The question

1. Which region are the lowest and highest prices of Avocado?
2. What is the highest region of avocado production?
3. What is the average avocado prices in each year?
4. What is the average avocado volume in each year?

Business undestanding first This dataset contains 13 columns: 1. Date - The date of the observation 2. AveragePrice: the average price of a single avocado 3. Total Volume: Total number of avocados sold 4. Total Bags: Total number o bags 5. Small Bags: Total number of Small bags 6. Large Bags: Total number of Large bags 7. XLarge Bags: Total number of XLarge bags 8. type: conventional or organic 9. year: the year 10. region: the city or region of the observation 11. 4046: Total number of avocados with level PLU 4046 sold 12. 4225: Total number of avocados with level PLU 4225 sold 13. 4770: Total number of avocados with level PLU 4770 sold

In [1]:

```
#First import all libraries require for analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

In [90]:

```
#import dataset
df=pd.read_csv("/Users/myyntiimac/Desktop/avocado.csv")
df.head()
```

Out[90]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Lar Ba
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.

In [6]:

```
1 len(df)
```

Out[6]:

```
18249
```

In [7]:

```
1 df.columns
```

Out[7]:

```
Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total Volume', '4046',
'4225',
       '4770', 'Total Bags', 'Small Bags', 'Large Bags', 'XLarge Bag
s', 'type',
       'year', 'region'],
      dtype='object')
```

In [8]:

```
1 df.shape
```

Out[8]:

```
(18249, 14)
```

In [11]:

```
1 df.columns=df.columns=['Unnamed', 'Date', 'AveragePrice', 'TotalVolume', '4046',
2        '4770', 'TotalBags', 'SmallBags', 'LargeBags', 'XLargeBags', 'type',
3        'year', 'region']
4 df.head()
```

Out[11]:

	Unnamed	Date	AveragePrice	TotalVolume	4046	4225	4770	TotalBags	SmallBag
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.6
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.0
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.2
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.4
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.2

In [12]:

```
df.isnull().any()
```

Out[12]:

```
Unnamed          False
Date             False
AveragePrice     False
TotalVolume      False
4046             False
4225             False
4770             False
TotalBags        False
SmallBags        False
LargeBags        False
XLargeBags       False
type             False
year             False
region           False
dtype: bool
```

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
Unnamed          0
Date             0
AveragePrice     0
TotalVolume      0
4046             0
4225             0
4770             0
TotalBags        0
SmallBags        0
LargeBags        0
XLargeBags       0
type             0
year             0
region           0
dtype: int64
```

Insight:There is no null values in our df, so we dont need to impute

In [15]:

```
#Descriptive ststistics
df.describe()
```

Out[15]:

	Unnamed	AveragePrice	TotalVolume	4046	4225	4770	
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.8
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.3
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.8
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.0
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.0
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.9
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.1
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.9

In [17]:

```
df.describe().transpose()
```

Out[17]:

	count	mean	std	min	25%	50%	75%	
Unnamed	18249.0	24.232232	1.548104e+01	0.00	10.00	24.00	38.00	
AveragePrice	18249.0	1.405978	4.026766e-01	0.44	1.10	1.37	1.66	
TotalVolume	18249.0	850644.013009	3.453545e+06	84.56	10838.58	107376.76	432962.29	62
4046	18249.0	293008.424531	1.264989e+06	0.00	854.07	8645.30	111020.20	22
4225	18249.0	295154.568356	1.204120e+06	0.00	3008.78	29061.02	150206.86	20
4770	18249.0	22839.735993	1.074641e+05	0.00	0.00	184.99	6243.42	2
TotalBags	18249.0	239639.202060	9.862424e+05	0.00	5088.64	39743.83	110783.37	19
SmallBags	18249.0	182194.686696	7.461785e+05	0.00	2849.42	26362.82	83337.67	13
LargeBags	18249.0	54338.088145	2.439660e+05	0.00	127.47	2647.71	22029.25	5
XLargeBags	18249.0	3106.426507	1.769289e+04	0.00	0.00	0.00	132.50	
year	18249.0	2016.147899	9.399385e-01	2015.00	2015.00	2016.00	2017.00	

In [23]:

```
# by business understanding we conclude that the below  attribute is not gne afi
df1=df.drop(['Unnamed','Date','4046', '4225','4770'],axis=1)
df1

```

Out[23]:

	AveragePrice	TotalVolume	TotalBags	SmallBags	LargeBags	XLargeBags	type	
0	1.33	64236.62	8696.87	8603.62	93.25	0.0	conventional	2
1	1.35	54876.98	9505.56	9408.07	97.49	0.0	conventional	2
2	0.93	118220.22	8145.35	8042.21	103.14	0.0	conventional	2
3	1.08	78992.15	5811.16	5677.40	133.76	0.0	conventional	2
4	1.28	51039.60	6183.95	5986.26	197.69	0.0	conventional	2
...	...	...	...	...	...	...	...	
18244	1.63	17074.83	13498.67	13066.82	431.85	0.0	organic	2
18245	1.71	13888.04	9264.84	8940.04	324.80	0.0	organic	2
18246	1.87	13766.76	9394.11	9351.80	42.31	0.0	organic	2
18247	1.93	16205.22	10969.54	10919.54	50.00	0.0	organic	2
18248	1.62	17489.58	12014.15	11988.14	26.01	0.0	organic	2

18249 rows × 9 columns

In [ ]:

```
#Now we answering the question
#Which region are the lowest and highest prices of Avocado?

```

In [30]:

```
sorted_data = df.groupby("region")["AveragePrice"].mean().sort_values(ascending=
sorted_data = sorted_data.reset_index().rename(columns={'region': 'region', 'Ave

sorted_data
```

Out[30]:

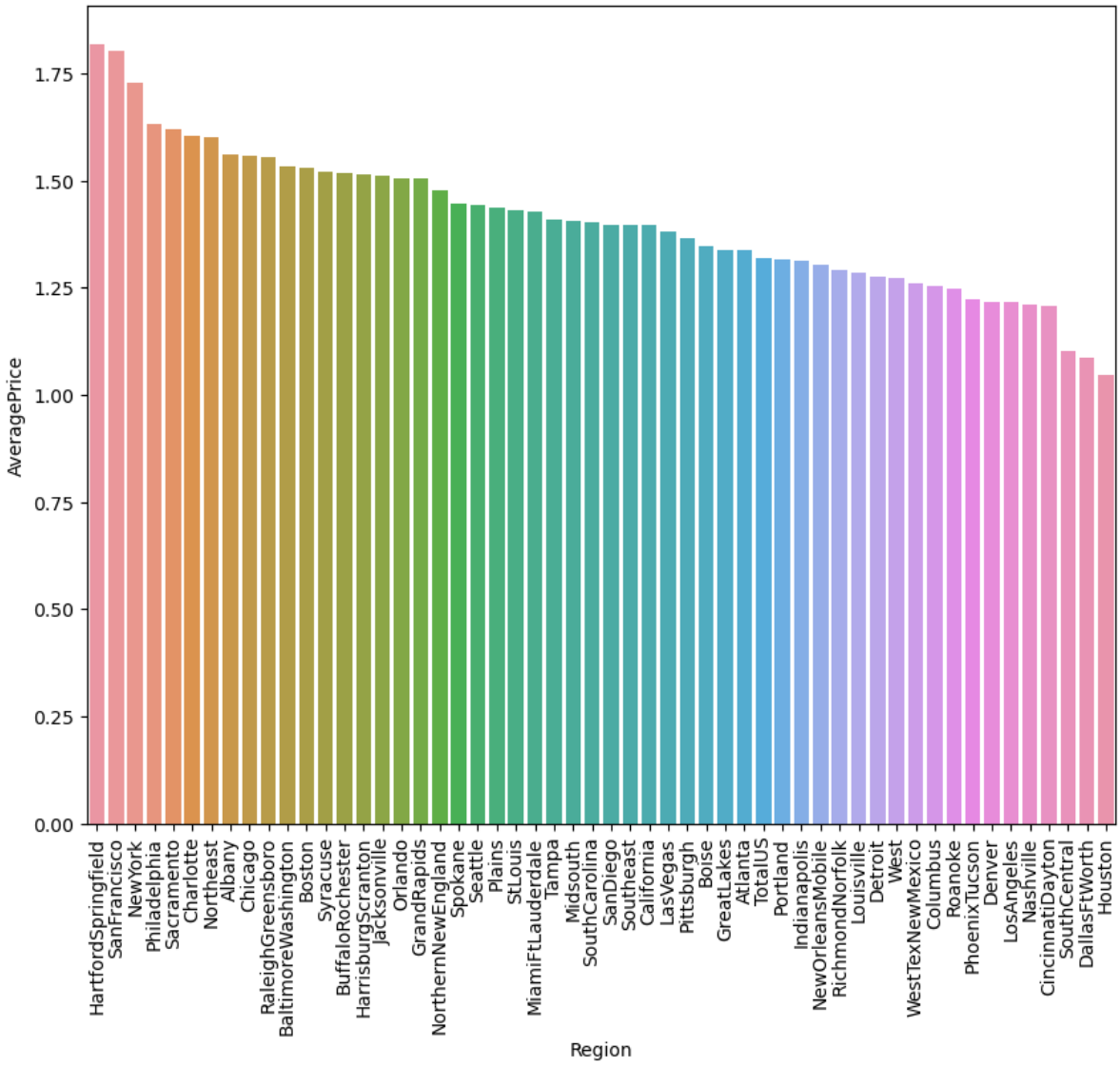
	region	AveragePrice
0	HartfordSpringfield	1.818639
1	SanFrancisco	1.804201
2	NewYork	1.727574
3	Philadelphia	1.632130
4	Sacramento	1.621568
5	Charlotte	1.606036
6	Northeast	1.601923
7	Albany	1.561036
8	Chicago	1.556775
9	RaleighGreensboro	1.555118
10	BaltimoreWashington	1.534231
11	Boston	1.530888
12	Syracuse	1.520325
13	BuffaloRochester	1.516834
14	HarrisburgScranton	1.513284
15	Jacksonville	1.510947
16	Orlando	1.506213
17	GrandRapids	1.505000
18	NorthernNewEngland	1.477396
19	Spokane	1.445592
20	Seattle	1.442574
21	Plains	1.436509
22	StLouis	1.430621
23	MiamiFtLauderdale	1.428491
24	Tampa	1.408846
25	Midsouth	1.404763
26	SouthCarolina	1.403284
27	SanDiego	1.398166
28	Southeast	1.398018
29	California	1.395325
30	LasVegas	1.380917
31	Pittsburgh	1.364320
32	Boise	1.348136
33	GreatLakes	1.338550
34	Atlanta	1.337959
35	TotalUS	1.319024

	region	AveragePrice
36	Portland	1.317722
37	Indianapolis	1.313994
38	NewOrleansMobile	1.304793
39	RichmondNorfolk	1.291331
40	Louisville	1.286686
41	Detroit	1.276095
42	West	1.272219
43	WestTexNewMexico	1.261701
44	Columbus	1.252781
45	Roanoke	1.247929
46	PhoenixTucson	1.224438
47	Denver	1.218580
48	LosAngeles	1.216006
49	Nashville	1.212101
50	CincinnatiDayton	1.209201
51	SouthCentral	1.101243
52	DallasFtWorth	1.085592
53	Houston	1.047929

In [36]:

```
#we can see the lowest and highest price region by barplot region
plt.figure(figsize=(10, 8))
sns.barplot(data=sorted_data,x="region",y="AveragePrice")
plt.xlabel("Region")
plt.xticks(rotation='vertical')

# Display the plot
plt.show()
```



Insight:we see Huosten have lowest Average price and hartfordspringfiled have highest average price

In [ ]:

```
#questin:What is the highest region of avocado production?
```

In [45]:

```
#first we create  group and filter the data on groupwise and shorted
#then we create a new data frame  by using the prvious result
# then create plot
highest_production = df.groupby("region")["TotalVolume"].mean().sort_values(asce
highest_production = highest_production.reset_index().rename(columns={'region':

highest_production
```

Out[45]:

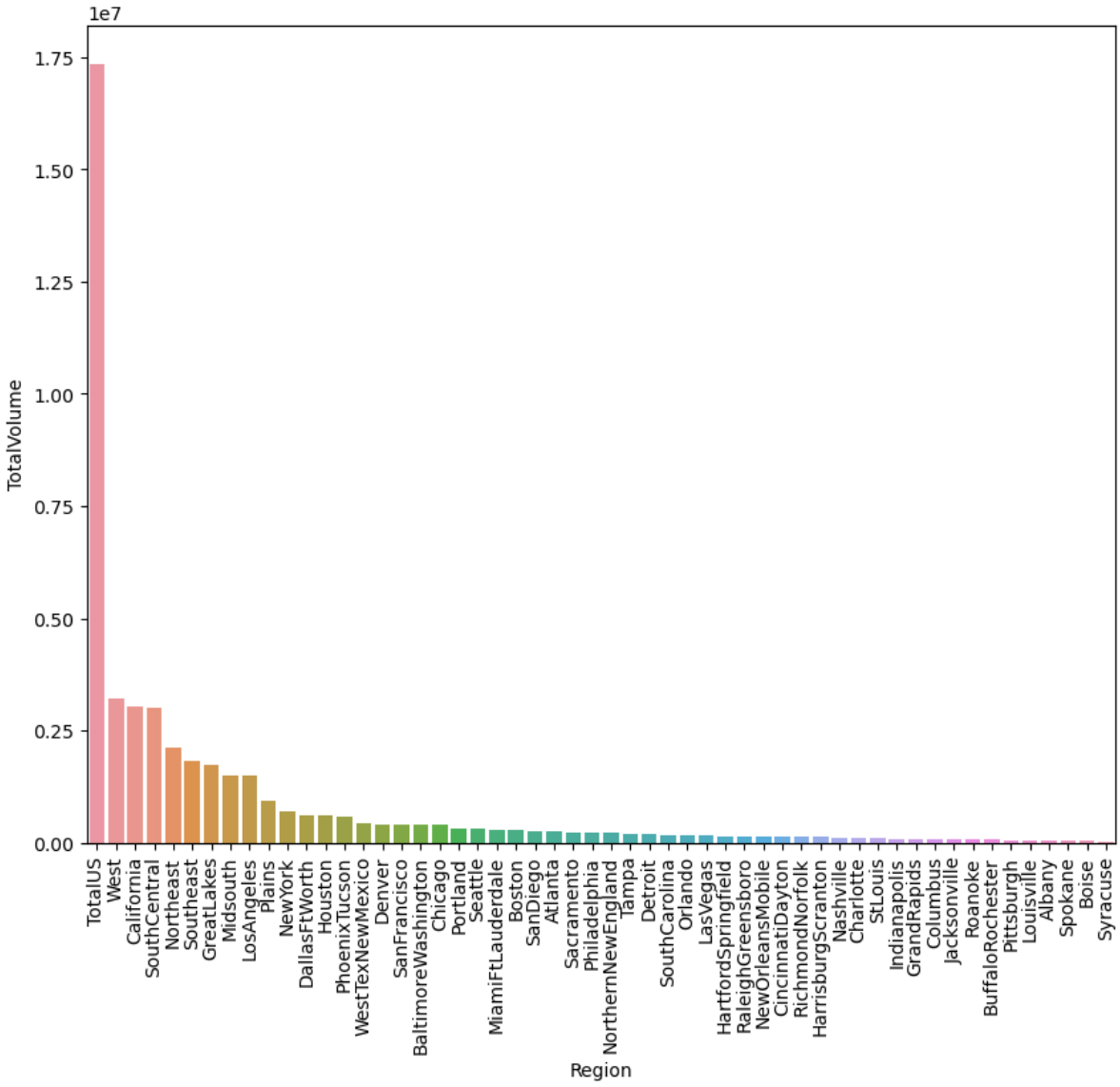
	region	TotalVolume
0	TotalUS	1.735130e+07
1	West	3.215323e+06
2	California	3.044324e+06
3	SouthCentral	2.991952e+06
4	Northeast	2.110299e+06
5	Southeast	1.820232e+06
6	GreatLakes	1.744505e+06
7	Midsouth	1.503992e+06
8	LosAngeles	1.502653e+06
9	Plains	9.206761e+05
10	NewYork	7.122311e+05
11	DallasFtWorth	6.166251e+05
12	Houston	6.010884e+05
13	PhoenixTucson	5.788264e+05
14	WestTexNewMexico	4.314085e+05
15	Denver	4.109542e+05
16	SanFrancisco	4.018645e+05
17	BaltimoreWashington	3.985619e+05
18	Chicago	3.955690e+05
19	Portland	3.270775e+05
20	Seattle	3.231189e+05
21	MiamiFtLauderdale	2.889740e+05
22	Boston	2.877929e+05
23	SanDiego	2.656566e+05
24	Atlanta	2.621453e+05
25	Sacramento	2.223779e+05
26	Philadelphia	2.125408e+05
27	NorthernNewEngland	2.116358e+05
28	Tampa	1.952797e+05
29	Detroit	1.876403e+05
30	SouthCarolina	1.797449e+05
31	Orlando	1.735524e+05
32	LasVegas	1.608784e+05
33	HartfordSpringfield	1.499128e+05
34	RaleighGreensboro	1.426116e+05
35	NewOrleansMobile	1.351927e+05

	region	TotalVolume
36	CincinnatiDayton	1.317219e+05
37	RichmondNorfolk	1.249433e+05
38	HarrisburgScranton	1.236948e+05
39	Nashville	1.053612e+05
40	Charlotte	1.051939e+05
41	StLouis	9.489004e+04
42	Indianapolis	8.953666e+04
43	GrandRapids	8.938383e+04
44	Columbus	8.873776e+04
45	Jacksonville	8.517753e+04
46	Roanoke	7.408879e+04
47	BuffaloRochester	6.793630e+04
48	Pittsburgh	5.564008e+04
49	Louisville	4.762427e+04
50	Albany	4.753787e+04
51	Spokane	4.605111e+04
52	Boise	4.264257e+04
53	Syracuse	3.237476e+04

In [47]:

```
plt.figure(figsize=(10, 8))
sns.barplot(data=highest_production,x="region",y="TotalVolume")
plt.xlabel("Region")
plt.xticks(rotation='vertical')

# Display the plot
plt.show()
```



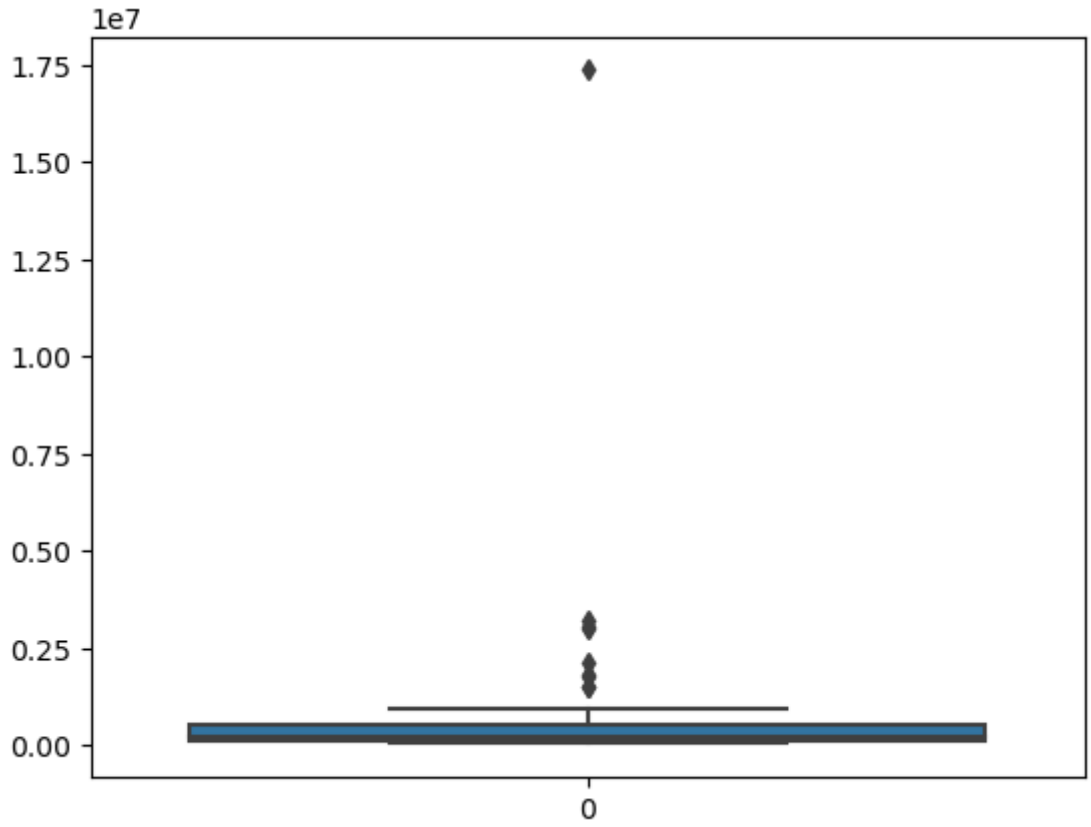
Insight:Totalus is hoigh production volume of avocado, From the figure we saw this bar is realy high than other region, it can be ouliar, Lets check in box plot to confirm

In [51]:

```
1 sns.boxplot(highest_production["TotalVolume"])
```

Out[51]:

```
<Axes: >
```



Insight:yes it is , so we should delete or seperate it sof that its not affect in model building

In [52]:

```
1 ### What is the average avocado prices in each year?
2 price_peryear= df.groupby("year")["AveragePrice"].mean().sort_values(ascending=F
3 price_peryear
```

Out[52]:

```
year
2017    1.515128
2015    1.375590
2018    1.347531
2016    1.338640
Name: AveragePrice, dtype: float64
```

In [53]:

```
# Lets creatte the new datframe with this result by reseting index and renaming
price_peryear = price_peryear.reset_index().rename(columns={'year': 'year', 'Ave
price_peryear
```

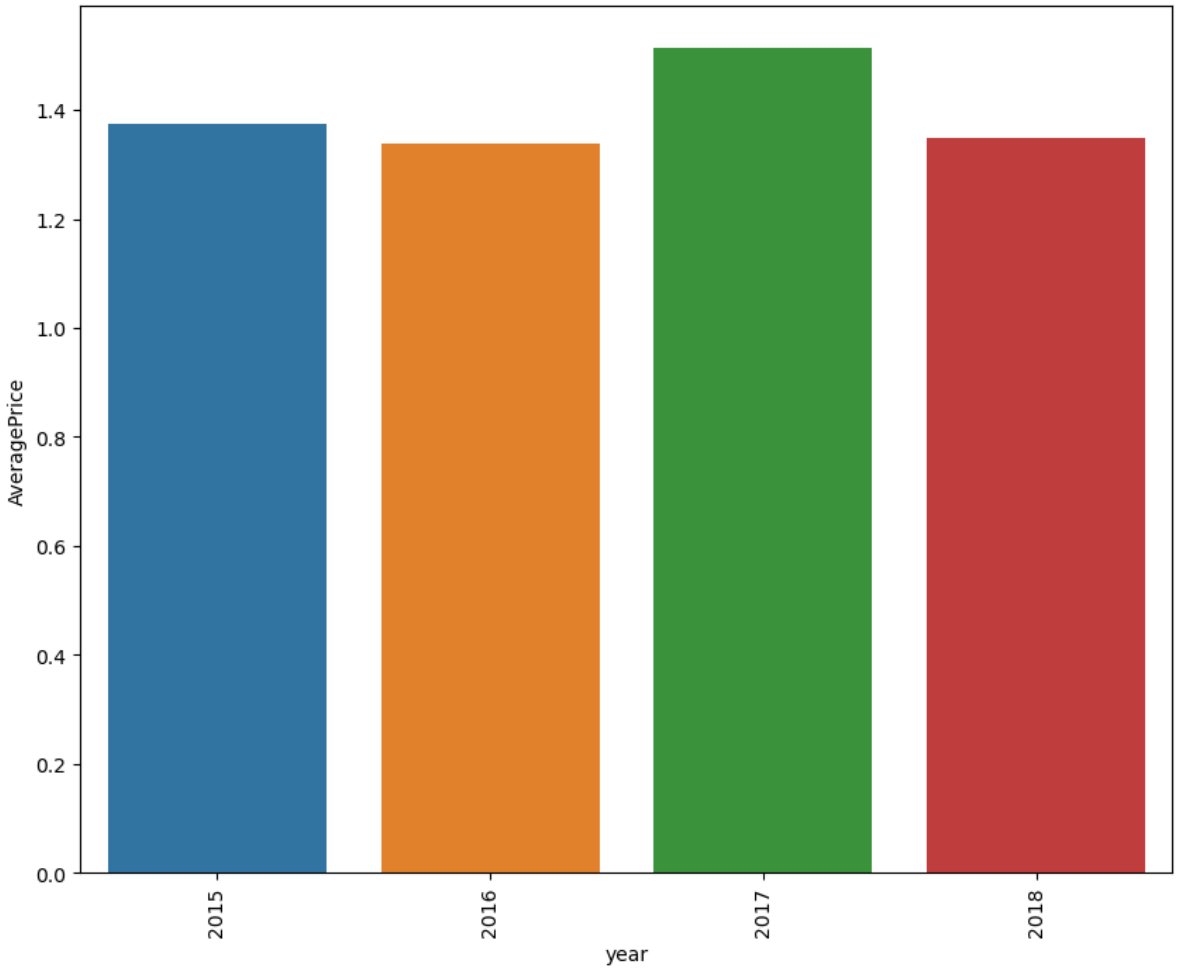
Out[53]:

	year	AveragePrice
0	2017	1.515128
1	2015	1.375590
2	2018	1.347531
3	2016	1.338640

In [54]:

```
plt.figure(figsize=(10, 8))
sns.barplot(data=price_peryear,x="year",y="AveragePrice")
plt.xlabel("year")
plt.xticks(rotation='vertical')

# Display the plot
plt.show()
```



In [ ]:

```
# we see the in 2017 is slightly high avcado price

```

What is the average avocado volume in each year?

In [55]:

```
voloume_peryear= df.groupby("year")["TotalVolume"].mean().sort_values(ascending=
voloume_peryear
```

Out[55]:

```
year
2018    1.066928e+06
2017    8.623393e+05
2016    8.584206e+05
2015    7.810274e+05
Name: TotalVolume, dtype: float64
```

In [56]:

```
voloume_peryear=voloume_peryear.reset_index().rename(columns={'year': 'year', 'T
voloume_peryear
```

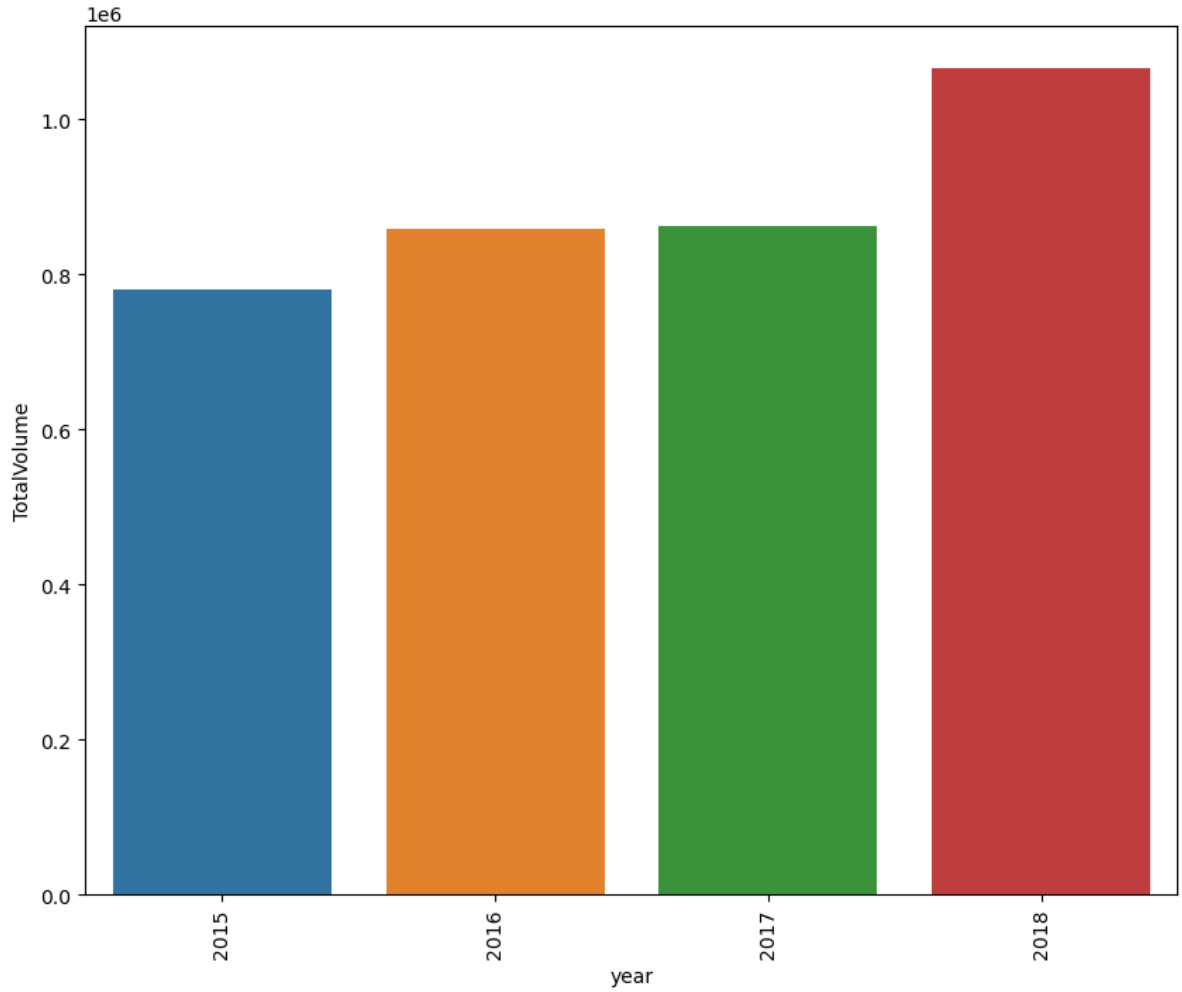
Out[56]:

	year	TotalVolume
0	2018	1.066928e+06
1	2017	8.623393e+05
2	2016	8.584206e+05
3	2015	7.810274e+05

In [57]:

```
plt.figure(figsize=(10, 8))
sns.barplot(data=voloume_peryear,x="year",y="TotalVolume")
plt.xlabel("year")
plt.xticks(rotation='vertical')

# Display the plot
plt.show()
```



Insight:in 2018 prodiction is high than the others years

Data modeling

#As dependent variable is continous , First we build linear regression model and use both _xtest and and y train for predictiong and find the accuracy of model

In [58]:

```
df1.head()
```

Out[58]:

	AveragePrice	TotalVolume	TotalBags	SmallBags	LargeBags	XLargeBags	type	year
0	1.33	64236.62	8696.87	8603.62	93.25	0.0	conventional	2015
1	1.35	54876.98	9505.56	9408.07	97.49	0.0	conventional	2015
2	0.93	118220.22	8145.35	8042.21	103.14	0.0	conventional	2015
3	1.08	78992.15	5811.16	5677.40	133.76	0.0	conventional	2015
4	1.28	51039.60	6183.95	5986.26	197.69	0.0	conventional	2015

In [72]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   AveragePrice  18249 non-null  float64
 1   TotalVolume   18249 non-null  float64
 2   TotalBags     18249 non-null  float64
 3   SmallBags     18249 non-null  float64
 4   LargeBags     18249 non-null  float64
 5   XLargeBags    18249 non-null  float64
 6   type          18249 non-null  object 
 7   year          18249 non-null  int64  
 8   region        18249 non-null  object 
dtypes: float64(6), int64(1), object(2)
memory usage: 1.3+ MB
```

In [73]:

```
#we have two object in data , we need to convert it to catagory and assign numer
df1['region'] = df1['region'].astype('category').cat.codes
df1['type'] = df1['type'].astype('category').cat.codes
```

In [74]:

```
#Lets define the variable
X=df1.drop(['AveragePrice'],axis=1)
X.head()
```

Out[74]:

	TotalVolume	TotalBags	SmallBags	LargeBags	XLargeBags	type	year	region
0	64236.62	8696.87	8603.62	93.25	0.0	0	2015	0
1	54876.98	9505.56	9408.07	97.49	0.0	0	2015	0
2	118220.22	8145.35	8042.21	103.14	0.0	0	2015	0
3	78992.15	5811.16	5677.40	133.76	0.0	0	2015	0
4	51039.60	6183.95	5986.26	197.69	0.0	0	2015	0

In [75]:

```
y = df1['AveragePrice']
```

In [76]:

```
#now split the variables
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_sta
```

In [77]:

```
X_train.head()
```

Out[77]:

	TotalVolume	TotalBags	SmallBags	LargeBags	XLargeBags	type	year	region
3984	2866855.58	993144.09	871562.92	104146.00	17435.17	0	2016	22
3163	6588277.70	2391861.16	2091747.51	282242.79	17870.86	0	2016	6
17247	2375.19	1332.62	312.22	1020.40	0.00	1	2017	47
8884	1354912.98	426983.93	200603.91	221593.36	4786.66	0	2018	33
16345	16661.59	16555.36	16555.36	0.00	0.00	1	2017	30

In [78]:

```
X_train.shape
```

Out[78]:

```
(12774, 8)
```

In [79]:

```
X_test.shape
```

Out[79]:

```
(5475, 8)
```

In [80]:

```
y_train.shape
```

Out[80]:

```
(12774,)
```

In [81]:

```
y_test.shape
```

Out[81]:

```
(5475,)
```

In [83]:

```
#We will scalized the data
# Scale the features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [85]:

```
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[85]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]:

```
#now our model build , now prediction
y_pre = model.predict(X_test)
test_score = r2_score(y_test,y_pre)
print("The accuracy of testing dataset ",test_score*100)
```

```
The accuracy of testing dataset  38.94980399703235
```

In [87]:

```
#use train data as predictor

train_pre = model.predict(X_train)
train_score = r2_score(y_train,train_pre)
print("The accuracy of training dataset ",train_score*100)

```

```
The accuracy of training dataset  39.54085418061671
```

In [88]:

```
#check the bias
bias = model.score(X_train, y_train)
bias
```

Out[88]:

```
0.39540854180616714
```

In [89]:

```
#check the variance
variance = model.score(X_test, y_test)
variance
```

Out[89]:

```
0.3894980399703235
```

```
Insight: the model accuracy is substantialy low , so this model is not
appropriate with this data
Lets try with other regressor model and find their accuray
```

In [91]:

```
#importing ML models from scikit-learn
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [93]:

```
#Then create a loop so that we can build the model one by one and find the r2_sc
regressors = {
    'Linear Regression' : LinearRegression(),
    'Decision Tree' : DecisionTreeRegressor(),
    'Random Forest' : RandomForestRegressor(),
    'Support Vector Machines' : SVR(gamma=1),
    'K-nearest Neighbors' : KNeighborsRegressor(n_neighbors=1),

}
results=pd.DataFrame(columns=['MAE','MSE','R2-score'])
for method,func in regressors.items():
    model = func.fit(X_train,y_train)
    pred = model.predict(X_test)
    results.loc[method]= [np.round(mean_absolute_error(y_test,pred),2),
                          np.round(mean_squared_error(y_test,pred),2),
                          np.round(r2_score(y_test,pred),2)
                         ]
```

In [94]:

```
results
```

Out[94]:

	MAE	MSE	R2-score
Linear Regression	0.24	0.10	0.39
Decision Tree	0.15	0.06	0.66
Random Forest	0.12	0.03	0.82
Support Vector Machines	0.32	0.16	-0.00
K-nearest Neighbors	0.25	0.13	0.21

```
Insight:Interpreting the table:

Linear Regression:

MAE: 0.24
MSE: 0.10
R2-score: 0.39
The linear regression model has a moderate MAE and MSE, indicating that the predictions are reasonably close to the true values. However, the R2-score of 0.39 suggests that the model explains only 39% of the variance in the dependent variable.
Decision Tree:

MAE: 0.15
MSE: 0.06
R2-score: 0.66
The decision tree model performs better than the linear regression model, with lower MAE and MSE values. The R2-score of 0.66 indicates that the decision tree model explains 66% of the variance in the dependent variable.
Random Forest:

MAE: 0.12
MSE: 0.03
R2-score: 0.82
The random forest model outperforms both the linear regression and decision tree models, with lower MAE and MSE values. The high R2-score of 0.82 suggests that the random forest model explains 82% of the variance in the dependent variable.
Support Vector Machines:

MAE: 0.32
MSE: 0.16
R2-score: -0.00
The support vector machines (SVM) model has a higher MAE and MSE compared to the previous models. The R2-score of -0.00 indicates that the SVM model does not explain the variance in the dependent variable and may not be a good fit for the data.
K-nearest Neighbors:

MAE: 0.25
MSE: 0.13
R2-score: 0.21
The k-nearest neighbors (KNN) model has moderate MAE and MSE values. The R2-score of 0.21 suggests that the KNN model explains 21% of the variance in the dependent variable.
In summary, the table provides a comparison of the performance of different regression models based on the evaluation metrics. Lower MAE and MSE values and higher R2-scores generally indicate better model performance. Therefore, based on the given metrics, the random forest model appears to be the best performer among the listed models.
```