In [2]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt


list_of_dicts = [
     {"name": "Ginger", "breed": "Dachshund", "height_cm": 22,"weight_kg": 10, "dat
    {"name": "Scout", "breed": "Dalmatian", "height_cm": 59,"weight_kg": 25, "date_
]
new_dogs = pd.DataFrame(list_of_dicts)
new_dogs
```

Out[2]:

	name	breed	height_cm	weight_kg	date_of_birth
0	Ginger	Dachshund	22	10	2019-03-14
1	Scout	Dalmatian	59	25	2019-05-09

In [3]:
```
dict_of_lists = {
     "name": ["Ginger", "Scout"],
     "breed": ["Dachshund", "Dalmatian"],
     "height_cm": [22, 59],
     "weight_kg": [10, 25],
     "date_of_birth": ["2019-03-14","2019-05-09"]  }
new_dogs = pd.DataFrame(dict_of_lists)
new_dogs
```

Out[3]:

	name	breed	height_cm	weight_kg	date_of_birth
0	Ginger	Dachshund	22	10	2019-03-14
1	Scout	Dalmatian	59	25	2019-05-09

In [5]:
```
# read CSV from using pandas
avocado = pd.read_csv(r"C:\Users\admin\Downloads\avocado.csv")
# print the first few rows of the dataframe
avocado.head()
```

Out[5]:

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

In [7]:
```
avocado = avocado.reset_index(drop=True)
avocado.head()
```

Out[7]:

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

In [9]:
```
avocado.to_csv("test_write.csv")
```

In [11]:
```
avocado = pd.read_csv(r"C:\Users\admin\Downloads\avocado.csv")
avocado.head()
```

Out[11]:

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

In [12]:
```
avocado.tail(10)
```

Out[12]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Sr B
18239	2	2018-03-11	1.56	22128.42	2162.67	3194.25	8.93	16762.57	1651
18240	3	2018-03-04	1.54	17393.30	1832.24	1905.57	0.00	13655.49	1340
18241	4	2018-02-25	1.57	18421.24	1974.26	2482.65	0.00	13964.33	1369
18242	5	2018-02-18	1.56	17597.12	1892.05	1928.36	0.00	13776.71	1355
18243	6	2018-02-11	1.57	15986.17	1924.28	1368.32	0.00	12693.57	1243
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	1306
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	894
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	935
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	1091
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	1198

In [13]:
```
avocado.info()
```

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

In [14]:
```
print(avocado.shape)
```

```
(18249, 14)
```

In [15]:
```
avocado.describe()
```

Out[15]:

	Unnamed: 0	AveragePrice	Total Volume	4046	4225	477
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+0
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+0
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+0
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+0
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+0
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+0
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+0
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+0

In [16]:
```
avocado.values
```

Out[16]:
```
array([[0, '2015-12-27', 1.33, ..., 'conventional', 2015, 'Albany'],
       [1, '2015-12-20', 1.35, ..., 'conventional', 2015, 'Albany'],
       [2, '2015-12-13', 0.93, ..., 'conventional', 2015, 'Albany'],
       ...,
       [9, '2018-01-21', 1.87, ..., 'organic', 2018, 'WestTexNewMexico'],
       [10, '2018-01-14', 1.93, ..., 'organic', 2018, 'WestTexNewMexico'],
       [11, '2018-01-07', 1.62, ..., 'organic', 2018, 'WestTexNewMexico']],
      shape=(18249, 14), dtype=object)
```

In [17]:
```
print(avocado.columns)
```

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

In [21]:
```
import pandas as pd

even = pd.Series([2, 4, 6, 8, 10])
odd = pd.Series([1, 3, 5, 7, 9])

res = pd.concat([even, odd])
print(res)
```

```
0     2
1     4
2     6
3     8
4    10
0     1
1     3
2     5
3     7
4     9
dtype: int64
```

In [22]:
```
res.reset_index(drop=True)
```

Out[22]:
```
0     2
1     4
2     6
3     8
4    10
5     1
6     3
7     5
8     7
9     9
dtype: int64
```

In [23]:
```
# sort values based on "AveragePrice" (ascending) and "year" (descending)
avocado.sort_values(["AveragePrice", "year"], ascending=[True, False])
```

Out[23]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	
15261	43	2017-03-05	0.44	64057.04	223.84	4748.88	0.00	5908
7412	47	2017-02-05	0.46	2200550.27	1200632.86	531226.65	18324.93	45036
15473	43	2017-03-05	0.48	50890.73	717.57	4138.84	0.00	4603
15262	44	2017-02-26	0.49	44024.03	252.79	4472.68	0.00	3929
1716	0	2015-12-27	0.49	1137707.43	738314.80	286858.37	11642.46	10089
...	...	...	...	...	...	...	...	
16720	18	2017-08-27	3.04	12656.32	419.06	4851.90	145.09	724
16055	42	2017-03-12	3.05	2068.26	1043.83	77.36	0.00	94
14124	7	2016-11-06	3.12	19043.80	5898.49	10039.34	0.00	310
17428	37	2017-04-16	3.17	3018.56	1255.55	82.31	0.00	168
14125	8	2016-10-30	3.25	16700.94	2325.93	11142.85	0.00	323

18249 rows × 14 columns

In [24]:
```
# Subsetting columns
avocado["AveragePrice"]
```

Out[24]:
```
0        1.33
1        1.35
2        0.93
3        1.08
4        1.28
         ...
18244    1.63
18245    1.71
18246    1.87
18247    1.93
18248    1.62
Name: AveragePrice, Length: 18249, dtype: float64
```

In [25]:
```
# Subsetting multiple columns
avocado[["AveragePrice","Date"]]
```

Out[25]:

	AveragePrice	Date
0	1.33	2015-12-27
1	1.35	2015-12-20
2	0.93	2015-12-13
3	1.08	2015-12-06
4	1.28	2015-11-29
...	...	...
18244	1.63	2018-02-04
18245	1.71	2018-01-28
18246	1.87	2018-01-21
18247	1.93	2018-01-14
18248	1.62	2018-01-07

18249 rows × 2 columns

In [26]:
```
# Subsetting rows
avocado["AveragePrice"]<1
```

Out[26]:
```
0        False
1        False
2         True
3        False
4        False
         ...
18244    False
18245    False
18246    False
18247    False
18248    False
Name: AveragePrice, Length: 18249, dtype: bool
```

In [27]:
```
# This will print only the rows with price < 1
avocado[avocado["AveragePrice"]<1]
```

Out[27]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35
6	6	2015-11-15	0.99	83453.76	1368.92	73672.72	93.26	8318.86
7	7	2015-11-08	0.98	109428.33	703.75	101815.36	80.00	6829.22
13	13	2015-09-27	0.99	106803.39	1204.88	99409.21	154.84	6034.46
43	43	2015-03-01	0.99	55595.74	629.46	45633.34	181.49	9151.45
...	...	...	...	...	...	...	...	...
17169	43	2017-03-05	0.99	155011.12	35367.23	5175.81	5.91	114462.17
17170	44	2017-02-26	0.99	171145.00	34520.03	6936.39	0.00	129688.58
17536	39	2017-04-02	0.98	402676.23	34093.33	58330.53	207.85	310044.52
17537	40	2017-03-26	0.90	456645.91	36169.35	51398.72	139.55	368938.29
17540	43	2017-03-05	0.99	367519.17	61166.48	55123.99	126.80	251101.90

2796 rows × 14 columns

In [28]:
```
# it will print all the rows with "type" = "organic"
avocado[avocado["type"]=="organic"]
```

Out[28]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Sr B
9126	0	2015-12-27	1.83	989.55	8.16	88.59	0.00	892.80	89
9127	1	2015-12-20	1.89	1163.03	30.24	172.14	0.00	960.65	96
9128	2	2015-12-13	1.85	995.96	10.44	178.70	0.00	806.82	80
9129	3	2015-12-06	1.84	1158.42	90.29	104.18	0.00	963.95	94
9130	4	2015-11-29	1.94	831.69	0.00	94.73	0.00	736.96	73
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	1306
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	894
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	935
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	1091
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	1198

9123 rows × 14 columns

In [29]:
```
# it will print all the rows with "Date" <= 2015-02-04
avocado[avocado["Date"]<="2015-02-04"]
```

Out[29]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	S
47	47	2015-02-01	0.99	70873.60	1353.90	60017.20	179.32	9323.18	91
48	48	2015-01-25	1.06	45147.50	941.38	33196.16	164.14	10845.82	101
49	49	2015-01-18	1.17	44511.28	914.14	31540.32	135.77	11921.05	116
50	50	2015-01-11	1.24	41195.08	1002.85	31640.34	127.12	8424.77	80
51	51	2015-01-04	1.22	40873.28	2819.50	28287.42	49.90	9716.46	91
...	...	...	...	...	...	...	...	...	
11928	46	2015-02-01	1.77	7210.19	1634.42	3012.44	0.00	2563.33	25
11929	47	2015-01-25	1.63	7324.06	1934.46	3032.72	0.00	2356.88	23
11930	48	2015-01-18	1.71	5508.20	1793.64	2078.72	0.00	1635.84	16
11931	49	2015-01-11	1.69	6861.73	1822.28	2377.54	0.00	2661.91	26
11932	50	2015-01-04	1.64	6182.81	1561.30	2958.17	0.00	1663.34	16

540 rows × 14 columns

In [30]:
```
# it will print all the rows with "Date" before 2015-02-04 and "type" == "organic"
avocado[(avocado["Date"]<"2015-02-04") & (avocado["type"]=="organic")]
```

Out[30]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags
9173	47	2015-02-01	1.83	1228.51	33.12	99.36	0.0	1096.03	1096.03
9174	48	2015-01-25	1.89	1115.89	14.87	148.72	0.0	952.30	952.30
9175	49	2015-01-18	1.93	1118.47	8.02	178.78	0.0	931.67	931.67
9176	50	2015-01-11	1.77	1182.56	39.00	305.12	0.0	838.44	838.44
9177	51	2015-01-04	1.79	1373.95	57.42	153.88	0.0	1162.65	1162.65
...	...	...	...	...	...	...	...	...	...
11928	46	2015-02-01	1.77	7210.19	1634.42	3012.44	0.0	2563.33	2563.33
11929	47	2015-01-25	1.63	7324.06	1934.46	3032.72	0.0	2356.88	2320.00
11930	48	2015-01-18	1.71	5508.20	1793.64	2078.72	0.0	1635.84	1620.00
11931	49	2015-01-11	1.69	6861.73	1822.28	2377.54	0.0	2661.91	2656.66
11932	50	2015-01-04	1.64	6182.81	1561.30	2958.17	0.0	1663.34	1663.34

270 rows × 14 columns

In [31]:
```
# subset the avocado in the region Boston or SanDiego
regionFilter = avocado["region"].isin(["Boston", "SanDiego"])
avocado[regionFilter]
```

Out[31]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags
208	0	2015-12-27	1.13	450816.39	3886.27	346964.70	13952.56	86012.86
209	1	2015-12-20	1.07	489802.88	4912.37	390100.99	5887.72	88901.80
210	2	2015-12-13	1.01	549945.76	4641.02	455362.38	219.40	89722.96
211	3	2015-12-06	1.02	488679.31	5126.32	407520.22	142.99	75889.78
212	4	2015-11-29	1.19	350559.81	3609.25	272719.08	105.86	74125.62
...	...	...	...	...	...	...	...	...
18100	7	2018-02-04	1.81	17454.74	1158.41	7388.27	0.00	8908.06
18101	8	2018-01-28	1.91	17579.47	1145.64	8284.41	0.00	8149.42
18102	9	2018-01-21	1.95	18676.37	1088.49	9282.37	0.00	8305.51
18103	10	2018-01-14	1.81	21770.02	3285.98	14338.52	0.00	4145.52
18104	11	2018-01-07	2.06	16746.82	5150.82	9366.31	0.00	2229.69

676 rows × 14 columns

In [33]:
```
# subset the avocado in the region Boston or SanDiego in the year 2016 or 2017
regionFilter = avocado["region"].isin(["Boston", "SanDiego"])
yearFilter = avocado["year"].isin(["2016", "2017"])
avocado[regionFilter & yearFilter]
```

Out[33]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	XLarge Bags

In [34]:
```
avocado.isna()
```

Out[34]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags	X
0	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	
18244	False	False	False	False	False	False	False	False	False	False	
18245	False	False	False	False	False	False	False	False	False	False	
18246	False	False	False	False	False	False	False	False	False	False	
18247	False	False	False	False	False	False	False	False	False	False	
18248	False	False	False	False	False	False	False	False	False	False	

18249 rows × 14 columns

In [35]:
```
avocado.isna().any()
```

Out[35]:
```
Unnamed: 0      False
Date            False
AveragePrice    False
Total Volume    False
4046            False
4225            False
4770            False
Total Bags      False
Small Bags      False
Large Bags      False
XLarge Bags     False
type            False
year            False
region          False
dtype: bool
```

In [36]:
```
avocado.isna().sum()
```

Out[36]:
```
Unnamed: 0      0
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
```

In [37]:
```
# ****  OR  ****

meanVal = avocado["AveragePrice"].mean()
avocado.fillna(meanVal)
```

Out[37]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11

18249 rows × 14 columns

In [38]:
```
avocado["AveragePricePer100"] = avocado["AveragePrice"] * 100
avocado
```

Out[38]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11

18249 rows × 15 columns

In [39]:
```
avocado.drop(["AveragePricePer100"],axis = 1)
```

Out[39]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11

18249 rows × 14 columns

In [40]:
```
# mean of the AveragePrice of avocado
avocado["AveragePrice"].mean()
```

Out[40]:
```
np.float64(1.405978409775878)
```

In [41]:
```
def pct30(column):
    #return the 0.3 quartile
    return column.quantile(0.3)
def pct50(column):
    #return the 0.5 quartile
    return column.quantile(0.5)

avocado[["AveragePrice","Total Bags"]].agg([pct30,pct50])
```

Out[41]:

	AveragePrice	Total Bags
pct30	1.15	7316.634
pct50	1.37	39743.830

In [42]:
```
temp = avocado.drop_duplicates(subset=["year"])
temp
```

Out[42]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87
2808	0	2016-12-25	1.52	73341.73	3202.39	58280.33	426.92	11432.09
5616	0	2017-12-31	1.47	113514.42	2622.70	101135.53	20.25	9735.94
8478	0	2018-03-25	1.57	149396.50	16361.69	109045.03	65.45	23924.33

In [43]:
```
# count number of avocado in each year in descending order
avocado["year"].value_counts(sort=True, ascending = False)
```

Out[43]:
```
year
2017    5722
2016    5616
2015    5615
2018    1296
Name: count, dtype: int64
```

In [47]:
```
import numpy as np

avocado.groupby(["year", "type"])["AveragePrice"].agg(["min", "max", "mean", "media
```

Out[47]:

year	type	min	max	mean	median
2015	conventional	0.49	1.59	1.077963	1.08
	organic	0.81	2.79	1.673324	1.67
2016	conventional	0.51	2.20	1.105595	1.08
	organic	0.58	3.25	1.571684	1.53
2017	conventional	0.46	2.22	1.294888	1.30
	organic	0.44	3.17	1.735521	1.72
2018	conventional	0.56	1.74	1.127886	1.14
	organic	1.01	2.30	1.567176	1.55

In [49]:
```
avocado.pivot_table(
    index=["year", "type"],
    values="AveragePrice",
    aggfunc=["min", "max", "mean", "median"]
)
```

Out[49]:

year	type	min	max	mean	median
		AveragePrice	AveragePrice	AveragePrice	AveragePrice
2015	conventional	0.49	1.59	1.077963	1.08
	organic	0.81	2.79	1.673324	1.67
2016	conventional	0.51	2.20	1.105595	1.08
	organic	0.58	3.25	1.571684	1.53
2017	conventional	0.46	2.22	1.294888	1.30
	organic	0.44	3.17	1.735521	1.72
2018	conventional	0.56	1.74	1.127886	1.14
	organic	1.01	2.30	1.567176	1.55

In [50]:
```
regionIndex = avocado.set_index(["region"])
regionIndex
```

Out[50]:

region	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770
Albany	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16
Albany	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33
Albany	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50
Albany	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58
Albany	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78
...	...	...	...	...	...	...	...
WestTexNewMexico	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00
WestTexNewMexico	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00
WestTexNewMexico	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94
WestTexNewMexico	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01
WestTexNewMexico	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53

18249 rows × 14 columns

In [51]:
```
# Insted of doing this
avocado[avocado["region"].isin(["Albany", "WestTexNewMexico"])]
```

Out[51]:

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5
...	...	...	...	...	...	...	...	...	
18244	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13
18245	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00	9264.84	8
18246	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94	9394.11	9
18247	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01	10969.54	10
18248	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53	12014.15	11

673 rows × 15 columns

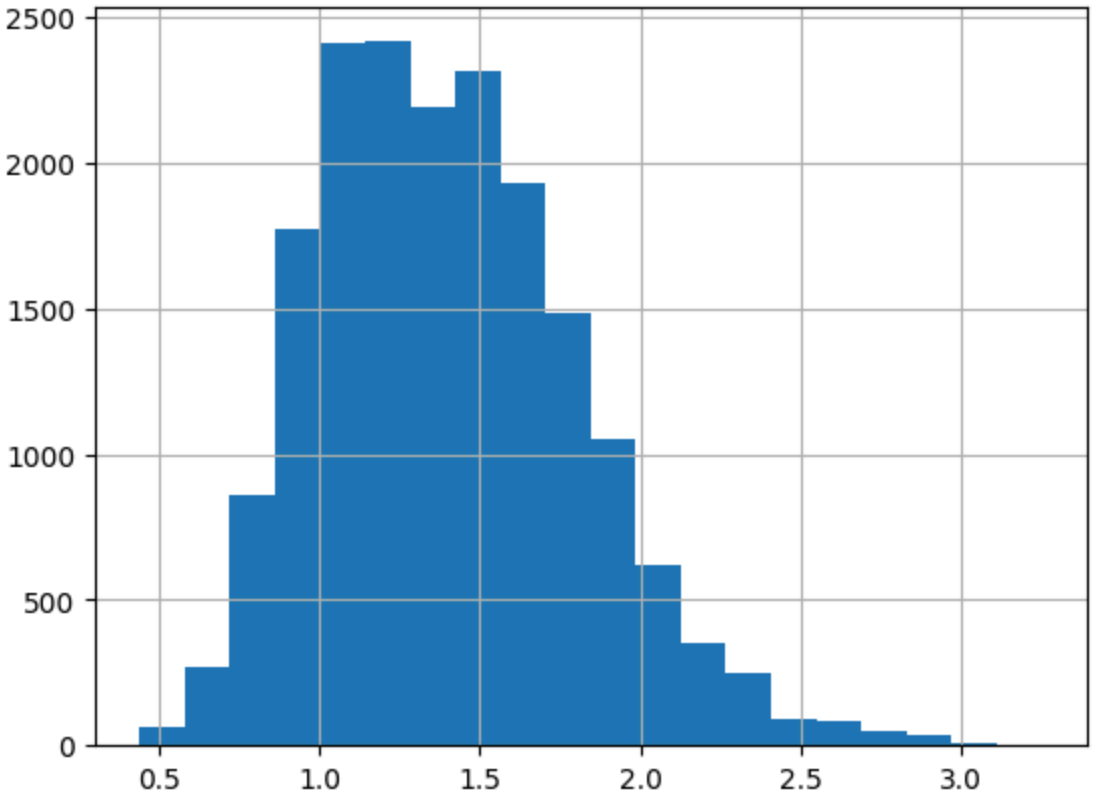
In [52]:
```
# we can simply do
regionIndex.loc[["Albany", "WestTexNewMexico"]]
```

Out[52]:

region	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770
Albany	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16
Albany	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33
Albany	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50
Albany	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58
Albany	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78
...	...	...	...	...	...	...	...
WestTexNewMexico	7	2018-02-04	1.63	17074.83	2046.96	1529.20	0.00
WestTexNewMexico	8	2018-01-28	1.71	13888.04	1191.70	3431.50	0.00
WestTexNewMexico	9	2018-01-21	1.87	13766.76	1191.92	2452.79	727.94
WestTexNewMexico	10	2018-01-14	1.93	16205.22	1527.63	2981.04	727.01
WestTexNewMexico	11	2018-01-07	1.62	17489.58	2894.77	2356.13	224.53

673 rows × 14 columns

In [53]:
```
avocado["AveragePrice"].hist(bins=20)
plt.show()
```

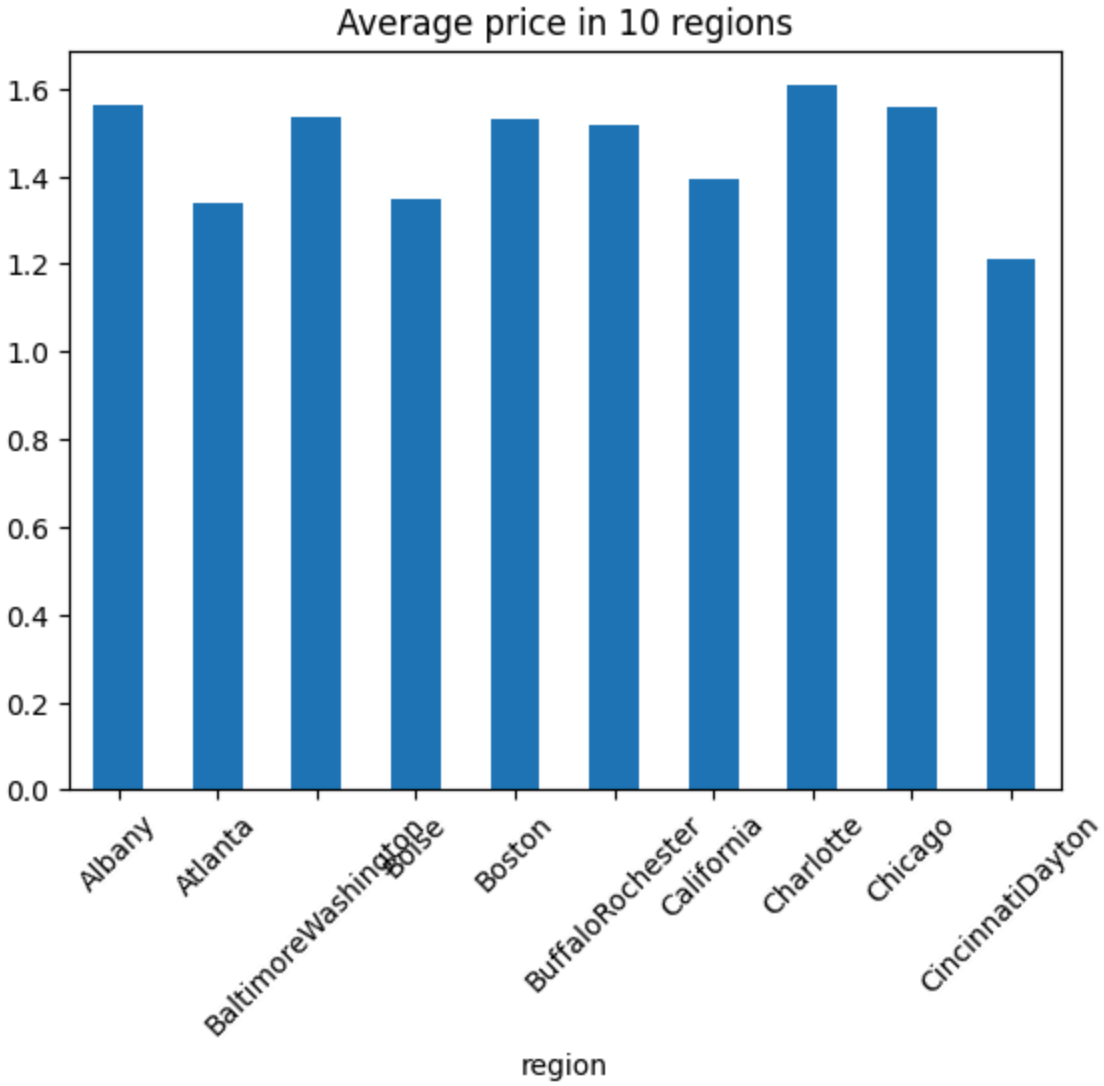


In [54]:
```
regionFilter = avocado.groupby("region")["AveragePrice"].mean().head(10)
regionFilter
```

Out[54]:
```
region
Albany                  1.561036
Atlanta                 1.337959
BaltimoreWashington     1.534231
Boise                   1.348136
Boston                  1.530888
BuffaloRochester        1.516834
California              1.395325
Charlotte               1.606036
Chicago                 1.556775
CincinnatiDayton        1.209201
Name: AveragePrice, dtype: float64
```

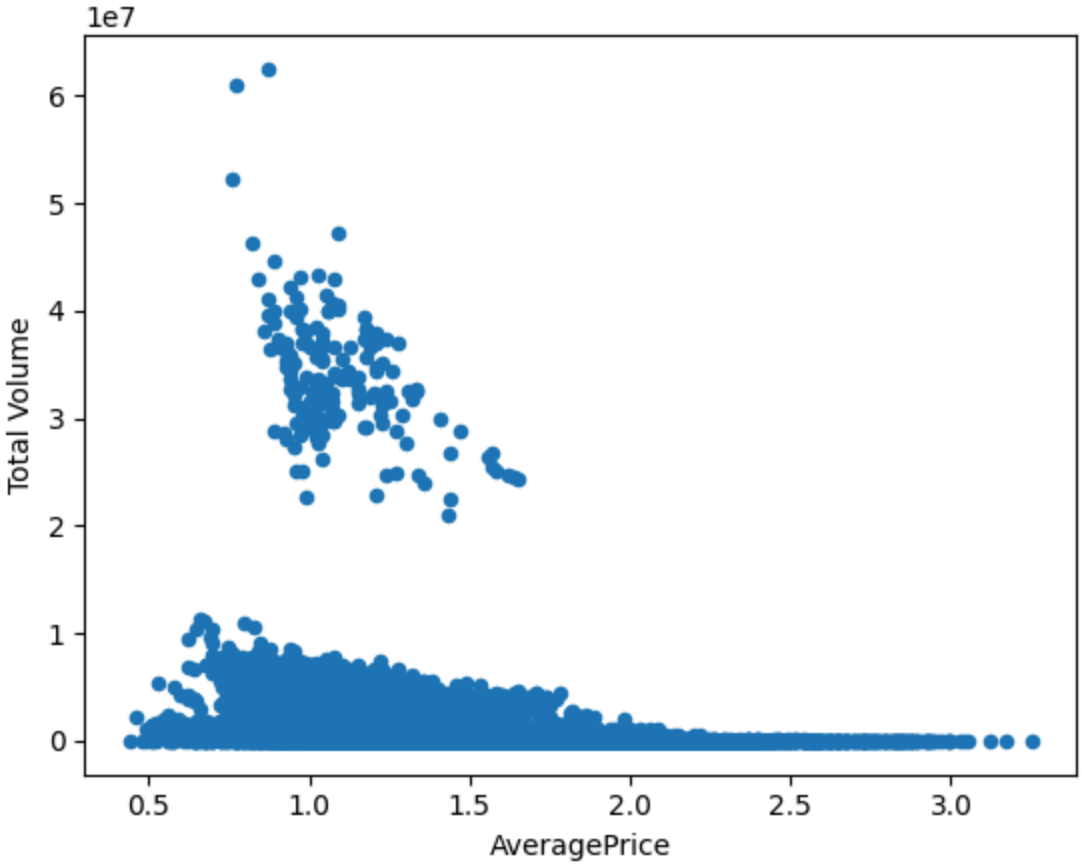
In [55]:
```
regionFilter.plot(kind = "bar",rot=45,title="Average price in 10 regions")
```

Out[55]:
```
<Axes: title={'center': 'Average price in 10 regions'}, xlabel='region'>
```



In [56]:
```
avocado.plot(x="AveragePrice", y="Total Volume", kind="scatter")
```

Out[56]:
```
<Axes: xlabel='AveragePrice', ylabel='Total Volume'>
```



In [57]:
```
# subtract AveragePrice with AveragePrice :P
# Dah its 0
avocado["AveragePrice"].sub(avocado["AveragePrice"])
```

Out[57]:
```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
18244    0.0
18245    0.0
18246    0.0
18247    0.0
18248    0.0
Name: AveragePrice, Length: 18249, dtype: float64
```

In [58]:
```
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 [59]:
```
df = pd.read_csv(r"C:\Users\admin\Downloads\avocado.csv")
```

In [60]:
```
df.info()
```

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

In [61]:
```
df.head()
```

Out[61]:

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

In [62]:
```
df.isnull().sum()
```

Out[62]:
```
Unnamed: 0      0
Date            0
AveragePrice    0
Total Volume    0
4046            0
4225            0
4770            0
Total Bags      0
Small Bags      0
Large Bags      0
XLarge Bags     0
type            0
year            0
region          0
dtype: int64
```

In [63]:
```
df = df.drop(['Unnamed: 0','4046','4225','4770','Date'],axis=1)
```

In [64]:
```
df.head()
```

Out[64]:

	AveragePrice	Total Volume	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	1.33	64236.62	8696.87	8603.62	93.25	0.0	conventional	2015	Albany
1	1.35	54876.98	9505.56	9408.07	97.49	0.0	conventional	2015	Albany
2	0.93	118220.22	8145.35	8042.21	103.14	0.0	conventional	2015	Albany
3	1.08	78992.15	5811.16	5677.40	133.76	0.0	conventional	2015	Albany
4	1.28	51039.60	6183.95	5986.26	197.69	0.0	conventional	2015	Albany

In [65]:
```
def get_avarage(df,column):
    """
    Description: This function to return the average value of the column

    Arguments:
        df: the DataFrame.
        column: the selected column.
    Returns:
        column's average
    """
    return sum(df[column])/len(df)
```

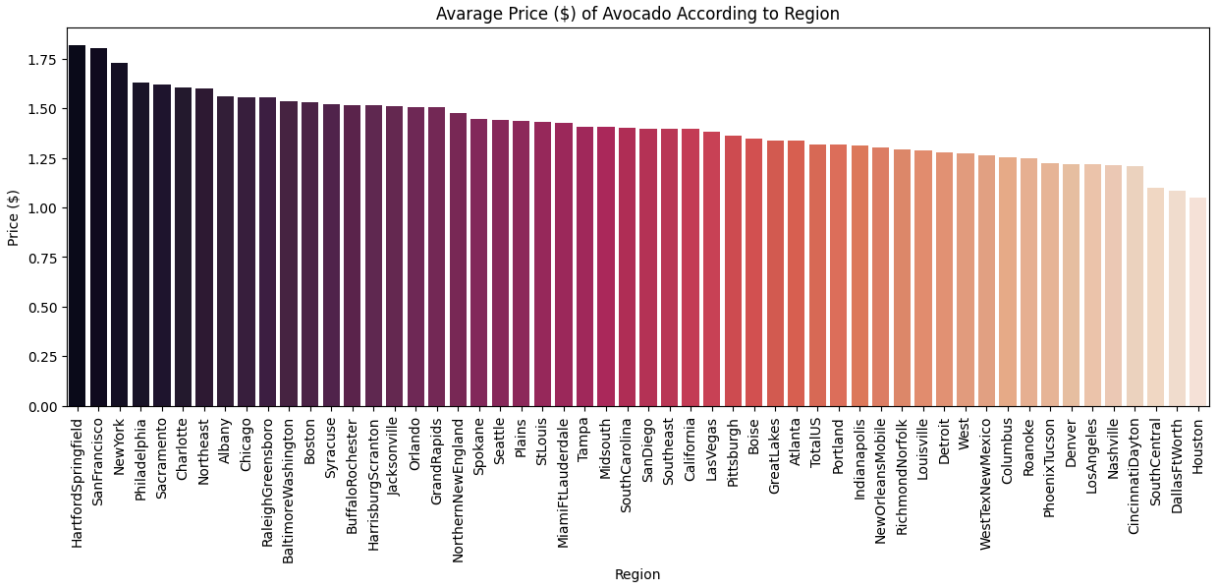
In [66]:
```
def get_avarge_between_two_columns(df,column1,column2):
    """
    Description: This function calculate the average between two columns in the dat

    Arguments:
        df: the DataFrame.
        column1:the first column.
        column2:the scond column.
    Returns:
        Sorted data for relation between column1 and column2
```

```
    """

    List=list(df[column1].unique())
    average=[]

    for i in List:
        x=df[df[column1]==i]
        column1_average= get_avarage(x,column2)
        average.append(column1_average)

    df_column1_column2=pd.DataFrame({'column1':List,'column2':average})
    column1_column2_sorted_index=df_column1_column2.column2.sort_values(ascending=F
    column1_column2_sorted_data=df_column1_column2.reindex(column1_column2_sorted_i

    return column1_column2_sorted_data
```

In [67]:
```
def plot(data,xlabel,ylabel):
    """
    Description: This function to draw a barplot

    Arguments:
        data: the DataFrame.
        xlabel: the label of the first column.
        ylabel: the label of the second column.
    Returns:
        None
    """

    plt.figure(figsize=(15,5))
    ax=sns.barplot(x=data.column1,y=data.column2,palette='rocket')
    plt.xticks(rotation=90)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(('Avarage '+ylabel+' of Avocado According to '+xlabel));
```

In [70]:
```
data1 = get_avarge_between_two_columns(df,'region','AveragePrice')
plot(data1,'Region','Price ($)')
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_9552\640296719.py:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax=sns.barplot(x=data.column1,y=data.column2,palette='rocket')
```

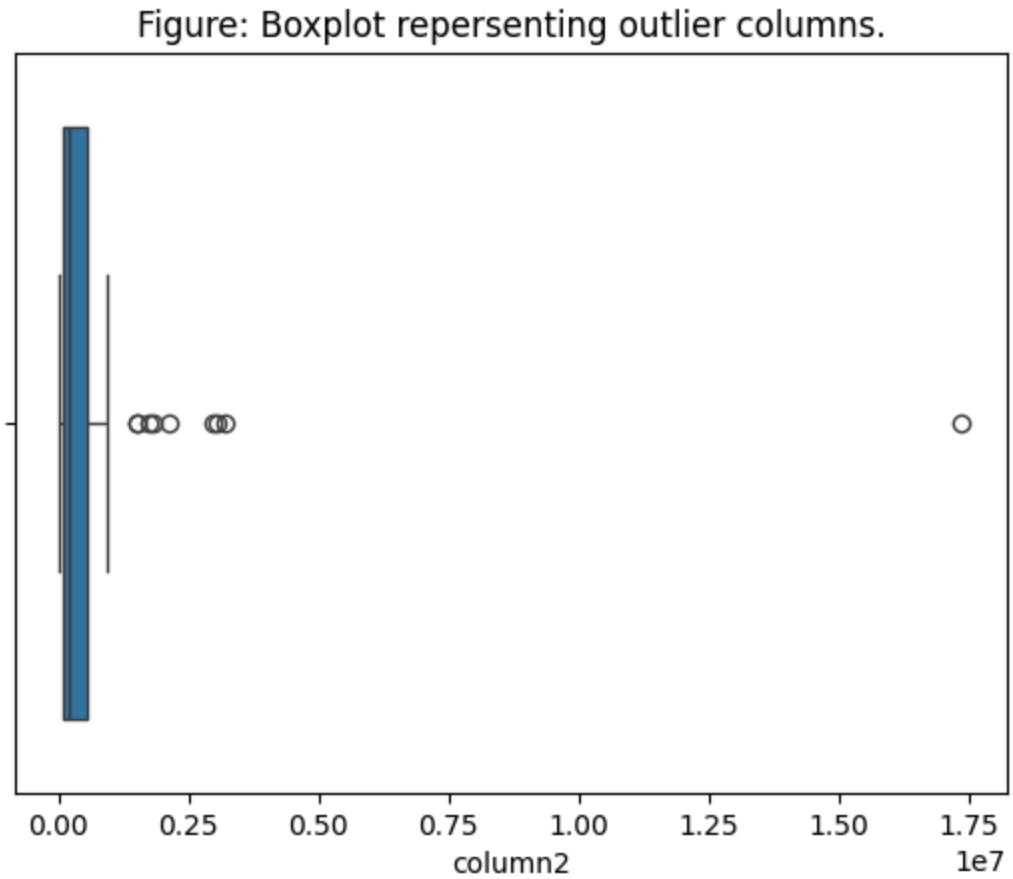


In [71]:
```
print(data1['column1'].iloc[-1], " is the region producing avocado with the lowest
```

```
Houston  is the region producing avocado with the lowest price.
```

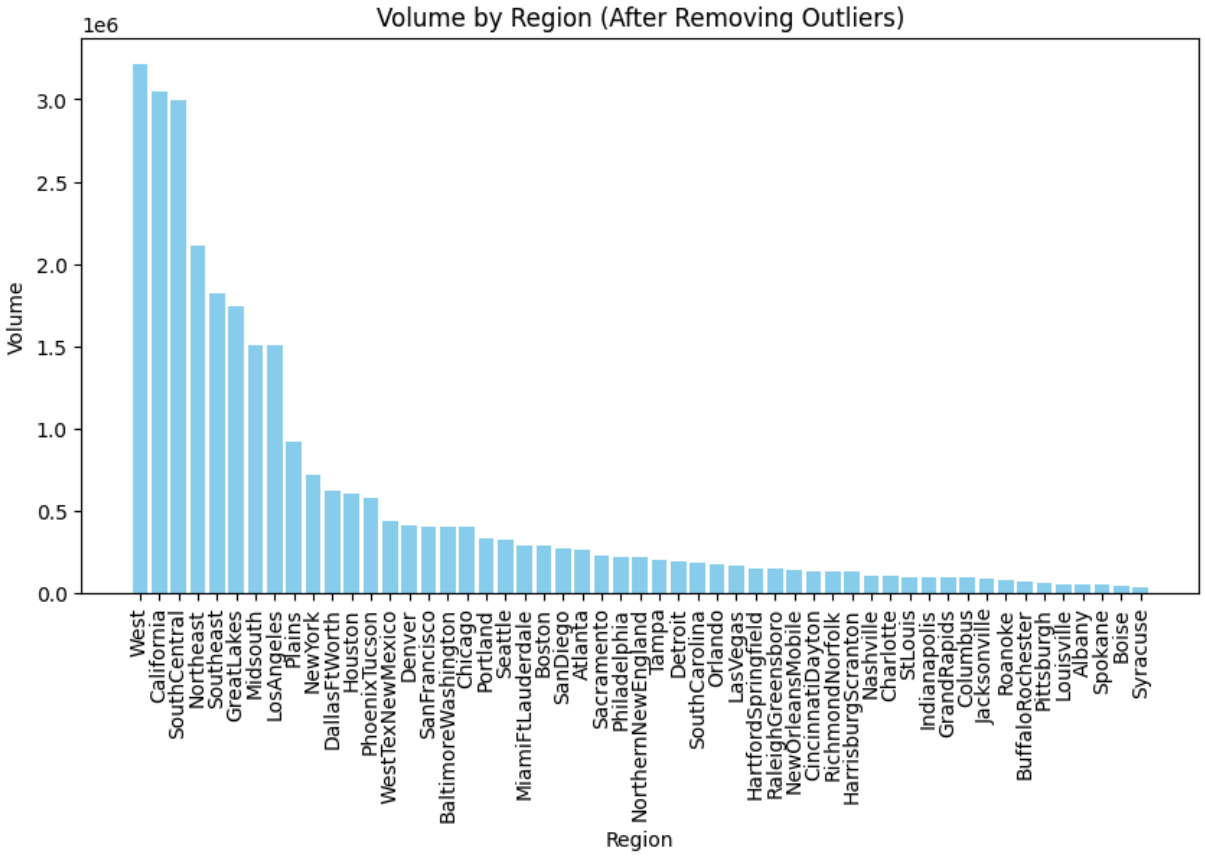
In [72]:
```
data2 = get_avarge_between_two_columns(df,'region','Total Volume')
sns.boxplot(x=data2.column2).set_title("Figure: Boxplot repersenting outlier column
```

Out[72]:
```
Text(0.5, 1.0, 'Figure: Boxplot repersenting outlier columns.')
```



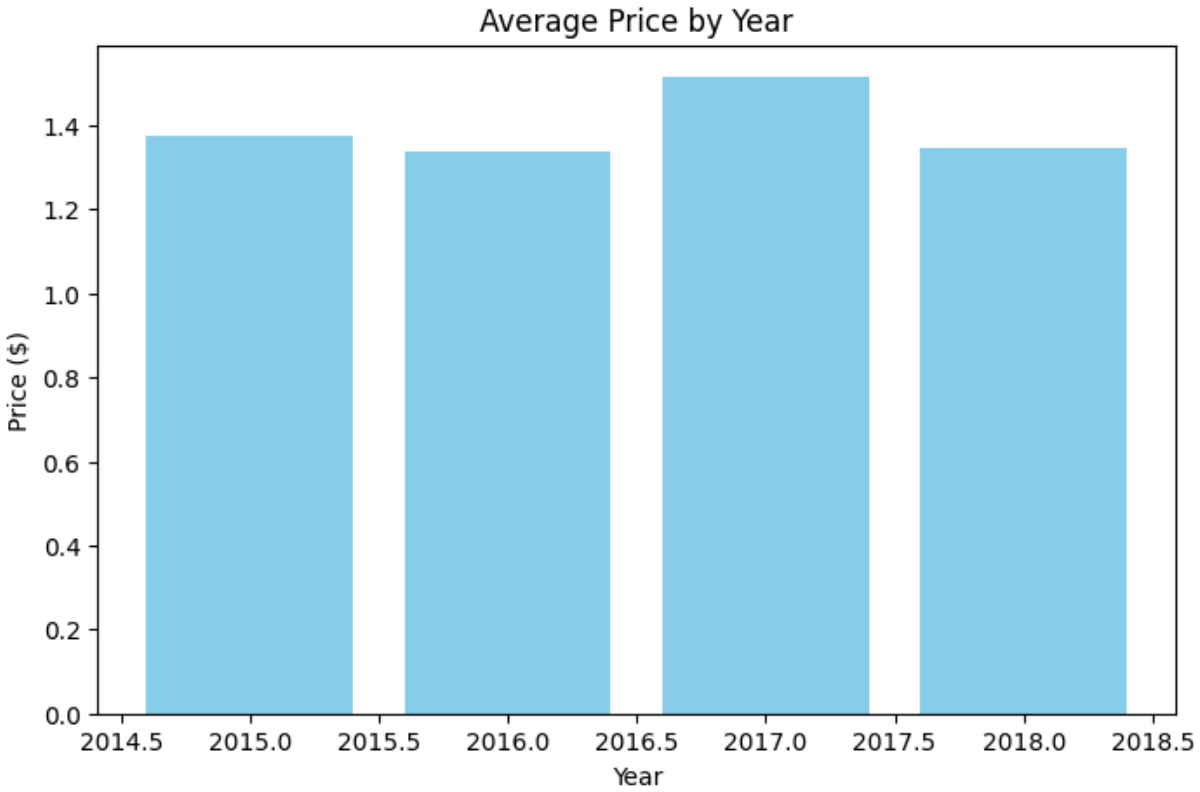
In [80]:
```
outlier_region = data2[data2['column2'] > 10000000]

if not outlier_region.empty:
    print(outlier_region['column1'].iloc[-1], "is an outlier value")
    data2 = data2.drop(outlier_region.index, axis=0)
else:
    print("No outlier found!")

import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))
plt.bar(data2['column1'], data2['column2'], color='skyblue')
plt.xlabel('Region')
plt.ylabel('Volume')
plt.title('Volume by Region (After Removing Outliers)')
plt.xticks(rotation=90)
plt.show()
```

```
No outlier found!
```

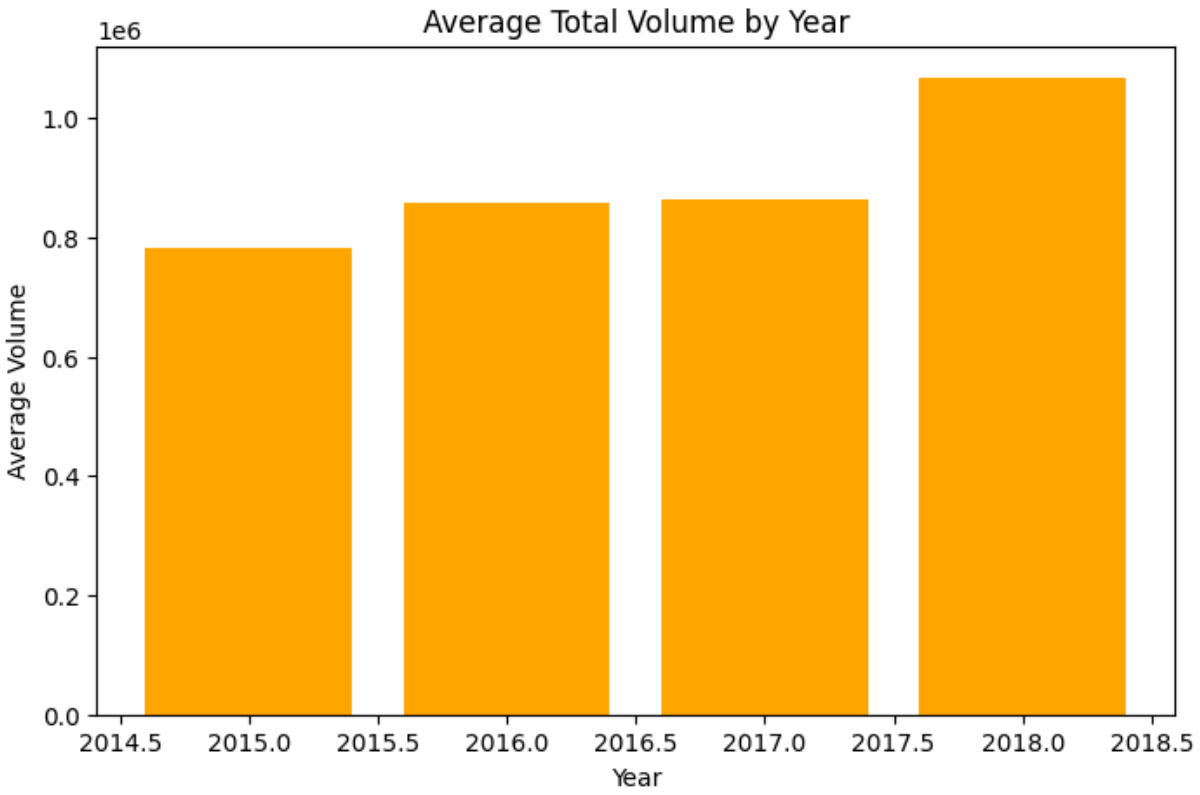


In [82]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Calculate average price by year
data3 = df.groupby('year')['AveragePrice'].mean().reset_index()

# Plot the result
plt.figure(figsize=(8,5))
plt.bar(data3['year'], data3['AveragePrice'], color='skyblue')
plt.xlabel('Year')
```

```
plt.ylabel('Price ($)')
plt.title('Average Price by Year')
plt.show()
```



In [84]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Calculate average total volume by year
data4 = df.groupby('year')['Total Volume'].mean().reset_index()

# Plot the result
plt.figure(figsize=(8,5))
plt.bar(data4['year'], data4['Total Volume'], color='orange')
plt.xlabel('Year')
plt.ylabel('Average Volume')
plt.title('Average Total Volume by Year')
plt.show()
```



In [85]:
```
df['region'] = df['region'].astype('category')
df['region'] = df['region'].cat.codes

df['type'] = df['type'].astype('category')
df['type'] = df['type'].cat.codes
```

In [86]:
```
df.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   Total Volume  18249 non-null  float64
 2   Total Bags    18249 non-null  float64
 3   Small Bags    18249 non-null  float64
 4   Large Bags    18249 non-null  float64
 5   XLarge Bags   18249 non-null  float64
 6   type          18249 non-null  int8   
 7   year          18249 non-null  int64  
 8   region        18249 non-null  int8   
dtypes: float64(6), int64(1), int8(2)
memory usage: 1.0 MB
```

In [87]:
```
df.head()
```

Out[87]:

	AveragePrice	Total Volume	Total Bags	Small Bags	Large Bags	XLarge Bags	type	year	region
0	1.33	64236.62	8696.87	8603.62	93.25	0.0	0	2015	0
1	1.35	54876.98	9505.56	9408.07	97.49	0.0	0	2015	0
2	0.93	118220.22	8145.35	8042.21	103.14	0.0	0	2015	0
3	1.08	78992.15	5811.16	5677.40	133.76	0.0	0	2015	0
4	1.28	51039.60	6183.95	5986.26	197.69	0.0	0	2015	0

In [88]:
```
# split data into X and y
X = df.drop(['AveragePrice'],axis=1)
y = df['AveragePrice']

# split data into traing and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.3,
                                                    random_state=15)
```

In [90]:
```
print("Training set:", X_train.shape, "-", y_train.shape[0], "samples")
print("Testing set:", X_test.shape, "-", y_test.shape[0], "samples")
```

```
Training set: (12774, 8) - 12774 samples
Testing set: (5475, 8) - 5475 samples
```

In [92]:
```
from sklearn.linear_model import LinearRegression

# Build and fit the model
model = LinearRegression()  # remove 'normalize=True'
model.fit(X_train, y_train)
```

Out[92]:

▾ LinearRegression

▶ Parameters

In [94]:
```
from sklearn.metrics import r2_score

# Prediction
test_pred = model.predict(X_test)

# Calculate R^2 score
test_score = r2_score(y_test, test_pred)

# Print as percentage (rounded to 2 decimal places)
print("The accuracy of testing dataset: {:.2f}%".format(test_score * 100))
```

```
The accuracy of testing dataset: 38.58%
```

In [95]:
```
#display image using python
from IPython.display import Image
```

```
url = 'https://img.etimg.com/thumb/msid-71806721,width-650,imgsize-807917,,resizemo
Image(url,height=300,width=400)
```

Out[95]:

In [97]:

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
#importing the dataset
data = pd.read_csv(r"C:\Users\admin\Downloads\avocado.csv")
# Check the data
```

In [98]:

```
data.info()
```

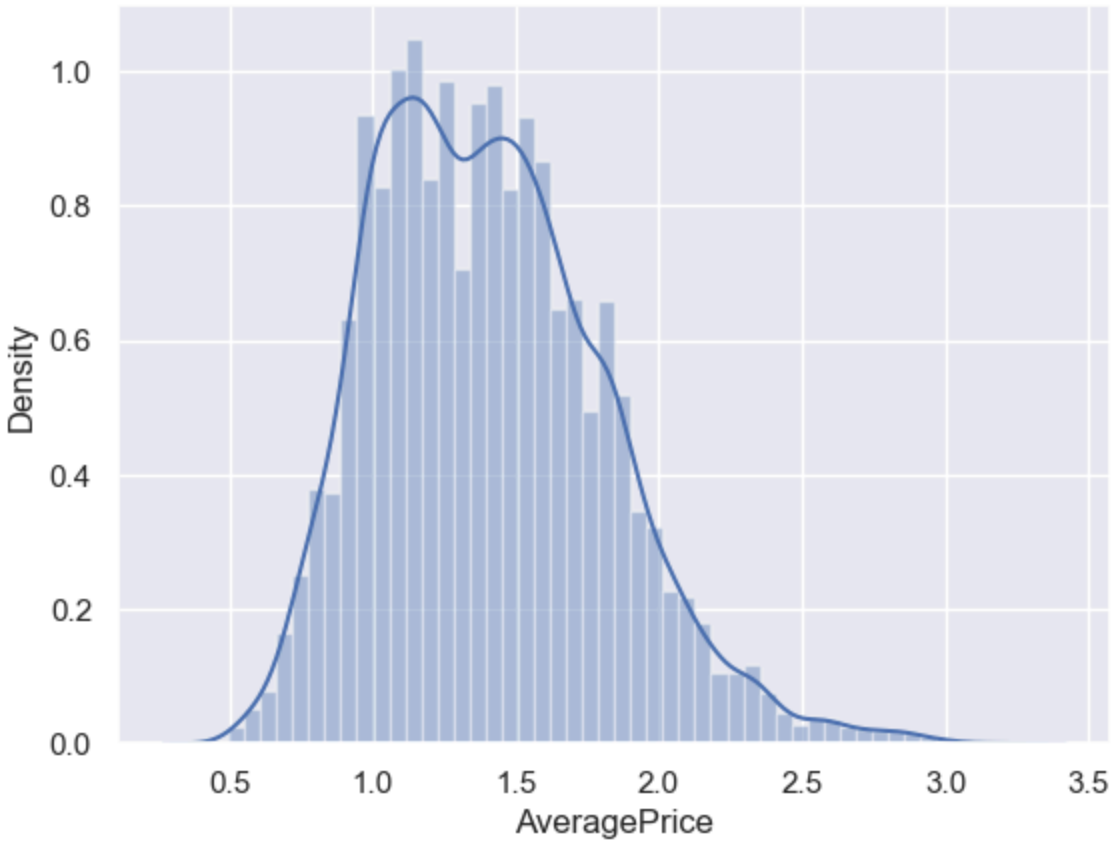
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18249 entries, 0 to 18248
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    18249 non-null  int64  
 1   Date          18249 non-null  object 
 2   AveragePrice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

In [99]:
```
data.head(3)
```

Out[99]:

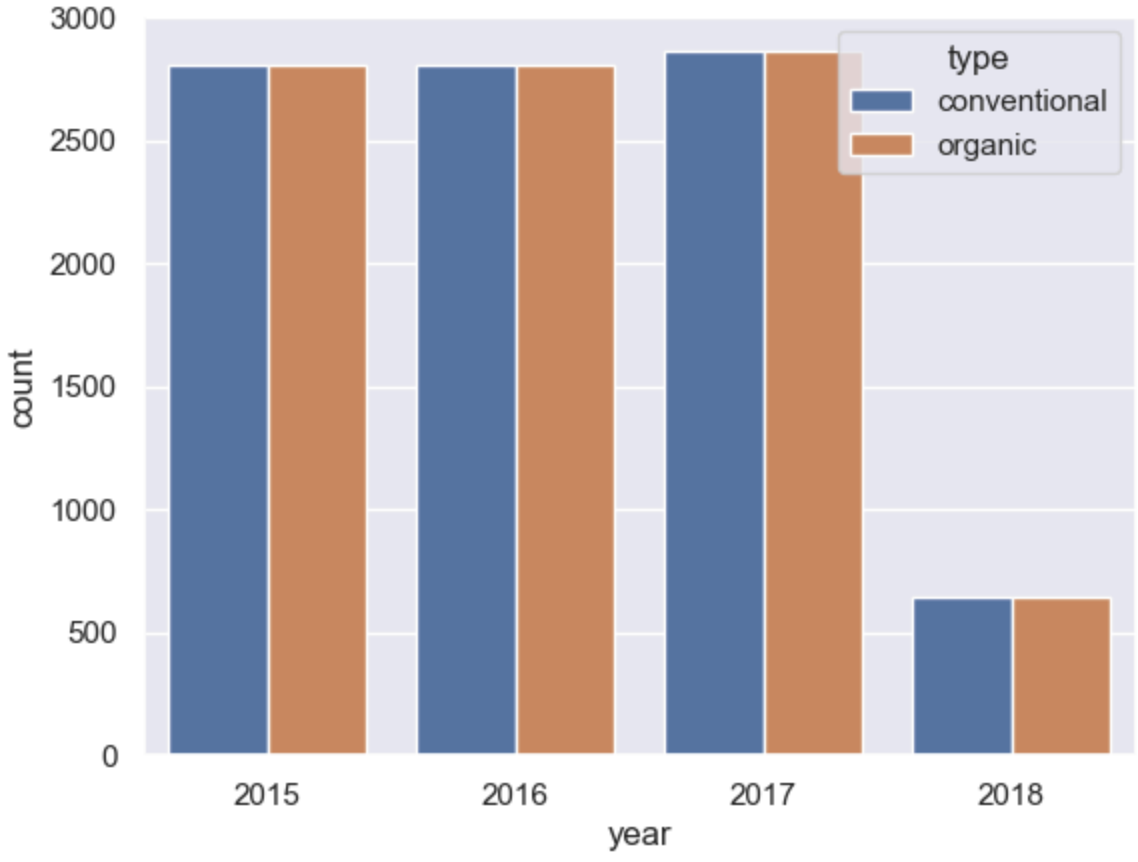
	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21

In [100…
```
sns.distplot(data['AveragePrice']);
```



In [101…

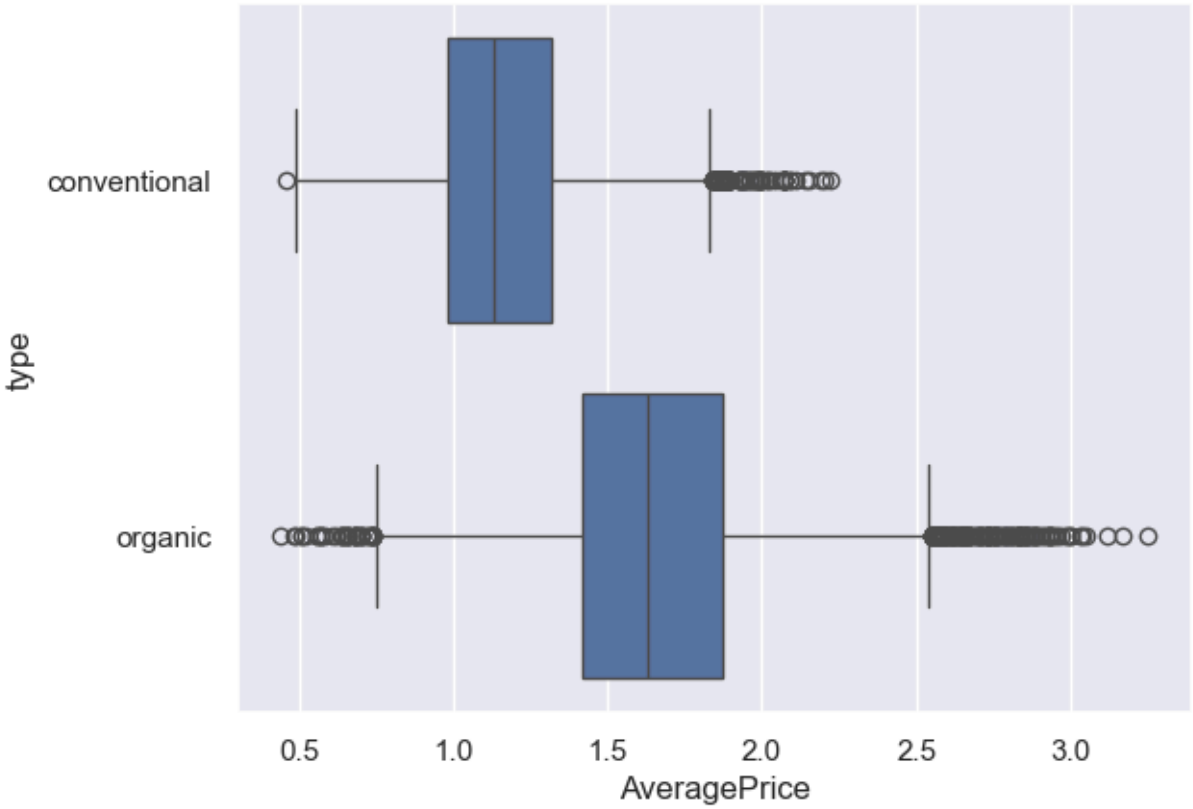
```
sns.countplot(x='year',data=data,hue='type');
```



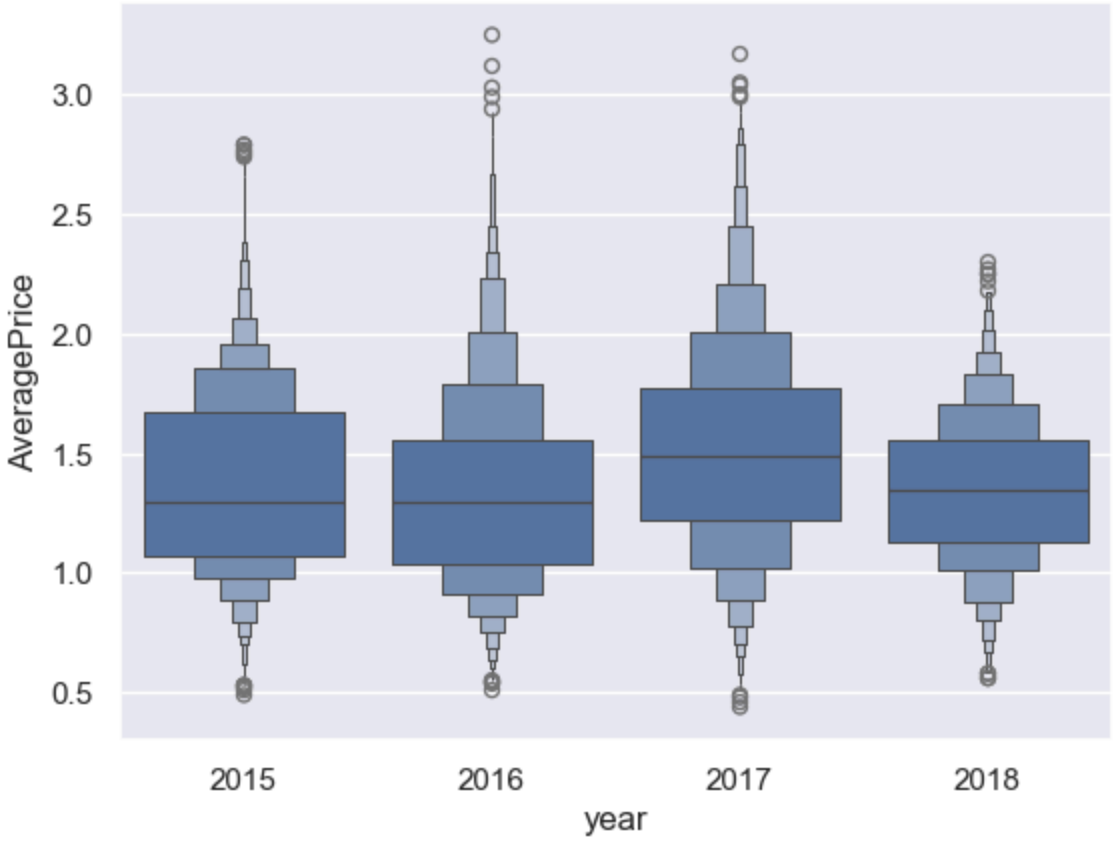
In [102…
```
data.year.value_counts()
```

Out[102…
```
year
2017    5722
2016    5616
2015    5615
2018    1296
Name: count, dtype: int64
```

In [103…
```
sns.boxplot(y="type", x="AveragePrice", data=data);
```

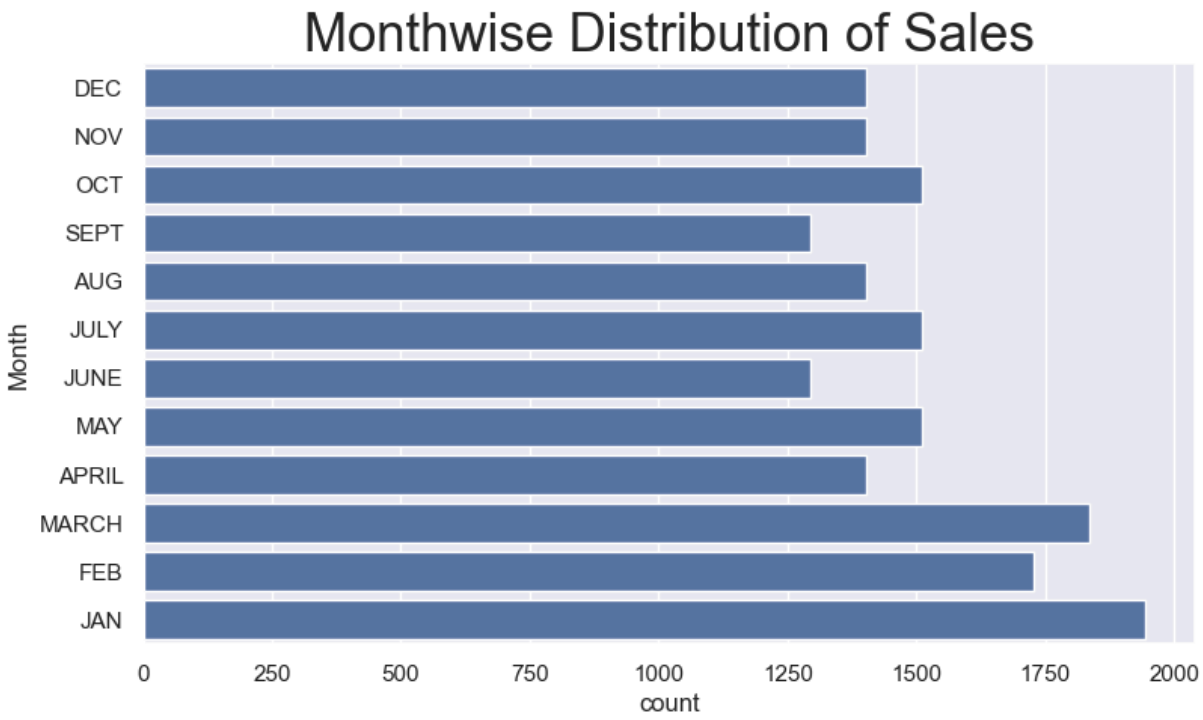


In [104…
```
data.year=data.year.apply(str)
sns.boxenplot(x="year", y="AveragePrice", data=data);
```



In [105…
```
data['type']= data['type'].map({'conventional':0,'organic':1})

# Extracting month from date column.
data.Date = data.Date.apply(pd.to_datetime)
data['Month']=data['Date'].apply(lambda x:x.month)
data.drop('Date',axis=1,inplace=True)
data.Month = data.Month.map({1:'JAN',2:'FEB',3:'MARCH',4:'APRIL',5:'MAY',6:'JUNE',7
```

In [106…
```
plt.figure(figsize=(9,5))
sns.countplot(data['Month'])
plt.title('Monthwise Distribution of Sales',fontdict={'fontsize':25});
```



In [107…
```
# Creating dummy variables
dummies = pd.get_dummies(data[['year','region','Month']],drop_first=True)
df_dummies = pd.concat([data[['Total Volume', '4046', '4225', '4770', 'Total Bags',
       'Small Bags', 'Large Bags', 'XLarge Bags', 'type']],dummies],axis=1)
target = data['AveragePrice']

# Splitting data into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_dummies,target,test_size=0.3

# Standardizing the data
cols_to_std = ['Total Volume', '4046', '4225', '4770', 'Total Bags', 'Small Bags','
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train[cols_to_std])
X_train[cols_to_std] = scaler.transform(X_train[cols_to_std])
X_test[cols_to_std] = scaler.transform(X_test[cols_to_std])
```

In [108…
```
#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 xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [109…
```
#to save time all models can be applied once using for loop
regressors = {
    'Linear Regression' : LinearRegression(),
    'Decision Tree' : DecisionTreeRegressor(),
    'Random Forest' : RandomForestRegressor(),
    'Support Vector Machines' : SVR(gamma=1),
```

```
    'K-nearest Neighbors' : KNeighborsRegressor(n_neighbors=1),
    'XGBoost' : XGBRegressor()
}
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),3),
                          np.round(mean_squared_error(y_test,pred),3),
                          np.round(r2_score(y_test,pred),3)
                         ]
```

In [112…

```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Split train set into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.20,

# Create the model
model = Sequential([
    Dense(76, activation='relu',
          kernel_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)
          bias_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)),

    Dense(200, activation='relu',
          kernel_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)
          bias_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)),

    Dropout(0.5),

    Dense(200, activation='relu',
          kernel_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)
          bias_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)),

    Dropout(0.5),

    Dense(200, activation='relu',
          kernel_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)
          bias_initializer=tf.random_uniform_initializer(minval=-0.1, maxval=0.1)),

    Dropout(0.5),

    Dense(1)  # output layer
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', mode='min', patience=10, verbose=1)
```

In [119…

```
# For features
X_train = X_train.apply(pd.to_numeric)
X_val = X_val.apply(pd.to_numeric)

# For target
y_train = pd.to_numeric(y_train)
y_val = pd.to_numeric(y_val)
```

In [120…

```
history = model.fit(
    x=X_train.values.astype(float),
    y=y_train.values.astype(float),
    validation_data=(X_val.values.astype(float), y_val.values.astype(float)),
    batch_size=100,
    epochs=150,
    callbacks=[early_stop]
)
```

```
Epoch 1/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 6s 20ms/step - loss: 0.3293 - val_loss: 0.1119
Epoch 2/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.1256 - val_loss: 0.0876
Epoch 3/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 24ms/step - loss: 0.1054 - val_loss: 0.0680
Epoch 4/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 19ms/step - loss: 0.0917 - val_loss: 0.0537
Epoch 5/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - loss: 0.0854 - val_loss: 0.0524
Epoch 6/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 24ms/step - loss: 0.0764 - val_loss: 0.0541
Epoch 7/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 18ms/step - loss: 0.0690 - val_loss: 0.0425
Epoch 8/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0661 - val_loss: 0.0408
Epoch 9/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - loss: 0.0621 - val_loss: 0.0435
Epoch 10/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 0.0590 - val_loss: 0.0436
Epoch 11/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0559 - val_loss: 0.0398
Epoch 12/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 0.0562 - val_loss: 0.0349
Epoch 13/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 0.0510 - val_loss: 0.0393
Epoch 14/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0500 - val_loss: 0.0335
Epoch 15/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0499 - val_loss: 0.0311
Epoch 16/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 16ms/step - loss: 0.0471 - val_loss: 0.0326
Epoch 17/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - loss: 0.0448 - val_loss: 0.0315
Epoch 18/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 24ms/step - loss: 0.0453 - val_loss: 0.0287
Epoch 19/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0442 - val_loss: 0.0291
Epoch 20/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0421 - val_loss: 0.0305
Epoch 21/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 18ms/step - loss: 0.0413 - val_loss: 0.0307
Epoch 22/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0422 - val_loss: 0.0269
Epoch 23/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0394 - val_loss: 0.0277
Epoch 24/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0383 - val_loss: 0.0349
Epoch 25/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0390 - val_loss: 0.0338
Epoch 26/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0380 - val_loss: 0.0265
Epoch 27/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0365 - val_loss: 0.0282
Epoch 28/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 18ms/step - loss: 0.0360 - val_loss: 0.0261
```

```
Epoch 29/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0355 - val_loss: 0.0327
Epoch 30/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - loss: 0.0352 - val_loss: 0.0247
Epoch 31/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0350 - val_loss: 0.0268
Epoch 32/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 23ms/step - loss: 0.0336 - val_loss: 0.0253
Epoch 33/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 20ms/step - loss: 0.0337 - val_loss: 0.0260
Epoch 34/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0317 - val_loss: 0.0259
Epoch 35/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 18ms/step - loss: 0.0324 - val_loss: 0.0261
Epoch 36/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 18ms/step - loss: 0.0321 - val_loss: 0.0253
Epoch 37/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - loss: 0.0306 - val_loss: 0.0250
Epoch 38/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - loss: 0.0303 - val_loss: 0.0244
Epoch 39/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0297 - val_loss: 0.0281
Epoch 40/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - loss: 0.0287 - val_loss: 0.0248
Epoch 41/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0300 - val_loss: 0.0268
Epoch 42/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0294 - val_loss: 0.0239
Epoch 43/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - loss: 0.0276 - val_loss: 0.0237
Epoch 44/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 16ms/step - loss: 0.0280 - val_loss: 0.0241
Epoch 45/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - loss: 0.0275 - val_loss: 0.0233
Epoch 46/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0263 - val_loss: 0.0231
Epoch 47/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 0.0256 - val_loss: 0.0253
Epoch 48/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - loss: 0.0250 - val_loss: 0.0249
Epoch 49/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - loss: 0.0251 - val_loss: 0.0236
Epoch 50/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0249 - val_loss: 0.0241
Epoch 51/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - loss: 0.0247 - val_loss: 0.0236
Epoch 52/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - loss: 0.0237 - val_loss: 0.0256
Epoch 53/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 0.0246 - val_loss: 0.0242
Epoch 54/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - loss: 0.0247 - val_loss: 0.0244
Epoch 55/150
82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - loss: 0.0242 - val_loss: 0.0238
Epoch 56/150
```

```
82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 16ms/step - loss: 0.0225 - val_loss: 0.0233
Epoch 56: early stopping
```

In [121…
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

In [124…
```
dnn_pred = model.predict(X_test)
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
```

In [123…
```
# Predict on test set
dnn_pred = model.predict(X_test.values.astype(float)).flatten()  # flatten to 1D ar
```

```
172/172 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
```

In [127…
```
results.loc['Deep Neural Network'] = [
    round(mean_absolute_error(y_test, dnn_pred), 3),
    round(mean_squared_error(y_test, dnn_pred), 3),
    round(r2_score(y_test, dnn_pred), 3)
]
```

In [129…
```
f"10% of mean of target variable is {np.round(0.1 * data.AveragePrice.mean(),3)}"
```

Out[129…
```
'10% of mean of target variable is 0.141'
```

In [130…
```
results.sort_values('R2-score',ascending=False).style.background_gradient(cmap='Gre
```

Out[130…

	MAE	MSE	R2-score
XGBoost	0.091000	0.016000	0.900000
Random Forest	0.092000	0.017000	0.890000
K-nearest Neighbors	0.098000	0.023000	0.856000
Deep Neural Network	0.110000	0.024000	0.847000
Support Vector Machines	0.116000	0.027000	0.828000
Decision Tree	0.130000	0.040000	0.746000
Linear Regression	0.182000	0.057000	0.637000