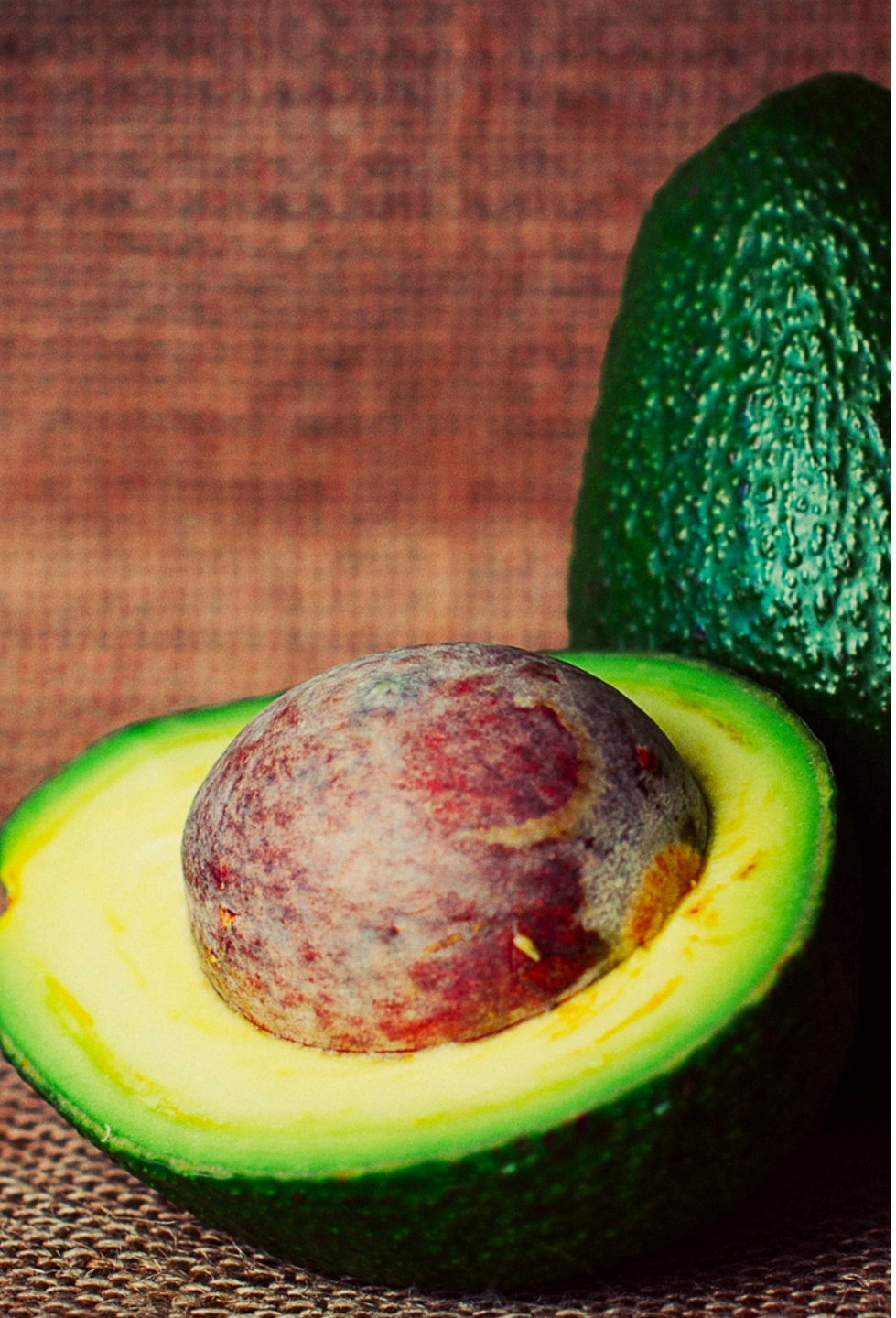
DS 540 Statistical Programming Project

Project in Python

BY

Bhagyashree Chavan



Goal and Task

Problem statement

- Goal is to analyse price with different features of avocado.
- The study focus on which is the best region to supply more avocado and which is the best average price avocado, like conventional or organic
- Analysing data to check total volume in store and sale price in store.
- Finding more about insight sale and try to predict price so that study will provide more strategical solution to marketer of avocado.

PROJECT ON AVOCADO DATASET

Project By Bhagyashree Chavan

In [183]:
```
# These are the following Library using to get our results


import pandas as pd

import matplotlib

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
```

In [184]:
```
# to get know what is our path
import pathlib
pathlib.Path().absolute()
```

Out[184]:
```
PosixPath('/Users/bhagyashreechavan/Desktop')
```

Problem statement on Avocado dataset with focus on which part of region supply more avocado in U.S?

What is best price in which region marketers get in market?

First step is reading file and try to understand data set.

In [236]:
```
# reading file

df = pd.read_csv('avocado.csv')
df
```

Out[236]:

	Unnamed: 0	Date	AveragePrice	Total_Volume	4046	4225	4770	Total_Bags	Small_Bags	Large_Bags	Xlarge_Bags	type	year
0	0	12/27/15	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015
1	1	12/20/15	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015
2	2	12/13/15	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015
3	3	12/6/15	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015
4	4	11/29/15	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18244	7	2/4/18	1.63	17074.83	2046.96	1529.20	0.00	13498.67	13066.82	431.85	0.0	organic	2018

Now checking any missing values in data set with following codes.

In [237]:
```
# checking head of data set
df.head()
```

Out[237]:

	Unnamed: 0	Date	AveragePrice	Total_Volume	4046	4225	4770	Total_Bags	Small_Bags	Large_Bags	Xlarge_Bags	type	year	regio
0	0	12/27/15	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventional	2015	Alban
1	1	12/20/15	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventional	2015	Alban
2	2	12/13/15	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventional	2015	Alban
3	3	12/6/15	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventional	2015	Alban
4	4	11/29/15	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventional	2015	Alban

In [238]:
```
# checking data types
df.dtypes
```

Out[238]:
```
Unnamed: 0        int64
Date             object
AveragePrice    float64
Total_Volume    float64
4046            float64
4225            float64
4770            float64
```

In [239]:
```
# now we will check what is our columns names
df.columns
```

Out[239]:
```
Index(['Unnamed: 0', 'Date', 'AveragePrice', 'Total_Volume', '4046', '4225',
       '4770', 'Total_Bags', 'Small_Bags', 'Large_Bags', 'Xlarge_Bags', 'type',
       'year', 'region'],
      dtype='object')
```

In [240]:
```
#total missing values and count of na's

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

Out[240]:
```
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
```

1]:
```
#description of the data
df.describe
```

1]:
```
<bound method NDFrame.describe of        Unnamed: 0      Date  AveragePrice  Total_Volume     4046
0                0  12/27/15          1.33      64236.62  1036.74   54454.85
1                1  12/20/15          1.35      54876.98   674.28   44638.81
2                2  12/13/15          0.93     118220.22   794.70  109149.67
3                3   12/6/15          1.08      78992.15  1132.00   71976.41
4                4  11/29/15          1.28      51039.60   941.48   43838.39
...            ...       ...           ...           ...      ...        ...
18244            7    2/4/18          1.63      17074.83  2046.96    1529.20
18245            8   1/28/18          1.71      13888.04  1191.70    3431.50
18246            9   1/21/18          1.87      13766.76  1191.92    2452.79
18247           10   1/14/18          1.93      16205.22  1527.63    2981.04
18248           11    1/7/18          1.62      17489.58  2894.77    2356.13

          4770  Total_Bags  Small_Bags  Large_Bags  Xlarge_Bags          type  \
0        48.16     8696.87     8603.62       93.25          0.0  conventional
1        58.33     9505.56     9408.07       97.49          0.0  conventional
2       130.50     8145.35     8042.21      103.14          0.0  conventional
3        72.58     5811.16     5677.40      133.76          0.0  conventional
4        75.78     6183.95     5986.26      197.69          0.0  conventional
...        ...         ...         ...         ...          ...           ...
18244     0.00    13498.67    13066.82      431.85          0.0       organic
18245     0.00     9264.84     8940.04      324.80          0.0       organic
18246   727.94     9394.11     9351.80       42.31          0.0       organic
18247   727.01    10969.54    10919.54       50.00          0.0       organic
18248   224.53    12014.15    11988.14       26.01          0.0       organic
```

- Now here when we apply describing code then in the result, we will get only those information who has data type in 'int64' or 'float'.
- we won't get result for object format.

```
18246  2018  WestTexNewMexico
18247  2018  WestTexNewMexico
18248  2018  WestTexNewMexico

[18249 rows x 14 columns]>
```

In [242]:
```
# if we use parenthesis then we will get same data in tabulaer format with statistics
df.describe()
```

Out[242]:

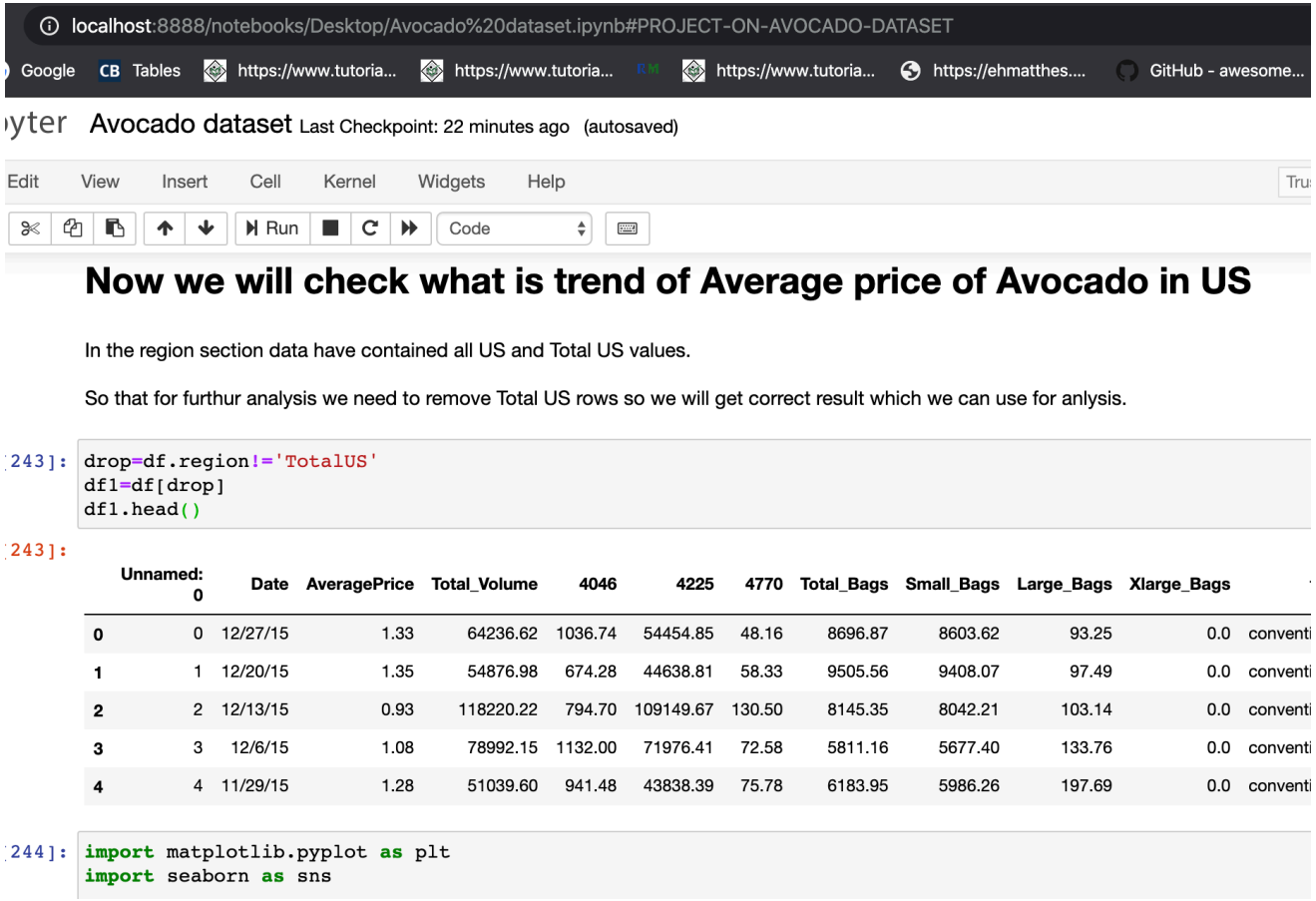
	Unnamed: 0	AveragePrice	Total_Volume	4046	4225	4770	Total_Bags	Small_Bags	Large_Bags	Xlarge_Bags	
count	18249.000000	18249.000000	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	1.824900e+04	18249.000000	18249.
mean	24.232232	1.405978	8.506440e+05	2.930084e+05	2.951546e+05	2.283974e+04	2.396392e+05	1.821947e+05	5.433809e+04	3106.426507	2016.
std	15.481045	0.402677	3.453545e+06	1.264989e+06	1.204120e+06	1.074641e+05	9.862424e+05	7.461785e+05	2.439660e+05	17692.894652	0.
min	0.000000	0.440000	8.456000e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	2015.
25%	10.000000	1.100000	1.083858e+04	8.540700e+02	3.008780e+03	0.000000e+00	5.088640e+03	2.849420e+03	1.274700e+02	0.000000	2015.
50%	24.000000	1.370000	1.073768e+05	8.645300e+03	2.906102e+04	1.849900e+02	3.974383e+04	2.636282e+04	2.647710e+03	0.000000	2016.
75%	38.000000	1.660000	4.329623e+05	1.110202e+05	1.502069e+05	6.243420e+03	1.107834e+05	8.333767e+04	2.202925e+04	132.500000	2017.
max	52.000000	3.250000	6.250565e+07	2.274362e+07	2.047057e+07	2.546439e+06	1.937313e+07	1.338459e+07	5.719097e+06	551693.650000	2018.

Now here when we apply describe code then in the result we will get only those information who has data type in 'int64' or 'float'.

we won't get result for object format.

Notebook result

- In the region section data have contained all US and Total US values.
- So that for further analysis we need to remove Total US rows so we will get correct result which we can use for analysis.



	Unnamed: 0	Date	AveragePrice	Total_Volume	4046	4225	4770	Total_Bags	Small_Bags	Large_Bags	Xlarge_Bags	
0	0	12/27/15	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25	0.0	conventi
1	1	12/20/15	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49	0.0	conventi
2	2	12/13/15	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conventi
3	3	12/6/15	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conventi
4	4	11/29/15	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conventi

Trend of Average price of Avocado

jupyter **Avocado dataset** Last Checkpoint: 30 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

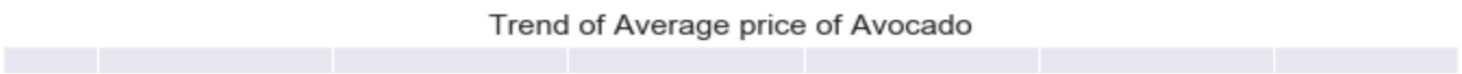
2	2	12/13/15	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14	0.0	conv
3	3	12/6/15	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76	0.0	conv
4	4	11/29/15	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69	0.0	conv

Now we will check what is trend of Average price of Avocado

For the trend I have used histogram with following Libraries.

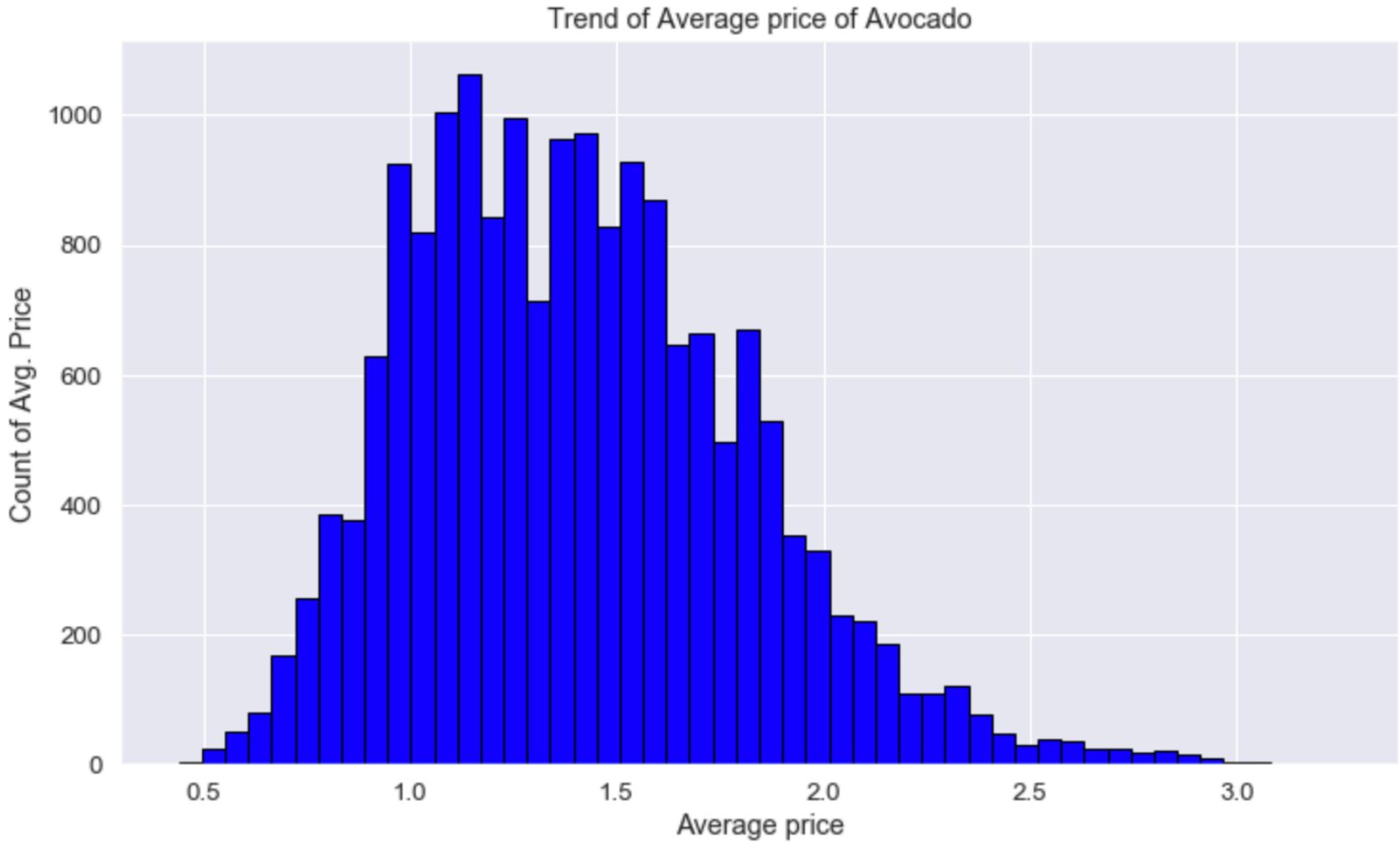
In [265]:
```
import matplotlib.pyplot as plt
import seaborn as sns

#creating histogram
plt.figure()
plt.figure(figsize=(12,7))
plt.hist(df1['AveragePrice'],color = 'blue', edgecolor = 'black', bins= 50)
plt.title('Trend of Average price of Avocado')
plt.xlabel('Average price')
plt.ylabel('Count of Avg. Price')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



```
plt.ylabel( 'Count of Avg. Price' )
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



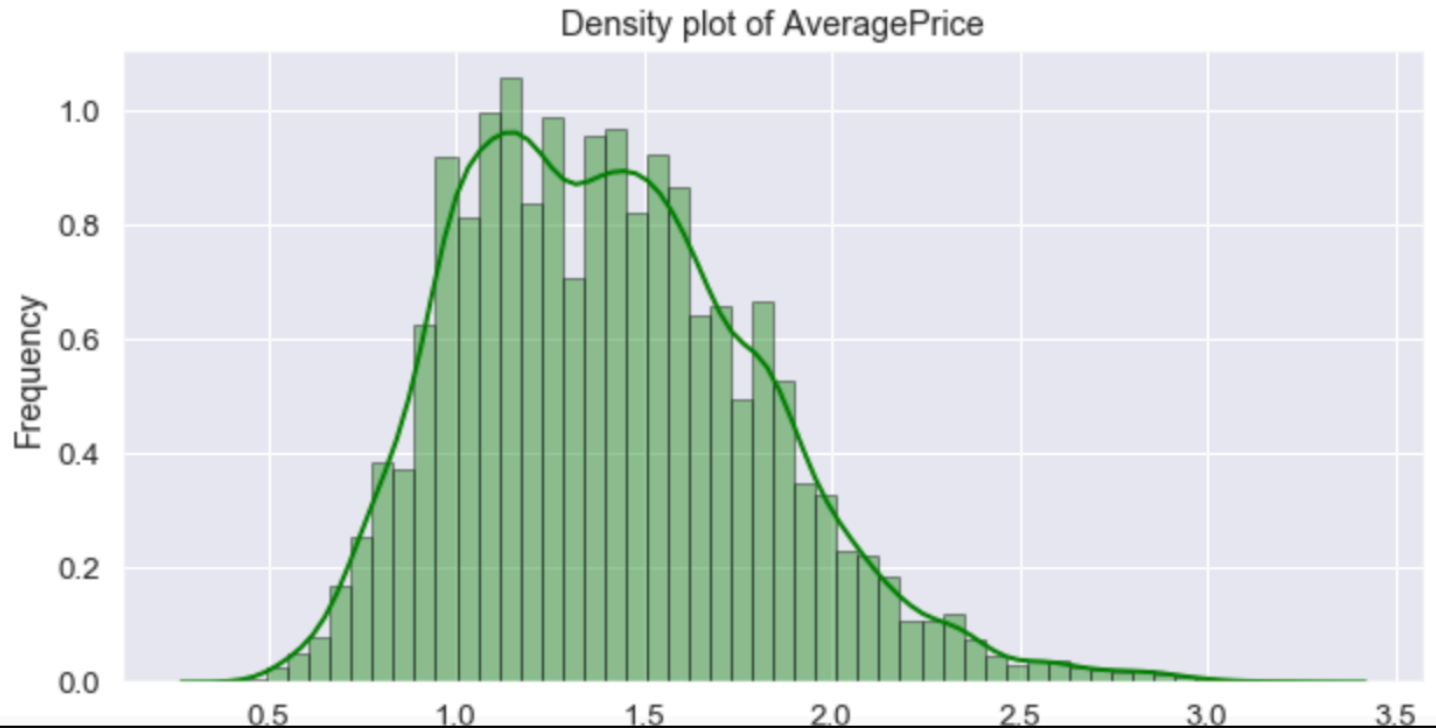
File Edit View Insert Cell Kernel Widgets Help

Run Code

#Now here we are checking what id density of Average price with using seaborne library

In [259]:
```
plt.figure()
plt.figure(figsize=(10,5))
sns.distplot(df1['AveragePrice'],hist = True, kde= True, color = 'green',
             hist_kws ={'edgecolor':'black'},kde_kws={'linewidth':2})
plt.title('Density plot of AveragePrice')
plt.xlabel('AveragePrice')
plt.ylabel('Frequency')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



After getting result of density of average price of avocado is USD 1.3 to USD 1.8

- Now we will understand what is average price for Avocado type ?

- According to see results Average price of conventional Avocado is USD 1.2 and Organic price is USD 1.7

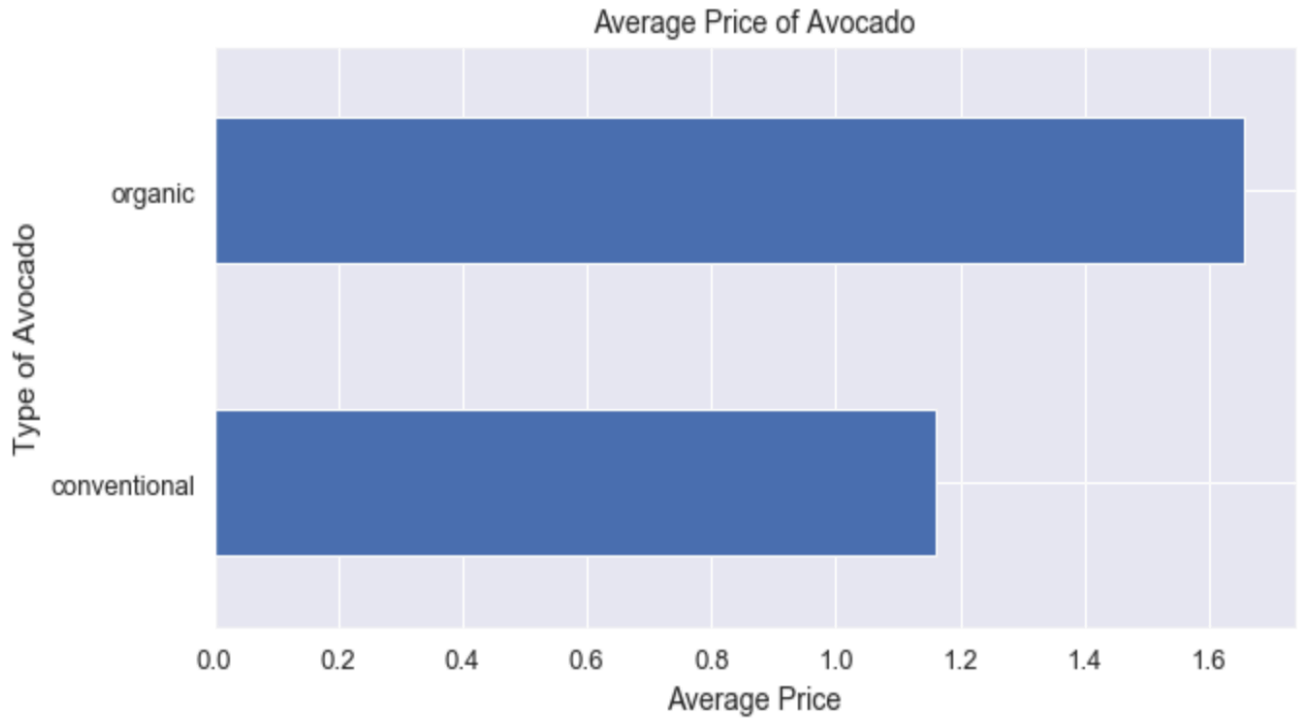
e Edit View Insert Cell Kernel Widgets Help

Run Code

Now we will understand what is average price for Avocado type ?

In [246]:
```
plt.figure()
plt.figure(figsize=(10,5))
df1.groupby('type').AveragePrice.mean().plot(kind='barh')
plt.title('Average Price of Avocado')
plt.xlabel('Average Price')
plt.ylabel('Type of Avocado')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

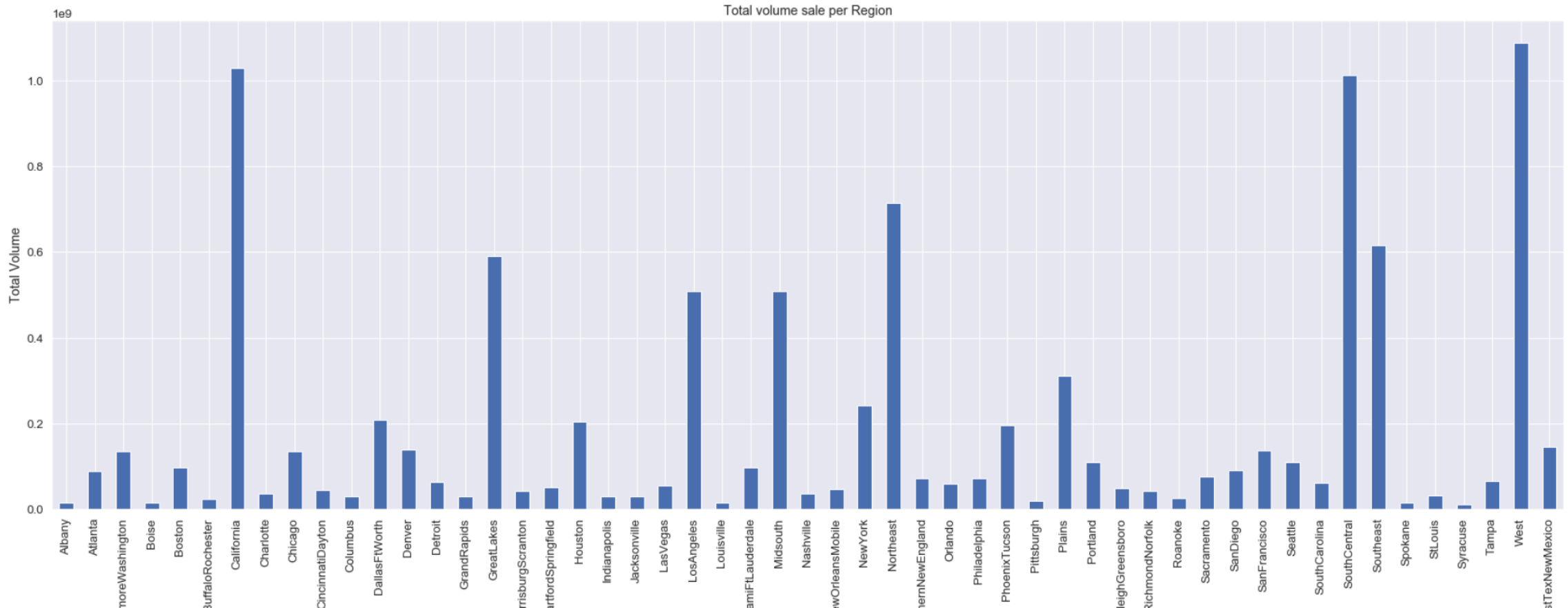


According to see results price of conventional Avocado is USD 1.2 and Organic price is USD 1.7

Now analysing Region of US with Total volume

In [267]:
```
plt.figure()
plt.figure(figsize=(30,10))
df1.groupby('region').Total_Volume.sum().plot(kind='bar')
plt.title('Total volume sale per Region')
plt.xlabel('Region in USA')
plt.ylabel('Total Volume')
plt.show()
```

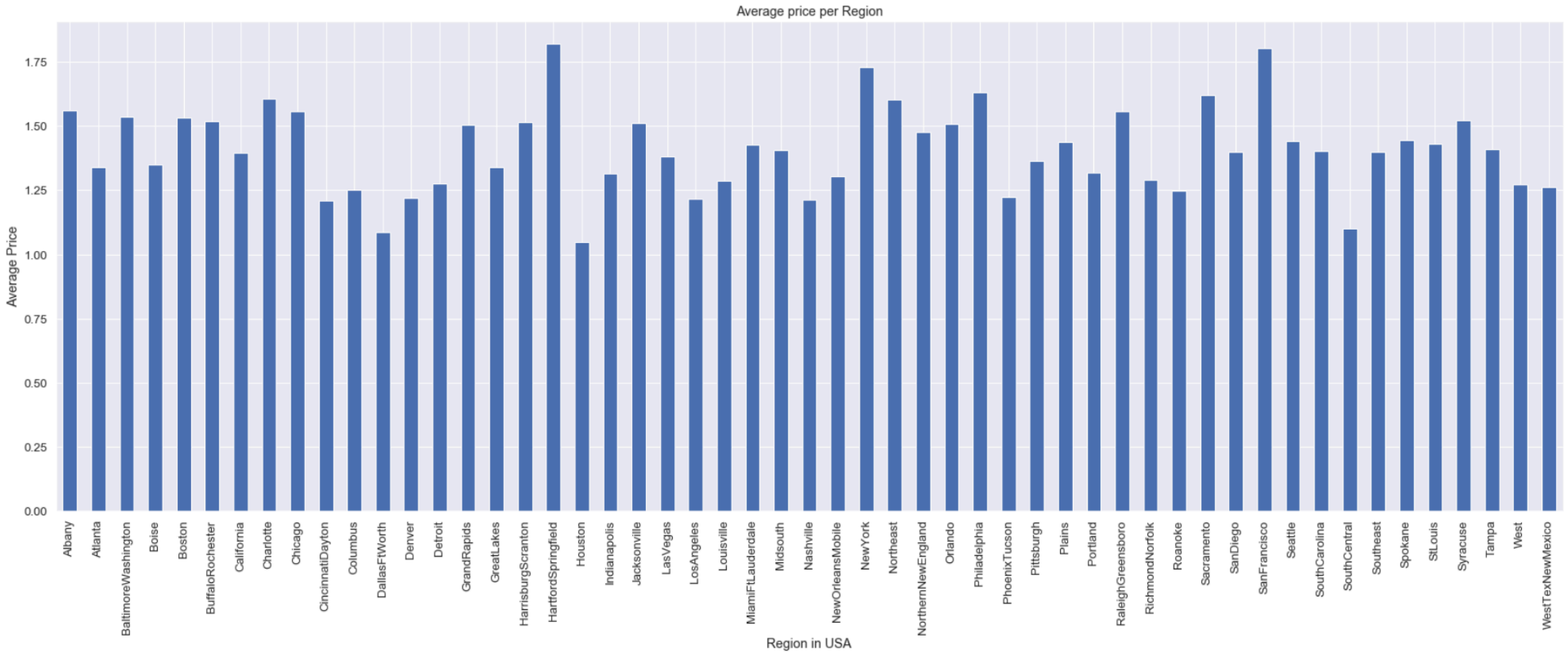
```
<Figure size 432x288 with 0 Axes>
```



Analysis of Regions with Average price of Avocado

In [248]:
```
plt.figure()
plt.figure(figsize=(30,10))
df1.groupby('region').AveragePrice.mean().plot(kind='bar')
plt.title('Average price per Region')
plt.xlabel('Region in USA')
plt.ylabel('Average Price')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



- So that in conclusion for best 5 region where the marketers, producers can supply more in order to get profit Top 5 regions are following:
- Hartforspringfield
- San Francisco
- New York
- Philadelphia
- Sacramento

jupyter Avocado dataset Last Checkpoint: 13 minutes ago (autosaved)

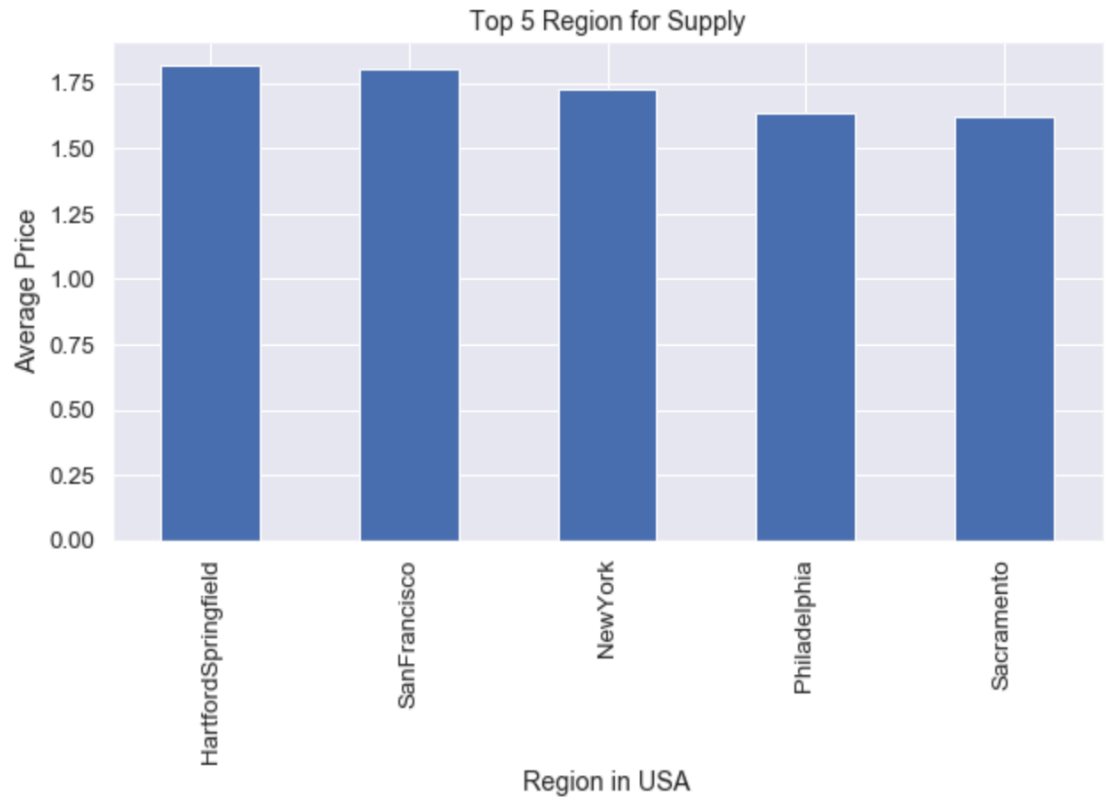
File Edit View Insert Cell Kernel Widgets Help

Run Code

Top 5 region where best price

In [272]:
```
plt.figure()
plt.figure(figsize=(10,5))
df1.groupby("region").AveragePrice.mean().sort_values(ascending=False)[:5].plot.bar()
plt.title('Top 5 Region for Supply')
plt.xlabel('Region in USA')
plt.ylabel('Average Price')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Top 5 region where the Volume comsumption most

- According to analysis of Total volume result for consumption region are different that best price region
- West
- California
- SouthCentral
- Northeast
- Southeast

In [250]:
```
plt.figure()
plt.figure(figsize=(10,5))
df1.groupby("region").Total_Volume.sum().sort_values(ascending=False)[:5].plot.bar()
plt.title('Top 5 Region as per Volume')
plt.xlabel('Region in USA')
plt.ylabel('Total Volume')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

- After getting the result comparing 2015 to other year sale has been increased.
- But in 2016 to 2017 volume had no changed and then major difference between 2017 to 2018.
- In 2018 Total volume has been increased up to 670k.

jupyter Avocado dataset Last Checkpoint: 14 minutes ago (autosaved)

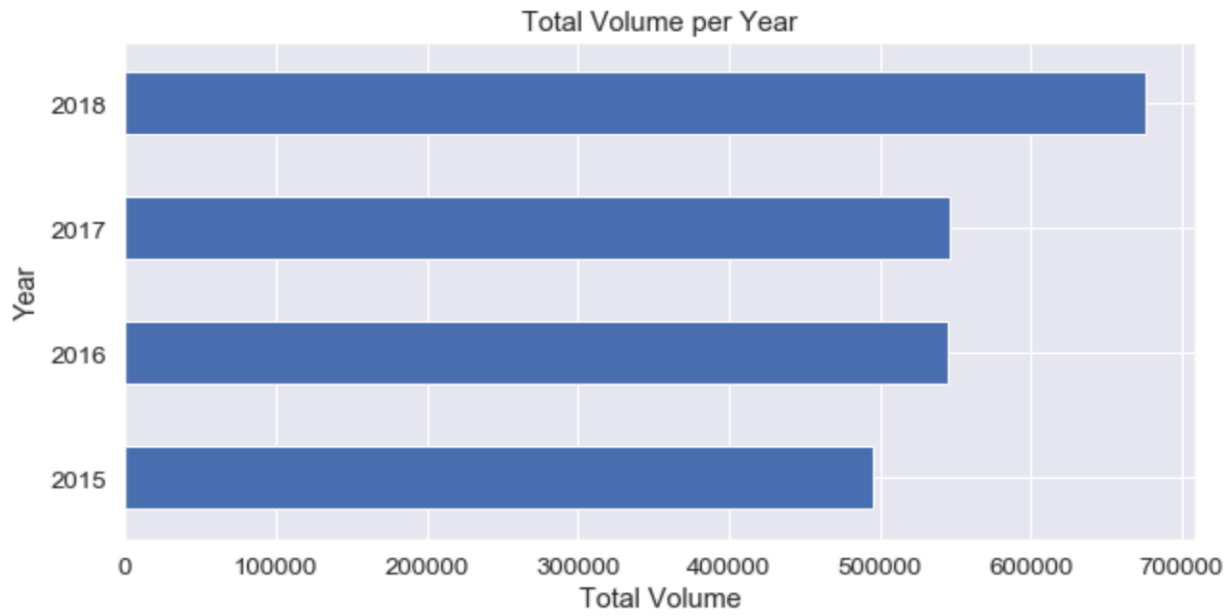
File Edit View Insert Cell Kernel Widgets Help

Run Code

Further more analysis according to year

In [286]:
```
plt.figure()
plt.figure(figsize=(10,5))
df1.groupby("year").Total_Volume.mean().plot.barh()
plt.title('Total Volume per Year')
plt.xlabel('Total Volume')
plt.ylabel('Year')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



After getting the result comparing 2015 to other year sale has been increased.

But in 2016 to 2017 volume had no changedand then major difference between 2017 to 2018.

In 2018 Total volume has been increased upto 670k.

- In graph has shown that price has less in between 2015 to 2016.
- But after starting year 2016 to 2017 price has been reached up to USD 1.5.
- In year 2018 price again slow down and reached in between USD 1.375 to USD 1.350.

jupyter Avocado dataset Last Checkpoint: 14 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

Price according to years

In [287]:
```
plt.figure()
plt.figure(figsize=(10,5))

df1.groupby('year').AveragePrice.mean().plot(kind='line')

plt.title('Average price per Year')
plt.xlabel('Years')
plt.ylabel('Average Price')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

In graph has shown that price has less in between 2015 to 2016.

But after starting year 2016 to 2017 price has been reached upto USD 1.5.

- For Prediction Linear regression model I have used.
- Here I have taking 20% of data to get train or to run our model
- Linear regression model we need to use LinearRegression() command.

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

# here we have taking 20% of data to get train or to run our model
#and random state can be any number we can take but mostly people take 0
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

```
#Now we are apploing Linear regression model in order to get result

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

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=
```

```
#To get intercept:
print(regres.intercept_)


#To get slope:
print(regres.coef_)
```

```
-76.60036402148224
[0.03869152]
```

```
# Here we are assing y pred of x test
y_pred = regres.predict(X_test)
```

```


import numpy as np
import pandas as pd
y_test = np.array(list(y_test))
y_pred = np.array(y_pred)
df4= pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatte
df4.head()
```

File Edit View Insert Cell Kernel Widgets Help Trusted

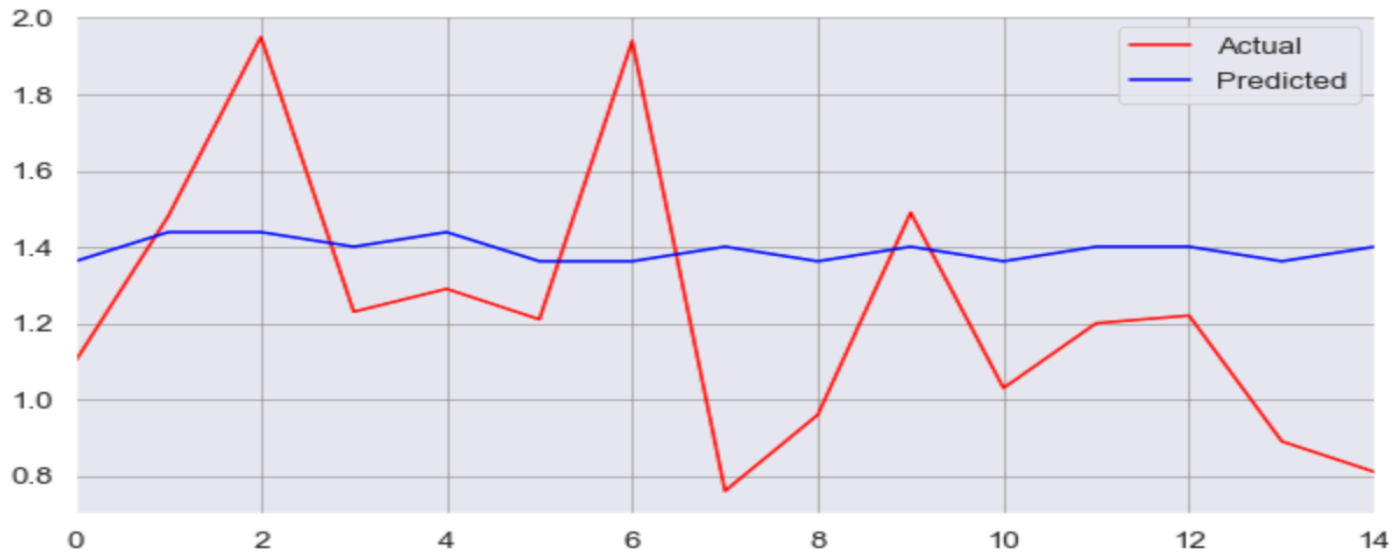
Run Code

Out[412]:

	Actual	Predicted
0	2.07	1.363043
1	1.26	1.440426
2	0.58	1.401734
3	1.43	1.479117
4	1.54	1.440426

In [413]:

```
# here we are considering only 15 results

df4 = df3.head(15)
df4.plot(kind='line',figsize=(10,5),color = ('red','blue'))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='gray')
plt.grid(which='minor', linestyle='dotted', linewidth='0.5', color='black')
plt.show()
```



Predicted Result

In [414]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.3193997805037335
Mean Squared Error: 0.157564993336592
Root Mean Squared Error: 0.3969445721213379
```

Mean square root is 0.3969 more than mean value of the percentage.

In [ ]:

Conclusion

- For supplies and marketers these are best places to get more profit:
- Hartforspringfield, SanFrancisco ,New York, Philadelphia and Sacramento
- For Volume consumption following region are best:
- West , California, SouthCentral , Northeast and Southeast


Thank You