Machine Learning Engineer Nanodegree

Unsupervised Learning

Project 3: Creating Customer Segments

Welcome to the third project of the Machine Learning Engineer Nanodegree! In this notebook, some template code has already been provided for you, and it will be your job to implement the additional functionality necessary to successfully complete this project. Sections that begin with **'Implementation'** in the header indicate that the following block of code will require additional functionality which you must provide. Instructions will be provided for each section and the specifics of the implementation are marked in the code block with a `'TODO'` statement. Please be sure to read the instructions carefully!

In addition to implementing code, there will be questions that you must answer which relate to the project and your implementation. Each section where you will answer a question is preceded by a **'Question X'** header. Carefully read each question and provide thorough answers in the following text boxes that begin with **'Answer:'**. Your project submission will be evaluated based on your answers to each of the questions and the implementation you provide.

> **Note:** Code and Markdown cells can be executed using the **Shift + Enter** keyboard shortcut. In addition, Markdown cells can be edited by typically double-clicking the cell to enter edit mode.

Getting Started

In this project, you will analyze a dataset containing data on various customers' annual spending amounts (reported in *monetary units*) of diverse product categories for internal structure. One goal of this project is to best describe the variation in the different types of customers that a wholesale distributor interacts with. Doing so would equip the distributor with insight into how to best structure their delivery service to meet the needs of each customer.

The dataset for this project can be found on the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Wholesale+customers). For the purposes of this project, the features `'Channel'` and `'Region'` will be excluded in the analysis — with focus instead on the six product categories recorded for customers.

Run the code block below to load the wholesale customers dataset, along with a few of the necessary Python libraries required for this project. You will know the dataset loaded successfully if the size of the dataset is reported.

```
In [1]:  # Import libraries necessary for this project
         import numpy as np
         import pandas as pd
         import renders as rs
         import matplotlib
         matplotlib.style.use('ggplot')

         from IPython.display import display # Allows the use of display() for DataFrames

         # Show matplotlib plots inline (nicely formatted in the notebook)
         %matplotlib inline

         # Load the wholesale customers dataset
         try:
             data = pd.read_csv("customers.csv")
             data.drop(['Region', 'Channel'], axis = 1, inplace = True)
             print "Wholesale customers dataset has {} samples with {} features each.".format(*d
         ata.shape)
         except:
             print "Dataset could not be loaded. Is the dataset missing?"
```

```
Wholesale customers dataset has 440 samples with 6 features each.
```

Data Exploration

In this section, you will begin exploring the data through visualizations and code to understand how each feature is related to the others. You will observe a statistical description of the dataset, consider the relevance of each feature, and select a few sample data points from the dataset which you will track through the course of this project.

Run the code block below to observe a statistical description of the dataset. Note that the dataset is composed of six important product categories: **'Fresh'**, **'Milk'**, **'Grocery'**, **'Frozen'**, **'Detergents_Paper'**, and **'Delicatessen'**. Consider what each category represents in terms of products you could purchase.

```
In [2]:  # Display a description of the dataset
         display(data.describe())
         #data.describe()
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

Implementation: Selecting Samples

To get a better understanding of the customers and how their data will transform through the analysis, it would be best to select a few sample data points and explore them in more detail. In the code block below, add **three** indices of your choice to the `indices` list which will represent the customers to track. It is suggested to try different sets of samples until you obtain customers that vary significantly from one another.

```
In [33]: # TODO: Select three indices of your choice you wish to sample from the dataset
         indices = [15,170,400]

         # Create a DataFrame of the chosen samples
         samples = pd.DataFrame(data.loc[indices], columns = data.keys()).reset_index(drop = Tru
         e)
         print "Chosen samples of wholesale customers dataset:"
         display(samples)
```

Chosen samples of wholesale customers dataset:

|	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	10253	1114	3821	397	964	412
1	260	8675	13430	1116	7015	323
2	4446	906	1238	3576	153	1014

Question 1

Consider the total purchase cost of each product category and the statistical description of the dataset above for your sample customers.
What kind of establishment (customer) could each of the three samples you've chosen represent?
Hint: Examples of establishments include places like markets, cafes, and retailers, among many others. Avoid using names for establishments, such as saying *"McDonalds"* when describing a sample customer as a restaurant.

In [34]:
```
import seaborn as sb
samples_vbar = samples.append(data.describe().loc['mean'])
samples_vbar.index = indices + ['mean']
samples_vbar.plot(kind='bar', figsize=(12,8))
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x20e1efd0>
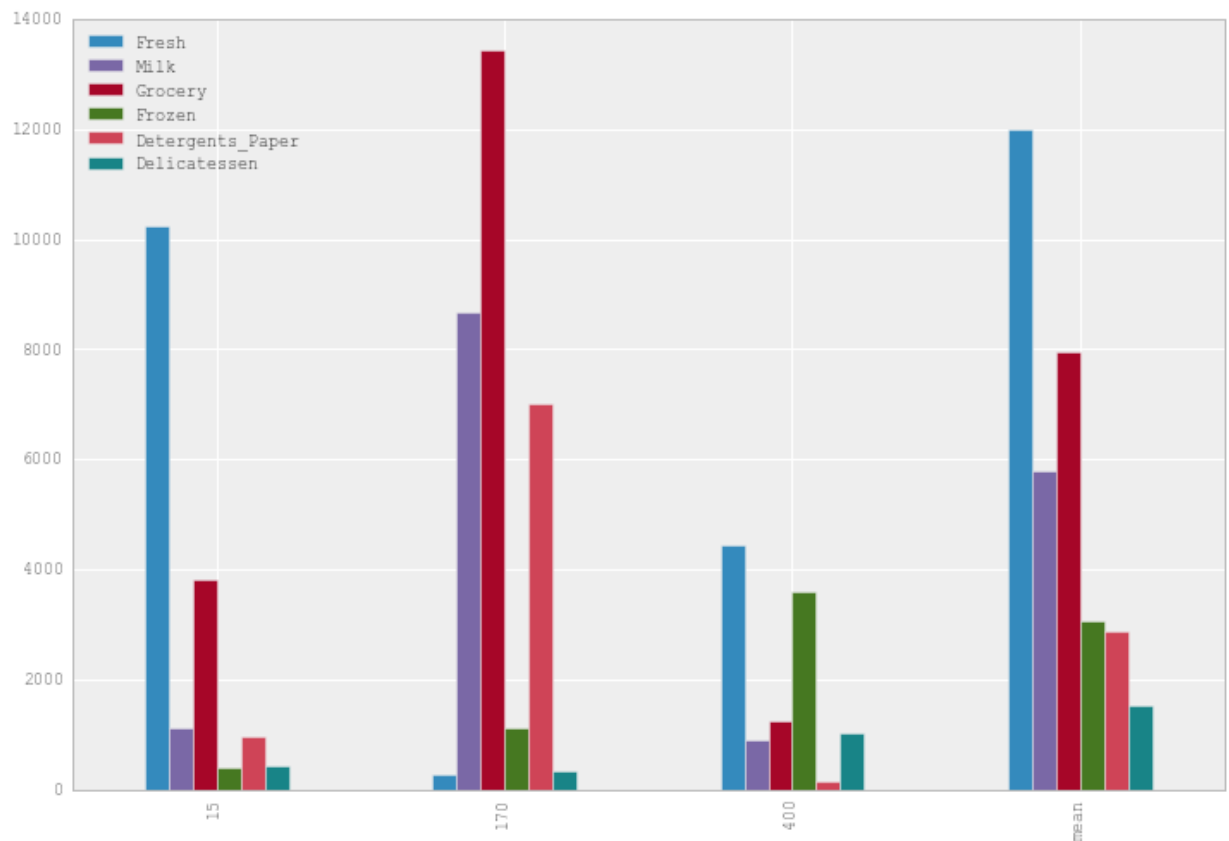
In [35]: `display(pd.DataFrame.mean(data))`

```
Fresh               12000.297727
Milk                 5796.265909
Grocery              7951.277273
Frozen               3071.931818
Detergents_Paper     2881.493182
Delicatessen         1524.870455
dtype: float64
```

In [36]: `display(pd.DataFrame.median(data))`

```
Fresh               8504.0
Milk                3627.0
Grocery             4755.5
Frozen              1526.0
Detergents_Paper     816.5
Delicatessen         965.5
dtype: float64
```

In [37]:
```python
# Compute the percentile ranking of the whole dataset.
percentiles = data.rank(pct=True)
# rank of the indices
percentiles = percentiles.iloc[indices]
percentiles
```

Out[37]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
15	0.572727	0.163636	0.440909	0.134091	0.547727	0.256818
170	0.040909	0.818182	0.825000	0.400000	0.890909	0.198864
400	0.309091	0.111364	0.084091	0.754545	0.139773	0.522727

In [38]:
```python
ax = sb.heatmap(percentiles,  vmax=1,annot=True, linewidths=.5,cmap="YlGnBu")
```
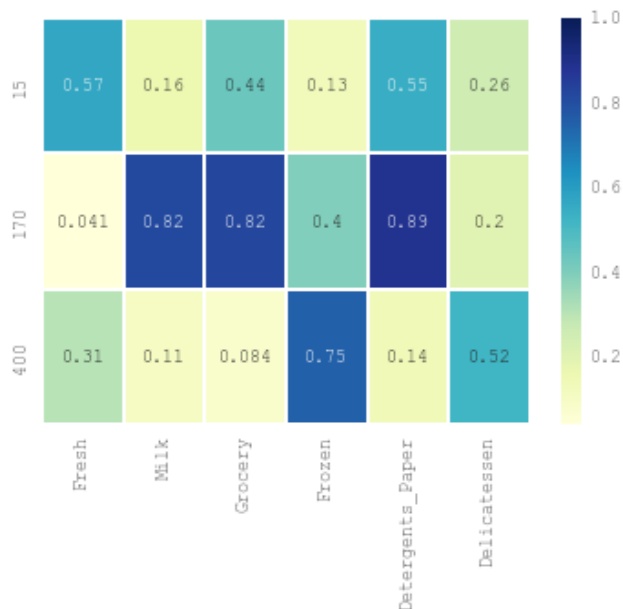

Answer: Looking into mean and median of the dataset, it can be concluded that the data is right skewed. Therefore, we used median to compare between the samples.

The first sample has slightly larger cost for Fresh and Detergent than the median values. Furthermore, cost for Grocery is very close to median. Therefore, it could be a restaurant which focus on selling fresh items.

The second sample has a much larger cost for Milk, Grocery and Detergents_Paper from their median values. This could likely be a retailers or superstore.

The last sample shows a very high Frozen cost and a slightly larger Delicatessen cost compared to the median values. This sample could be a cafe/restaurant that prepares set menu items from frozen.

Implementation: Feature Relevance

One interesting thought to consider is if one (or more) of the six product categories is actually relevant for understanding customer purchasing. That is to say, is it possible to determine whether customers purchasing some amount of one category of products will necessarily purchase some proportional amount of another category of products? We can make this determination quite easily by training a supervised regression learner on a subset of the data with one feature removed, and then score how well that model can predict the removed feature.

In the code block below, you will need to implement the following:

- Assign `new_data` a copy of the data by removing a feature of your choice using the `DataFrame.drop` function.
- Use `sklearn.cross_validation.train_test_split` to split the dataset into training and testing sets.
  - Use the removed feature as your target label. Set a `test_size` of `0.25` and set a `random_state`.
- Import a decision tree regressor, set a `random_state`, and fit the learner to the training data.
- Report the prediction score of the testing set using the regressor's `score` function.

```
In [39]:  # TODO: Make a copy of the DataFrame, using the 'drop' function to drop the given featu
          re
          new_data = data.drop(['Grocery'],axis=1)

          # TODO: Split the data into training and testing sets using the given feature as the ta
          rget
          from sklearn.cross_validation import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(new_data, data['Fresh'],
          test_size=0.25, random_state=84)

          # TODO: Create a decision tree regressor and fit it to the training set
          #from sklearn.metrics import make_scorer
          from sklearn.tree import DecisionTreeRegressor
          #from sklearn.metrics import mean_squared_error
          from sklearn.metrics import r2_score
          #from sklearn.grid_search import GridSearchCV

          #scoring_function = make_scorer(mean_squared_error, greater_is_better=False)

          regressor = DecisionTreeRegressor(random_state=84)
          #parameters = {'max_depth':(1,2,3,4)}
          #reg = GridSearchCV(regressor,parameters,cv = 5, scoring=scoring_function)
          regressor.fit(X_train, y_train)
          y_predict = regressor.predict(X_test)
          # TODO: Report the score of the prediction using the testing set
          score = regressor.score(X_test, y_test)
          score
```

Out[39]: 0.75672330167431845

Question 2

Which feature did you attempt to predict? What was the reported prediction score? Is this feature is necessary for identifying customers' spending habits?
Hint: The coefficient of determination, R^2, is scored between 0 and 1, with 1 being a perfect fit. A negative R^2 implies the model fails to fit the data.

```
In [40]:  r2_score(y_test,y_predict)
```

```
Out[40]:  0.75672330167431845
```

Answer: I tried to predict the feature: Grocery using decision tree regressor method. The prediction score was 0.7567, which inidcates that the decision tree regressor method fits the dataset to quite accurately to predict Grocery cost. Since Grocery cost can be predicted from other features, this feature is not necessary for identifying customers' spending habits.

Visualize Feature Distributions

To get a better understanding of the dataset, we can construct a scatter matrix of each of the six product features present in the data. If you found that the feature you attempted to predict above is relevant for identifying a specific customer, then the scatter matrix below may not show any correlation between that feature and the others. Conversely, if you believe that feature is not relevant for identifying a specific customer, the scatter matrix might show a correlation between that feature and another feature in the data. Run the code block below to produce a scatter matrix.

```
In [41]:  # Produce a scatter matrix for each pair of features in the data
          pd.options.display.mpl_style = 'default'
          pd.scatter_matrix(data, alpha = 0.8, color='DarkGreen',figsize = (14,8), diagonal = 'kd
          e');
```
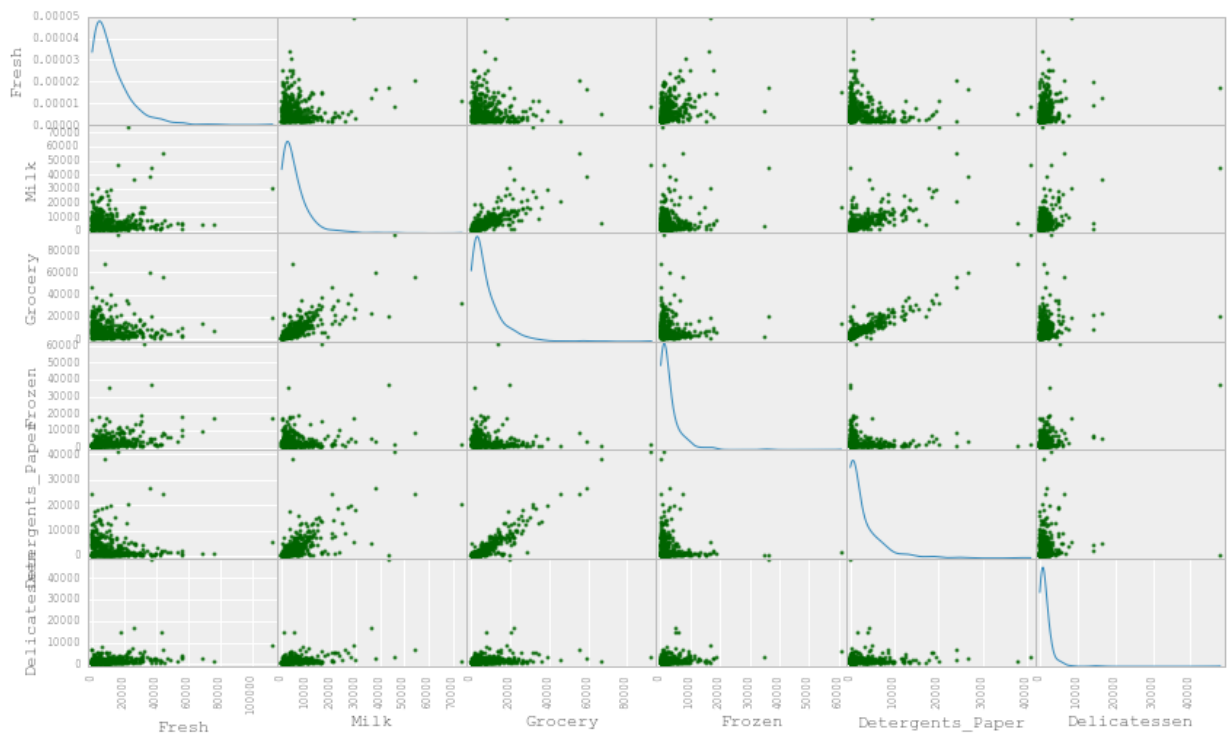
```
In [42]:  ## correlation matrix between the features
          corrmat = data.corr()
          # Draw the heatmap of the correlation using seaborn
          sb.heatmap(corrmat, vmax=1, square=True,annot=True)
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x23a04be0>
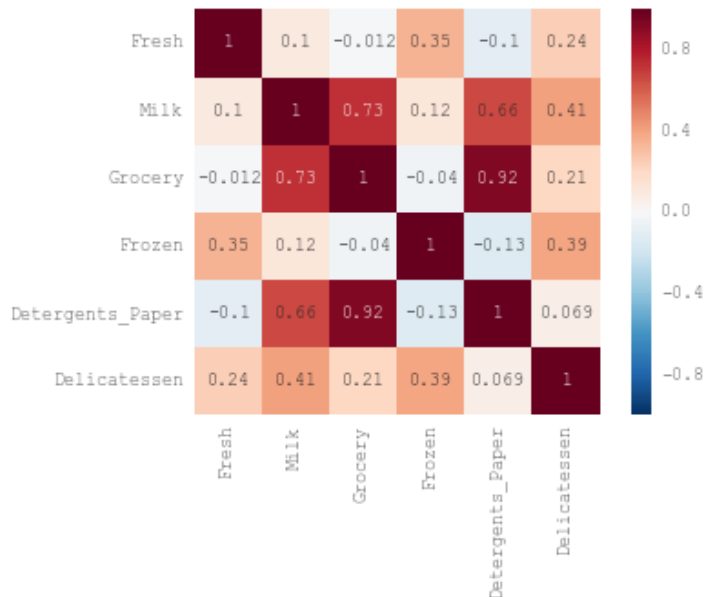
Question 3

Are there any pairs of features which exhibit some degree of correlation? Does this confirm or deny your suspicions about the relevance of the feature you attempted to predict? How is the data for those features distributed?
Hint: Is the data normally distributed? Where do most of the data points lie?

Answer: From scatter plot and correlation matrix heatmap, Grocery and Detergents_Paper are strongly correlated (correlation value =0.92)with each other. Moderate correlation (correlation value > 0.5 and < 0.9) is observed between Milk and Grocery, Milk and Detergents_Paper. Delicatessen, Fresh, and Frozen show poor correlation (correlation value < 0.5) with any of the other features. Furthermore, the data is not normally distributed. Instead, it is skewed to the right.

Data Preprocessing

In this section, you will preprocess the data to create a better representation of customers by performing a scaling on the data and detecting (and optionally removing) outliers. Preprocessing data is often times a critical step in assuring that results you obtain from your analysis are significant and meaningful.

Implementation: Feature Scaling

If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate (http://econbrowser.com/archives/2014/02/use-of-logarithms-in-economics) to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test (http://scipy.github.io/devdocs/generated/scipy.stats.boxcox.html), which calculates the best power transformation of the data that reduces skewness. A simpler approach which can work in most cases would be applying the natural logarithm.

In the code block below, you will need to implement the following:

- Assign a copy of the data to `log_data` after applying a logarithm scaling. Use the `np.log` function for this.
- Assign a copy of the sample data to `log_samples` after applying a logrithm scaling. Again, use `np.log`.

```
In [43]:  # TODO: Scale the data using the natural logarithm
          log_data = np.log(data)

          # TODO: Scale the sample data using the natural logarithm
          log_samples = np.log(samples)
          pd.options.display.mpl_style = 'default'
          # Produce a scatter matrix for each pair of newly-transformed features
          pd.scatter_matrix(log_data, alpha = 0.8,color='DarkBlue', figsize = (14,8), diagonal =
          'kde');
```
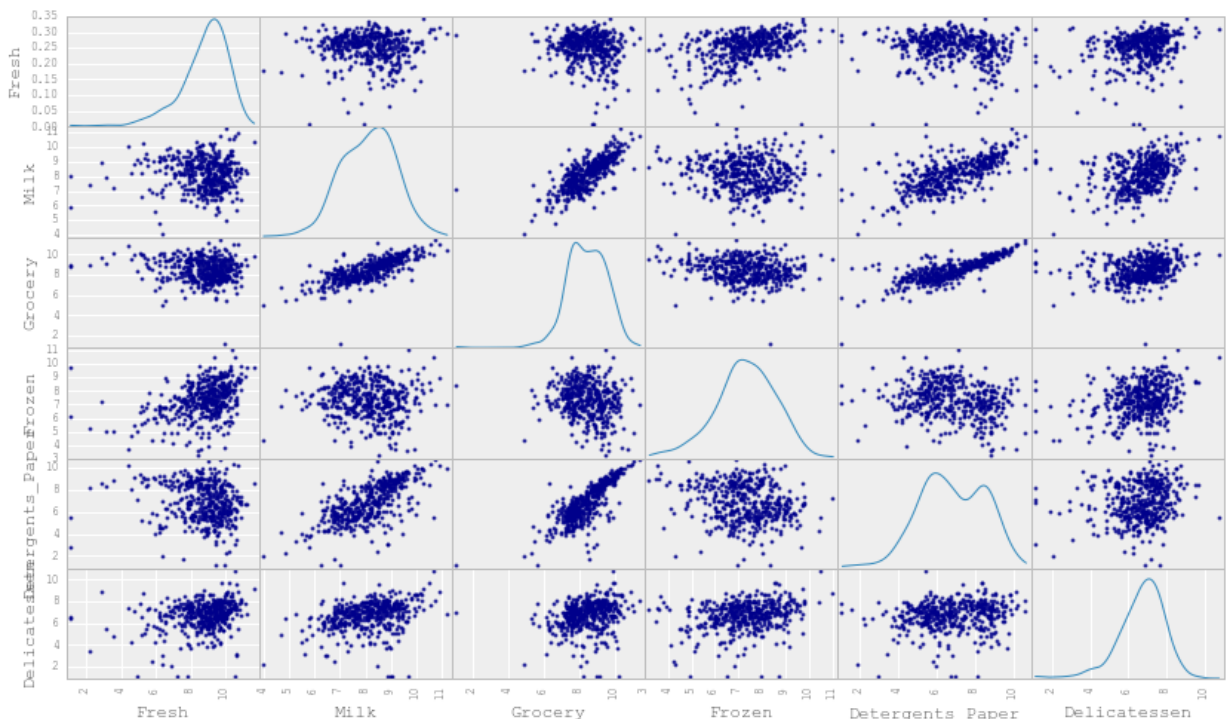

Observation

After applying a natural logarithm scaling to the data, the distribution of each feature should appear much more normal. For any pairs of features you may have identified earlier as being correlated, observe here whether that correlation is still present (and whether it is now stronger or weaker than before).

Run the code below to see how the sample data has changed after having the natural logarithm applied to it.

```
In [44]:   # Display the log-transformed sample data
           display(log_samples)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9.235326	7.015712	8.248267	5.983936	6.871091	6.021023
1	5.560682	9.068201	9.505246	7.017506	8.855806	5.777652
2	8.399760	6.809039	7.121252	8.182000	5.030438	6.921658

Implementation: Outlier Detection

Detecting outliers in the data is extremely important in the data preprocessing step of any analysis. The presence of outliers can often skew results which take into consideration these data points. There are many "rules of thumb" for what constitutes an outlier in a dataset. Here, we will use Tukey's Method for identfying outliers (http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/): An *outlier step* is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal.

In the code block below, you will need to implement the following:

- Assign the value of the 25th percentile for the given feature to Q1. Use `np.percentile` for this.
- Assign the value of the 75th percentile for the given feature to Q3. Again, use `np.percentile`.
- Assign the calculation of an outlier step for the given feature to `step`.
- Optionally remove data points from the dataset by adding indices to the `outliers` list.

NOTE: If you choose to remove any outliers, ensure that the sample data does not contain any of these points! Once you have performed this implementation, the dataset will be stored in the variable `good_data`.

In [45]:
```python
# For each feature find the data points with extreme high or low values
outliers  = []
for feature in log_data.keys():

    # TODO: Calculate Q1 (25th percentile of the data) for the given feature
    Q1 = np.percentile(log_data[feature], 25)

    # TODO: Calculate Q3 (75th percentile of the data) for the given feature
    Q3 = np.percentile(log_data[feature], 75)

    # TODO: Use the interquartile range to calculate an outlier step (1.5 times the int
erquartile range)
    step = 1.5 * (Q3 - Q1)

    # Display the outliers
    print "Data points considered outliers for the feature '{}':".format(feature)
    display(log_data[~((log_data[feature] >= Q1 - step) & (log_data[feature] <= Q3 + st
ep))])
    outliers.extend(log_data[~((log_data[feature] >= Q1 - step) & (log_data[feature] <=
 Q3 + step))].index)

# OPTIONAL: Select the indices for data points you wish to remove
print "Removing the following outliers from the dataset: {}".format(np.sort(list(set(ou
tliers))))

# Remove the outliers, if any were specified
good_data = log_data.drop(log_data.index[outliers]).reset_index(drop = True)
```

Data points considered outliers for the feature 'Fresh':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
65	4.442651	9.950323	10.732651	3.583519	10.095388	7.260523
66	2.197225	7.335634	8.911530	5.164786	8.151333	3.295837
81	5.389072	9.163249	9.575192	5.645447	8.964184	5.049856
95	1.098612	7.979339	8.740657	6.086775	5.407172	6.563856
96	3.135494	7.869402	9.001839	4.976734	8.262043	5.379897
128	4.941642	9.087834	8.248791	4.955827	6.967909	1.098612
171	5.298317	10.160530	9.894245	6.478510	9.079434	8.740337
193	5.192957	8.156223	9.917982	6.865891	8.633731	6.501290
218	2.890372	8.923191	9.629380	7.158514	8.475746	8.759669
304	5.081404	8.917311	10.117510	6.424869	9.374413	7.787382
305	5.493061	9.468001	9.088399	6.683361	8.271037	5.351858
338	1.098612	5.808142	8.856661	9.655090	2.708050	6.309918
353	4.762174	8.742574	9.961898	5.429346	9.069007	7.013016
355	5.247024	6.588926	7.606885	5.501258	5.214936	4.844187
357	3.610918	7.150701	10.011086	4.919981	8.816853	4.700480
412	4.574711	8.190077	9.425452	4.584967	7.996317	4.127134

Data points considered outliers for the feature 'Milk':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
86	10.039983	11.205013	10.377047	6.894670	9.906981	6.805723
98	6.220590	4.718499	6.656727	6.796824	4.025352	4.882802
154	6.432940	4.007333	4.919981	4.317488	1.945910	2.079442
356	10.029503	4.897840	5.384495	8.057377	2.197225	6.306275

Data points considered outliers for the feature 'Grocery':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
75	9.923192	7.036148	1.098612	8.390949	1.098612	6.882437
154	6.432940	4.007333	4.919981	4.317488	1.945910	2.079442

Data points considered outliers for the feature 'Frozen':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
38	8.431853	9.663261	9.723703	3.496508	8.847360	6.070738
57	8.597297	9.203618	9.257892	3.637586	8.932213	7.156177
65	4.442651	9.950323	10.732651	3.583519	10.095388	7.260523
145	10.000569	9.034080	10.457143	3.737670	9.440738	8.396155
175	7.759187	8.967632	9.382106	3.951244	8.341887	7.436617
264	6.978214	9.177714	9.645041	4.110874	8.696176	7.142827
325	10.395650	9.728181	9.519735	11.016479	7.148346	8.632128
420	8.402007	8.569026	9.490015	3.218876	8.827321	7.239215
429	9.060331	7.467371	8.183118	3.850148	4.430817	7.824446
439	7.932721	7.437206	7.828038	4.174387	6.167516	3.951244

Data points considered outliers for the feature 'Detergents_Paper':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
75	9.923192	7.036148	1.098612	8.390949	1.098612	6.882437
161	9.428190	6.291569	5.645447	6.995766	1.098612	7.711101

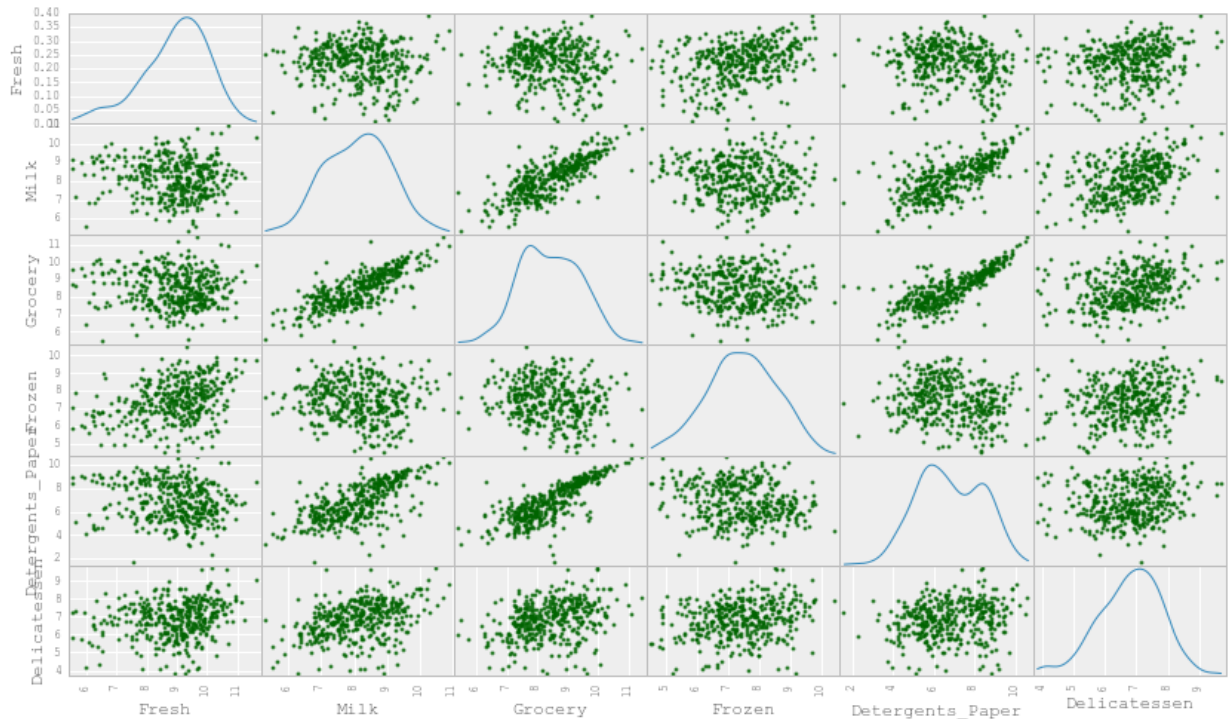Data points considered outliers for the feature 'Delicatessen':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
66	2.197225	7.335634	8.911530	5.164786	8.151333	3.295837
109	7.248504	9.724899	10.274568	6.511745	6.728629	1.098612
128	4.941642	9.087834	8.248791	4.955827	6.967909	1.098612
137	8.034955	8.997147	9.021840	6.493754	6.580639	3.583519
142	10.519646	8.875147	9.018332	8.004700	2.995732	1.098612
154	6.432940	4.007333	4.919981	4.317488	1.945910	2.079442
183	10.514529	10.690808	9.911952	10.505999	5.476464	10.777768
184	5.789960	6.822197	8.457443	4.304065	5.811141	2.397895
187	7.798933	8.987447	9.192075	8.743372	8.148735	1.098612
203	6.368187	6.529419	7.703459	6.150603	6.860664	2.890372
233	6.871091	8.513988	8.106515	6.842683	6.013715	1.945910
285	10.602965	6.461468	8.188689	6.948897	6.077642	2.890372
289	10.663966	5.655992	6.154858	7.235619	3.465736	3.091042
343	7.431892	8.848509	10.177932	7.283448	9.646593	3.610918

Removing the following outliers from the dataset: [38 57 65 66 75 81 86 95 96
 98 109 128 137 142 145 154 161 171
 175 183 184 187 193 203 218 233 264 285 289 304 305 325 338 343 353 355
 356 357 412 420 429 439]

```
In [46]:   from collections import Counter
           outliers_count = Counter(outliers)
           multi_feature_outliers = list({indice for indice, count in outliers_count.items() if co
           unt > 1})
           print "Outliers for more than one feature: ", np.sort(multi_feature_outliers)
```

```
Outliers for more than one feature:  [ 65  66  75 128 154]
```

```
In [47]:   pd.scatter_matrix(good_data, alpha = 0.8, color='DarkGreen',figsize = (14,8), diagonal
           = 'kde');
```


Question 4

Are there any data points considered outliers for more than one feature? Should these data points be removed from the dataset? If any data points were added to the outliers *list to be removed, explain why.*

Answer: Five data points are identified as outliers for more than one feature set. The data points are: 65, 66, 75, 128, and 154. These five points are removed from the dataset. Furthermore, several other outliers are also found in the feature sets. They are also removed form the dataset. These outliers are the main reason behind skewed distribution of the dataset. After the removal of these points, the skewness in the dataset is almost gone, and distribution looks pretty close to normal distribution (as shown in the scatter plots of the good data). The removal of outliers will help us to develop more accurate predictive model.

Feature Transformation

In this section you will use principal component analysis (PCA) to draw conclusions about the underlying structure of the wholesale customer data. Since using PCA on a dataset calculates the dimensions which best maximize variance, we will find which compound combinations of features best describe customers.

Implementation: PCA

Now that the data has been scaled to a more normal distribution and has had any necessary outliers removed, we can now apply PCA to the good_data to discover which dimensions about the data best maximize the variance of features involved. In addition to finding these dimensions, PCA will also report the *explained variance ratio* of each dimension — how much variance within the data is explained by that dimension alone. Note that a component (dimension) from PCA can be considered a new "feature" of the space, however it is a composition of the original features present in the data.

In the code block below, you will need to implement the following:

- Import `sklearn.decomposition.PCA` and assign the results of fitting PCA in six dimensions with `good_data` to `pca`.
- Apply a PCA transformation of the sample log-data `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

```
In [48]:   # TODO: Apply PCA to the good data with the same number of dimensions as features
           from sklearn.decomposition import PCA
           pca = PCA(n_components=6).fit(good_data)

           # TODO: Apply a PCA transformation to the sample log-data
           pca_samples = pca.transform(log_samples)
           # Generate PCA results plot
           pca_results = rs.pca_results(good_data, pca)
```
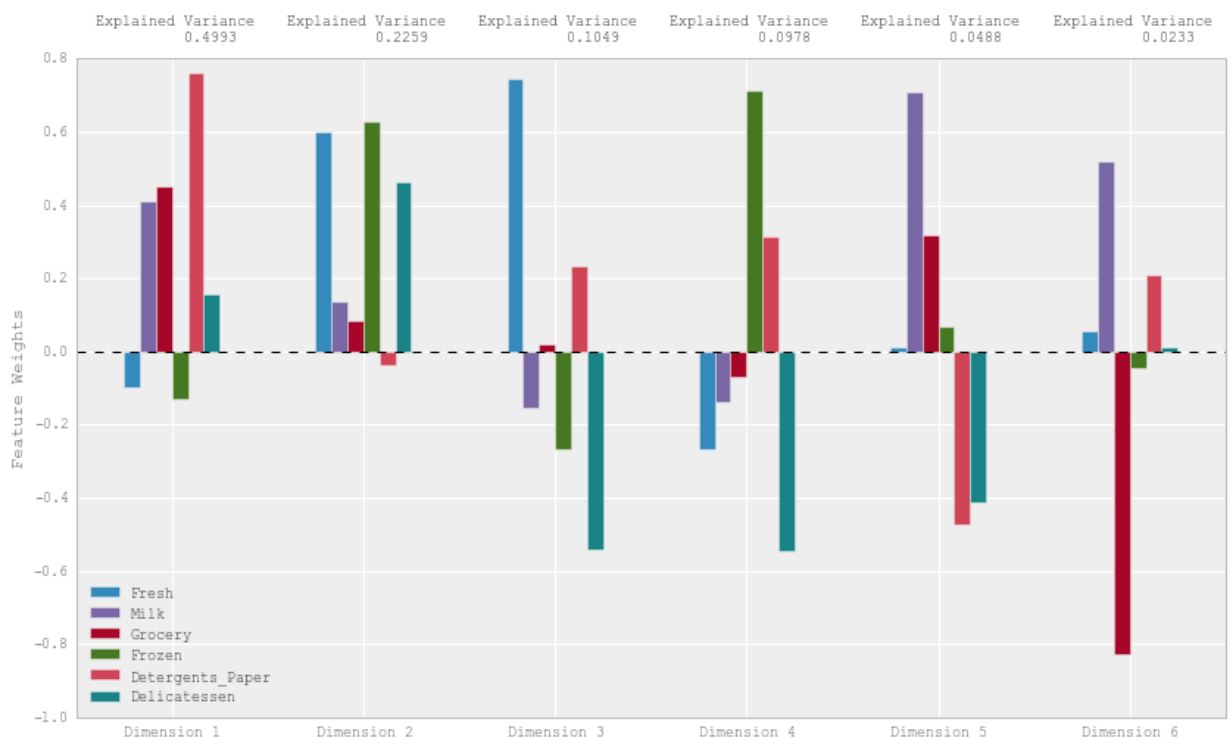


```
In [49]:   #calclulate the cumulative sum of variance explained
           cumsum_var= np.cumsum(pca.explained_variance_ratio_)

           print "Explained variance: ", cumsum_var

           Explained variance:  [ 0.49930484  0.7252529   0.83013599  0.92795361  0.9767447   1.
             ]
```

Question 5

*How much variance in the data is explained **in total** by the first and second principal component? What about the first four principal components? Using the visualization provided above, discuss what the first four dimensions best represent in terms of customer spending.*
Hint: A positive increase in a specific dimension corresponds with an *increase* of the *positive-weighted* features and a *decrease* of the *negative-weighted* features. The rate of increase or decrease is based on the indivdual feature weights.

Answer: The first two principal component and the first four principal components explain 0.7252529 and 0.92795361 of total variance in the data respectively. The first principal component shows positive correlation to Milk, Grocery and Detergents_Paper in terms of total spending and the second dimension shows strong positive correlation of Fresh, Frozen and Delicatessen. These positive correaltion suggests that increasing cost of one feature might also lead to an increase in cost in the other positive correlated features. The third dimension shows a strong positive correlation to Fresh and a strong negative correlation to Delicatessen; that suggests an increase in one might lead to a decrease in the other. The fourth dimension is very similar to the third dimension where strong positive correlation is observed in Frozen and high negative correlation is observed in Delicatessen.

Observation

Run the code below to see how the log-transformed sample data has changed after having a PCA transformation applied to it in six dimensions. Observe the numerical value for the first four dimensions of the sample points. Consider if this is consistent with your initial interpretation of the sample points.

```
In [50]:  # Display sample log-data after having a PCA transformation applied
          display(pd.DataFrame(np.round(pca_samples, 4), columns = pca_results.index.values))
```

	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5	Dimension 6
0	-0.4351	-1.2598	1.2207	-0.4961	-0.6487	-0.3370
1	2.6704	-2.6155	-1.4868	1.6088	0.3924	-0.1510
2	-2.4842	-0.0147	-0.9005	0.3315	-0.5147	-0.0357

Implementation: Dimensionality Reduction

When using principal component analysis, one of the main goals is to reduce the dimensionality of the data — in effect, reducing the complexity of the problem. Dimensionality reduction comes at a cost: Fewer dimensions used implies less of the total variance in the data is being explained. Because of this, the *cumulative explained variance ratio* is extremely important for knowing how many dimensions are necessary for the problem. Additionally, if a signifiant amount of variance is explained by only two or three dimensions, the reduced data can be visualized afterwards.

In the code block below, you will need to implement the following:

- Assign the results of fitting PCA in two dimensions with `good_data` to `pca`.
- Apply a PCA transformation of `good_data` using `pca.transform`, and assign the reuslts to `reduced_data`.
- Apply a PCA transformation of the sample log-data `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

```
In [51]:  # TODO: Apply PCA by fitting the good data with only two dimensions
          pca = PCA(n_components=2).fit(good_data)

          # TODO: Transform the good data using the PCA fit above
          reduced_data = pca.transform(good_data)

          # TODO: Transform the sample log-data using the PCA fit above
          pca_samples = pca.transform(log_samples)

          # Create a DataFrame for the reduced data
          reduced_data = pd.DataFrame(reduced_data, columns = ['Dimension 1', 'Dimension 2'])
```

Observation

Run the code below to see how the log-transformed sample data has changed after having a PCA transformation applied to it using only two dimensions. Observe how the values for the first two dimensions remains unchanged when compared to a PCA transformation in six dimensions.

```
In [52]:  # Display sample log-data after applying PCA transformation in two dimensions
          display(pd.DataFrame(np.round(pca_samples, 4), columns = ['Dimension 1', 'Dimension
          2']))
```

	Dimension 1	Dimension 2
0	-0.4351	-1.2598
1	2.6704	-2.6155
2	-2.4842	-0.0147

Clustering

In this section, you will choose to use either a K-Means clustering algorithm or a Gaussian Mixture Model clustering algorithm to identify the various customer segments hidden in the data. You will then recover specific data points from the clusters to understand their significance by transforming them back into their original dimension and scale.

Question 6

What are the advantages to using a K-Means clustering algorithm? What are the advantages to using a Gaussian Mixture Model clustering algorithm? Given your observations about the wholesale customer data so far, which of the two algorithms will you use and why?

Answer:rewrite Advantages of K Means clustering:

- K-means clustering execution time is linear t0 the number of data objects, however execution time for hierarchical clustering algorithms is quadratic.
- k-means performs really well with round shaped, approximately equal density clusters.

Advantages of Gaussian Mixture Models (GMM):

- Fastest algorithm for learning mixture models
- GMM maximizes only the likelihood, therefore it will not bias the means towards zero.
- Provide a density estimation for each cluster

For the wholesame customer data, GMM probably the appropriate model to use since dataset is small and is not divided into tight round shape.

References:

1. http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/mixture.html (http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/mixture.html)
2. https://www.quora.com/What-are-the-advantages-to-using-a-Gaussian-Mixture-Model-clustering-algorithm (https://www.quora.com/What-are-the-advantages-to-using-a-Gaussian-Mixture-Model-clustering-algorithm)
3. http://scikit-learn.org/stable/modules/mixture.html (http://scikit-learn.org/stable/modules/mixture.html)
4. https://www.quora.com/What-are-the-pros-and-cons-of-kmeans-vs-hierarchical-clustering (https://www.quora.com/What-are-the-pros-and-cons-of-kmeans-vs-hierarchical-clustering)

Implementation: Creating Clusters

Depending on the problem, the number of clusters that you expect to be in the data may already be known. When the number of clusters is not known *a priori*, there is no guarantee that a given number of clusters best segments the data, since it is unclear what structure exists in the data — if any. However, we can quantify the "goodness" of a clustering by calculating each data point's *silhouette coefficient*. The silhouette coefficient (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html) for a data point measures how similar it is to its assigned cluster from -1 (dissimilar) to 1 (similar). Calculating the *mean* silhouette coefficient provides for a simple scoring method of a given clustering.

In the code block below, you will need to implement the following:

- Fit a clustering algorithm to the `reduced_data` and assign it to `clusterer`.
- Predict the cluster for each data point in `reduced_data` using `clusterer.predict` and assign them to `preds`.
- Find the cluster centers using the algorithm's respective attribute and assign them to `centers`.
- Predict the cluster for each sample data point in `pca_samples` and assign them `sample_preds`.
- Import sklearn.metrics.silhouette_score and calculate the silhouette score of `reduced_data` against `preds`.
    - Assign the silhouette score to `score` and print the result.

```
In [53]: from sklearn.mixture import GMM
         from sklearn.metrics import silhouette_samples, silhouette_score

         for n in xrange(10, 1, -1):
             # TODO: Apply your clustering algorithm of choice to the reduced data
             clusterer = GMM(n_components=n).fit(reduced_data)

             # TODO: Predict the cluster for each data point
             preds = clusterer.predict(reduced_data)

             # TODO: Find the cluster centers
             centers = clusterer.means_

             # TODO: Predict the cluster for each transformed sample data point
             sample_preds = clusterer.predict(pca_samples)

             # TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
             score = silhouette_score(reduced_data, preds, metric='euclidean')
             print "For {} clusters, the mean silhouette coefficient is {:.3f}.".format(n,
         score)
```

```
For 10 clusters, the mean silhouette coefficient is 0.269.
For 9 clusters, the mean silhouette coefficient is 0.284.
For 8 clusters, the mean silhouette coefficient is 0.306.
For 7 clusters, the mean silhouette coefficient is 0.294.
For 6 clusters, the mean silhouette coefficient is 0.307.
For 5 clusters, the mean silhouette coefficient is 0.266.
For 4 clusters, the mean silhouette coefficient is 0.292.
For 3 clusters, the mean silhouette coefficient is 0.357.
For 2 clusters, the mean silhouette coefficient is 0.444.
```

Question 7

Report the silhouette score for several cluster numbers you tried. Of these, which number of clusters has the best silhouette score?

Answer: Best silhouette score is 0.444 and the obtained from 2 clusters.

Cluster Visualization

Once you've chosen the optimal number of clusters for your clustering algorithm using the scoring metric above, you can now visualize the results by executing the code block below. Note that, for experimentation purposes, you are welcome to adjust the number of clusters for your clustering algorithm to see various visualizations. The final visualization provided should, however, correspond with the optimal number of clusters.

```
In [54]:  # Display the results of the clustering from implementation
          rs.cluster_results(reduced_data, preds, centers, pca_samples)
```
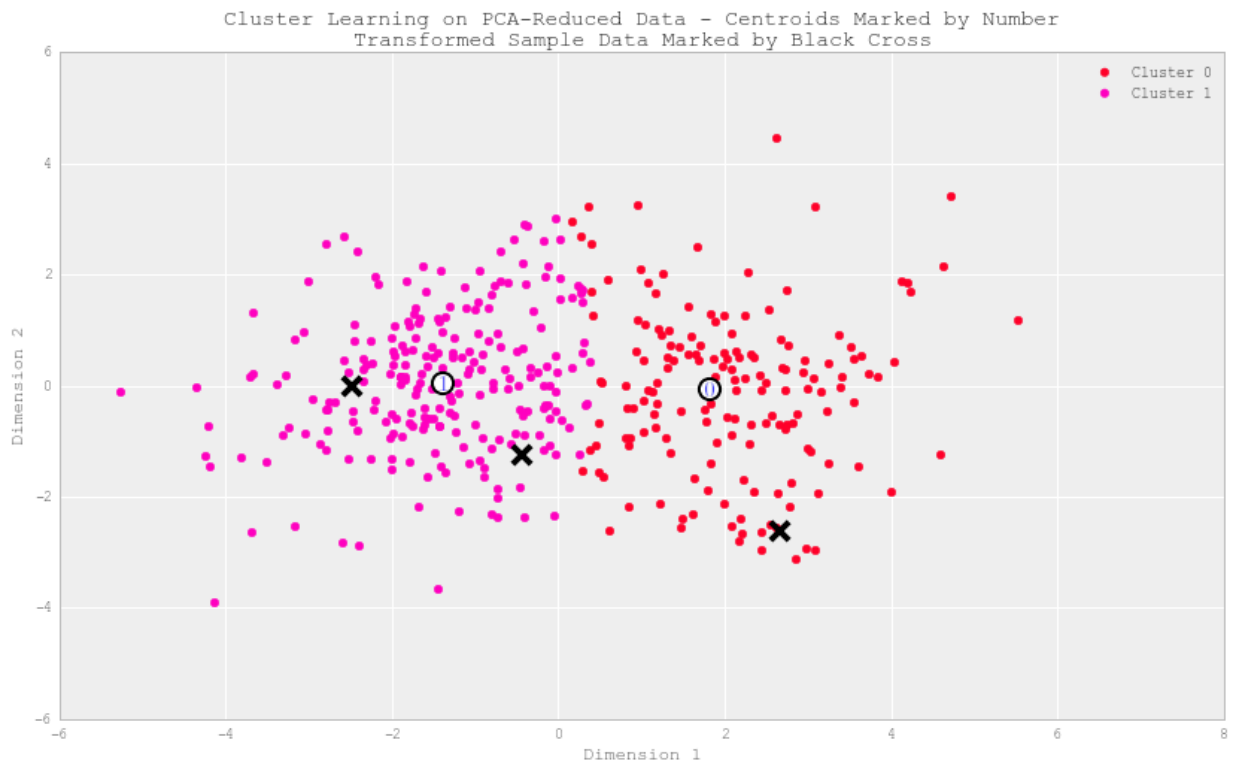

Implementation: Data Recovery

Each cluster present in the visualization above has a central point. These centers (or means) are not specifically data points from the data, but rather the *averages* of all the data points predicted in the respective clusters. For the problem of creating customer segments, a cluster's center point corresponds to *the average customer of that segment*. Since the data is currently reduced in dimension and scaled by a logarithm, we can recover the representative customer spending from these data points by applying the inverse transformations.

In the code block below, you will need to implement the following:

- Apply the inverse transform to `centers` using `pca.inverse_transform` and assign the new centers to `log_centers`.
- Apply the inverse function of `np.log` to `log_centers` using `np.exp` and assign the true centers to `true_centers`.

```
In [55]:  # TODO: Inverse transform the centers
          log_centers = pca.inverse_transform(centers)

          # TODO: Exponentiate the centers
          true_centers = np.exp(log_centers)

          # Display the true centers
          segments = ['Segment {}'.format(i) for i in range(0,len(centers))]
          true_centers = pd.DataFrame(np.round(true_centers), columns = data.keys())
          true_centers.index = segments
          display(true_centers)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Segment 0	6079	7042	10241	1275	3546	1159
Segment 1	8967	1920	2437	2081	309	741

Question 8

Consider the total purchase cost of each product category for the representative data points above, and reference the statistical description of the dataset at the beginning of this project. *What set of establishments could each of the customer segments represent?*
Hint: A customer who is assigned to 'Cluster X' should best identify with the establishments represented by the feature set of 'Segment X'.

```
In [56]:  display(pd.DataFrame.median(data))
          Fresh             8504.0
          Milk              3627.0
          Grocery           4755.5
          Frozen            1526.0
          Detergents_Paper   816.5
          Delicatessen       965.5
          dtype: float64
```

Answer:

For Segment 0, Milk, Grocery, Detergents_Paper and Delicatessen are higher than their respective medians and could represent retailers.

For Segment 1, Fresh and Frozen are higher than their respective medians and could represent restaurants.

```
In [60]:  samples
```
Out[60]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	10253	1114	3821	397	964	412
1	260	8675	13430	1116	7015	323
2	4446	906	1238	3576	153	1014

Question 9

*For each sample point, which customer segment from **Question 8** best represents it? Are the predictions for each sample point consistent with this?*

Run the code block below to find which cluster each sample point is predicted to be.

```
In [58]: # Display the predictions
         for i, pred in enumerate(sample_preds):
             print "Sample point", i, "predicted to be in Cluster", pred

         Sample point 0 predicted to be in Cluster 1
         Sample point 1 predicted to be in Cluster 0
         Sample point 2 predicted to be in Cluster 1
```

Answer: The predictions are consistent with the shape of the datapoints. Sample 1 has a higher spending in Milk, Grocery and Detergents_Paper than their respective median. Therefore. Sample 0 has higher spending in Fresh, Grocery and Detergents_Paper which indicates it might be a restaurant specilizes in fresh food. Sample 2 has a very high spend in Frozen and Delicatessen than their respective medians which indicates it is a cafe which serves menu items from frozen.

Conclusion

Question 10

Companies often run [A/B tests (https://en.wikipedia.org/wiki/A/B_testing)](https://en.wikipedia.org/wiki/A/B_testing) when making small changes to their products or services. If the wholesale distributor wanted to change its delivery service from 5 days a week to 3 days a week, how would you use the structure of the data to help them decide on a group of customers to test?
Hint: Would such a change in the delivery service affect all customers equally? How could the distributor identify who it affects the most?

Answer: We have defined two groups of customers from the analysis: Restaurants and Retailers. Now, we can use this classification method to design better A/B tests. For example, if the company were to change its delivery service from 5 days a week to 3 days a week, they would respectively run the A/B test on the two customer segments separately. A random set of samples from a group will be divided into two equal subgroups: A and B. Group A will remain on the 5 days a week delivery schedule while group B delivery will be changed to the 3 days a week delivery. After a certain period, we can anlayze the two groups' spending to determine if they were statistically significant.

Impacts of change in delivery service will be different for two customer groups. For Retailers, there will be no statistically significant change or decreasae in spending.However, significant decrease in spending (as well as customer dissatisfacton and complain) might be observed for customers in Restaurant segment.

Question 11

Assume the wholesale distributor wanted to predict a new feature for each customer based on the purchasing information available. How could the wholesale distributor use the structure of the data to assist a supervised learning analysis?
Hint: What other input feature could the supervised learner use besides the six product features to help make a prediction?

Answer: Two defined customer groups/clusters are classified in dataset: Retailer (0) and Restaurant (1). This customer group label can be used as the target in a supervised learner. Therfore, a classifier algorithm can be trained and tuned on existing dataset. The we can use this trained model to predict the customer segments for new customers.

Visualizing Underlying Distributions

At the beginning of this project, it was discussed that the `'Channel'` and `'Region'` features would be excluded from the dataset so that the customer product categories were emphasized in the analysis. By reintroducing the `'Channel'` feature to the dataset, an interesting structure emerges when considering the same PCA dimensionality reduction applied earlier on to the original dataset.

Run the code block below to see how each data point is labeled either `'HoReCa'` (Hotel/Restaurant/Cafe) or `'Retail'` the reduced space. In addition, you will find the sample points are circled in the plot, which will identify their labeling.

```
In [75]:  # Display the clustering results based on 'Channel' data
          rs.channel_results(reduced_data, outliers, pca_samples)
```
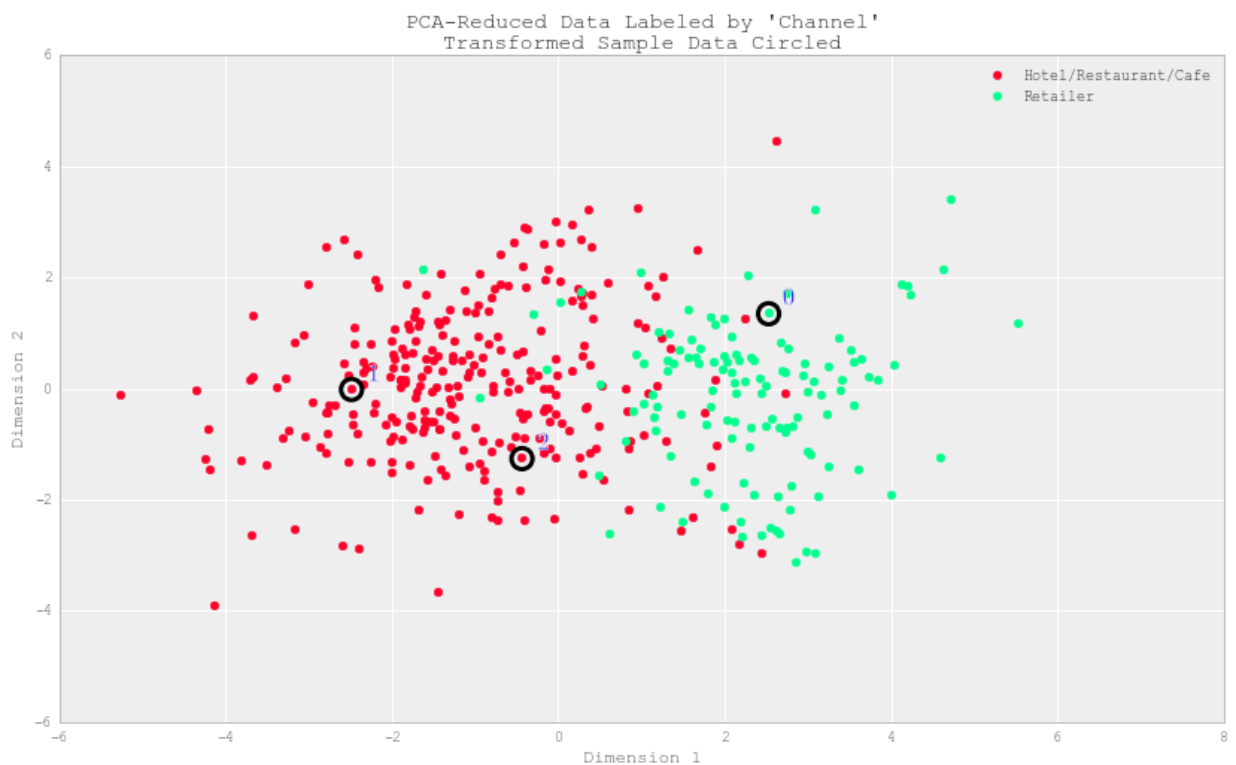

Question 12

How well does the clustering algorithm and number of clusters you've chosen compare to this underlying distribution of Hotel/Restaurant/Cafe customers to Retailer customers? Are there customer segments that would be classified as purely 'Retailers' or 'Hotels/Restaurants/Cafes' by this distribution? Would you consider these classifications as consistent with your previous definition of the customer segments?

Answer: From the above graph, we can observe two clusters but they are not distinct. There are some overlap in the two clusters. Therefore, pure classification of the customer groups is not possible. That's why, GMM algorithm with two clusters is used instead of K-means. These classifications are also consistent with my previous definition of restaurant/cafe and retailer segments.

Note: Once you have completed all of the code implementations and successfully answered each question above, you may finalize your work by exporting the iPython Notebook as an HTML document. You can do this by using the menu above and navigating to
File -> Download as -> HTML (.html). Include the finished document along with this notebook as your submission.