0.172940	0.880640	0.838645	0.810015	0.698471	0.943009	0.902589	1.000000	0.710858	0.087891	-0.020187	-0.008352
XLarge Bags	0.081033	-0.117592	0.747157	0.699377	0.688809	0.679861	0.804233	0.806845	0.710858	1.000000	0.081033	-0.012969	0.000319
year	1.000000	0.093197	0.017193	0.003353	-0.009559	-0.036531	0.071552	0.063915	0.087891	0.081033	1.000000	-0.177050	0.004475
Month	-0.177050	0.162409	-0.024689	-0.025803	-0.022108	-0.033424	-0.022724	-0.023126	-0.020187	-0.012969	-0.177050	1.000000	0.010621
Year	0.004475	0.027386	-0.009747	-0.010159	-0.012393	-0.009009	-0.004988	-0.003870	-0.008352	0.000319	0.004475	0.010621	1.000000

By plotting this in heat map will give us a clear idea,

```
In [29]: 1  plt.figure(figsize=(10,10))
         2  sns.heatmap(df.corr(),annot=True,linewidths=1,linecolor="white",fmt=".3f")
```

Out[29]: <AxesSubplot:>



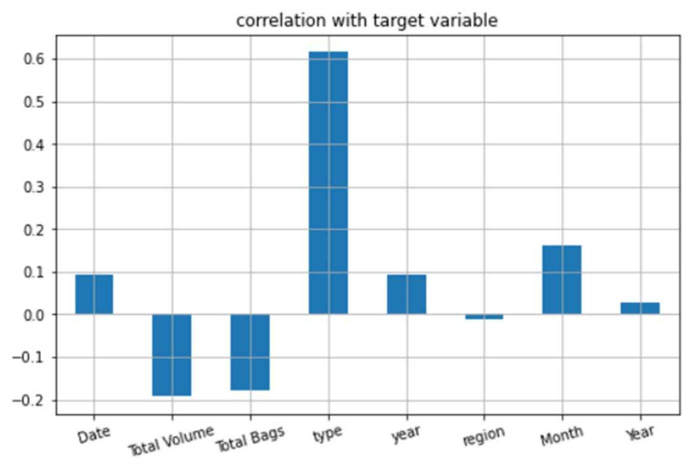
The column 4046, 4225, 4770 and small bags, medium bags and large bags columns are highly correlated with the total volume and total bags respectively so by dropping these columns can solve the multicollinearity problem. So, I have dropped these columns.
And again, shifted the data from df1 to df.

I have renamed some column like 'Type' and 'Region', just to avoid dropping these columns and prevent from the data loss, as these are the main columns for our data analysis.

Now checking the correlation of all features with the target variable, by plotting the bar graph.

```
In [36]:
plt.figure(figsize=(8,5))
df.drop('AveragePrice',axis=1).corrwith(df['AveragePrice']).plot(kind='bar',grid=True)
plt.xticks(rotation=15)
plt.title('correlation with target variable')
```

```
Out[36]: Text(0.5, 1.0, 'correlation with target variable')
```



From the above bar graph, we can say that the 'Total volume', 'total bags', and 'region' columns shows the negative correlation with target variable.
The 'Type' column is highly correlated with the target variable followed by month, date, year.

Skewness and outliers:

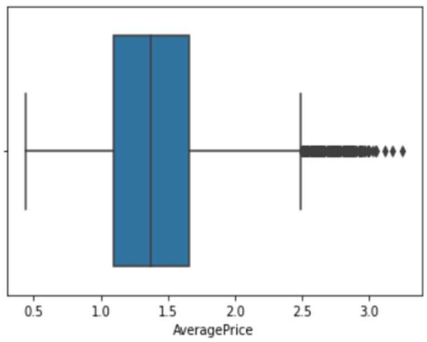
```
In [37]:
df.skew()
```

```
Out[37]: Date            0.215339
         AveragePrice    0.580303
         Total Volume    9.007687
         Total Bags      9.756072
         type            0.000329
         year            0.215339
         region          0.000030
         Month           0.106617
         Year            0.014310
         dtype: float64
```

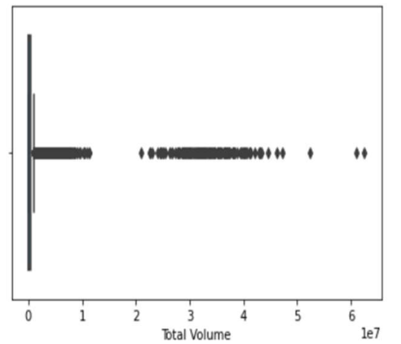
There is no normal distribution in any of the column.

Now checking for the outliers,

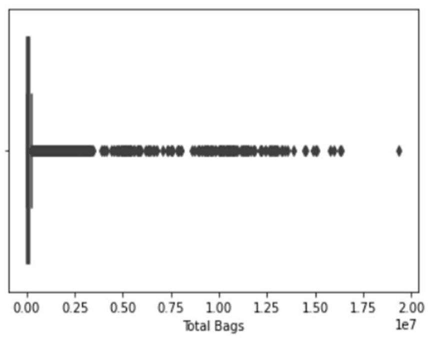
```
In [39]: sns.boxplot(df['AveragePrice'])
Out[39]: <AxesSubplot:xlabel='AveragePrice'>
```



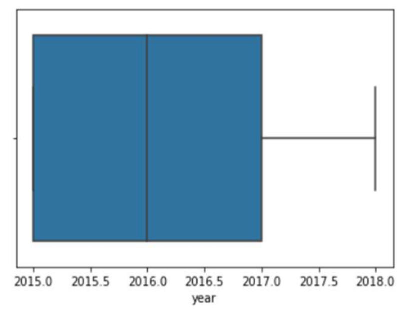

```
In [40]: sns.boxplot(df['Total Volume'])
Out[40]: <AxesSubplot:xlabel='Total Volume'>
```



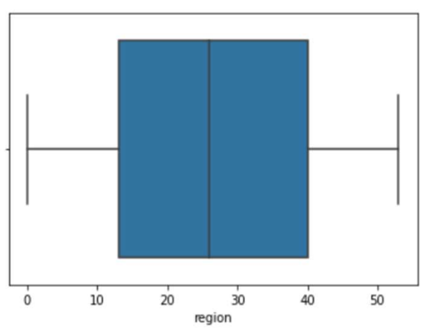

```
In [41]: sns.boxplot(df['Total Bags'])
Out[41]: <AxesSubplot:xlabel='Total Bags'>
```



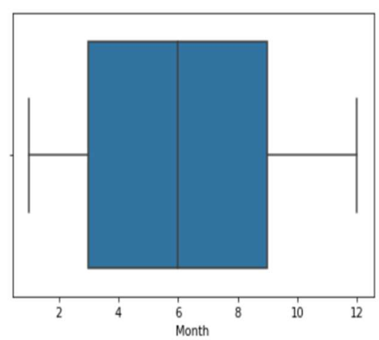

```
In [42]: sns.boxplot(df['year'])
Out[42]: <AxesSubplot:xlabel='year'>
```




```
In [43]: sns.boxplot(df['region'])
Out[43]: <AxesSubplot:xlabel='region'>
```




```
In [44]: sns.boxplot(df['Month'])
Out[44]: <AxesSubplot:xlabel='Month'>
```



From the above box plot,

Region, month, year columns do not have any outlier.

Column Total bags, Total volume and average price have the outliers so we need to remove these outliers for correct prediction.

For removing these outliers I am using the Z-score method,

Z-score Method

In [45]:
```python
from scipy.stats import zscore

z= np.abs(zscore(df))
print(np.where(z>3))
```

```
(array([ 2652,  2652,  2653,  2653,  2654,  2654,  2655,  2655,  2656,
        2656,  2657,  2657,  2658,  2658,  2659,  2659,  2660,  2660,
        2661,  2661,  2662,  2662,  2663,  2663,  2664,  2664,  2665,
        2665,  2666,  2666,  2667,  2667,  2668,  2668,  2669,  2669,
        2670,  2670,  2671,  2671,  2672,  2672,  2673,  2673,  2674,
        2674,  2675,  2675,  2676,  2676,  2677,  2677,  2678,  2678,
        2679,  2679,  2680,  2680,  2681,  2681,  2682,  2682,  2683,
        2683,  2684,  2684,  2685,  2685,  2686,  2686,  2687,  2687,
        2688,  2688,  2689,  2689,  2690,  2690,  2691,  2691,  2692,
        2692,  2693,  2693,  2694,  2694,  2695,  2695,  2696,  2696,
        2697,  2697,  2698,  2698,  2699,  2699,  2700,  2700,  2701,
        2701,  2702,  2702,  2703,  2703,  5460,  5460,  5461,  5461,
        5462,  5462,  5463,  5463,  5464,  5464,  5465,  5465,  5466,
        5466,  5467,  5467,  5468,  5468,  5469,  5469,  5470,  5470,
        5471,  5471,  5472,  5472,  5473,  5473,  5474,  5474,  5475,
        5475,  5476,  5476,  5477,  5477,  5478,  5478,  5479,  5479,
        5480,  5480,  5481,  5481,  5482,  5482,  5483,  5483,  5484,
        5484,  5485,  5485,  5486,  5486,  5487,  5487,  5488,  5488,
        5489,  5489,  5490,  5490,  5491,  5491,  5492,  5492,  5493,
        5493,  5494,  5494,  5495,  5495,  5496,  5496,  5497,  5497,
        5498,  5498,  5499,  5499,  5500,  5500,  5501,  5501,  5502,
        5502,  5503,  5503,  5504,  5504,  5505,  5505,  5506,  5506,
        5507,  5507,  5508,  5508,  5509,  5509,  5510,  5510,  5511,
        5511,  5514,  5524,  5981,  5981,  8319,  8319,  8320,  8320,
        8321,  8321,  8322,  8322,  8323,  8323,  8324,  8324,  8325,
```

In [46]:
```python
df_1 = df[(z<3).all(axis=1)]
print("with outliers::",df.shape)
print("After removing outliers::",df_1.shape)
```

```
with outliers:: (18249, 9)
After removing outliers:: (17931, 9)
```

So, after before removing the outliers, we have 18249 rows and 9 columns, but after removal of the outliers we have 17931 rows and number of columns are same.

318 rows are removed from the given dataset. To check the percentage data loss, we can apply the formula,

Percentage data loss= [(no of rows with outliers- no of rows after removing outliers) / no of rows with outliers] *100

With this we got **1.7%** which is acceptable range for the data loss.

Now moving forward-

We are splitting the data into X and Y, as ‘average price’ is our target variable so I am dropping this column from the dataset and labelling Y as ‘average price’.

Splitting data into X and Y:

In [47]:
```
x = df_1.drop("AveragePrice", axis=1)
y = df_1["AveragePrice"]
```

In [48]:
```
x.shape
```

Out[48]: (17931, 8)

In [49]:
```
y.shape
```

Out[49]: (17931,)

In [50]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression

```

I have imported some libraries for the model buildings.

splitting into x_train and y_train:

In [51]:
```
lr=LinearRegression()
for i in range (0,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=i)
    lr.fit(x_train,y_train)
    lr_predict_train=lr.predict(x_train)
    lr_predict_test=lr.predict(x_test)
    print(f'At random state {i}, The training accuracy is :-{r2_score(y_train,lr_predict_train)}')
    print(f'At random state {i}, The test accuracy is :-{r2_score(y_test,lr_predict_test)}')
    print('\n')
```

```
At random state 44, The training accuracy is :-0.4304049991227228
At random state 44, The test accuracy is :-0.4281167890164572


At random state 45, The training accuracy is :-0.43000855209755806
At random state 45, The test accuracy is :-0.43014825306244076


At random state 46, The training accuracy is :-0.42496150642594177
At random state 46, The test accuracy is :-0.4503875091694196


At random state 47, The training accuracy is :-0.43456088832347606
At random state 47, The test accuracy is :-0.4110240276538537
```

Here we can choose any random state, I have chosen random state 45 which is showing almost same training and testing accuracy which is -0.43. On this basis I got the following output.

In [52]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=45)
```

In [53]:
```
x_train.shape
```

Out[53]: (14344, 8)

In [54]:
```
y_train.shape
```

Out[54]: (14344,)

In [55]:
```
x_test.shape
```

Out[55]: (3587, 8)

In [56]:
```
y_test.shape
```

Out[56]: (3587,)

Model Building and Prediction:

The above problem statement clearly explains that the target variable is continuous and it's a regression problem as we need to predict the average price. So, this can be solved by any of the regression model (Regression Machine learning algorithms):

i) Linear Regression Model (lr)
ii) Decision Tree Regressor (DTR)
iii) Random Forest Regressor (rdr)
iv) Support Vector Regression (svr)
v) Gradient Boosting Classifier (GBR)

There are two data sets that are given one is training and another one is testing data.

1) Train file will be used for training the model, i.e. the model will learn from this file. It contains all the independent variables and the target variable. Size of training set: 14344 records.
2) Test file contains all independent variables but not the target variable. We will predict the target variable for test data by using model. The size of test data set: 3587 records.

Here we are going to check the root mean square error RMSE.

In regression analysis we can understand the relationship between one or more predictor variable and response variable.

By calculating the RMSE we can say how well a regression model fits a dataset. RMSE tells us the distance between predicted values from the model and the actual values in the dataset.

Lower RMSE value gives us a best fit of the model. So now predicting with the help of following models.

1. **Linear Regression Model:**

In [57]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_train,y_train))
lr_predict=lr.predict(x_test)
```

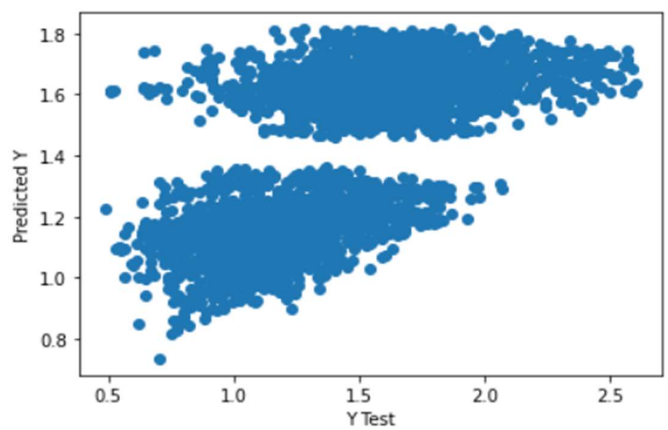
```
0.43000855209755806
```

In [58]:
```python
print('MSE:',mean_squared_error(lr_predict,y_test))
print('MAE:',mean_absolute_error(lr_predict,y_test))
print('r2_score:',r2_score(lr_predict,y_test))
print('RMSE:', np.sqrt(mean_squared_error(lr_predict,y_test)))
```

```
MSE: 0.08511411600276199
MAE: 0.22740799535703005
r2_score: -0.3115651768785075
RMSE: 0.2917432364301904
```

In [59]:
```python
plt.scatter(x=y_test,y=lr_predict)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[59]: `Text(0, 0.5, 'Predicted Y')`



- The lr_predict for the given linear regression model is 0.43
- The RMSE is 0.29, from this we can say our model is best fit. But for the confirmation I have plotted the scatter plot for Y Test, that is not showing the straight line so we **can't predict whether this model is best fit or not.**

2. Decision Tree Regressor:

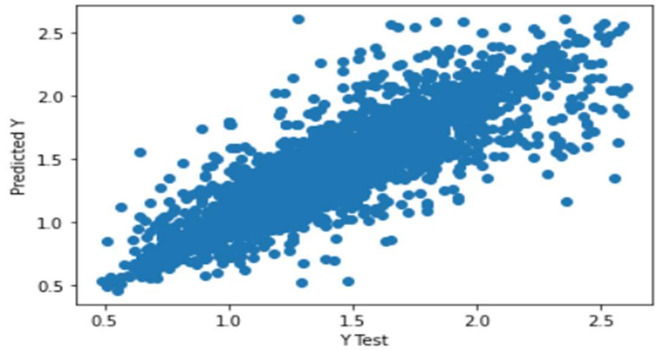
In [60]:
```python
from sklearn.tree import DecisionTreeRegressor

DTR = DecisionTreeRegressor()
DTR.fit(x_train,y_train)
print(DTR.score(x_train,y_train))
DTR_Predict=DTR.predict(x_test)
```

```
1.0
```

In [61]:
```python
print('MSE:',mean_squared_error(DTR_Predict,y_test))
print('MAE:',mean_absolute_error(DTR_Predict,y_test))
print('r2_score:',r2_score(DTR_Predict,y_test))
print('RMSE:', np.sqrt(mean_squared_error(DTR_Predict,y_test)))
```

```
MSE: 0.03758441594647337
MAE: 0.12640367995539448
r2_score: 0.7463923504981574
RMSE: 0.19386700582222177
```

In [62]:
```
plt.scatter(x=y_test,y=DTR_Predict)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```
Out[62]: Text(0, 0.5, 'Predicted Y')



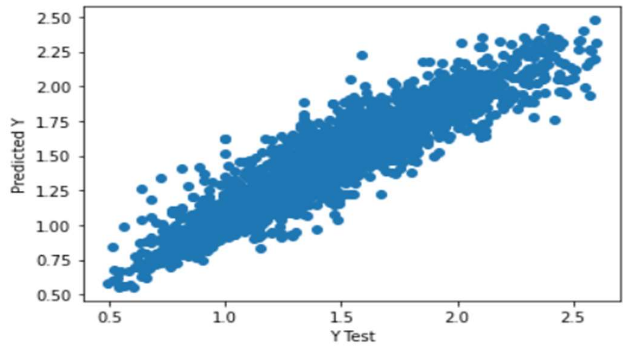
- The DTR_predict for the given decision tree regression model is 1.
- The RMSE is 0.193, from this we can say our model is best fit. But for the confirmation I have plotted the scatter plot for Y Test, which showing the straight line so we can say that this model is **best fit.**

3. Random Forest Regressor:

In [63]:
```
from sklearn.ensemble import RandomForestRegressor
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
print(rdr.score(x_train,y_train))
rdr_Predict=rdr.predict(x_test)
```
```
0.9833205817589906
```

In [64]:
```
print('MSE:',mean_squared_error(rdr_Predict,y_test))
print('MAE:',mean_absolute_error(rdr_Predict,y_test))
print('r2_score:',r2_score(rdr_Predict,y_test))
print('RMSE:', np.sqrt(mean_squared_error(rdr_Predict,y_test)))
```
```
MSE: 0.01761658085865626
MAE: 0.09489461945915806
r2_score: 0.8544503461589636
RMSE: 0.13272746836527946
```

```
In [65]: plt.scatter(x=y_test,y=rdr_Predict)
         plt.xlabel('Y Test')
         plt.ylabel('Predicted Y')

Out[65]: Text(0, 0.5, 'Predicted Y')
```



- The rdr_predict for the random forest regressor model is 0.98.
- The RMSE is 0.13 which actually very low, so by plotting the scatter plot for Y Test tells us whether the model is best fit or not. So, the scatter plot is in the straight line so this model is also **the best fit model**.

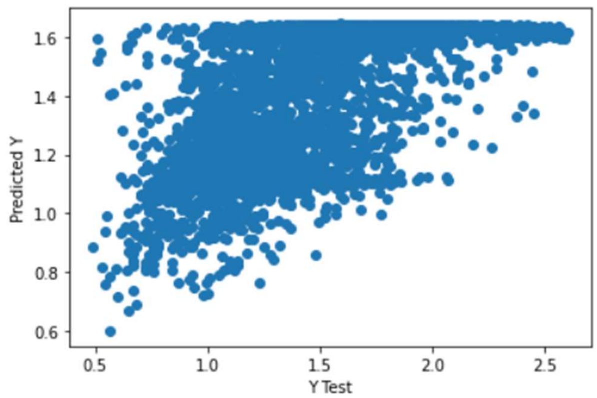
4. Support Vector Regression:

```
In [66]: from sklearn.svm import SVR

         svr = SVR()
         svr.fit(x_train,y_train)
         print(svr.score(x_train,y_train))
         svr_Predict=svr.predict(x_test)

0.3619459608256048
```

```
In [67]: print('MSE:',mean_squared_error(svr_Predict,y_test))
         print('MAE:',mean_absolute_error(svr_Predict,y_test))
         print('r2_score:',r2_score(svr_Predict,y_test))
         print('RMSE:', np.sqrt(mean_squared_error(svr_Predict,y_test)))

MSE: 0.09425820051064812
MAE: 0.24052608829986646
r2_score: -0.6934553571329747
RMSE: 0.3070149841793526
```

```
In [68]:
plt.scatter(x=y_test,y=svr_Predict)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Out[68]: Text(0, 0.5, 'Predicted Y')
```



- The svr_predict is 0.36.
- The RMSE for this model is 0.3 which is low but as compare to the other regression model this RMSE is high. By plotting Y test, we can say that the plot is not linear so this model is **not the best fit model**.

5. Gradient Boosting Classifier:

```
In [69]:
from sklearn.ensemble import GradientBoostingRegressor

GBR = GradientBoostingRegressor()
GBR.fit(x_train,y_train)
print(GBR.score(x_train,y_train))
GBR_Predict=GBR.predict(x_test)
```

```
0.7203363391203127
```

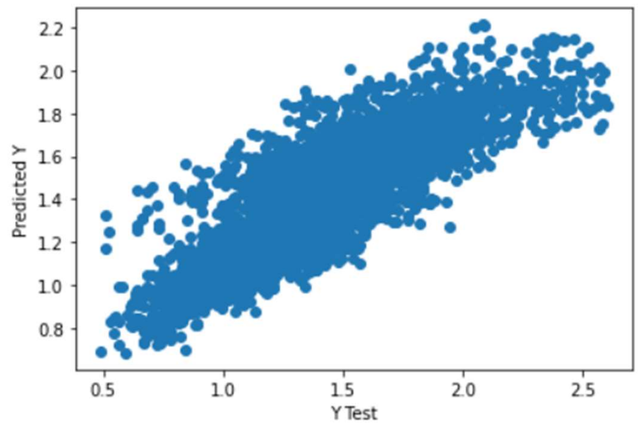
```
In [70]:
print('MSE:',mean_squared_error(GBR_Predict,y_test))
print('MAE:',mean_absolute_error(GBR_Predict,y_test))
print('r2_score:',r2_score(GBR_Predict,y_test))
print('RMSE:', np.sqrt(mean_squared_error(GBR_Predict,y_test)))
```

```
MSE: 0.04240093581734881
MAE: 0.15646220184706838
r2_score: 0.5256190196509728
RMSE: 0.20591487517260332
```

In [71]:

```
plt.scatter(x=y_test,y=GBR_Predict)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[71]: `Text(0, 0.5, 'Predicted Y')`



- The GBR_predict is 0.72.
- The RMSE is 0.2 which is good. So, by plotting the Y test, we got the straight line so we can say that this is the **best fit model**.

We have predicted the average price by using the above regression model, almost all models are the best fit. But we are going to choose the one with low RMSE and perfect linear plot.

So, the Random Forest Regressor model is the one with very low RSME (0.13) and with linear scatter plot. So we are choosing this plot and cross validating it.

Cross Validation:

In [72]:

```
from sklearn.model_selection import cross_val_score
```

In [73]:

```
from sklearn.model_selection import GridSearchCV
```

In [74]:

```
rdr = RandomForestRegressor()

param ={
       'n_estimators':[100,200],
      'criterion':['mse','mae'],
     'min_samples_split':[2],
    'min_samples_leaf':[1],
}
```

In [75]:

```
rdr_grid=GridSearchCV(RandomForestRegressor(),param,cv=4,scoring='accuracy',n_jobs=-1,verbose=2)
```

In [78]:

```
rdr_grid.fit(x_train,y_train)
rdr_grid_PRED=rdr_grid.best_estimator_.predict(x_test)
```

```
Fitting 4 folds for each of 4 candidates, totalling 16 fits
```

```
In [90]: rdr_grid.best_params_
Out[90]: {'criterion': 'mse',
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 100}
```

```
In [92]: print('MSE:',mean_squared_error(rdr_grid_PRED,y_test))
print('MAE:',mean_absolute_error(rdr_grid_PRED,y_test))
print('r2_score:',r2_score(rdr_grid_PRED,y_test))

MSE: 0.017933938341232226
MAE: 0.09568962921661556
r2_score: 0.8512726922323649
```

```
In [96]: RF = RandomForestRegressor()

param ={
        'n_estimators':[100],
       'criterion':['mse'],
      'min_samples_split':[2],
     'min_samples_leaf':[1],
}
```

```
In [97]: RF_grid=GridSearchCV(RandomForestRegressor(),param,cv=4,scoring='accuracy',n_jobs=-1,verbose=2)
```

```
In [98]: RF_grid.fit(x_train,y_train)
RF_grid_PRED=RF_grid.best_estimator_.predict(x_test)

Fitting 4 folds for each of 1 candidates, totalling 4 fits
```

As we have seen, we have less difference of train & test score, and the predicted value & test value is normally distributed, Also, in the scatter plot the test value & the prediction value is linearly distributed. The test & prediction values are almost close to each other.

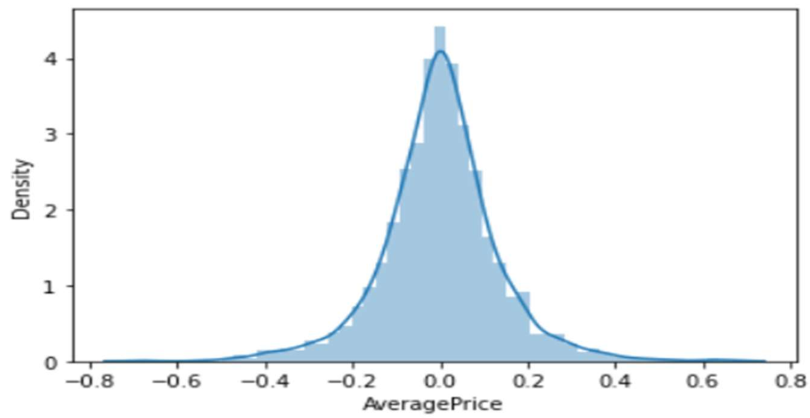
Since Random Forest Regressor is the best model in terms of model score, cross-validation difference, test & train r2 score difference, also as per the evaluation metrics, we choose Random Forest Regressor to be the final model. Let's see if we can increase the score by using hyper-parameter tuning.

```
In [99]: print('MSE:',mean_squared_error(RF_grid_PRED,y_test))
print('MAE:',mean_absolute_error(RF_grid_PRED,y_test))
print('r2_score:',r2_score(RF_grid_PRED,y_test))

MSE: 0.017668631212712574
MAE: 0.09509919152495122
r2_score: 0.8539991353383092
```

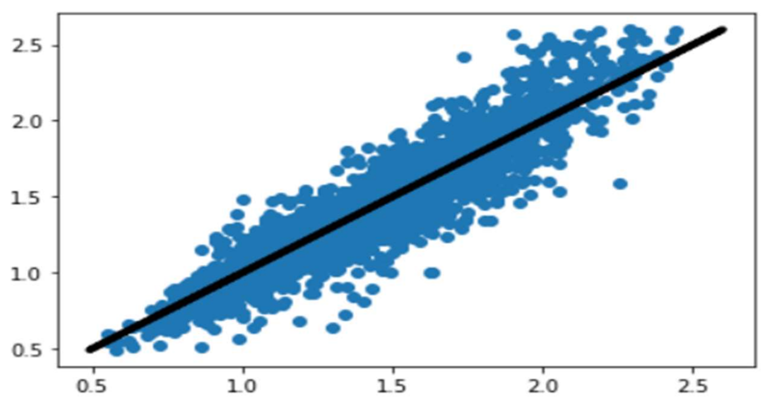
```
In [100]: RF_grid_PRED
Out[100]: array([0.7821, 1.3247, 0.9754, ..., 1.637 , 1.6471, 1.0618])
```

As we can see the r2_score is 85%.

In [101]:
```
sns.distplot(RF_grid_PRED-y_test)
```
Out[101]: <AxesSubplot:xlabel='AveragePrice', ylabel='Density'>



In [102]:
```
plt.scatter(RF_grid_PRED,y_test)
plt.plot(y_test,y_test,linewidth=4,color='black')
```
Out[102]: [<matplotlib.lines.Line2D at 0x22ecb04b520>]



The random forest prediction for Y test is showing the normal distribution.

So, saving this as the best fit model.

In [103]:
```
import joblib
```

In [104]:
```
joblib.dump(RF_grid.best_estimator_,'Avocado_Prediction_Project.obj')
```
Out[104]: ['Avocado_Prediction_Project.obj']

So, the model has been saved.

Concluding remarks:

1. Saving the model: The model is ready & we have saved the model in 'obj' format by using "joblib".
2. Test data: It is having 3587 rows & 8 columns.
3. We need to perform all the same steps as we performed for the training dataset, which includes Data analysis, EDA, & Pre-processing. The test dataset is having all the same columns except the target variable.
4. We don't need to build any model using the test dataset. Hence no need to perform a train test split, only cleaning the data is required.

Actual Prediction:

As the predicted model and values are matching with the actual model and values. So, from that we can say that from conventional and organic avocado, the customer prefers the organic avocado. This prediction will help to the retailers to decide **type of avocado they can sell**.

The average price of the avocado is also based on the type of the avocado so the predicted price and actual price is matching so we can say that this model helps retailers to **decide the price of avocado.**

Almost of all the region having same price and demand for the organic type.

It also helps to the customer to predict the future average price of the avocado.

Conclusion:

- The best model for the prediction is the random forest regression model.
- It is having the r2_score 85%.
- The plot for Y test prediction of Random Forest shows the normal distribution.
- This will help to retailers and customer to predict the average price of the avocado.

THE END