customer_segments

March 30, 2016

1 Creating Customer Segments

In this project you, will analyze a dataset containing annual spending amounts for internal structure, to understand the variation in the different types of customers that a wholesale distributor interacts with.

Instructions:

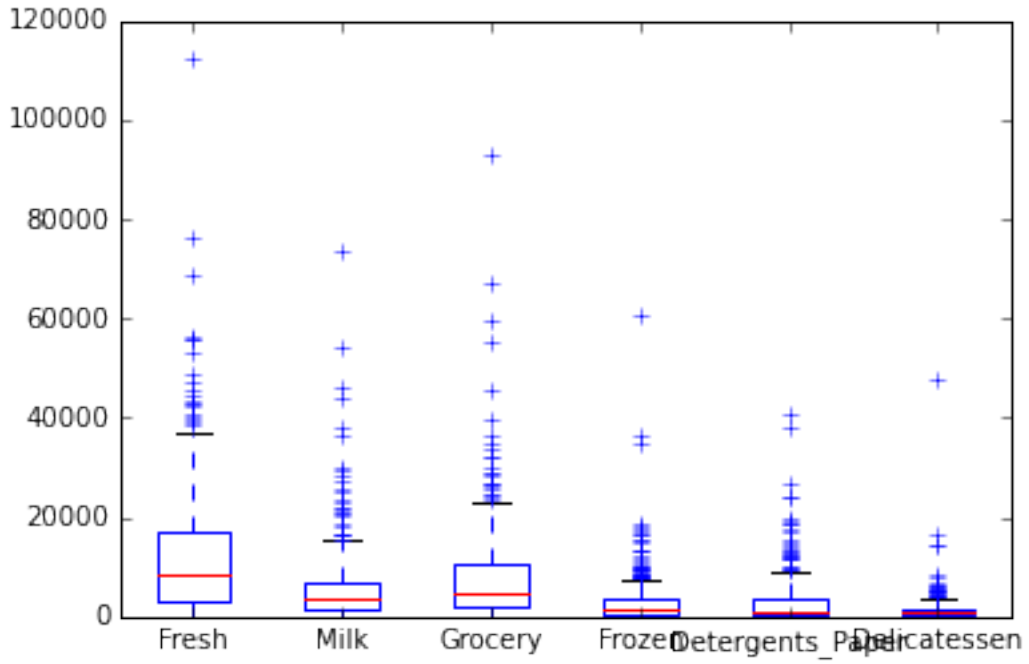
- Run each code block below by pressing **Shift+Enter**, making sure to implement any steps marked with a TODO.
- Answer each question in the space provided by editing the blocks labeled "Answer:".
- When you are done, submit the completed notebook (.ipynb) with all code blocks executed, as well as a .pdf version (File > Download as).

In [1]:
```
import warnings
warnings.filterwarnings('ignore')

# Import libraries: NumPy, pandas, matplotlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Tell iPython to include plots inline in the notebook
%matplotlib inline

# Read dataset
data = pd.read_csv("wholesale-customers.csv")
print "Dataset has {} rows, {} columns".format(*data.shape)
print data.head()  # print the first 5 rows
print data.describe()
data.plot(kind="box")
```

```
Dataset has 440 rows, 6 columns
   Fresh  Milk  Grocery  Frozen  Detergents_Paper  Delicatessen
0  12669  9656     7561     214              2674          1338
1   7057  9810     9568    1762              3293          1776
2   6353  8808     7684    2405              3516          7844
3  13265  1196     4221    6404               507          1788
4  22615  5410     7198    3915              1777          5185
                Fresh          Milk       Grocery        Frozen  \
count      440.000000    440.000000    440.000000    440.000000
mean     12000.297727   5796.265909   7951.277273   3071.931818
std      12647.328865   7380.377175   9503.162829   4854.673333
min          3.000000     55.000000      3.000000     25.000000
25%       3127.750000   1533.000000   2153.000000    742.250000
```

```
50%      8504.000000   3627.000000   4755.500000   1526.000000
75%     16933.750000   7190.250000  10655.750000   3554.250000
max    112151.000000  73498.000000  92780.000000  60869.000000

       Detergents_Paper  Delicatessen
count        440.000000    440.000000
mean        2881.493182   1524.870455
std         4767.854448   2820.105937
min            3.000000      3.000000
25%          256.750000    408.250000
50%          816.500000    965.500000
75%         3922.000000   1820.250000
max        40827.000000  47943.000000
```

Out[1]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7336af6ed0>



1.1 Feature Transformation

1) In this section you will be using PCA and ICA to start to understand the structure of the data. Before doing any computations, what do you think will show up in your computations? List one or two ideas for what might show up as the first PCA dimensions, or what type of vectors will show up as ICA dimensions.

Answer:

PCA helps us to reduce the dimensionality of the data by projecting the higher dimension features to lower dimensions keeping the maximum variance of data possible. Looking at the box plot of data above, PCA should return first dimension corresponding to category "Fresh" and the second dimension corresponding to category "Grocery".

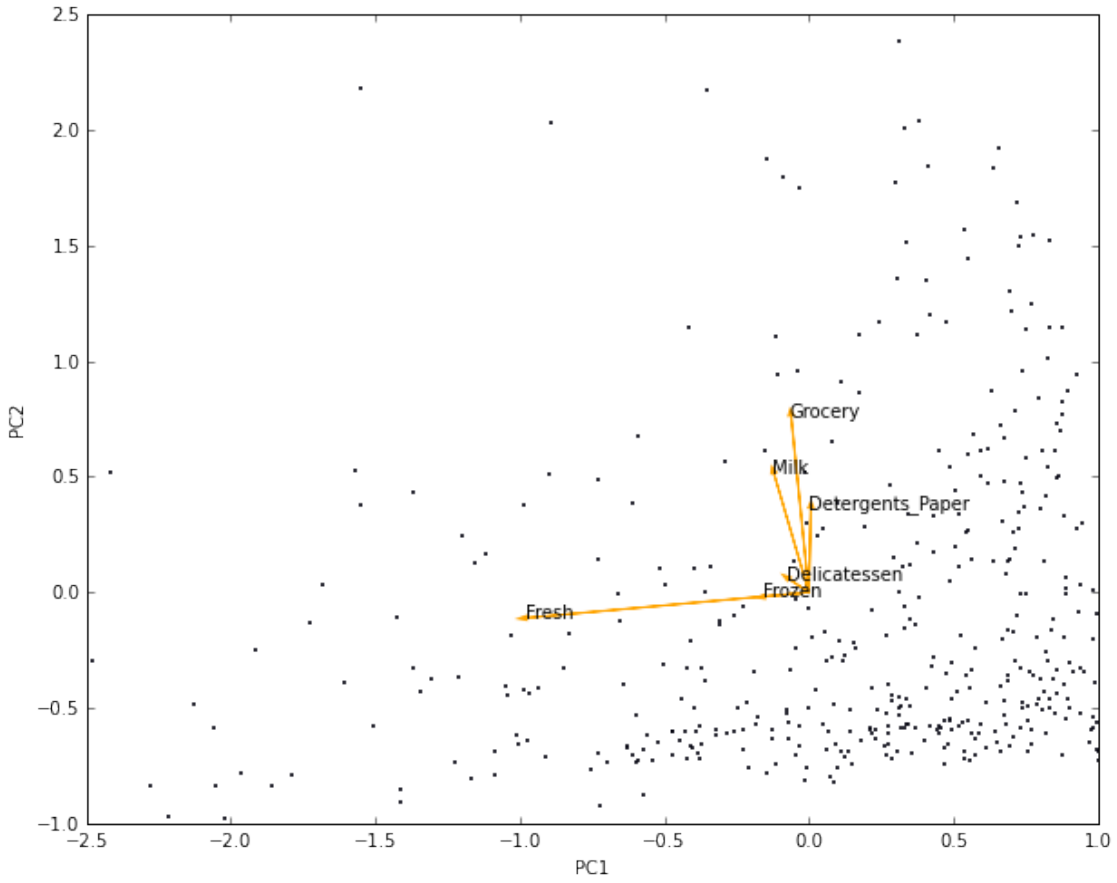
ICA helps to separate source signals from composite signals. In this problem, the amount of money spent on different product categories by 440 customers of a wholesale distributor is given. The types of vectors resulting from ICA will be indicators for different types of customers of the distributor.

1.1.1 PCA

In [2]:
```
# TODO: Apply PCA with the same number of dimensions as variables in the dataset
from sklearn.decomposition import PCA

pca = PCA (n_components=2, whiten=True)

# we don't need scaling here because the input data represents
# annual spending in monetary units and hence are on same scale.

Z = pd.DataFrame(pca.fit_transform(data), columns=["PC1", "PC2"])

# Print the components and the amount of variance in the data contained in each dimension
print pca.components_
print pca.explained_variance_ratio_

ax = Z.plot(kind='scatter', x="PC1", y="PC2", figsize=(10, 8), s=1)
loading = pca.components_

for i, (x,y) in enumerate(zip(loading[0], loading[1])):
    ax.arrow(0, 0, x,y, width=0.001, fc='orange', ec='orange')
    ax.annotate(data.columns[i], (x,y))
ax.set_xlim([-2.5, 1])
ax.set_ylim([-1, 2.5])
```

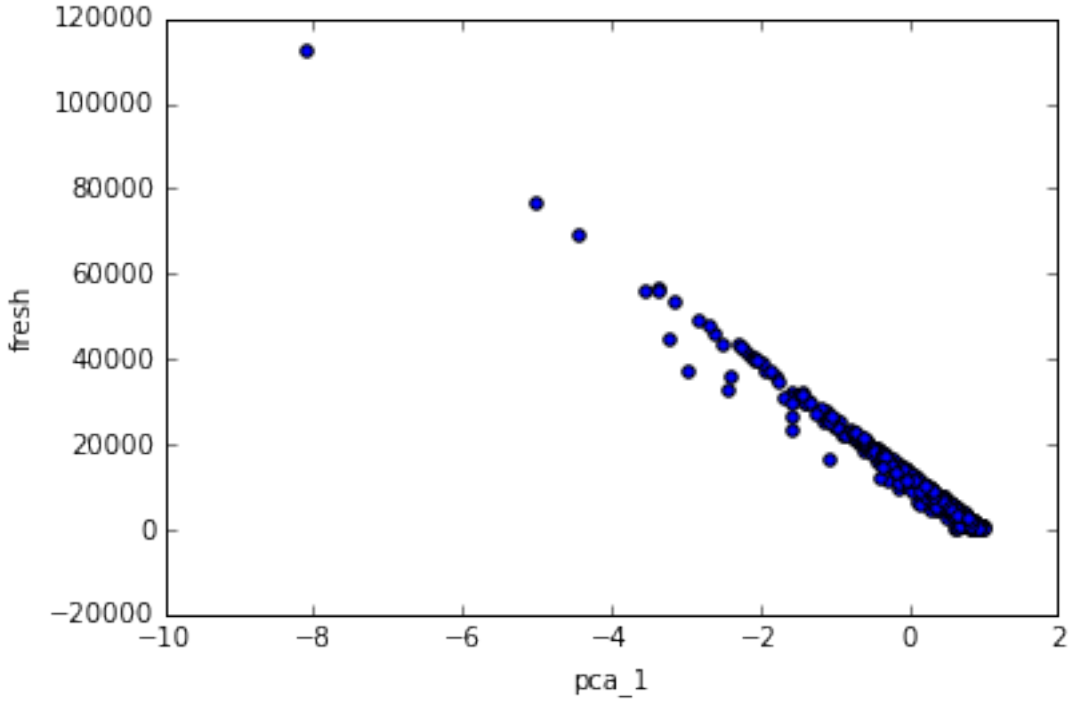
```
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.06810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.05707921]]
[ 0.45961362  0.40517227]
```

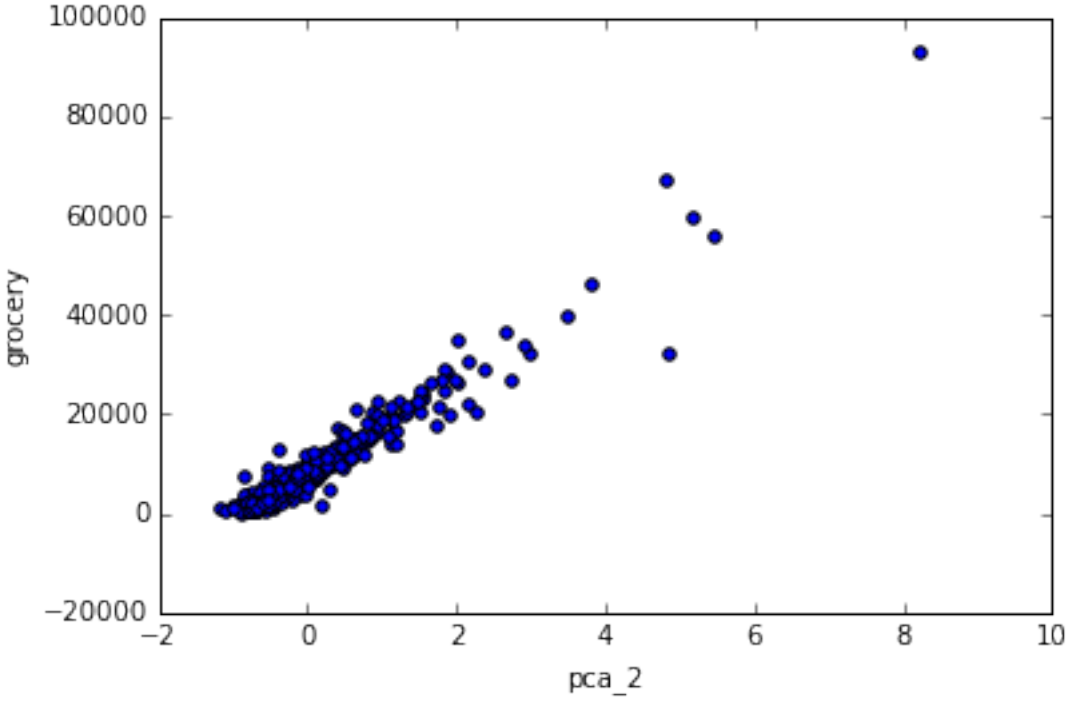
Out[2]: (-1, 2.5)



2) How quickly does the variance drop off by dimension? If you were to use PCA on this dataset, how many dimensions would you choose for your analysis? Why?

```
In [3]: pca = PCA (n_components=6, whiten=True)

        # we don't need scaling here because the input data represents
        # annual spending in monetary units and hence are on same scale.

        Z = pd.DataFrame(pca.fit_transform(data))
        # print the amount of variance explained by each component
        variances = pca.explained_variance_ratio_
        print variances

        np.cumsum(variances)

        pd.DataFrame({'pca_1': Z[:][0], 'fresh': data["Fresh"]}).plot(x='pca_1', y='fresh', kind='scatt
        pd.DataFrame({'pca_2': Z[:][1], 'grocery': data["Grocery"]}).plot(x='pca_2', y='grocery', kind=

[ 0.45961362  0.40517227  0.07003008  0.04402344  0.01502212  0.00613848]

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7333b10690>
```





Answer: Most of the variance is explained by 2 dimensions. For the third dimension, the variance dropped by almost 98% to 0.04. I will use 3 dimensions, because I would like my dimensions to capture atleast 90% of variance in the data.

3) What do the dimensions seem to represent? How can you use this information?

Answer: The dimensions seem to represent the product groups in the decreasing order of importance i.e. the product that is sold the most.

The first dimension output from the PCA algorithm is the loading vector of the first principal component indicating the maximum variance in product category "Fresh" than anything else. The component that follows, is the loading vector of the second principal component that indicates that second maximum variance is along the feature "Grocery". Also, it has significant contributions from Milk and Detergents.

Also, there seems to be some correlation among Frozen foods and Delicatesen i.e. customers buying Frozen food is likely to buy Delicatessen. Another product categories with correlation are Milk, Groceries and Detergent_Paper. Product category Fresh doesn't have any correlation with any other product groups. We can use these information to introduce discounted sales on the correlated product categories to maximize the sales.

Looking at the biplot and the scatter plots above, we can see that first principal component has high negative correlation with Fresh foods (˜-0.97) i.e. it captures the fact that there is high variance in the Fresh foods and also how the volume of Fresh foods purchased reduces with the increase in the first principal component. It is therefore a measure of how Fresh foods is of less favor among high volume customers.

Similary, the second principal component shows a moderate level of positive correlation (˜0.5) with the amount of Groceries bought from the distributor. Also Grocery is a favourite item among the high volume buyers unlike Fresh foods.

1.1.2 ICA

```
In [4]: # TODO: Fit an ICA model to the data
        # Note: Adjust the data to have center at the origin first!
        from sklearn.decomposition import FastICA
        from sklearn import preprocessing

        # scaling the data to align the mean to 0 and to have unit variance
        scaler = preprocessing.StandardScaler()
        data_std = pd.DataFrame(scaler.fit_transform(data))

        ica = FastICA(n_components=6, random_state=42)
        ica.fit(data_std)

        # Print the independent components
        print ica.components_
```

```
[[-0.0109083  -0.00108579  0.00730777  0.05405594 -0.00254136 -0.01675677]
 [ 0.00253788 -0.0123283   0.06912878  0.00142375 -0.01374853 -0.00544097]
 [-0.00490605 -0.00153897 -0.00562146 -0.002525    0.00238444  0.05092947]
 [-0.00336282  0.01863001  0.10899024 -0.00723244 -0.13338644 -0.0160228 ]
 [-0.05026646  0.00647203  0.00748246  0.00322414 -0.01147139  0.0027079 ]
 [-0.00193854 -0.07245463  0.05647623  0.0016736  -0.0171404   0.01695592]]
```

4) For each vector in the ICA decomposition, write a sentence or two explaining what sort of object or property it corresponds to. What could these components be used for?

Answer:

The components can be used for separating out the source signals i.e. the type of customers of the wholesale distributor.

- First component seems to indicate that the customer predominantly sells Frozen foods followed by grocery. It looks like a Deli store. The do very little or none of other product categories.
- Second component indicates a grocery store. They do predominantly grocery followed by some Fresh foods and frozen items. They do a very little Dairy and other stuff.

- Third component indicates Deli store. They mostly sell fine foods and some detergents and paper. They do little or nothing of other product categories.
- Fourth component indicates a Local store. They do a lot of Groceries and Dairy items. Other product categories are not of much prevalent here.

The components output from ICA indicate how each customer weigh different product categories in terms of may be ... space allocated to them, the kind of marketing done and general demand. One potential use of this idea for the distributor is to pre-allocate appropriate portions of different product categories and pack them so that they will be ready to be shipped on demand relatively quickly.

References:

http://whatwhy.in/featured/what-is-the-difference-between-a-hyper-market-super-market-departmental-store-and-a-general-store/457/ https://en.wikipedia.org/wiki/Appetizing_store

Note: Above conclusions were derived from looking for the feature with maximum absolute value and then comparing the remaining features relative to the magnitude and sign of the maximum value.

1.2 Clustering

In this section you will choose either K Means clustering or Gaussian Mixed Models clustering, which implements expectation-maximization. Then you will sample elements from the clusters to understand their significance.

1.2.1 Choose a Cluster Type

5) What are the advantages of using K Means clustering or Gaussian Mixture Models?

Answer:

K-Means clustering or GMM lets us to segregate data points into sub groups within which the data points are mostly identical. In a sense, the advantage is "finding" these subgroups within the data that helps in exploratory analysis. These subgroups can then be used to study the effect of other variables on them helping us make effective decisions using the data.

K-means clustering uses a notion of distance of the data points in order to segregate them. GMM assumes that the observed data is a mixture of probability distributions where each distribution is Gaussian or Normal. It then tries to identify sub groups of data based on probability that a particular data point will land on a distribution.

Comparing KMeans and GMM algorithms, KMeans is scalable with even cluster sizes and works well with flat structures and medium number of clusters. GMM on the other hand is not scalable, also works with flat geometry and good for density estimation purposes. GMM is generally fast to learn mixture models but fails when the number of samples is really high. KMeans is generally good with large scale data and so is going to be the model of choice here.

Looking at the results above, it seems that there are distinct groups of customers and we can go with the hard assignment of clusters using K-means. GMM, does a soft assignment where a particular data point belongs to different clusters with varying degrees of probability. I feel that k-means will work well in this case.

6) Below is some starter code to help you visualize some cluster data. The visualization is based on this demo from the sklearn documentation.

```
In [5]: # Import clustering modules
        from sklearn.cluster import KMeans
        from sklearn.mixture import GMM

In [6]: # TODO: First we reduce the data to two dimensions using PCA to capture variation
        from sklearn.decomposition import PCA
        pca = PCA(n_components = 2, whiten=True)
        reduced_data = pca.fit_transform(data)
        print reduced_data[:10]  # print upto 10 elements
```

```
[[-0.05066239  0.13161505]
 [ 0.34502287  0.33556674]
 [ 0.37738285  0.21406486]
 [-0.07718708 -0.5212911 ]
 [-0.83067886 -0.17928035]
 [ 0.2155776  -0.07967954]
 [ 0.05576966 -0.16710073]
 [ 0.34874672  0.11866355]
 [ 0.52313722 -0.18311407]
 [ 0.37595155  1.11903068]]
```

In [7]:
```
# TODO: Implement your clustering algorithm here, and fit it to the reduced data for visualizat
# The visualizer below assumes your clustering object is named 'clusters'

kmeans = KMeans(n_clusters=4)

clusters = kmeans.fit(reduced_data)
print clusters
```

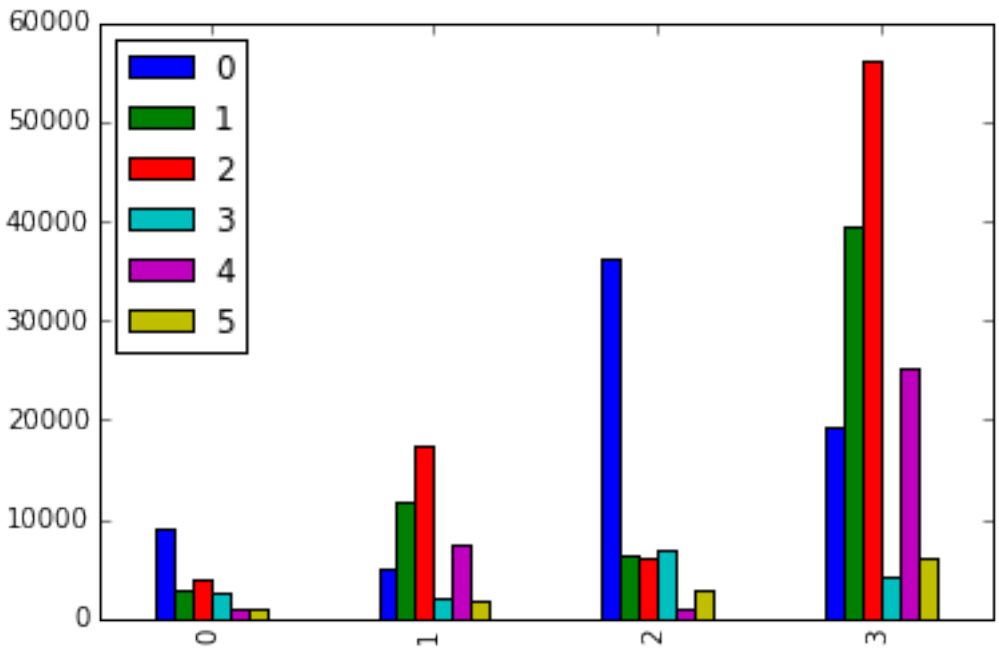
```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=4, n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
```

In [8]:
```
# Plot the decision boundary by building a mesh grid to populate a graph.
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
hx = (x_max-x_min)/1000.
hy = (y_max-y_min)/1000.
xx, yy = np.meshgrid(np.arange(x_min, x_max, hx), np.arange(y_min, y_max, hy))

# Obtain labels for each point in mesh. Use last trained model.
Z = clusters.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [9]:
```
# TODO: Find the centroids for KMeans or the cluster means for GMM

centroids = kmeans.cluster_centers_
print centroids

pd.DataFrame(pca.inverse_transform(centroids)).plot(kind = 'bar')
```

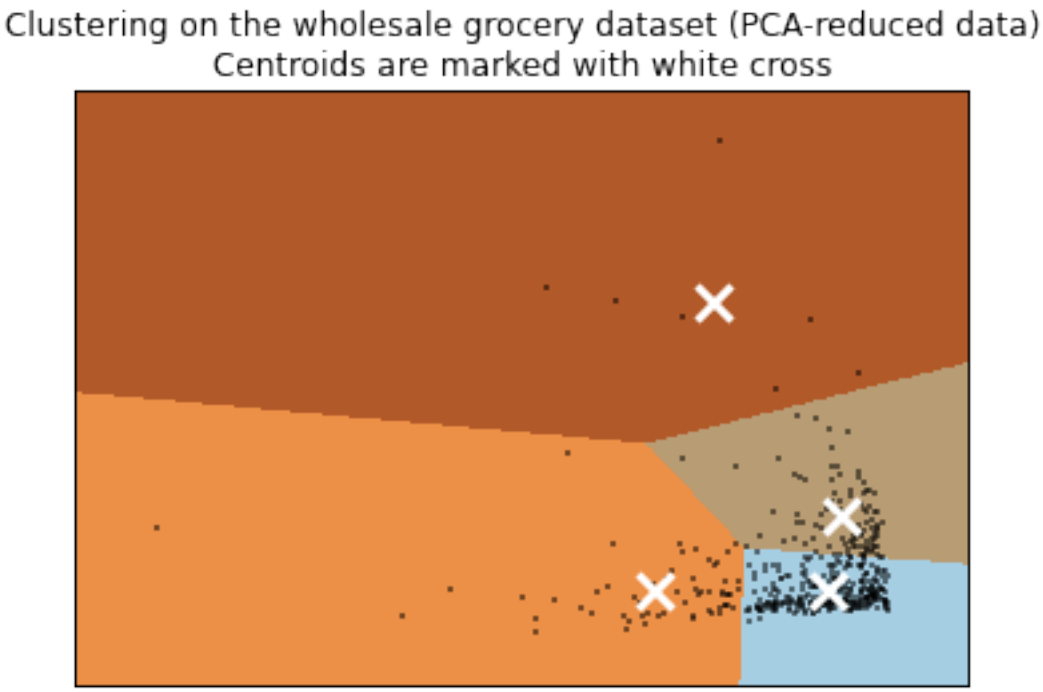
```
[[ 0.27768495 -0.40733618]
 [ 0.44209446  1.05923614]
 [-1.88774966 -0.36229652]
 [-1.13306214  5.12306113]]
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7333666ad0>

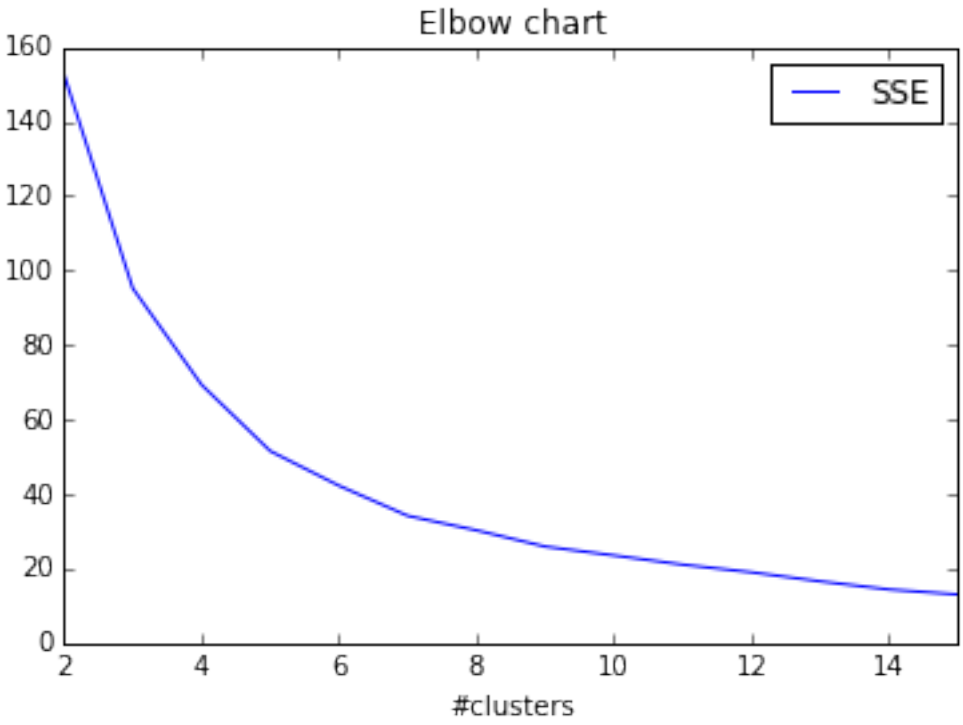


- The customers in first cluster buy Groceries mostly followed by Milk and Detergents.
- In second cluster, the customers buy some Fresh food and small quantities of everything else.
- Third cluster consists predominantly of customers buying large quantities of Fresh food and very small quantities of everything else.
- Fourth cluster customers buy huge quantity of Grocery followed by large quantity of Milk and Detergents. Fresh foods are also their important purchase.

```
In [10]: # Put the result into a color plot
         Z = Z.reshape(xx.shape)
         plt.figure(1)
         plt.clf()
         plt.imshow(Z, interpolation='nearest',
                    extent=(xx.min(), xx.max(), yy.min(), yy.max()),
                    cmap=plt.cm.Paired,
                    aspect='auto', origin='lower')

         plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
         plt.scatter(centroids[:, 0], centroids[:, 1],
                     marker='x', s=169, linewidths=3,
                     color='w', zorder=10)
         plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
                   'Centroids are marked with white cross')
         plt.xlim(x_min, x_max)
         plt.ylim(y_min, y_max)
         plt.xticks(())
         plt.yticks(())
         plt.show()
```




```
In [11]: # using elbow chart to determine the #clusters
         def fitAndScore(num_clusters):
             kmeans = KMeans(n_clusters=num_clusters)

             clusters = kmeans.fit(reduced_data)

             clusters_ = range(num_clusters)
             for i, _ in enumerate(clusters_):
                 clusters_[i] = []

             for l,d in zip(clusters.predict(reduced_data), reduced_data):
                 clusters_[l].append(d)

             mu = kmeans.cluster_centers_
             Wk = sum([np.linalg.norm(mu[k]-c)**2/(2*len(c)) for k in range(num_clusters) for c in clus
             return Wk

         cluster_sizes = range(2,16,1)

         # sum of squared errors
         sse = [fitAndScore(x) for x in cluster_sizes]
         df = pd.DataFrame({'#clusters': cluster_sizes, 'SSE': sse})
         df.head()
         df.plot(x='#clusters', y='SSE', title='Elbow chart')

         # From the chart below, the optimal number of clusters is 4

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f73334f1190>
```



7) What are the central objects in each cluster? Describe them as customers.

Answer:

The central object in bottom left cluster is an average convenience store, the one on bottom right is a Super Market and the one on top is a Grocery store. These customer types were identified by how much of the important product groups they buy relative to the average quantities.

1.2.2 Conclusions

8) Which of these techniques did you feel gave you the most insight into the data?

Answer: Principal Component Analysis. Visualizing first two principal components told the story about which is the important product category to differentiate among customers i.e. the amount of Fresh food that they typically buy. The next differentiating factor is the amount of Grocery bought. Also the correlation coefficients told the story about what high volume customers like to buy than low volume customers. The other techniques help to reinforce the ideas derived from PCA.

9) How would you use that technique to help the company design new experiments?

Answer: Since the data helps us to identify the best selling product groups and not so best selling ones, experiments can be designed around introducing discounted sales on product groups that go well together, stacking the items in the store, promoting the less favorable product groups and so forth.

One A/B test could be, once we identify the imporant product group, we can run a following experiment. Randomly separate the customers into two groups i.e. control and test sets, introduce a sale offer on the most popular product to control group. This will help us to ascertain the importance of the product which can assist in a decision of whether efficient marketing on a already popular product improves sales or not. The experiment can then be repeated with slow moving goods to evaluate what kind of marketing helps in the sales of those products.

10) How would you use that data to help you predict future customer needs?

Answer: Now that we have looked at 440 distinct customers in distinct groups, I guess this information can be used to train a supervised learning classifier to identify the customer group any future customer may fall into. The management of the distributor can focus on setting up tailored product offerings appealing to

each customer groups. Using the focused and tailor-made product offers will result in increased turn around and improved loyalty of the customer.

Also, the purchase history of the customers in each group can be used in the distributor's favor as they can informed decisions about the demands of the new customer. All these advantages result from classifying the customer into a group.