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 [2]:
```
# Import libraries necessary for this project
import numpy as np
import pandas as pd
import renders as rs
from IPython.display import display # Allows the use of display() for DataFr

# 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
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 [3]:
```
# Display a description of the dataset
display(data.describe())
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delica
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.0
mean	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.
std	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.
min	3.000000	55.000000	3.000000	25.000000	3.000000	3.000
25%	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.2
50%	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.5
75%	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.
max	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943

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 [4]:
```
# TODO: Select three indices of your choice you wish to sample from the data
indices = [30, 53, 92]

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

```
Chosen samples of wholesale customers dataset:
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	18815	3610	11107	1148	2134	2963
1	491	10473	11532	744	5611	224
2	9198	27472	32034	3232	18906	5130

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.

Answer: From the above results we can see that:

Before we begin one can explain how to approach this analysis We can analyse the customers standout purchased segment as it relates to the mean and the 25% and 75% interquartile range (IQ). For example, if the customer purchased a segment that was above the mean purchases of segment in the dataset it is considered to be an outlier because it is above the 75% IQ range. We will analyze the three customers below.

From the above table We can analyze all three customers: Firstly, Analyzing Customer 0. From the table we can see that this customer stands out purchasing the segment Fresh. If we analyze a bit more in detail and make a comparison with the Dataset we can see that this customer purchases Fresh above the average , as well as above the percentage quartiles 25, 50 and 75 respectively. So we can successfully say that this consumer greatly prefers Fresh items so we can assume that this customer is a Vegetarian and Shops at a Organic Market. Secondly, Analyzing Customer 1. From the table we can see that this customer stands out purchasing the segment Grocery. If we analyze a bit more in detail and make a comparison with the Dataset we can see that this customer purchases Grocery above the average, as well as above the Percentage Quartiles of 25, 50 and 75 respectively. So we can successfully say that this consumer greatly prefers Grocery items. So we can assume

that this customer shops at a supermarket. Third and Final, Analyzing Customer 2. From the tabel we can see that this customer stands out purchasing the Segment Grocery. Similar to Customer 1 However this customer purchases far more than the Average and Percentage Quartiles of 25, 50, and 75 of Grocery proporionately greater than Customer 1.

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 [5]:
```
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.cross_validation import cross_val_score


# TODO: Make a copy of the DataFrame, using the 'drop' function to drop the
new_data = data.drop(labels="Milk", axis=1, level=None, inplace=False, erro

X_train = new_data
y_train = data[["Milk"]]

# TODO: Split the data into training and testing sets using the given featu
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_s
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# TODO: Create a decision tree regressor and fit it to the training set
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(X_train, y_train)


# TODO: Report the score of the prediction using the testing set
score = regressor.score(X_test, y_test)
print(score)
print ("The mean score:"), (np.mean(score))
```

```
((330, 5), (110, 5), (330, 1), (110, 1))
0.173438009379
The mean score: 0.173438009379
```

Question 2

Which feature did you attempt to predict? What was the reported prediction score? Is this feature relevant for identifying a specific customer?
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.

Answer:

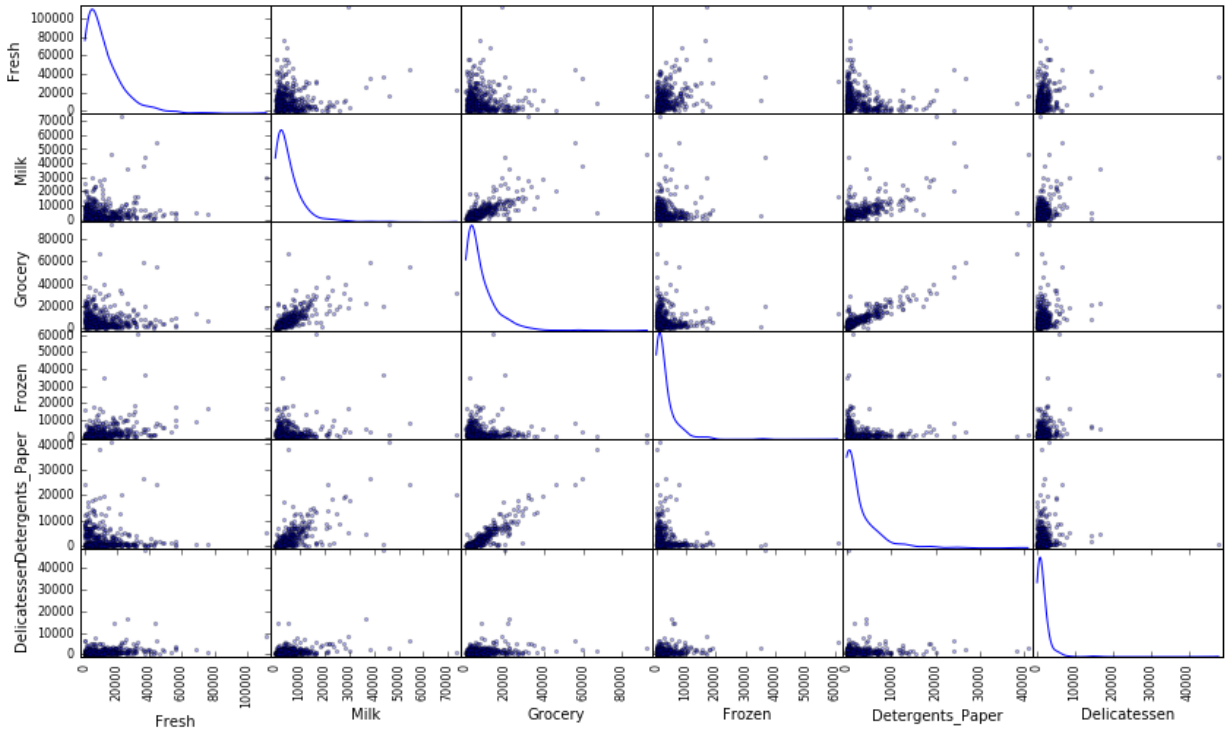
The feature we attempted to predict was "Milk". The reported prediction score is an average score of 0.1734

From the analysis one can say that Milk is an important segment. Why so? Because when we drop the Milk Segment the average score dropped tremendously . One can confidentially say MIlk is a relevant Customer Segment. From the model we ran a small experimental test to derive the importance of milk. How did we find this out? We did this by dropping another segment such as Grocery. When we ran this we got a an average score of 0.963. What this showed us is that Grocery was not as relevant in comparison to the relevance of the segment Milk. Where as when dropping Milk we got an average of 0.1734 So one can clearly see the relevance of MIlk in the Dataset and the outcome of it, it were to be dropped it affects the average value greatly.

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 [6]:
```
# Produce a scatter matrix for each pair of features in the data
pd.scatter_matrix(data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
```



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 the above diagram there are few pairs of features in which that we can state exhibit some degree of correlation but only will be specified. The first is the correlation between Milk and Delicatessen. From assessing the diagram we can determine hypothetically that there is a form of correlation The second is the correlation between Grocery and Milk. From assessing the diagram we can determine hypothetically that there is a postive correlation between the two.

However much we analyze the graphs one can confidentally state that there are abnormalities with the data diagrams.We can look and see that the data is skewed which means in simple terms that the data is biased. We can also see that the data is not Normally distributed.

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

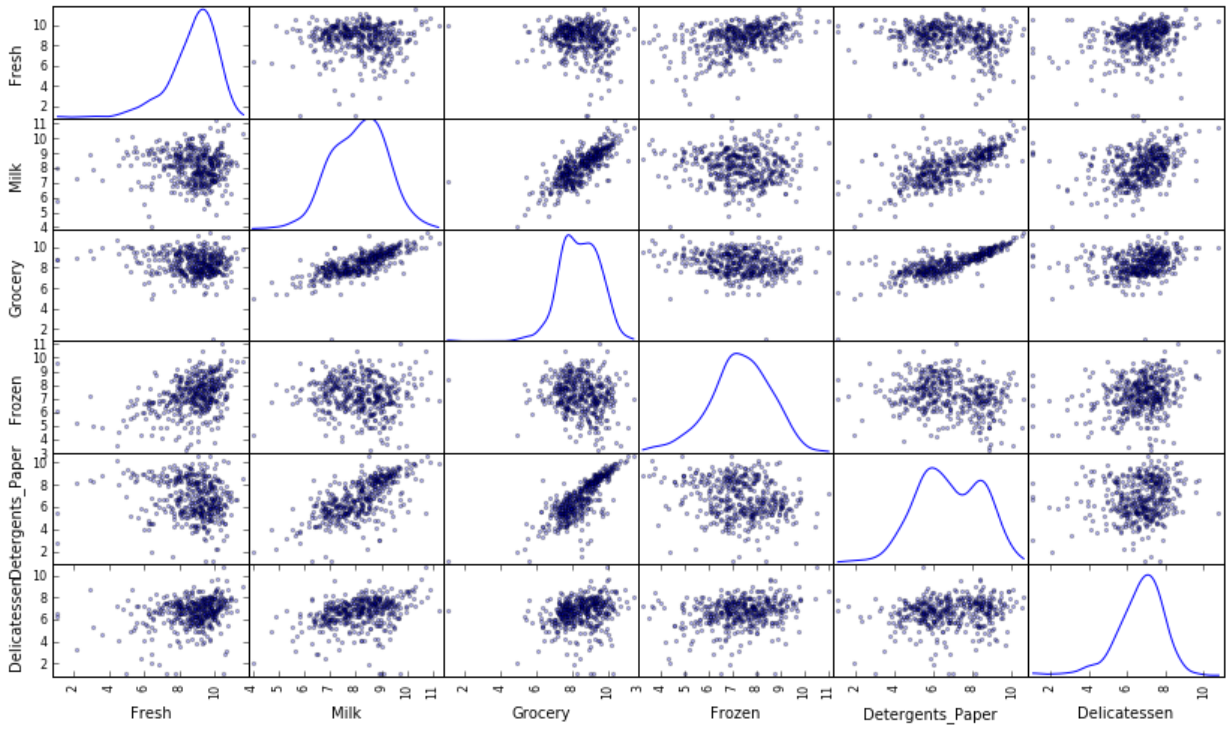
If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate (http://econbrowser.com/archives/2014/02/use-of-logarithms-in-economics) to apply a non-linear scaling — particularly for financial data. One way to achieve 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 [7]:
```
# 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)
print(log_samples)

# Produce a scatter matrix for each pair of newly-transformed features
pd.scatter_matrix(log_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde')
```

```
      Fresh       Milk    Grocery    Frozen  Detergents_Paper  Delicatessen
0  9.842410   8.191463   9.315331  7.045777          7.665753      7.993958
1  6.196444   9.256556   9.352881  6.612041          8.632484      5.411646
2  9.126741  10.220923  10.374553  8.080856          9.847235      8.542861
```



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 [8]:
```
# Display the log-transformed sample data
display(log_samples)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9.842410	8.191463	9.315331	7.045777	7.665753	7.993958
1	6.196444	9.256556	9.352881	6.612041	8.632484	5.411646
2	9.126741	10.220923	10.374553	8.080856	9.847235	8.542861

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 [9]:
```
# For each feature find the data points with extreme high or low values
import numpy as np

# Keep outlier indices in a list and examine after looping thru the features
idx = []

for feature in log_data.keys():

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

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

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

    # Gather the indexes of all the outliers
    idx += log_data[~((log_data[feature] >= Q1 - step) & (log_data[feature]

    # Display the outliers
    #print "Data points considered outliers for the feature '{}':".format(fe
    #dataset = display(log_data[~((log_data[feature] >= Q1 - step) & (log_da

print(sorted(idx))
# OPTIONAL: Select the indices for data points you wish to remove
outliers  = idx


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

```
[38, 57, 65, 65, 66, 66, 75, 75, 81, 86, 95, 96, 98, 109, 128, 128, 137,
 142, 145, 154, 154, 154, 161, 171, 175, 183, 184, 187, 193, 203, 218, 23
3, 264, 285, 289, 304, 305, 325, 338, 343, 353, 355, 356, 357, 412, 420,
 429, 439]
         Fresh       Milk    Grocery    Frozen  Detergents_Paper  Delicat
essen
0     9.446913   9.175335   8.930759  5.365976          7.891331      7.1
98931
1     8.861775   9.191158   9.166179  7.474205          8.099554      7.4
82119
2     8.756682   9.083416   8.946896  7.785305          8.165079      8.9
67504
3     9.492884   7.086738   8.347827  8.764678          6.228511      7.4
88853
4    10.026369   8.596004   8.881558  8.272571          7.482682      8.5
53525
5     9.149847   9.019059   8.542081  6.501290          7.492760      7.2
80008
6     9.403107   8.070594   8.850088  6.173786          8.051978      6.3
00786
7     8.933137   8.508354   9.151227  7.419980          8.108021      7.8
50104
8     8.693329   8.201934   8.731013  6.052089          7.447751      6.6
20073
9     8.700514   9.314070   9.845911  7.055313          8.912608      7.6
48740
10    8.121480   8.594710   9.470703  8.389360          8.695674      7.4
63937
11    9.483873   7.024649   8.416931  7.258412          6.308098      6.2
08590
12   10.364514   9.418898   9.372204  5.659482          8.263848      7.9
83099
13    9.962558   8.733594   9.614605  8.037543          8.810907      6.4
00257
14   10.112654   9.155356   9.400217  5.683580          8.528726      7.6
81560
15    9.235326   7.015712   8.248267  5.983936          6.871091      6.0
21023
16    6.927558   9.084324   9.402695  4.897840          8.413609      6.9
84716
17    8.678632   8.725345   7.983781  6.732211          5.913503      8.4
06932
18    9.830971   8.752581   9.220192  7.698483          7.925519      8.0
64951
19    8.959312   7.822044   9.155250  6.505784          7.831220      6.2
16606
20    9.772581   8.416046   8.434246  6.971669          7.722678      7.6
61056
21    8.624612   6.769642   7.605890  8.126518          5.926926      6.3
43880
22   10.350606   7.558517   8.404920  9.149316          7.775276      8.3
74246
23   10.180096  10.502956   9.999661  8.547528          8.374938      9.7
12509
24   10.027783   9.187686   9.531844  7.977625          8.407825      8.6
61813
25    9.690604   8.349957   8.935245  5.303305          8.294799      4.0
```

```
43051
26    9.200088   6.867974   7.958926  8.055475          5.488938      6.7
25034
27    9.566335   6.688355   8.021256  6.184149          4.605170      6.2
49975
28    8.321908   9.927399  10.164197  7.054450          9.059982      8.5
57567
29   10.671000   7.649693   7.866722  7.090077          7.009409      6.7
12956
..         ...        ...        ...       ...               ...
  ...
368  10.129268   8.519191   8.522380  9.190750          6.995766      6.8
66933
369   8.904087   9.460632   9.835369  7.956477          8.972464      6.3
15358
370   9.018817   8.263590   8.765146  7.406103          7.912057      5.8
40642
371   9.071997   8.198089   8.716044  7.761745          7.660585      8.5
44225
372   8.799812   7.647786   8.425736  7.236339          7.528332      7.5
45390
373   7.661998   8.098339   8.095904  7.336286          5.459586      8.3
81373
374   8.513787   8.488588   8.799812  9.790655          6.815640      7.7
97702
375   8.694335   7.595890   8.136518  8.644530          7.034388      5.6
69881
376   8.967249   8.707152   9.053920  7.433075          8.171882      7.5
35830
377   8.386857   9.300181   9.297252  6.742881          8.814033      6.9
00731
378   8.530109   8.612322   9.310638  5.897154          8.156223      6.9
68850
379   6.492240   9.047115   9.832099  4.890349          8.815815      6.6
54153
380   9.089415   8.238273   7.706613  6.450470          7.365180      7.3
27123
381   9.744668   8.486115   9.110851  6.938284          8.135933      7.4
86613
382  10.181119   7.227662   8.336151  6.721426          6.854355      7.1
04965
383   9.773664   8.212297   8.446127  6.965080          7.497207      6.5
04288
384   9.739791   7.966933   9.411811  6.773080          8.074960      5.5
17453
385   9.327501   7.786552   7.860571  9.638740          4.682131      7.5
42213
386   9.482960   9.142811   9.569133  8.052296          8.532870      7.5
46446
387  10.342130   9.722385   8.599510  9.621257          6.084499      7.0
58758
388   8.021913   8.694502   8.499029  7.695303          6.745236      5.7
58902
389   8.038189   8.349957   9.710085  6.354370          5.484797      7.6
40123
390   9.051696   8.613594   8.548692  9.509407          7.227662      7.3
11886
```

```
391   9.957834   7.057898   8.466742  5.594711          7.191429      5.9
78886
392   7.591862   8.076515   7.308543  7.340187          5.874931      7.2
78629
393   9.725019   8.274357   8.986447  6.533789          7.771067      6.7
31018
394  10.299003   9.396903   9.682030  9.483036          5.204007      7.6
98029
395  10.577146   7.266129   6.638568  8.414052          4.532599      7.7
60467
396   9.584040   9.647821  10.317020  6.079933          9.605149      7.5
32088
397   9.238928   7.591357   7.710653  6.945051          5.123964      7.6
61527

[398 rows x 6 columns]
```

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:

One can say, yes there are data points considered to be ouliers for more than one feature, we can see confidentially say this by looking at the results generated in the table above

The multipe category outlier values are:[65, 66, 75, 128, 154, 154]

If we normalize the data, the outliers won't necessarily need to be removed, given that all the datapoints will be centered with a mean of 0 and will all fall within a set range with a normal distribution.

Data points were added to the outlier list to be removed because points which are outliers can skew the data by not creating Normal Distribution. In simple terms in Normal Distribution everything should be on the same scale if not this causes the data to be Biased. In conclusion one or more outliers create distortion in the data so hence we remove them because generally it will give a better understanding of the customer base.

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.

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

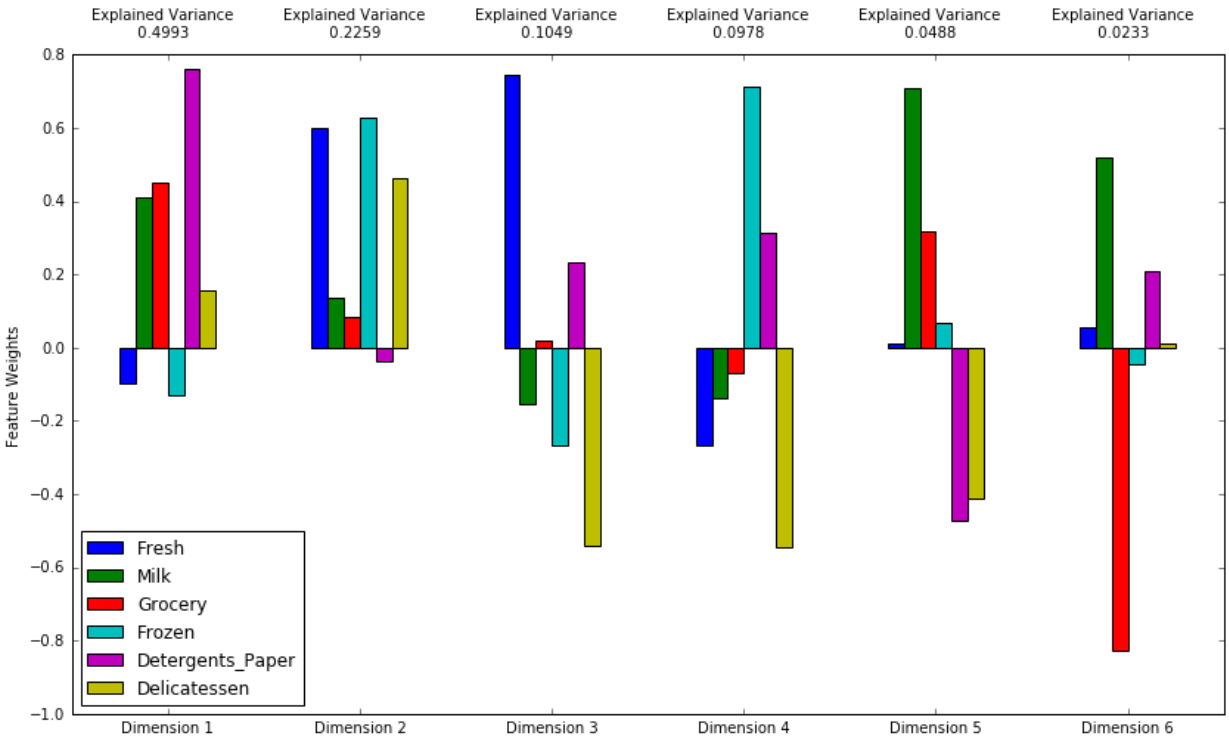
- Import `sklearn.preprocessing.PCA` and assign the results of fitting PCA in six dimensions with `good_data` to `pca`.
- Apply a PCA transformation of the sample log-data `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

In [10]:
```
# TODO: Apply PCA to the good data with the same number of dimensions as fea
from sklearn.decomposition import PCA
pca = PCA(n_components=6, copy=True, whiten=False)
pca.fit(good_data)
print (good_data)

# TODO: Apply a PCA transformation to the sample log-data
pca_samples = pca.transform(log_samples)
print (pca_samples)

# Generate PCA results plot
pca_results = rs.pca_results(good_data, pca)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
2	8.756682	9.083416	8.946896	7.785305	8.165079	8.967504
3	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853
4	10.026369	8.596004	8.881558	8.272571	7.482682	8.553525
5	9.149847	9.019059	8.542081	6.501290	7.492760	7.280008
6	9.403107	8.070594	8.850088	6.173786	8.051978	6.300786
7	8.933137	8.508354	9.151227	7.419980	8.108021	7.850104
8	8.693329	8.201934	8.731013	6.052089	7.447751	6.620073
9	8.700514	9.314070	9.845911	7.055313	8.912608	7.648740
10	8.121480	8.594710	9.470703	8.389360	8.695674	7.463937
11	9.483873	7.024649	8.416931	7.258412	6.308098	6.208590
12	10.364514	9.418898	9.372204	5.659482	8.263848	7.983099
13	9.962558	8.733594	9.614605	8.037543	8.810907	6.400257
14	10.112654	9.155356	9.400217	5.683580	8.528726	7.681560
15	9.235326	7.015712	8.248267	5.983936	6.871091	6.021023
16	6.927558	9.084324	9.402695	4.897840	8.413609	6.984716
17	8.678632	8.725345	7.983781	6.732211	5.913503	8.406932
18	9.830971	8.752581	9.220192	7.698483	7.925519	8.064951
19	8.959312	7.822044	9.155250	6.505784	7.831220	6.216606
20	9.772581	8.416046	8.434246	6.971669	7.722678	7.661056
21	8.624612	6.769642	7.605890	8.126518	5.926926	6.343880
22	10.350606	7.558517	8.404920	9.149316	7.775276	8.374246
23	10.180096	10.502956	9.999661	8.547528	8.374938	9.712509
24	10.027783	9.187686	9.531844	7.977625	8.407825	8.661813
25	9.690604	8.349957	8.935245	5.303305	8.294799	4.043051
26	9.200088	6.867974	7.958926	8.055475	5.488938	6.725034
27	9.566335	6.688355	8.021256	6.184149	4.605170	6.2

```
49975
28    8.321908   9.927399  10.164197  7.054450          9.059982      8.5
57567
29   10.671000   7.649693   7.866722  7.090077          7.009409      6.7
12956
..         ...        ...        ...       ...               ...
  ...
368  10.129268   8.519191   8.522380  9.190750          6.995766      6.8
66933
369   8.904087   9.460632   9.835369  7.956477          8.972464      6.3
15358
370   9.018817   8.263590   8.765146  7.406103          7.912057      5.8
40642
371   9.071997   8.198089   8.716044  7.761745          7.660585      8.5
44225
372   8.799812   7.647786   8.425736  7.236339          7.528332      7.5
45390
373   7.661998   8.098339   8.095904  7.336286          5.459586      8.3
81373
374   8.513787   8.488588   8.799812  9.790655          6.815640      7.7
97702
375   8.694335   7.595890   8.136518  8.644530          7.034388      5.6
69881
376   8.967249   8.707152   9.053920  7.433075          8.171882      7.5
35830
377   8.386857   9.300181   9.297252  6.742881          8.814033      6.9
00731
378   8.530109   8.612322   9.310638  5.897154          8.156223      6.9
68850
379   6.492240   9.047115   9.832099  4.890349          8.815815      6.6
54153
380   9.089415   8.238273   7.706613  6.450470          7.365180      7.3
27123
381   9.744668   8.486115   9.110851  6.938284          8.135933      7.4
86613
382  10.181119   7.227662   8.336151  6.721426          6.854355      7.1
04965
383   9.773664   8.212297   8.446127  6.965080          7.497207      6.5
04288
384   9.739791   7.966933   9.411811  6.773080          8.074960      5.5
17453
385   9.327501   7.786552   7.860571  9.638740          4.682131      7.5
42213
386   9.482960   9.142811   9.569133  8.052296          8.532870      7.5
46446
387  10.342130   9.722385   8.599510  9.621257          6.084499      7.0
58758
388   8.021913   8.694502   8.499029  7.695303          6.745236      5.7
58902
389   8.038189   8.349957   9.710085  6.354370          5.484797      7.6
40123
390   9.051696   8.613594   8.548692  9.509407          7.227662      7.3
11886
391   9.957834   7.057898   8.466742  5.594711          7.191429      5.9
78886
392   7.591862   8.076515   7.308543  7.340187          5.874931      7.2
78629
```

```
393   9.725019   8.274357   8.986447  6.533789          7.771067      6.7
31018
394  10.299003   9.396903   9.682030  9.483036          5.204007      7.6
98029
395  10.577146   7.266129   6.638568  8.414052          4.532599      7.7
60467
396   9.584040   9.647821  10.317020  6.079933          9.605149      7.5
32088
397   9.238928   7.591357   7.710653  6.945051          5.123964      7.6
61527

[398 rows x 6 columns]
[[  1.24907260e+00   9.10271687e-01   3.46694943e-01  -9.61462718e-01
   -5.88253640e-01  -4.43645646e-01]
 [  2.44155830e+00  -2.63739678e+00  -7.91009216e-01   1.26375378e+00
    7.13975166e-01   7.61547397e-02]
 [  4.24152406e+00   1.67293072e+00  -5.40010312e-01   3.26500605e-03
   -1.17368876e-02   1.02845434e-01]]
```



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 total explained variance for the first and second principal components is by finding the sum of the two explained variance values which are:(0.4424 + 0.2766). The sum of these two numbers gives us the number (0.719) which is the total of the first and second principal component. The same method how we found out the total variance of the first and second principal components is also the same method that we will calculate the first four principal components. Which is simply finding the sum of:(0.4424 + 0.2766 + 0.1162 +0.0962). Which amounts too the value (0.9314).

From the vizualization above one can analyze and discuss about the First Four Dimensions determining customer spending:

From the First Graphical Dimension, Dimension 1. We can see a correlation that there is a major increase in spending on Detergents_Paper which significantly stands out over the others as the most positive increase, followed by a fairly positive increase in Grocery, closely followed by Milk and then somewhat of a satisfactory increase in Delicatessen. However we can see negative decreases in spending on segments such as Milk and the most negative decrease on Frozen.

From the Second Graphical Dimension, Dimension 2. From this correlation we do not see a significant standout 'towering' as we saw in Dimension 1. However the most positive increase in spending on a segment in this case is Frozen. This is followed by Fresh which can be concerned second which isn't too far and Delicatessen in third which has a fairly positive increase, Milk and Grocery have positive but satisfactory increases. There is only negative decrease in this Dimension and that is the only decrease of the segment Detergents_Paper which is not a major decrease in comparison to other segments like we saw in Dimension1.

Thirdly the Third Graphical Dimension, Dimension 3. From this correlation we can clearly see two "towering" opposites of one being positive and negative. The positive increase in spending in Fresh which uniquely stands out over the fairly positive increase in Detergents_Paper and Grocery just barely being a positive increase. However Unlike Dimension 1 and Dimension 2 which stood out proportionately greater on the positive side, We can see that Dimension 3 stands out proportionately greater on the negative sides with a large decrease in Milk followed by larger Decrease in Frozen and the largest Decrease in Delicatessen. But we can make another analysis and see that the positive increase in Fresh is proportionately greater than the negative decrease in Delicatessen.

Fourth and Final the Fourth Graphical Dimension, Dimension 4. From this correlation, we can see quite a bit of negative decreases even more than Dimension 3. We can first by looking at the only 2 positives. The major increase we can see "towering" is the Frozen segment and then further down not competing with Frozen is Detergents_Paper which stands fairly positive. But as we said earlier our main center of attraction to focus on are the negative decreases starting with the Delicatessen which stands out as the most negative decrease followed by Fresh, Milk and Grocery in this order. We can definitely see that this Dimension is far more proportionately negative then it is positive in comparison to the other 3 Dimensions.

In general we get an idea how customer spending correlates on one segment and greatly affects the other segment within the same Dimension.

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 [11]:
```
# Display sample log-data after having a PCA transformation applied
display(pd.DataFrame(np.round(pca_samples, 4), columns = pca_results.index.v
```

	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5	Dimension 6
0	1.2491	0.9103	0.3467	-0.9615	-0.5883	-0.4436
1	2.4416	-2.6374	-0.7910	1.2638	0.7140	0.0762
2	4.2415	1.6729	-0.5400	0.0033	-0.0117	0.1028

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 [12]:
```
import numpy as np
from sklearn.decomposition import PCA

# TODO: Fit PCA to the good data using only two dimensions
pca = PCA(n_components=2, copy=True, whiten=False)
pca.fit(good_data)
print (good_data)

# TODO: Apply a PCA transformation the good data
reduced_data = pca.transform(good_data)
print (reduced_data)

# TODO: Apply a PCA transformation to the sample log-data
pca_samples = pca.transform(log_samples)
print (pca_samples)

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

```
         Fresh       Milk    Grocery    Frozen  Detergents_Paper  Delicat
essen
0     9.446913   9.175335   8.930759  5.365976          7.891331      7.1
98931
1     8.861775   9.191158   9.166179  7.474205          8.099554      7.4
82119
2     8.756682   9.083416   8.946896  7.785305          8.165079      8.9
67504
3     9.492884   7.086738   8.347827  8.764678          6.228511      7.4
88853
4    10.026369   8.596004   8.881558  8.272571          7.482682      8.5
53525
5     9.149847   9.019059   8.542081  6.501290          7.492760      7.2
80008
6     9.403107   8.070594   8.850088  6.173786          8.051978      6.3
00786
7     8.933137   8.508354   9.151227  7.419980          8.108021      7.8
50104
8     8.693329   8.201934   8.731013  6.052089          7.447751      6.6
```

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 [13]:
```
# Display sample log-data after applying PCA transformation in two dimensio
display(pd.DataFrame(np.round(pca_samples, 4), columns = ['Dimension 1', 'Di
```

	Dimension 1	Dimension 2
0	1.2491	0.9103
1	2.4416	-2.6374
2	4.2415	1.6729

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:

The Advantages of Using K-means algorithm are:

-It is very easy to implement

-It is computationally very efficient compared to other cluster algorithms

-It is very effective in identifying clusters of spherical shape

The Advantages of using Gaussian Mixture Model algorithm:

-The Gaussian Process relys more on the family of functions rather than parameters. -(Speed) It is the fastest algorithm for learning mixture models -(Agnostic) AS the algorithm maximizes only the likelihood, it will not bias the means towards zero, or bias the cluster sizes to have specific structures that might or might not apply.

The main difference between the two is that K means is a form of hard clustering and Gaussian is a form of soft clustering. We can remind ourselves that soft clustering is where one sample is assigned to one or more clusters and that hard clustering is where one dataset set is assigned to exactly one cluster

We can use both algorithims, However one would prefer K-Means because of its simplicity.

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 [14]:
```
# TODO: Apply your clustering algorithm of choice to the reduced data
from sklearn.cluster import KMeans

# fit cluster to reduced data
clusterer = KMeans(n_clusters = 2, init='k-means++', n_init=10, max_iter=300,
clusterer.fit(reduced_data)

# predict cluster for each data point
preds = clusterer.predict(reduced_data)

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

#print (centers)

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

# TODO: Calculate the mean silhouette coefficient for the number of clusters
from sklearn.metrics import silhouette_score
score = silhouette_score(reduced_data, preds)
print (score)
```

```
[1 1 1]
0.447157742293
```

Question 7

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

Answer:

In this case the algorithm was run several times using the numbers 1 through to 10 in the n_clusterers.

When n_clusterer = 2, The silhouette_score = 0.447

When n_clusterer = 3, The silhouette_score = 0.364

When n_clusterer = 4, The silhouette_score = 0.331

When n_clusterer = 5, The silhouette_score = 0.352

When n_clusterer = 6, The silhouette_score = 0.363

When n_clusterer = 7, The silhouette_score = 0.355

When n_clusterer = 8, The silhouette_score = 0.367

When n_clusterer = 9, The silhouette_score = 0.367

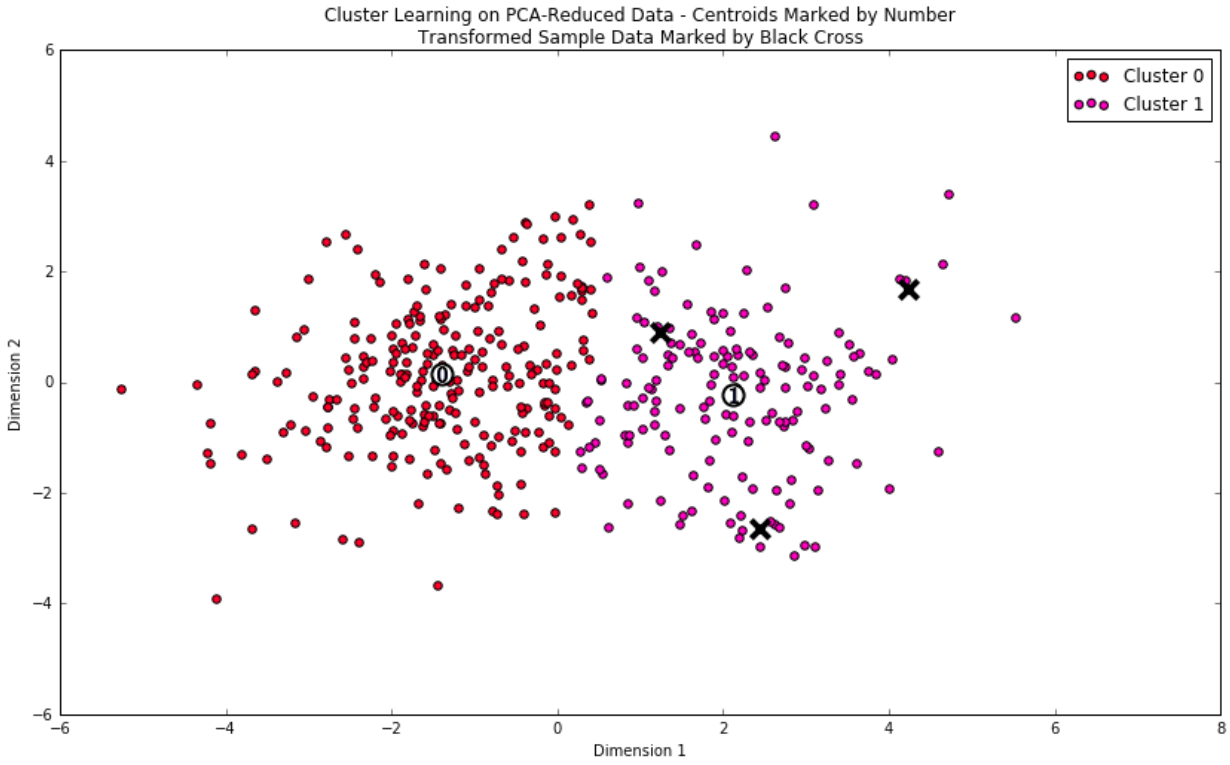
When n_clusterer = 10,The silhouette_score = 0.352

From the several outcomes generated one can confidentially say that the clusteres are most accurately defined when the n_clusteres = 2 which generates the score with a value of 0.447

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 [15]:
```
# 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 [16]:
```
# TODO: Inverse transform the centers
log_centers = pca.inverse_transform(centers)
print (log_centers)

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

```
[[ 9.15385615  7.56949933  7.80360017  7.69608444  5.72648239  6.6471685
7]
 [ 8.59862529  8.95935377  9.352869    7.02384337  8.39920005  7.0351473
1]]
[[  9450.81398866   1938.16965459   2449.40439917   2199.71797803
     306.88785424    770.59933661]
 [  5424.19778156   7780.32799336  11531.86089437   1123.09480755
    4443.51073162   1135.86222037]]
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Segment 0	9451.0	1938.0	2449.0	2200.0	307.0	771.0
Segment 1	5424.0	7780.0	11532.0	1123.0	4444.0	1136.0

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'`.

Answer:

Each cluster represents a certain customer on average. Basically the cluster algorithim shows a good representation of each customer and what category they are primarily spending on the most. To conclude we can succesfully say, Based on the customers spending we can categorize and determine what type of customers they really are.

Such as, Segment 0 which shows us that there is a higher preference towards the customer spending on Fresh in comparison to the other categories. However we can make another small analyses that customers who purchased Fresh also purchased quite a bit of Grocery and minimal amounts of Milk, Frozen and Delicatessen .We can also compare this data to the customer dataset on how much they spend on the categories. Customer(s) spends above the 1st Quartiles but still

consumes Fresh far below the 2nd and 3rd Quartile. Although he/she spends quite a bit on Grocery customer(s) still purchases below the average in comparison to the mean purchase of Fresh. Basically, customer(s) purchases under the average of all the categories within this Segment.

From the Analyses of Segment 1 we can see that there is a higher preference towards the customer spending on Grocery in comparison to the other categories. We can also see that customers also spend fairly on Milk, Fresh and Detergents_Paper in that order but not as much on Frozen and least on Delicatessen. We can compare this data to the customer dataset on how much they spend on all categories. Customer(s) spends far above average on Grocery and Milk and Delicatessen. But Spends below average on Fresh, Frozen and Detergents_Paper.

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 [17]:
```
# 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 1
Sample point 2 predicted to be in Cluster 1
```

```
**Answer:**

Sample 0,1 and 2 were predicted to fall in cluster 1.  Fresh and Frozen
were below average while milk and groceries were above average.

Yes, one can successfully say that the predictions for each sample point
are consistent with this. We can say this because earlier we found out
that sample_preds prints an array (1, 1, 1).
```

Conclusion

Question 10

Companies often run A/B tests (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: To perform A/B on the data, test the change on delivery to one group (experimental group) and see how they respond to it. If they respond well, apply the delivery service to the 'control group'. For instance, if the distributor wants to roll out a bulk delivery service to customers that largely

purchases a customer segment, then run the service on an experimental group, or a subset of the dataset or a sample of customers from a particular cluster. Then if the experiment goes well, test the service on the control group or the entire cluster. For example if we plan to change the delivery service for customers who purchase Grocery, Wholesale Distributors main clientel are the SuperMarkets. The Distributor can try it on a few supermarkets first and analyze the results to determine whether the service receives positve feedback or negative feedback. If the results are positve and the customer which is the Supermarket is satsified then the Distributor can go ahead and implement this with all of its customers.

Question 11

Assume the wholesale distributor wanted to predict some other 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?

Answer:

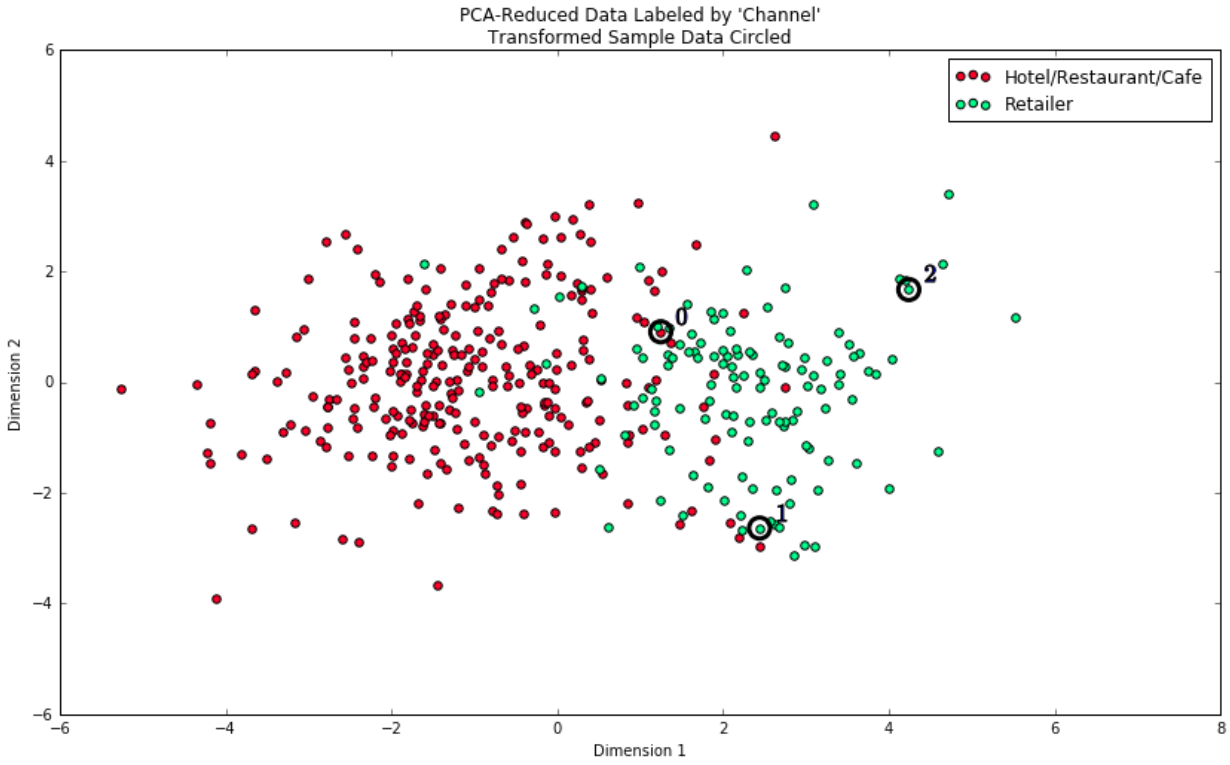
Given that we classified the customers into clusters or segments, we now have a new feature to add to the dataset. So adding to the previous known features, we have added the cluster label classifications as an additional feature. This updated dataset can be used towards learning some new feature using supervised learning. For example, we could train a supervised learning model to predict whether a particular segment responds positively or negatively to a new bulk delivery service given the additional cluster information.

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 [18]:
```
# 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 graph we cannot correctly classify samples within the dataset because we see samples overlapping, For example we see some red data points in areas alongside with green points and vice-versa. Unlike the PCA Reduced_Data which can be classified this cannot be classified. One can only see one pure cluster which is Data point 2. Apart from that all data clusters look mixed up or chaotic.

Finally, we can say that these classifications are not consistent because the clusters cannot be classified.

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.