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 [21]:
```
# 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
```

```
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
```

In [22]:

```
feature_list = list(data.columns)
feature_stat = pd.DataFrame(columns=['Minimum', 'Maximum', 'Mean_Value', 'Std_Dev'])

print "============Basic Statistical data from features============="
for feature in feature_list:
    feature_stat.loc[feature] = [np.min(data[feature]),np.max(data[feature]),np.mean(data[feature]),np.std(data[feature])]

print feature_stat
```

```
=============Basic Statistical data from features=============
                  Minimum  Maximum    Mean_Value       Std_Dev
Fresh                   3   112151  12000.297727  12632.948725
Milk                   55    73498   5796.265909   7371.985612
Grocery                 3    92780   7951.277273   9492.357638
Frozen                 25    60869   3071.931818   4849.153520
Detergents_Paper        3    40827   2881.493182   4762.433350
Delicatessen            3    47943   1524.870455   2816.899449
```

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: If we use PCA(n_componets = n), we will obtain n vectors. These vectors have two important properties. First these n vectors are orthogonal each other. It means these vectors are part of "basis of feature data's dimension". The data set's dimension can represent with some basis. These vectors are the 'n' most important basis among the basis. As you can see above, I calculate some basic statiscal data from features. In terms of Primary componont, (standard)deviation is very important because the bigger deviation means the more primary component. So I can predict that Fresh would be an first important feature for PCA. I mean the first basis vector of PCA is close to vector of Fresh. Similarly Grocery feature will impact a lot to second component of PCA.

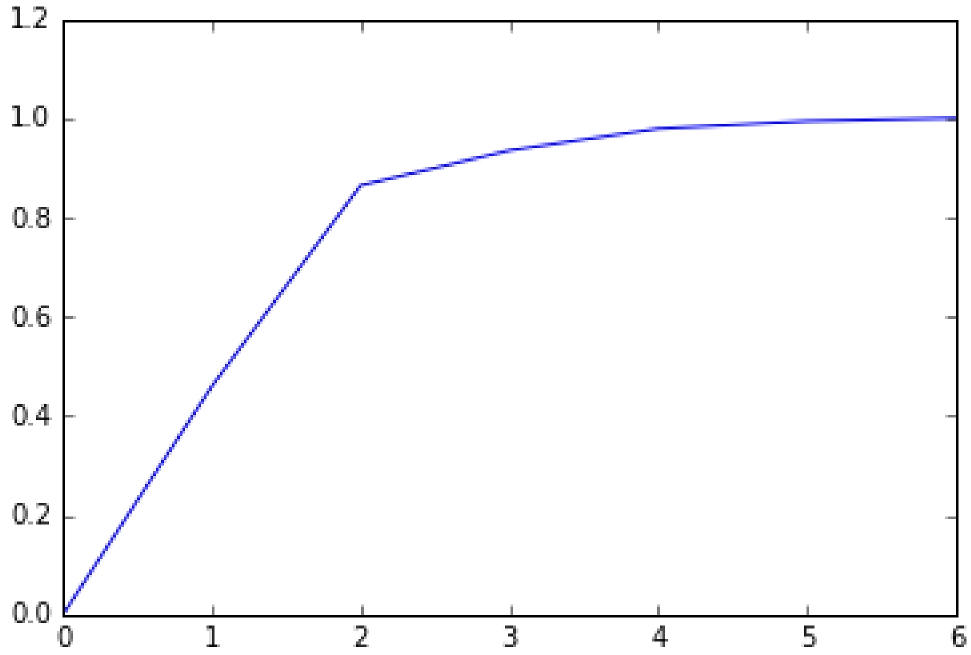
ICA will return independent components set. We have 6 features now. But these are not (statistical) independent features. ICA algorithm will return (statistical) independent features. This means one feature doesn't affect other features value(statistically). For example, When we see original features, if someone buy fresh, then they are prone to buy milk. So the features in original data set is not independent each other. But ICA can find (hidden new) independent features of data set.

For example in this case, Many features can be a candidate of ICA components. Like Store size, Store type, location, or price, ... This is why we call them hidden layer. At first these factors are very hard to find, but usinc ICA we can find them and these are very meaningful.

PCA

In [23]:
```
# TODO: Apply PCA with the same number of dimensions as variables in the data
set
from sklearn.decomposition import PCA
pca = PCA(n_components=6)
pca.fit(data)

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

# Visualize the percent of variance explained
x = np.arange(7)
plt.plot(x, np.cumsum(np.insert(pca.explained_variance_ratio_, 0, 0)), '-')
plt.show()
```

```
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.06810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.05707921]
 [-0.17855726  0.50988675 -0.27578088  0.71420037 -0.20440987  0.28321747]
 [-0.04187648 -0.64564047  0.37546049  0.64629232  0.14938013 -0.02039579]
 [ 0.015986    0.20323566 -0.1602915   0.22018612  0.20793016 -0.91707659]
 [-0.01576316  0.03349187  0.41093894 -0.01328898 -0.87128428 -0.26541687]]
[ 0.45961362  0.40517227  0.07003008  0.04402344  0.01502212  0.00613848]
```



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?

Answer: The value drops really quickly as components go backward. Particularly, the first two components shows large percent of variance explained by that dimension. From third elements, we can see the variance drops dramatically. So based on elbow method, I will choose two components as new eigenvectors. So our new frame would have two dimensions.

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

Answer: Let's talk about two major basis vectors. First one is [-0.97653685 -0.12118407 -0.06154039 -0.15236462 0.00705417 -0.06810471]. As we can see this one is almost similar to firstfeature(Fresh). So this information says that fresh_feature will be a important factor. Second major component is [-0.11061386 0.51580216 0.76460638 -0.01872345 0.36535076 0.05707921]. From this vector, we can check that the weighted sum of milk, grocery and detegernt_paper would be a major factor. we call these new componets 'eigenvectors'. Eigenvectors have given us a much more useful axis to frame the data in. So eigenvectors can be thought as a new imortant features. When we see second eigenvector, we can check that milk, grocery, and detergent got lumped together. It means they are tend to move together, and this amount is important to divide types of customers.

So How can we use this information? we can use it to divide customer segmentations. For simple example we can divide four groups according to their consumption of 'fresh' and 'milk,grocery,detergent'. Let's say A, B, C, D groups. So people in group A rarely buy Fresh, but they really like to buy milk. We can select two different people in A, and we can perform A/B testing! For example providing different type of delivery services to them, we can check which service is better.

ICA

In [24]:

```
# TODO: Fit an ICA model to the data
# Note: Adjust the data to have center at the origin first!
from sklearn.decomposition import FastICA
ica = FastICA(n_components=6, random_state = 5)
ica.fit(data-np.mean(data))

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

```
[[  1.53874502e-07   9.84415888e-06  -5.81376140e-06  -3.63202844e-07
    3.32648293e-06  -6.05563906e-06]
 [ -3.86521701e-07  -2.19551336e-07  -5.99708422e-07  -5.22097463e-07
    5.09161121e-07   1.80922044e-05]
 [ -2.98165594e-07   2.31859174e-06   1.20290960e-05  -1.46456246e-06
   -2.82002691e-05  -5.72900128e-06]
 [  8.65114959e-07   1.40705484e-07  -7.73807907e-07  -1.11462494e-05
    5.53637745e-07   5.95198452e-06]
 [  3.97607953e-06  -8.60599847e-07  -6.33167248e-07  -6.76694970e-07
    2.08016287e-06  -1.03862789e-06]
 [  2.09727144e-07  -1.87724758e-06   6.47877605e-06   4.04429646e-07
   -9.28349244e-07  -1.47958685e-06]]
```

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: Let's see second, third, fourth, and sixth vectors from ICA. Second vector is mostly consisted of delicatessen feature. Third vector is mostly consisted of Detergents. Fourth vector is mostly consisted of Frozen. Grocery feature chiefly consists of the sixth vector.

So if customer has high value of second component, they would like to purchase delicatessen products. If customer has higher value of thir component, they are likely to purchase detergetns. Likewise customer with high value of fourth and sixth components are tend to purchase frozen and grocery respectively. And interesting thing is these tendency is (statistically) independent to each other.

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.

Choose a Cluster Type

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

Answer: First let's talk about K means algorithm.

- (Pros) Simple and fast for data with low dimensionality.
- (Cons) K means cannot discern outliers.

Follwing properties are Gaussian EM models'.

- (Pros) Soft clustering is enabled. (calculate probablity of belonging to each group)
- (Pros) Obtain a density estimation for each cluster (also can discern outliers)
- (Cons) With a large data set, calculation can be slow.

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 [25]:
```
# Import clustering modules
from sklearn.cluster import KMeans
from sklearn.mixture import GMM
```

In [26]:
```
# TODO: First we reduce the data to two dimensions using PCA to capture varia
tion
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(data)

reduced_data = pca.fit_transform(data)
print reduced_data[:10]  # print upto 10 elements
```

```
[[  -650.02212207   1585.51909007]
 [  4426.80497937   4042.45150884]
 [  4841.9987068    2578.762176  ]
 [  -990.34643689  -6279.80599663]
 [-10657.99873116  -2159.72581518]
 [  2765.96159271   -959.87072713]
 [   715.55089221  -2013.00226567]
 [  4474.58366697   1429.49697204]
 [  6712.09539718  -2205.90915598]
 [  4823.63435407  13480.55920489]]
```

In [27]:

```
# TODO: Implement your clustering algorithm here, and fit it to the reduced d
ata for visualization
# The visualizer below assumes your clustering object is named 'clusters'
# implemnt two algorithm (KMean & GMM) so that we can compare two algorithms'
results.
gmm_clusters_7 = GMM(n_components=7)
gmm_clusters = GMM(n_components=6)
gmm_clusters_7.fit(reduced_data)
gmm_clusters.fit(reduced_data)

km_clusters_7 = KMeans(n_clusters=7)
km_clusters = KMeans(n_clusters=6)
km_clusters_7.fit(reduced_data)
km_clusters.fit(reduced_data)
print gmm_clusters
```

```
GMM(covariance_type='diag', init_params='wmc', min_covar=0.001,
  n_components=6, n_init=1, n_iter=100, params='wmc', random_state=None,
  thresh=None, tol=0.001, verbose=0)
```

In [28]:
```
# 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.
gmm_Z_7 = gmm_clusters_7.predict(np.c_[xx.ravel(), yy.ravel()])
gmm_Z = gmm_clusters.predict(np.c_[xx.ravel(), yy.ravel()])
km_Z_7 = km_clusters_7.predict(np.c_[xx.ravel(), yy.ravel()])
km_Z = km_clusters.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [29]:
```
# TODO: Find the centroids for KMeans or the cluster means for GMM
gmm_centroids_7 = gmm_clusters_7.means_
gmm_centroids = gmm_clusters.means_
km_centroids_7 = km_clusters_7.cluster_centers_
km_centroids = km_clusters.cluster_centers_
```

In [30]:
```
# Put the result into a color plot
gmm_Z_7 = gmm_Z_7.reshape(xx.shape)
gmm_Z = gmm_Z.reshape(xx.shape)
km_Z_7 = km_Z_7.reshape(xx.shape)
km_Z = km_Z.reshape(xx.shape)

# plot for GMM algorithm
print "-------------Customer_segmentaion by GMM algorithm--------------"
plt.figure(1)
plt.clf()
plt.imshow(gmm_Z_7, 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(gmm_centroids_7[:, 0], gmm_centroids_7[:, 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()

plt.figure(1)
plt.clf()
plt.imshow(gmm_Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')
```

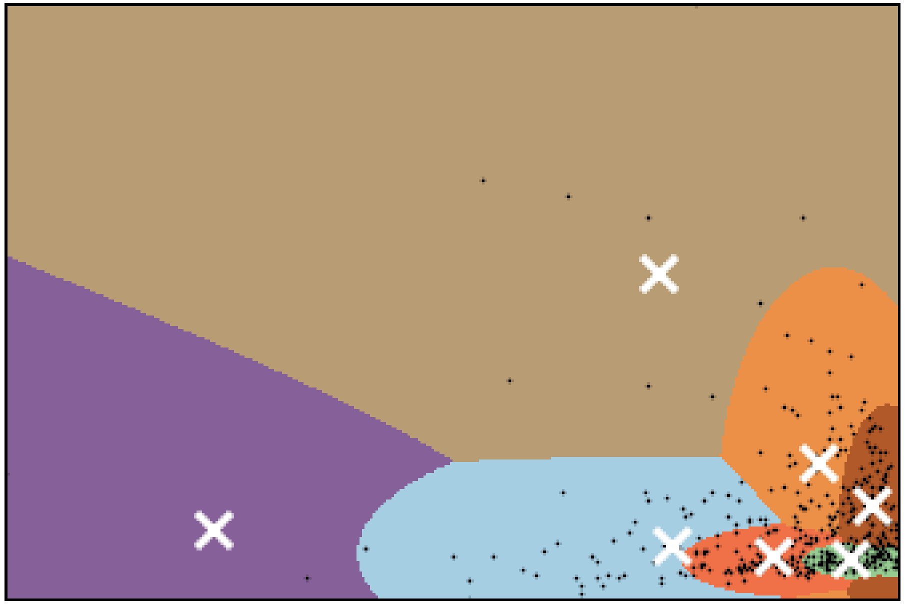
```
plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
plt.scatter(gmm_centroids[:, 0], gmm_centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          'Centroids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

# analyze the centroids coordinate
print pca.inverse_transform(gmm_centroids)
gmm_center_data = pd.DataFrame(pca.inverse_transform(gmm_centroids), columns=
data.columns)
plt.figure(1)
plt.clf()
gmm_center_data.plot(kind = 'bar')
plt.show()
print "----------------------------------------------------------------------"

# plot for KMeans algorithm
print "-------------Customer_segmentaion by KMeans algorithm---------------"
plt.figure(1)
plt.clf()
plt.imshow(km_Z_7, 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)
```

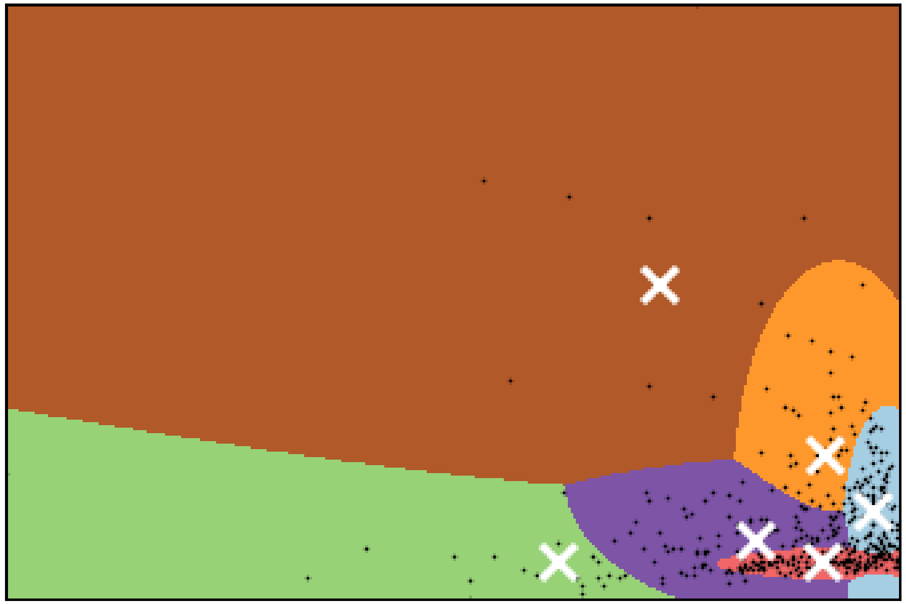
```
plt.scatter(km_centroids_7[:, 0], km_centroids_7[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

plt.figure(1)
plt.clf()
plt.imshow(km_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(km_centroids[:, 0], km_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()

# analyze the centroids coordinate
print pca.inverse_transform(km_centroids)
km_center_data = pd.DataFrame(pca.inverse_transform(km_centroids), columns=da
```

```
ta.columns)
plt.figure(1)
plt.clf()
km_center_data.plot(kind = 'bar')
plt.show()
```

```
-------------Customer_segmentaion by GMM algorithm---------------
```

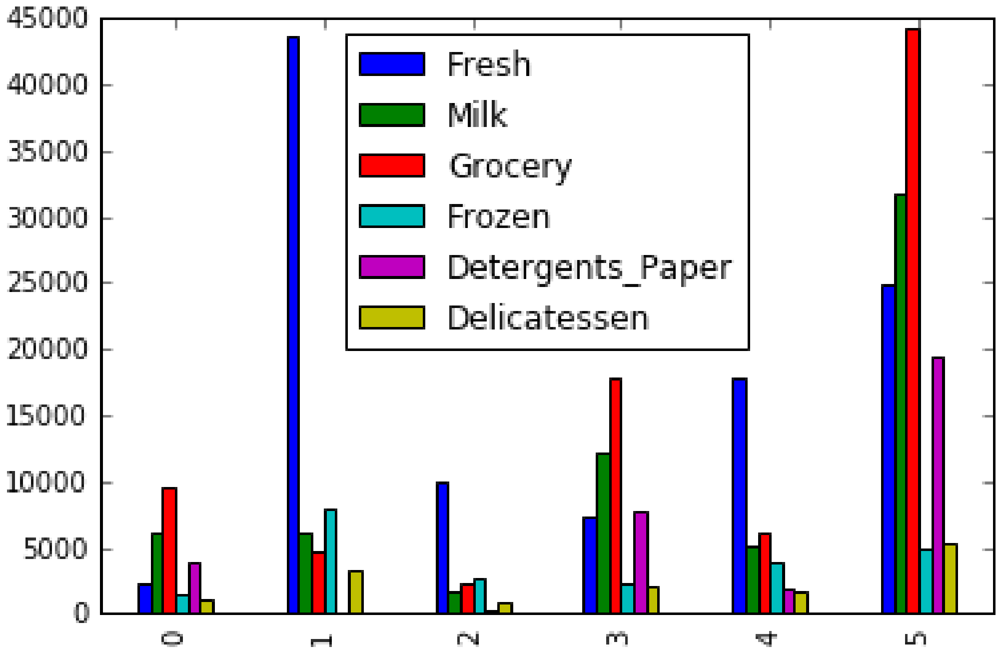
Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross





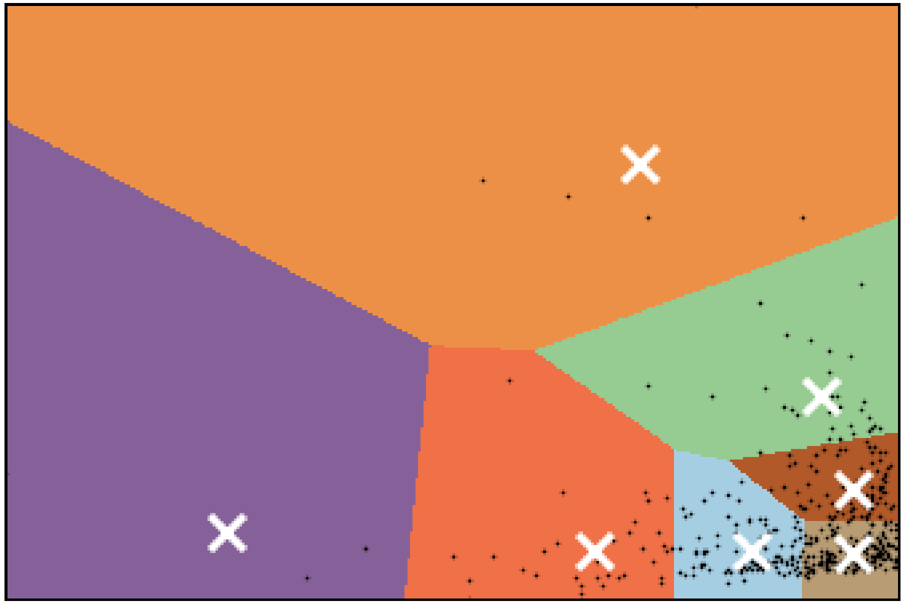

```
[[  2370.9682822    6124.72177668   9564.21582971   1565.2968383
    3999.84010218   1039.71699671]
 [ 43575.59647918   6115.48202315   4696.80414589   8008.43919867
     175.5894889    3286.57119042]
 [  9940.81902601   1741.93161244   2286.31055893   2761.10948999
     281.13002791    916.42096559]
 [  7430.81717654  12264.67595601  17914.69507062   2339.51373834
    7758.03397613   2067.05448529]
 [ 17895.39791117   5047.32170557   6165.54077126   3995.81321111
    1819.66477533   1754.84576693]
 [ 24912.67365842  31699.70180441  44174.11398574   5019.36545396
   19518.17544259   5398.88929234]]
```

```
<matplotlib.figure.Figure at 0x112b03550>
```

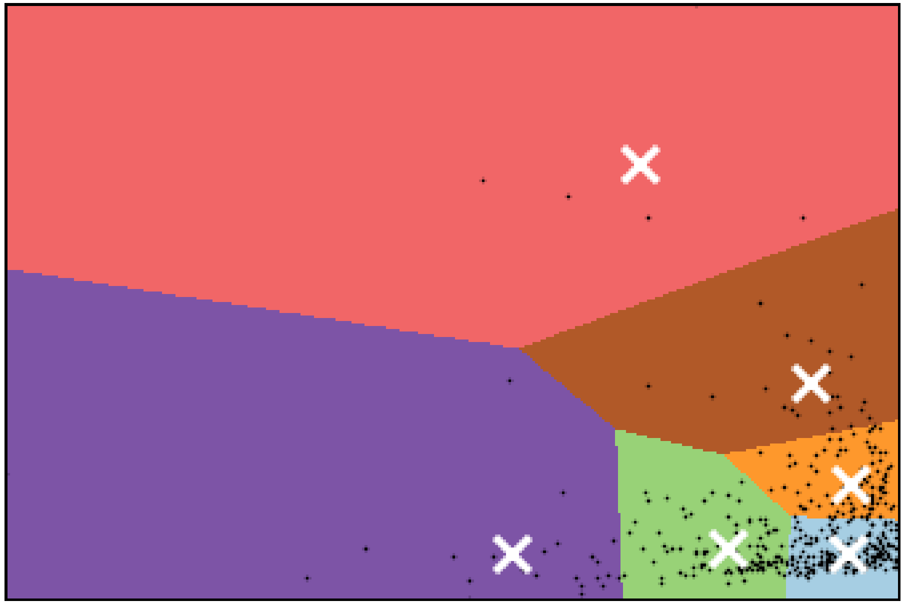



```
-------------------------------------------------------------------
-------------Customer_segmentaion by KMeans algorithm---------------
```

Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross

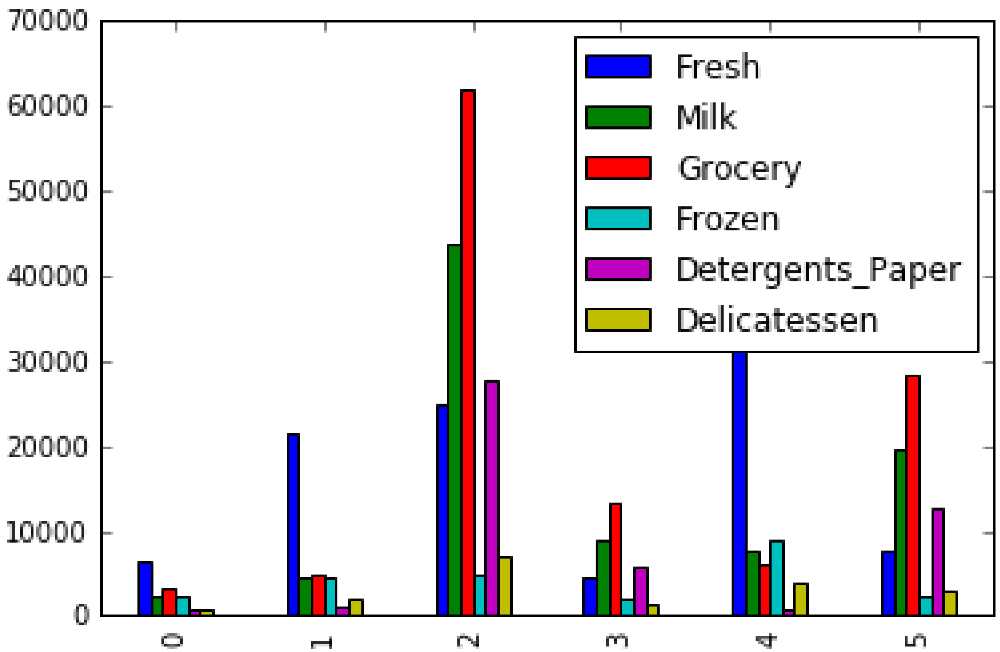


Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross



```
[[  6530.27762696   2206.36621968   3364.80414817   2226.52254481
     916.87780541    787.18195192]
 [ 21581.49165304   4488.34429429   4916.82839149   4573.74947658
    1093.29763515   1887.54965516]
 [ 24860.69170589  43833.94858552  61861.06334885   4977.66921543
   27876.75857623   6880.80866079]
 [  4435.66620993   8953.7595231   13443.18661029   1880.32492103
    5756.16985802   1498.52304524]
 [ 49427.35748872   7534.56299422   6075.20517973   8919.5461264
     610.34375945   3779.46321341]
 [  7548.033033    19464.05960205  28391.11764097   2337.92643736
   12703.5558441    2954.37124317]]
```

```
<matplotlib.figure.Figure at 0x1135db890>
```



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

Answer: These central points represent each clusters. I divided to 6 groups. I can say that two people who are belonging to same group have similar pattern of consumption. And also they will respond very similarly to some events. Furthermore each group can be represented by their centroid. Using inverse_transform, I printed the features' information of centroids.

As you can see above bar-graph(gmm), You can visually check the properties of each group. For example group 5 is heavy consumer and they like to buy grocery products. Group 6 usually buy fresh a lot. Group 0 and Group 2 normally spend not too much. Likewise we can analyze the centroid data and this would represent each segmentation of customers.

Conclusions

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

Answer: As you can see on second picture I tried to classify using KMeans algorithm. But the segmentation seems not that meaningful. I believe that the segmentation with GMM classifier seems more meaningful. Gaussian Mixture Model. It considers not only distance but also probability. So the GMM is more natural separation. On the other hands, KMeans algorithm's separation looks not meaningful. It just divide the area geometricaly.

Let's talk about PCA and ICA. Basically ICA is not used for reducing dimensionality. It is userful when separate mixed signals. If we want to discern featured signal from others, we can use ICA. On the other hand, PCA is the optimized method to reduce dimensionality or to compress the data size. So in this case I would like to choose PCA method so that compress data and reduce dimensionality which can prevent the overfitting problem.

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

Answer: Customer segmentation is really important task to marketing part. Using this data we can apply suitable advertisement or promotion to each segmentation group. For example, for green group customers, we can give them to sale promotion code or coupon of fresh food. Or delivery service for another group.

Also we can manage group respectively. If we manage whole people at once, there are many problems. Group with large number(like light green, brown group)'s tendency is much more powerfull than small groups tendency. So, in this case people of group(light green)'s dissatisfaction can be ignored by noise of major group. However using a segmentation and managing respectively can prevent these kinds of problems.

Moreover, once we classify the data set, A/B testing become really useful. When we test something, there are really lots of variables. And it is very important to manage variables. Execpt target variables, we should keep other variables same values. As we classify the data set to some groups, we can perform the A/B test on same group. It will maintain other feautres same and only variable related to testing would be different. It is ideal environment to A/B testing.

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

Answer: I will store data separately based on their group. I can take a suvey from people who belongs to different groups. And their opinion can represent their group. So I can predict their needs likewise.

People in same group are prone to behave similar. So when we apply machine learning, training should not be weighted on one group. I mean, when we split the training data we can choose data uniformly in each sections.

```
In [ ]:
```