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

Inspiration /Label

The dataset can be seen in two angles to find the region and find the average price .

Task: One of Classification and other of Regression

Do both tasks in the same .ipynb file and submit at single file.

In [14]:

```
#Importing the relevant libraries

import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
scaler = StandardScaler()
from sklearn.model_selection import RepeatedStratifiedKFold

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score


from sklearn import metrics

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
#reading the train datset
df_ = pd.read_csv("https://raw.githubusercontent.com/dsrscientist/Data-Science-
df=df_.drop(["Unnamed: 0"],axis=1)
df=df.dropna()
df
```

Out[2]:

	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Large Bags
0	27-12-2015	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.25
1	20-12-2015	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.49
2	13-12-2015	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.14
3	06-12-2015	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.76
4	29-11-2015	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.69
...	...	...	...	...	...	...	...	...	...
1512	16-10-2016	1.39	190846.01	57529.11	56366.66	17531.78	59418.46	48823.53	10354.65
1513	09-10-2016	1.51	178235.75	43325.87	52189.61	19419.57	63300.70	54704.14	8596.56
1514	02-10-2016	1.48	178410.82	46364.75	52893.38	16736.92	62415.77	53332.61	8258.16
1515	25-09-2016	1.47	189131.52	54110.79	53593.58	17495.42	63931.73	55653.47	8278.26
1516	18-09-2016	1.43	182978.30	43116.41	54193.42	16563.91	69104.56	57456.21	11648.35

1517 rows × 13 columns

In [3]:
```
#Finding out the column informations and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1516
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Date          1517 non-null   object 
 1   AveragePrice  1517 non-null   float64
 2   Total Volume  1517 non-null   float64
 3   4046          1517 non-null   float64
 4   4225          1517 non-null   float64
 5   4770          1517 non-null   float64
 6   Total Bags    1517 non-null   float64
 7   Small Bags    1517 non-null   float64
 8   Large Bags    1517 non-null   float64
 9   XLarge Bags   1517 non-null   float64
 10  type          1517 non-null   object 
 11  year          1517 non-null   float64
 12  region        1517 non-null   object 
dtypes: float64(10), object(3)
memory usage: 165.9+ KB
```

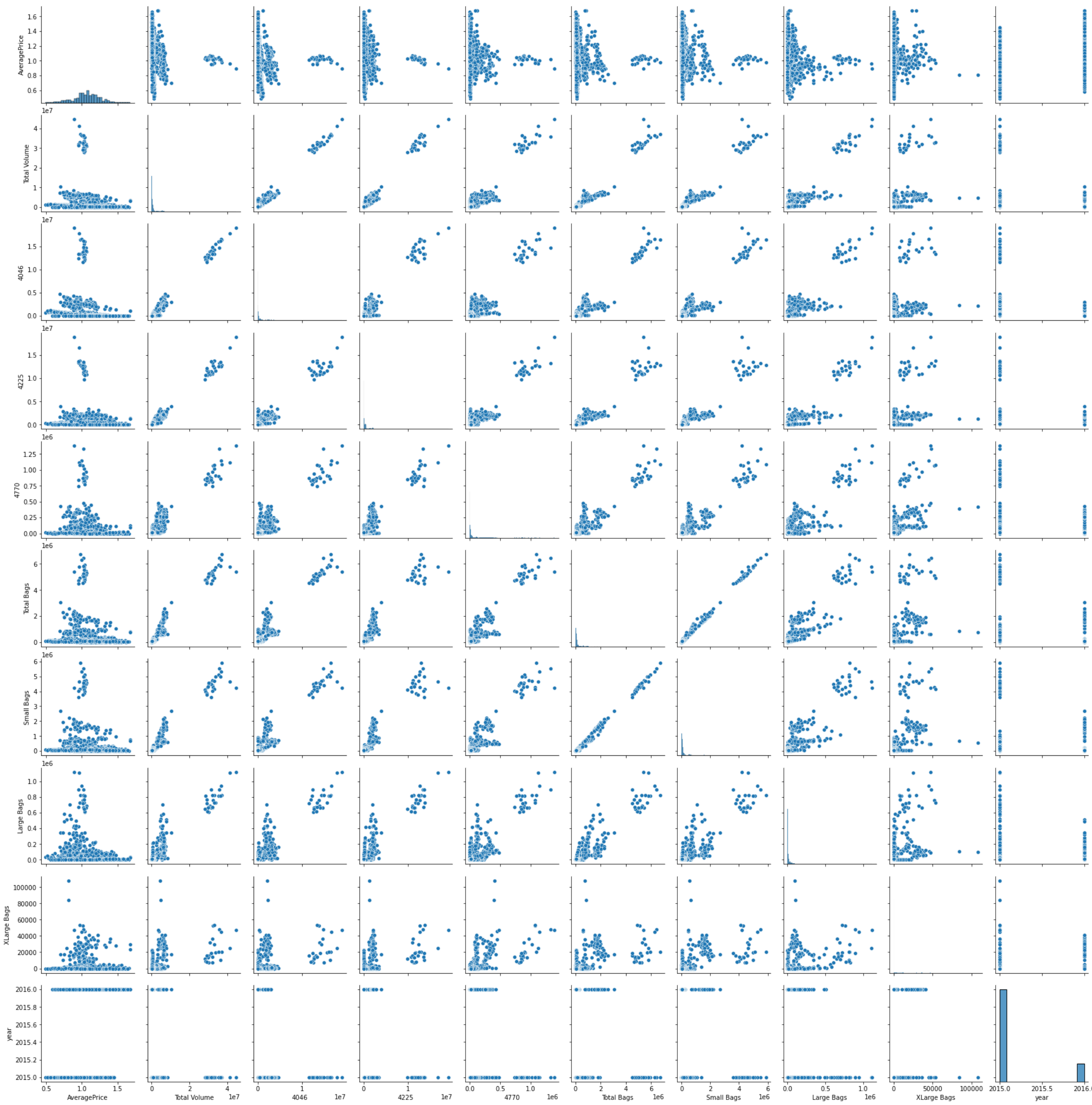
In [4]:
```
#finding out the descriptiove statistics
df.describe()
```

Out[4]:

	AveragePrice	Total Volume	4046	4225	4770	Total Bags	S
count	1517.000000	1.517000e+03	1.517000e+03	1.517000e+03	1.517000e+03	1.517000e+03	1.51
mean	1.074990	1.601879e+06	6.464387e+05	6.114375e+05	5.040550e+04	2.935974e+05	2.48
std	0.188891	4.433143e+06	1.947614e+06	1.672906e+06	1.377812e+05	7.579765e+05	6.47
min	0.490000	3.875074e+04	4.677200e+02	1.783770e+03	0.000000e+00	3.311770e+03	3.31
25%	0.980000	1.474700e+05	2.040034e+04	4.147606e+04	9.112500e+02	3.620689e+04	2.97
50%	1.080000	4.027919e+05	8.175117e+04	1.186649e+05	7.688170e+03	7.397906e+04	6.23
75%	1.190000	9.819751e+05	3.775785e+05	4.851503e+05	2.916730e+04	1.576097e+05	1.46
max	1.680000	4.465546e+07	1.893304e+07	1.895648e+07	1.381516e+06	6.736304e+06	5.89

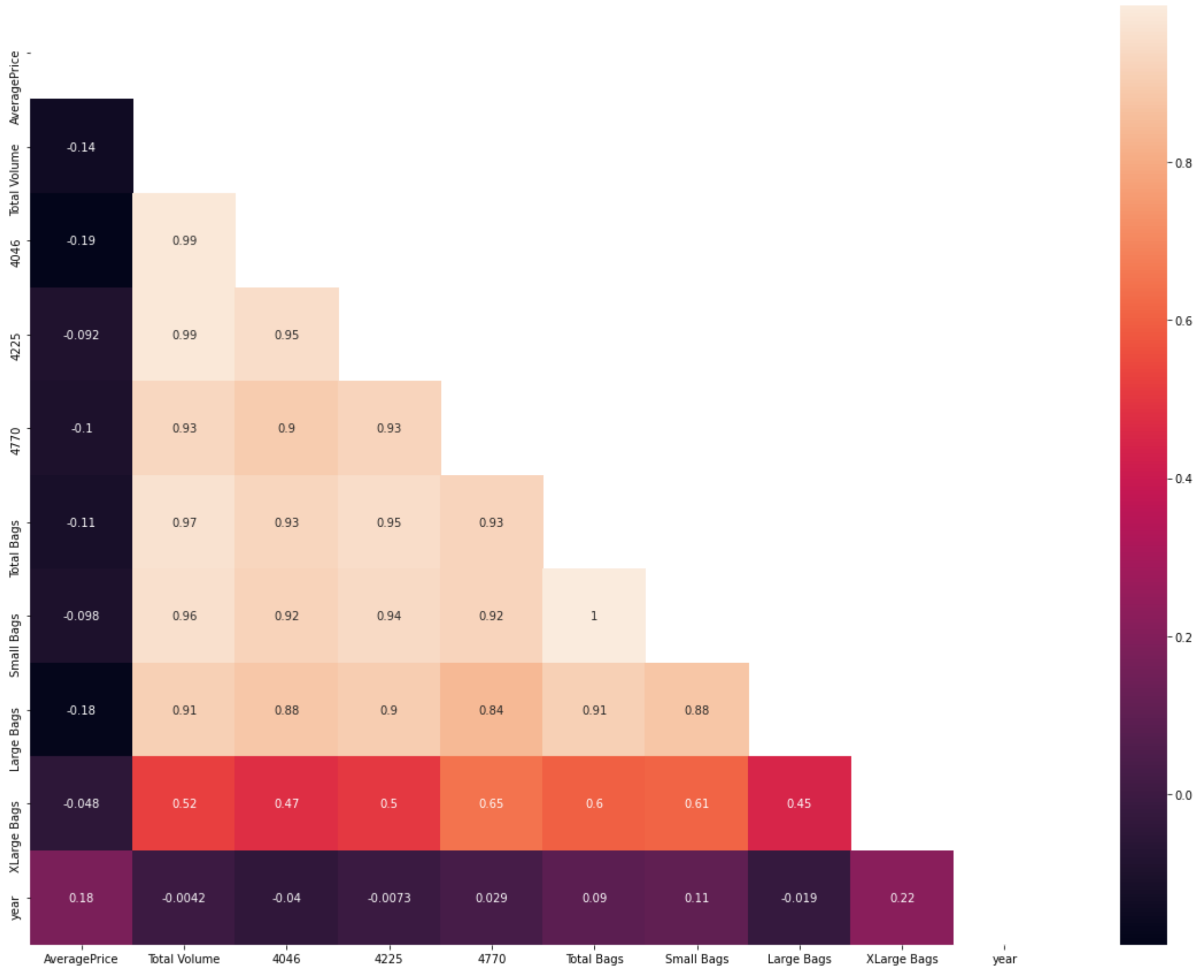
In [5]:
```
#Using Scatter plots to understand relationships between various features
sns.pairplot(df,diag_kind="hist")
```

Out[5]: <seaborn.axisgrid.PairGrid at 0x1e24a3fa580>

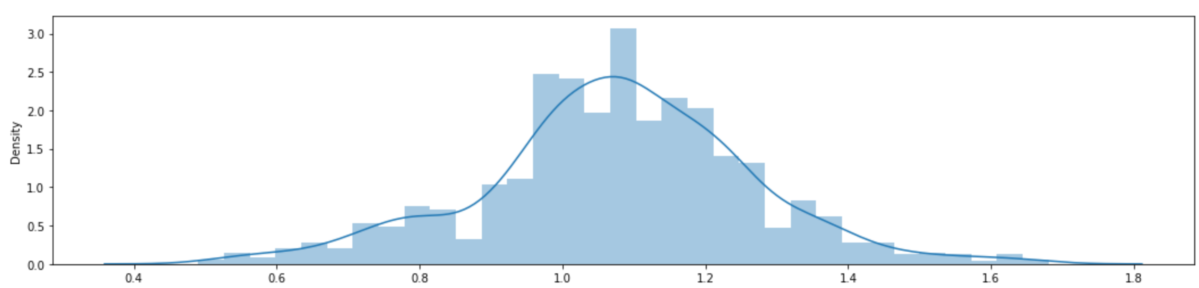


In [6]:
```python
#correlation matrix
plt.figure(figsize=(20,15))
#ax=subplot(111)
matrix = np.triu(df.corr())
sns.heatmap(df.corr(), annot=True, mask=matrix)
```

Out[6]:
```
<AxesSubplot:>
```



In [7]:
```python
#understanding the density distribution is gaussian or not
conventional = df[df['type'] == 'conventional']
organic = df[df['type'] == 'organic']
# df_organic.shape
f, ax = plt.subplots(nrows=1, ncols=1, figsize=(18, 4))
sns.distplot(conventional['AveragePrice']) # histogram
sns.distplot(organic['AveragePrice']) # histogram
plt.show()
```



In [9]:
```
#Feature Engineering an imputations
df['type']= df['type'].map({'conventional':0,'organic':1})

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

In [10]:
```
# Creating dummy variables
dummies = pd.get_dummies(df[['year','region','Month']],drop_first=True)
df_dummies = pd.concat([df[['Total Volume', '4046', '4225', '4770', 'Total Bags
       'Small Bags', 'Large Bags', 'XLarge Bags', 'type']],dummies],axis=1)
target = df['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

# Standardizing the data
cols_to_std = ['Total Volume', '4046', '4225', '4770', 'Total Bags', 'Small Bag
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 [15]:
```
#Spot checking various algorithms for scores
models = []
models.append(('LR',LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('GBM', GradientBoostingRegressor()))
models.append(('XGB', XGBRegressor()))

results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, random_state=21,shuffle=True)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='r2
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.628409 (0.067169)
LASSO: -0.011906 (0.014649)
EN: -0.011906 (0.014649)
KNN: 0.600566 (0.060251)
CART: 0.641610 (0.078379)
GBM: 0.718360 (0.044951)
XGB: 0.805490 (0.037373)
```

In [16]:
```
#Selecting XGBoost as the baseline model
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
model = XGBRegressor()
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)
score = model.score(X_train, y_train)
print('Training Score:', score)
score = model.score(X_test, y_test)
print('Testing Score:', score)
output = pd.DataFrame({'Predicted':Y_pred})
```

```
Training Score: 0.9975604420427843
Testing Score: 0.8170136921359821
```

In [17]:
```
#Finding out the mean absolute error
mae = np.round(mean_absolute_error(y_test,Y_pred),3)
print('Mean Absolute Error:', mae)
```

```
Mean Absolute Error: 0.059
```

In [18]:
```
#Finding out the mean Squarred Error
mse = np.round(mean_squared_error(y_test,Y_pred),3)
print('Mean Squared Error:', mse)
```

```
Mean Squared Error: 0.007
```

In [19]:
```
#Finding out the R2 Score
score = np.round(r2_score(y_test,Y_pred),3)
print('R2 Score:', score)
```

```
R2 Score: 0.817
```

In [ ]: