Machine Learning Engineer Nanodegree

Unsupervised Learning

Project: Creating Customer Segments

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

# Import supplementary visualizations code visuals.py
import visuals as vs

# Pretty display for notebooks
%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.".fo
rmat(*data.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())
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	
count	440.000000	440.000000	440.000000	440.000000	440.000000	4
mean	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1
std	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2
min	3.000000	55.000000	3.000000	25.000000	3.000000	3
25%	3127.750000	1533.000000	2153.000000	742.250000	256.750000	4
50%	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	9
75%	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1
max	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	4

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 [3]:

```
# TODO: Select three indices of your choice you wish to sample from the dataset
indices = [21, 152, 338]

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

Chosen samples of wholesale customers dataset:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	5567	871	2010	3383	375	569
1	18840	1371	3135	3001	352	184
2	3	333	7021	15601	15	550

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, delis, wholesale retailers, among many others. Avoid using names for establishments, such as saying "*McDonalds*" when describing a sample customer as a restaurant. You can use the mean values for reference to compare your samples with. The mean values are as follows:

- Fresh: 12000.2977
- Milk: 5796.2
- Grocery: 7951.3
- Detergents_paper: 2881.4
- Delicatessen: 1524.8

Knowing this, how do your samples compare? Does that help in driving your insight into what kind of establishments they might be?

Answer:

- The customer at index 21 (displayed at row 0 in the chart above) has spending amounts that fall just below or just above the 25th percentile for the delicatessen, detergents/paper, grocery, and milk categories. The fact that this customer spends 3,383 monetary units on the frozen category is interesting in that this puts them above the 75th percentile for that category. Finally, although the amount this customer spends on fresh foods is less than the mean amount spent on this category by all customers, at 5,567 units, we see that this customer spends more here than in any other category.

  To sum it up, this customer spends the majority of their funds on fresh items, and buys more frozen goods than the majority of other customers. I would posit that this customer is some sort of mid-sized restaurant, likely independent (not a franchise), that serves a decent variety of dishes. The money spent on frozen foods could be for staples or basic ingredients that can be kept for a long time. The funds spent on fresh food could be for the key ingredients that go into entrees and appetizers that this restaurant serves. The relatively smaller amounts spent in other categories could be for sundry supplies and or minor ingredients.

- The customer at index 152 (displayed at row 1 in the chart above) spends an overwhelmingly large amount on fresh foods (well above the 75th percentile) at 18,840 monetary units. This is roughly six times what this customer spends on groceries, six times what is spent on frozen foods, 18 times what is spent on milk, more than 50 times what's spent on detergents/paper, and over 100 times what's spent on delicatessen.

Because this customer by far and away cares about fresh goods, with a little bit of extra attention paid to some extra grocery, frozen, and milk items, I hypothesize that this customer is a decent-sized market that sells fresh fruit/vegetables and possibly meat/fish as well. Folks primarily go to this market to buy healthy, fresh food, and may grab a small carton of milk or a small box of crackers on their way out the door, in order to save themselves a trip to a more traditional super market.

- The customer at index 338 (displayed at row 2 in the chart above), by contrast, is all about frozen food. While far less than the maximum spend of 60,869 monetary units seen in this category, at an amount of 15,601 units, this customer still spends almost five times the amount at the 75th percentile for frozen foods.

  All in all, this customer spends more money than most businesses on frozen foods, and then adds to this a non-trivial amount of money that they spend on groceries. All other categories are an afterthought. This tells me this customer is somewhat larger in size, and needs mostly frozen foods, with some other groceries as well. I wouldn't be surprised if this customer is a fast food restaurant, with a need for large amounts of frozen beef patties, chicken, etc. Their spend on groceries could represent bread and other condiments such as ketchup and mustard. The fact that this customer spends nearly nothing on fresh foods tells me that their offerings possibly aren't terribly healthy (e.g. no lettuce, fresh tomatoes, etc.) -- maybe they serve a kind of no-frills, "meat-and-potatoes" equivalent of fast food that has an appeal to a certain segment of the Portuguese population.

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 [4]:

```
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor

# TODO: Make a copy of the DataFrame, using the 'drop' function to drop the give
n feature
new_data = data.drop(['Detergents_Paper'], axis = 1)

# TODO: Split the data into training and testing sets(0.25) using the given feat
ure as the target
# Set a random state.
target_data = data['Detergents_Paper']
X_train, X_test, y_train, y_test = train_test_split(new_data,
                                                    target_data,
                                                    test_size = 0.25,
                                                    random_state = 42)

# TODO: Create a decision tree regressor and fit it to the training set
regressor = DecisionTreeRegressor(random_state=42)
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("The reported prediction score for estimating 'Detergents_Paper' category
spending amounts is {}.".format(score))
```

```
The reported prediction score for estimating 'Detergents_Paper' cate
gory spending amounts is 0.27166698062685013.
```

Question 2

- Which feature did you attempt to predict?
- What was the reported prediction score?
- Is this feature 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. If you get a low score for a particular feature, that lends us to believe that that feature point is hard to predict using the other features, thereby making it an important feature to consider when considering relevance.

Answer:

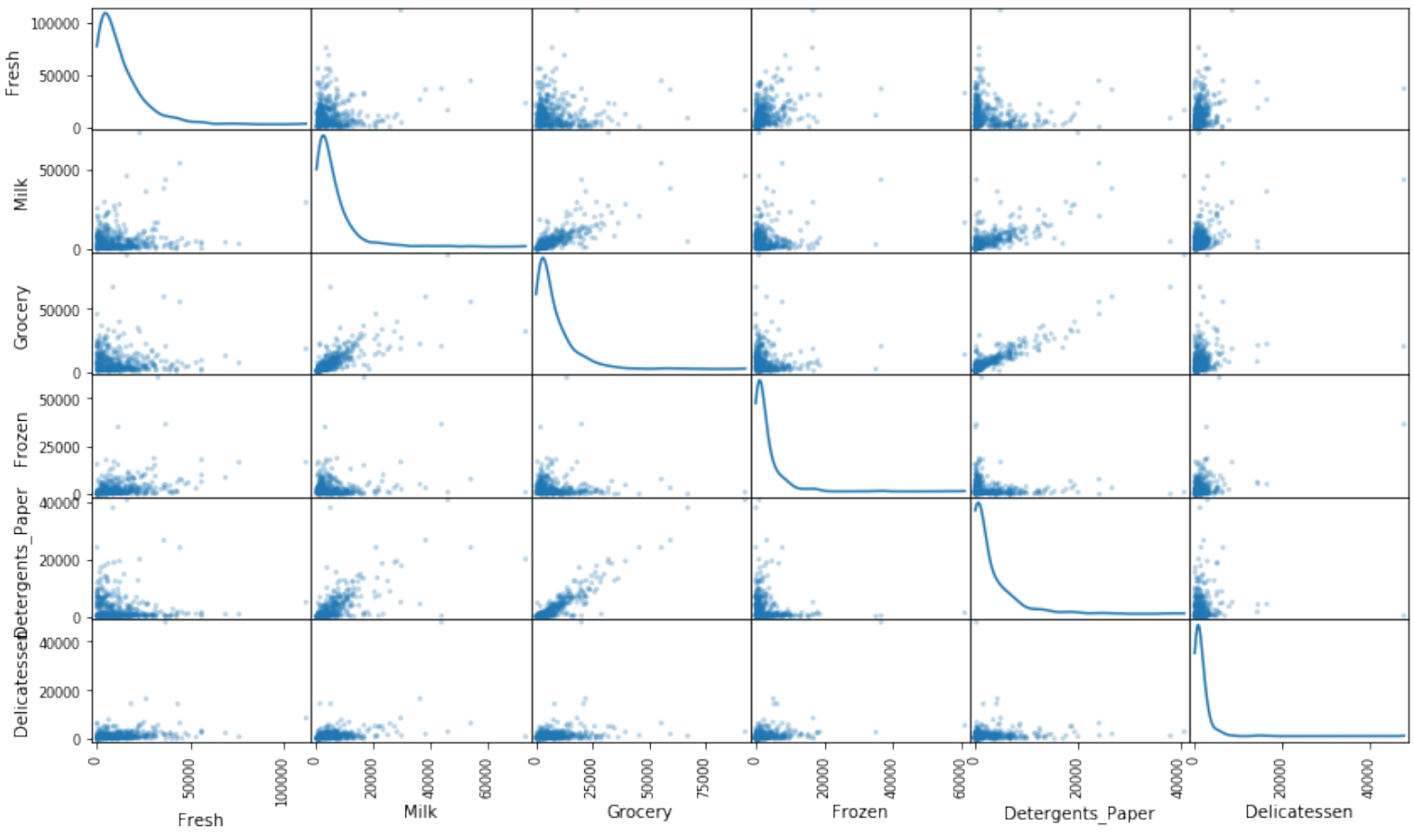
- I attempted to predict the `Detergents_Paper` feature.
- The reported prediction score for estimating `Detergents_Paper` category spending amounts is 0.27166698062685013.
- With its `R^2` score of 0.27 being closer to 0 than to 1, we see that only just over one fourth of the `Detergents_Paper` feature's variance is explained by the variances of the other features -- the `Detergents_Paper` feature's values do not, in fact, correlate with those of the other five features together. This means that the `Detergents_Paper` feature is likely *not* redundant, and may indeed be useful for our task of segmenting customers according to their 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 [5]:

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



Question 3

- Using the scatter matrix as a reference, discuss the distribution of the dataset, specifically talk about the normality, outliers, large number of data points near 0 among others. If you need to sepearate out some of the plots individually to further accentuate your point, you may do so as well.
- 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? You can use corr() (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html) to get the feature correlations and then visualize them using a heatmap (http://seaborn.pydata.org/generated/seaborn.heatmap.html)(the data that would be fed into the heatmap would be the correlation values, for eg: `data.corr()`) to gain further insight.

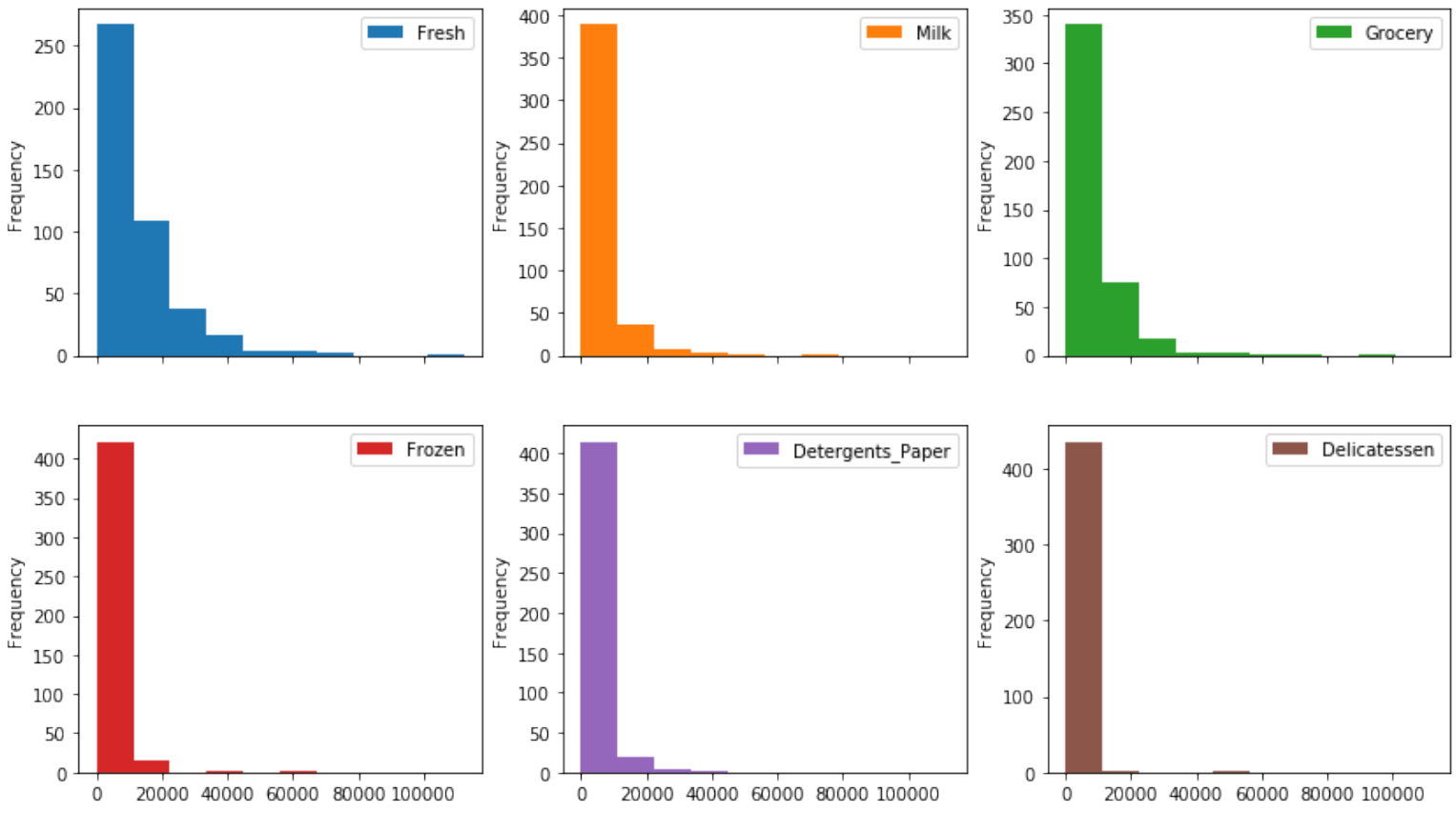
Answer:

- The six features are *not* normally distributed; these features all have positively skewed distributions. From the histograms below, for each category we can see that the overwhelming majority of customers spend a relatively small amount of monetary units on that category. This is especially true for `Frozen`, `Detergents_Paper`, and `Delicatessen` categories.

The majority of the data points for each category fall below its mean, and we can observe that there are a small number of extreme outliers all of our categories, save for perhaps `Detergents_Paper`.

In [6]:

```
data.plot(kind='hist', subplots=True, layout=(2,3), figsize=(14,8));
```

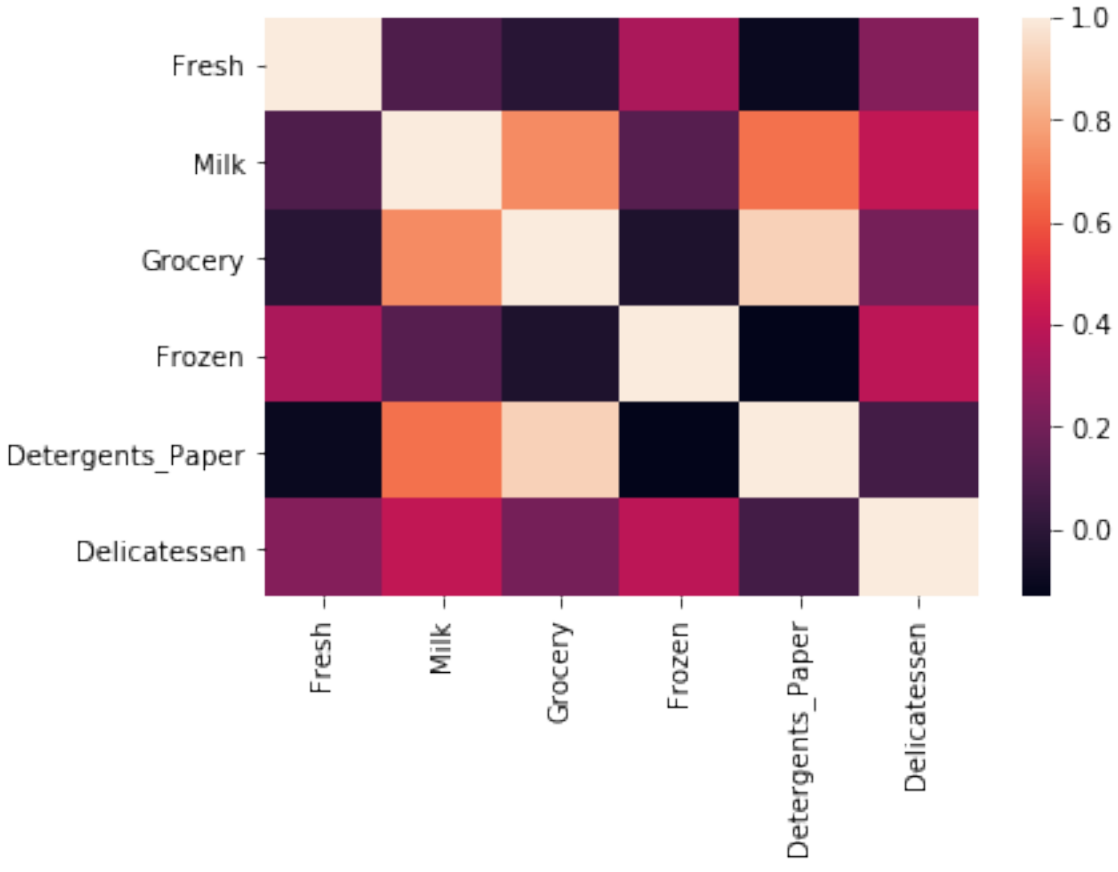


- Referring to the scatter matrix above, I observe the strongest positive correlation between the `Detergents_Paper--Grocery` feature pair. To a lesser extent, I see some correlation between the `Milk--Grocery` pair. Finally, there also appears to be some slight correlation in the `Detergents_Paper--Milk` feature pair.

Indeed, these observations are confirmed by the heatmap displayed below, with `Detergents_Paper--Grocery` clearly having the highest correlation, followed by `Milk--Grocery` that looks to have a correlation around 0.8, itself followed by `Detergents_Paper--Milk` with an ever so slightly lower correlation.

In [7]:

```
import seaborn
correlations = data.corr()
seaborn.heatmap(correlations);
```



- After getting an `R^2` score of only 0.27 for the five other features' ability to explain the variation in `Detergents_Paper`, I am surprised to see that `Detergents_Paper` has clearly observable positive correlations with two other features. This would appear to lend some credence to my initial hunch that `Detergents_Paper` could in fact be removed, seeing as how it does, in fact, correlate with two of the remaining five other features.
- `Detergents_Paper`, `Milk`, and `Grocery` are all positively skewed, with both `Milk` and `Grocery` features showing evidence of having a small number of really large, isolated outliers.

  Because all features have most of their data points close to zero, and because the majority of the feature pairs don't exhibit a correlation that's visible to the human eye, this data set is a good candidate for analysis using unsupervised learning tools. I am optimistic that I will be able to make more headway by using PCA to analyze this data and cluster customers, than would have been possible were I to rely solely on my own human intuition for feature selection. Also, it may be helpful to purge some of the extreme outliers we see in the `Fresh`, `Milk`, `Frozen`, `Grocery`, and `Delicatessen` categories.

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

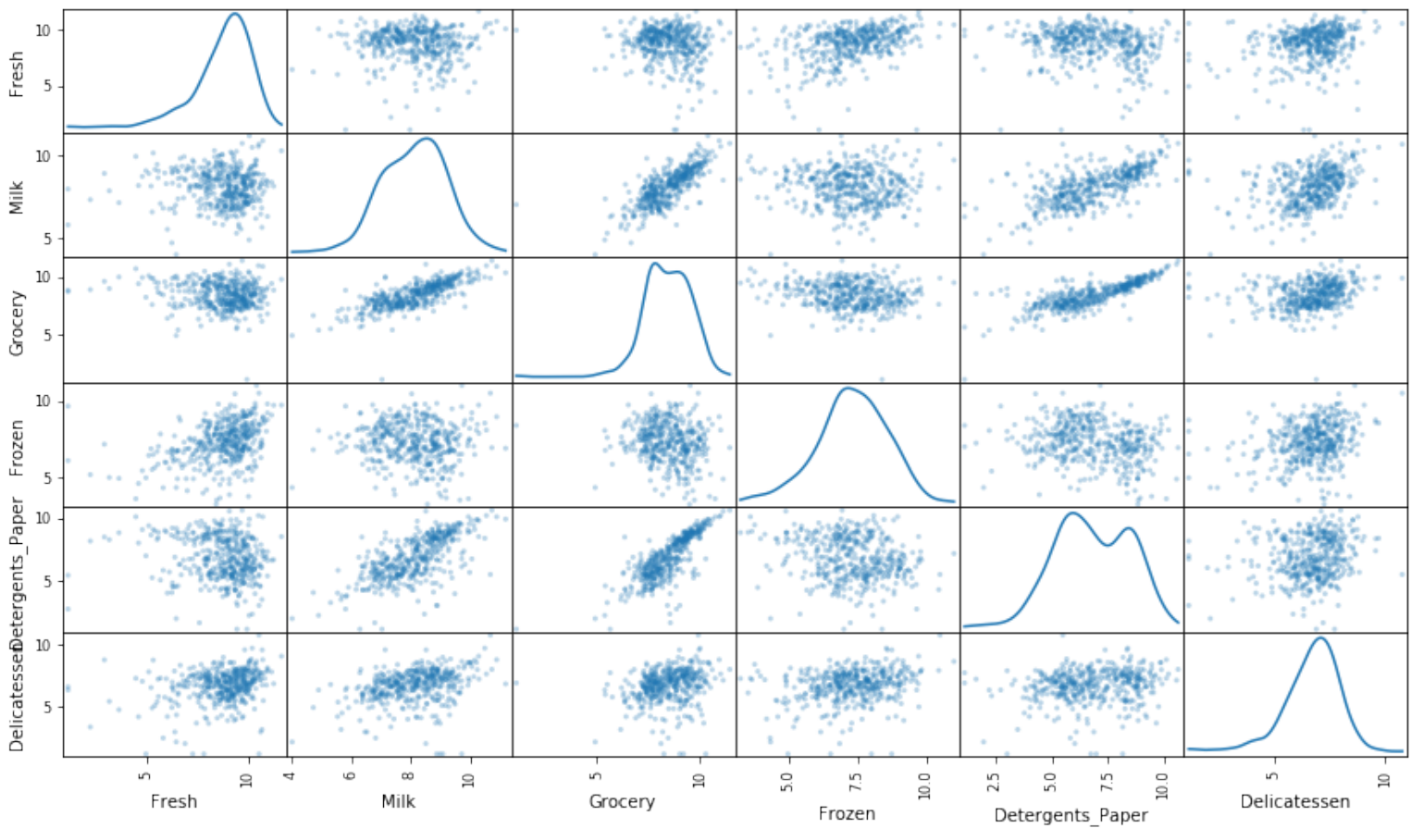
If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate (http://econbrowser.com/archives/2014/02/use-of-logarithms-in-economics) to apply a non-linear scaling — particularly for financial data. One way to achieve 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 logarithmic scaling. Use the `np.log` function for this.
- Assign a copy of the sample data to `log_samples` after applying logarithmic scaling. Again, use `np.log`.

In [8]:

```
# 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)

# Produce a scatter matrix for each pair of newly-transformed features
pd.plotting.scatter_matrix(log_data, alpha = 0.3, 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 [9]:

```
# Display the log-transformed sample data
display(log_samples)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	8.624612	6.769642	7.605890	8.126518	5.926926	6.343880
1	9.843738	7.223296	8.050384	8.006701	5.863631	5.214936
2	1.098612	5.808142	8.856661	9.655090	2.708050	6.309918

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 [11]:

```
# For each feature find the data points with extreme high or low values
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 interquartile 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 + step))])

# OPTIONAL: Select the indices for data points you wish to remove
outliers  = [65, 75, 154]

# 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

Question 4

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

Hint: If you have datapoints that are outliers in multiple categories think about why that may be and if they warrant removal. Also note how k-means is affected by outliers and whether or not this plays a factor in your analysis of whether or not to remove them.

Answer:

- Data points at the following five indices are considered outliers for more than one feature based on the definition above:

  154 (`Delicatessen`, `Grocery`, `Milk`)
  128 (`Delicatessen`, `Fresh`)
  66 (`Delicatessen`, `Fresh`)
  75 (`Detergents_Paper`, `Grocery`)
  65 (`Frozen`, `Fresh`)

- I am hesitant to remove data points (customers) who are outliers for just one feature. I haven't yet run PCA and I can't yet say with certainty which compounds of features will ultimately be most helpful in clustering the customers into segments. I am concerned that if I remove data points that each are outliers in just one dimension, I might in doing so be removing points that could contain useful clustering information in their other five dimensions. My judgment is that such data points likely provide a greater benefit to my analysis if they are allowed to remain, than if they are purged.

  On the other other hand, data points that are outliers in two or more dimensions have a greater chance of hampering my ability to cluster my data, especially if I use K-means. K-means positions clusters' centers in locations that minimize their respective loss functions, which is the sum of the squares of the distance between each point in the cluster and its center. If an outlier is far enough away from the rest of the points, K-means will conclude that the optimal location for cluster center that minimizes the loss function is somewhere in between this outlier and the rest of the points in the cluster. In worst-case scenarios, K-means may actually make the outlier itself the center of a cluster.

  This tendency to pay too much attention to outliers is exactly the opposite of how we wish K-means would behave in these situations -- it would be better if K-means understood that it should simply ignore the one outlier and only pay attention to those data points that comprise the cluster as we would delineate it by our own human intuition.

- Data points that are outliers in two or more dimensions have a greater influence or pull on K-means' attention than their counterparts who are outliers in just one dimension. Thus, the five data points I identified above that are outliers in two dimensions are therefore all candidates for removal.

  However, two of these points (at indices 66 and 128) are outliers in both the `Delicatessen` and `Fresh` dimensions. Seeing as how these two points are outliers in the same two dimensions, I am wondering if these points perhaps indicate some semblance of a signal to which I might want my clustering to pay attention. I will not remove these two outliers.

  I will remove the remaining three outliers (at indices 65, 75, and 154). In particular, since point 154 is an outlier in three dimensions (`Delicatessen`, `Grocery`, `Milk`), I believe that out of all the outliers, it would exert the greatest disruptive influence on my K-means clustering.

  None of the three points at indices 65, 75, and 154 are outliers in the same two or more features, and so I believe that these points have a much greater likelihood of simply being unhelpful noise (as opposed to containing a useful signal). I would do my analysis more harm than good were I to leave any of these points in my data set.

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

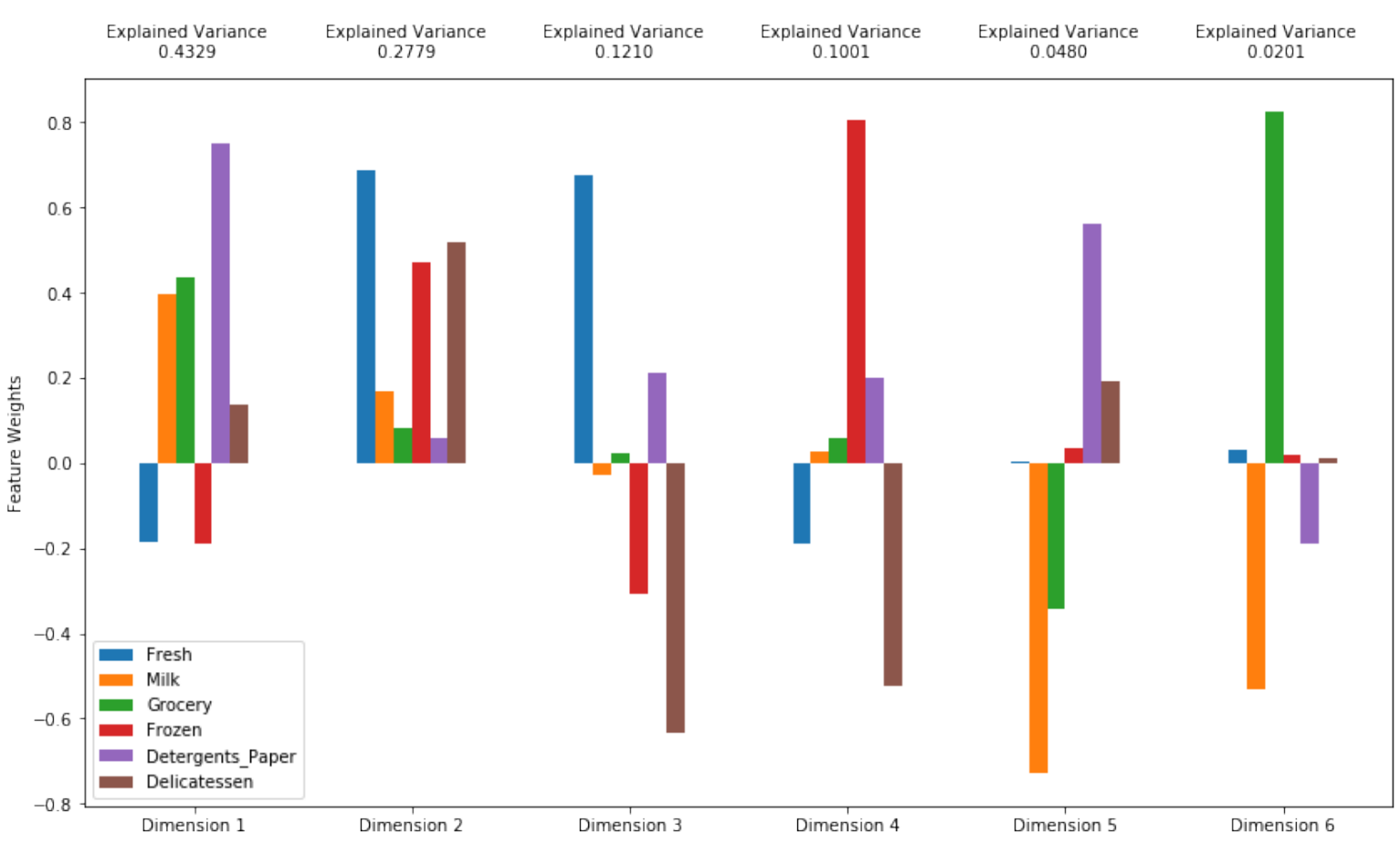
Now that the data has been scaled to a more normal distribution and has had any necessary outliers removed, we can now apply PCA to the `good_data` to discover which dimensions about the data best maximize the variance of features involved. In addition to finding these dimensions, PCA will also report 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 `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

In [12]:

```
from sklearn.decomposition import PCA

# TODO: Apply PCA by fitting the good data with the same number of dimensions as features
pca = PCA()
pca.fit(good_data)

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

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



Question 5

- How much variance in the data is explained ***in total*** by the first and second principal component?
- How much variance in the data is explained by the first four principal components?
- Using the visualization provided above, talk about each dimension and the cumulative variance explained by each, stressing upon which features are well represented by each dimension(both in terms of positive and negative variance explained). 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 individual feature weights.

Answer:

- 0.4329 + 0.2779 = 0.7108; 71.08% of the variance of the data in total is explained by the first and second principal component.
- 0.4329 + 0.2779 + 0.1210 + 0.1001 = 0.9319; 93.19% of the variance of the data in total is explained by the first four principal components.
- Based on the graphs above, the first four dimensions indicate that overall customer spending patterns tend to correlate with the amounts customers spend on the `Detergents_Paper`, `Milk`, `Grocery`, and `Fresh` categories:

  Positive increases in dimension 1 are primarily driven by an increase in the amount spent on `Detergents_Paper`, and to a lesser extent by what's spent on `Milk` and `Grocery`. For dimension 2, the most important categories are `Fresh`, `Frozen`, and `Delicatessen`. However, in dimension 3, *decreases* in the amounts spent on `Delicatessen` and `Frozen` along with, again, an

increase in the amount spent on `Fresh` coincides with a positive increase in this dimension. Positive changes in dimension 4 seem to correspond to mostly to increases in amount spent on `Frozen`, to decreases in the amount spent on `Delicatessen`, and to a much smaller extent to decreases in amount spent on `Fresh`. Finally, in dimensions 5 and 6, we observe that positive increases in these dimensions correspond to increases in amount spent on `Detergents_Paper` and `Grocery`, respectively. Both dimension 5 and 6 also see a noticeable positive increases when there is a decrease in the amount spent on `Milk`. Also, in dimensions 5 and 6 we see that `Detergents_Paper` and `Grocery` behave inversely: an increase in amount spent on `Detergents_Paper` and a corresponding (but lower-weighted) decrease in amount spent `Grocery` lead to a positive increase in dimension 5, while an increase in amount spent on `Grocery` and a corresponding (but lower-weighted) decrease in amount spent `Detergents_Paper` lead to a positive increase in dimension 6.

After looking at feature weights across the first four dimensions, and weighing the overall significance of each dimension's set of weights by the fraction of overall variance explained by that dimension, I find that `Detergents_Paper` seems to be the most important feature overall (it's by far the largest weight in dimension 1, which itself explains 43.29% of the variance). After this, `Milk` and `Grocery` appear to my eye to both be equally important -- they have decently large weights in dimension 1, the most meaningful dimension, and have smaller positive weights in dimensions 2, 3, and 4 as well (dimension 3 is a slight exception in that `Milk` has a tiny negative weight, but I don't believe this impacts the feature's overall importance). Finally, `Fresh` rounds out the top four features. Though it does have low negative weights in dimensions 1 and 4, this is more than overcome by the extent to which `Fresh` is very highly positively weighted in dimensions 2 and 3, which together explain about 30% of the variance.

The `Delicatessen` feature is meaningfully positively weighted in dimension 2, but this is more than cancelled out by the feature's large negative weights in dimensions 3 and 4. Based on dimensions 2 and 4, `Frozen` at first glance looks to be important thanks to its large positive weights in both those dimensions. Unfortunately, `Frozen`'s highest positive weight is in the less important dimension 4 (only explains 10% of the variance). Therefore, this influence is somewhat counter-acted by `Frozen`'s slight negative weight in the more important dimension 1, as well as its greater negative weight in dimension 3.

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 [13]:

```
# 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	-1.7458	-0.2177	-0.2807	0.5857	0.7576	0.1901
1	-1.7768	0.0870	1.2802	0.8738	0.0199	0.3491
2	-2.8896	-4.9309	-6.4384	2.6534	-0.7433	2.1547

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 results to `reduced_data`.
- Apply a PCA transformation of `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

In [14]:

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

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

# TODO: Transform log_samples 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 [15]:

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

	Dimension 1	Dimension 2
0	-1.7458	-0.2177
1	-1.7768	0.0870
2	-2.8896	-4.9309

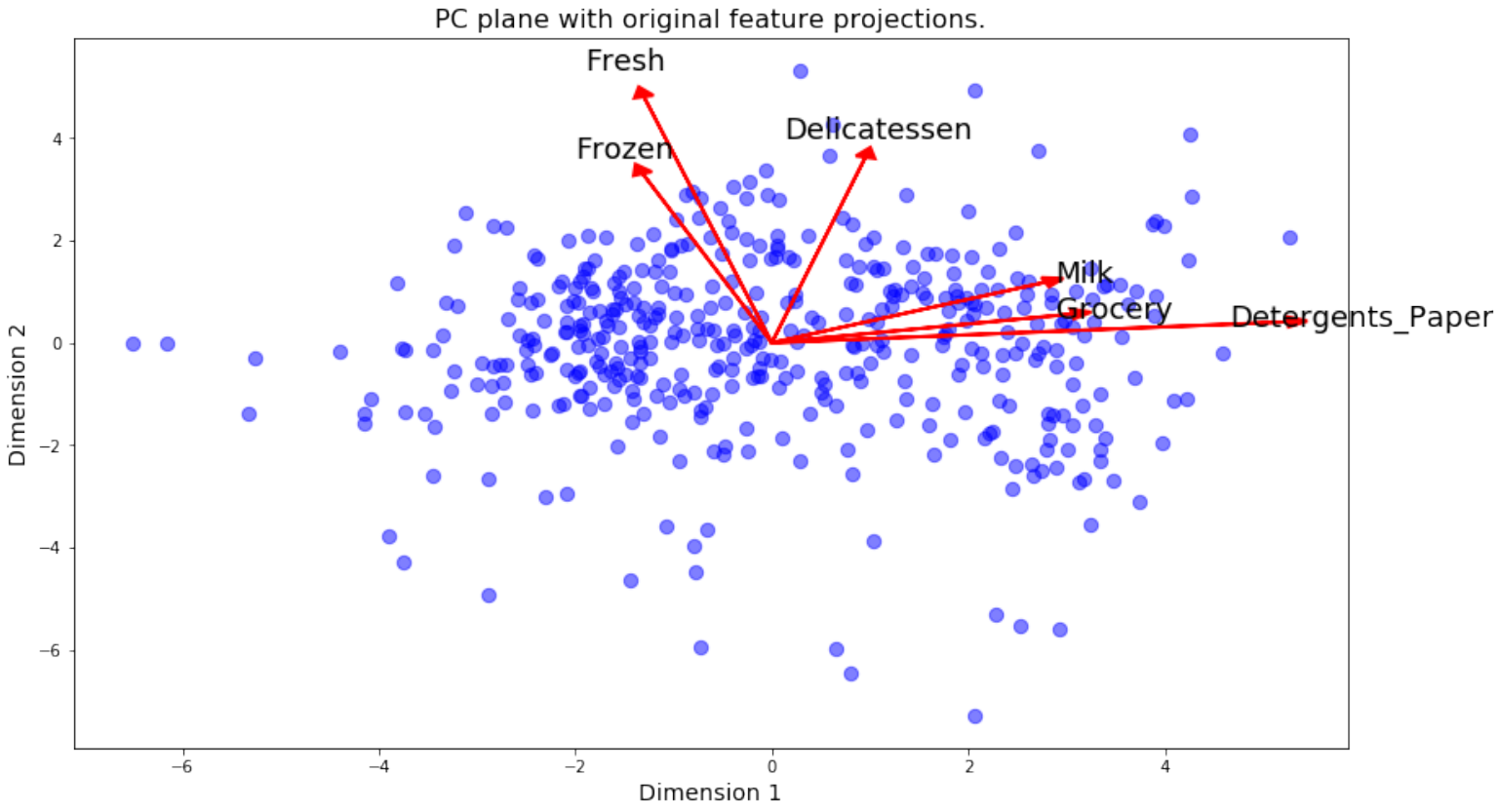
Visualizing a Biplot

A biplot is a scatterplot where each data point is represented by its scores along the principal components. The axes are the principal components (in this case `Dimension 1` and `Dimension 2`). In addition, the biplot shows the projection of the original features along the components. A biplot can help us interpret the reduced dimensions of the data, and discover relationships between the principal components and original features.

Run the code cell below to produce a biplot of the reduced-dimension data.

In [16]:

```
# Create a biplot
vs.biplot(good_data, reduced_data, pca);
```



Observation

Once we have the original feature projections (in red), it is easier to interpret the relative position of each data point in the scatterplot. For instance, a point the lower right corner of the figure will likely correspond to a customer that spends a lot on `'Milk'`, `'Grocery'` and `'Detergents_Paper'`, but not so much on the other product categories.

From the biplot, which of the original features are most strongly correlated with the first component? What about those that are associated with the second component? Do these observations agree with the pca_results plot you obtained earlier?

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?

Hint: Think about the differences between hard clustering and soft clustering and which would be appropriate for our dataset.

Answer:

- K-means works well on data sets whose points are distributed in such a way that they fall into a set of dense clusters, each having a circular/spherical/hyper-spherical shape. Each of these clusters ideally would be isolated enough from the other clusters to an extent that would allow us to verify this isolation by visual inspection. We could then determine, before running K-means, how many clusters we should expect our data to be clustered into. K-means is most effective (there would be no need for running K-means using various numbers of clusters and then choosing the number that gives the highest mean silhouette coefficient) when we know the number of expected clusters beforehand.

  Unfortunately, because K-means defines a cluster's data points by their distance to that cluster's centroid, K-means is not effective on data sets that have dense concentrations of points where these dense concentrations are of any shape other than the circle/sphere/hyper-sphere for which K-means is optimized. For example, if we had a data set whose points were arranged in a series of dense, concentric rings, K-means would not be able to correctly discern that each ring should be a separate cluster.

  K-means also assumes hard clustering: each data point can belong to one and only one cluster. K-means is therefore ineffective on data sets whose points distributed relatively uniformly, or at least distrbuted broadly enough such that we cannot separate the points into clusters by visual inspection. This would be because the data set, to our naked eyes, appears to be a nebulous cloud of data points, or at most, a merging of a few nebulous clouds of data points. In these scenarious, we could get extremely different clustering results each time we run K-means. This is because K-means starts by placing centroids at random, and then re-locates each of these centroids until the loss function is minimized. For (near) uniformly distributed data, our K-means cluster results are at the mercy the initial random placement of centroids.

- Thankfully, the Gaussian Mixture Model clustering algorithm is much better at working with data sets whose distributions are too "cloudy" or vague for the constraints of hard clustering. GMM allows for soft clustering: that each data point can belong to multiple clusters with different probabilities. GMM ultimately places a data point inside the cluster with which GMM determines that particular point has the highest probability of being a member. This is done via the process of

expectation maximization, the process by which GMM iteratively discovers the ideal mean and variance for each cluster's distrbution that will maximize the probabilities of each cluster's points belonging to that cluster.

One downside is that, unlike K-means where we can have a visual confirmation of the optimal number of clusters before we run the algorithm, with GMM there's really no way for us to know what this ideal number of clusters would be, and yet, GMM requires us to tell it how many clusters we want it to use before we run it. The way around this limitation could be to run GMM a few times, each time using a different number of clusters, and then choosing to group our data set into the number of clusters that results in our GMM results having the highest density-based clustering validation score (http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=83C3BD5E078B1444CB26E243975507E1?doi=10.1.1.707.9034&rep=rep1&type=pdf (http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=83C3BD5E078B1444CB26E243975507E1?doi=10.1.1.707.9034&rep=rep1&type=pdf)). Note: I could also use the mean silhouette coefficient to score my various GMMs against each other, but if I were scoring a GMM against a K-means, I would not want to use the silhouette coefficient -- a superior clustering with GMM can sometimes get a poorer silhouette score when compared to an inferior clustering done by K-means. Also note: I must use an internal validation index for this data set because I do not have a set of labels I can use to confirm that I've properly clustered my data. Seeing as how this is unsupervised learning, I'm creating and defining my data's clusters from scratch.

Finally, GMM assumes that each cluster will have a normal distrbution. This is not the same thing as assuming that the entire data set itself has one gaussian distrbution. Rather, if each cluster does indeed have a normal distribution, the entire dataset will be distrbuted as *a combination* of these individual gaussian distributions.

- Since our data does not appear to already be naturally positioned into dense clusters which we could describe by visual inspection, I do not believe that K-means and its hard clustering would be ideal for clustering our dataset. I expect that Gaussian Mixture Modeling, with its soft clustering, will do a much better job of describing for me the most likely clustering arrangement of our data set. Since I've already log-normalized the distributions of each feature in this data set, I have no concerns about violating GMM's assumption that each cluster it defines will have a normal distribution.

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.

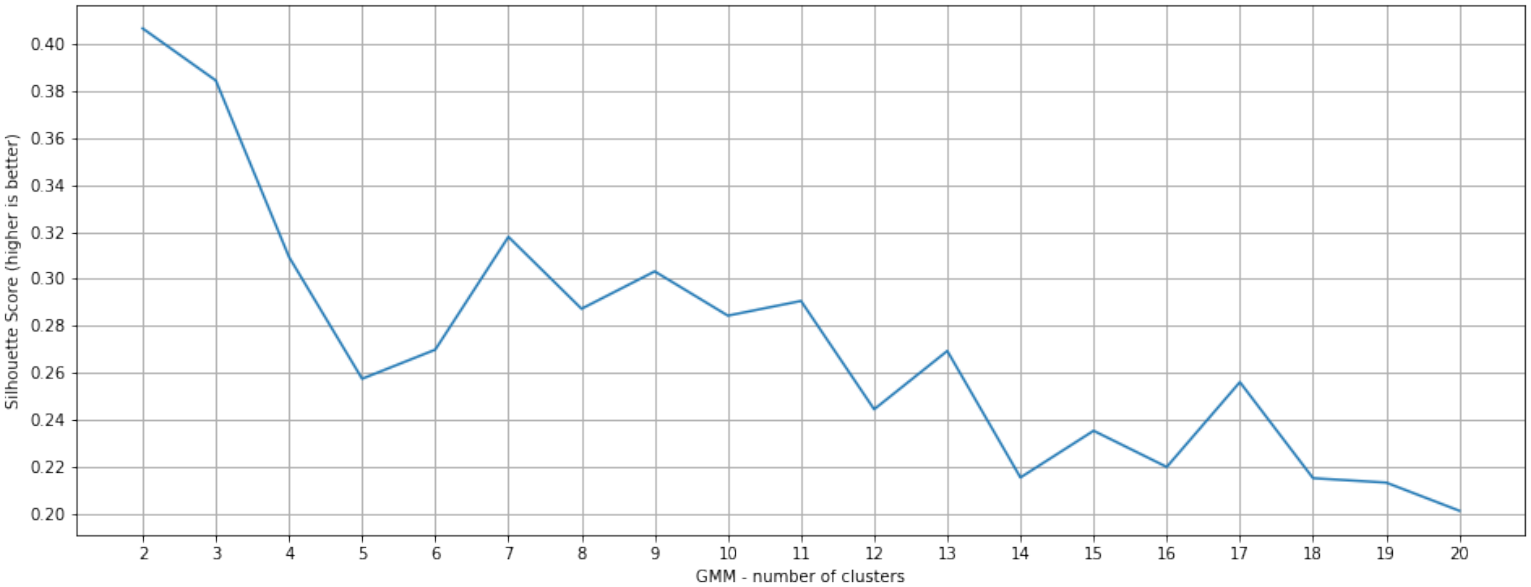
In [19]:

```
from sklearn import mixture
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt


## A list of the different numbers of clusters (the 'n_components' parameter) wi
th
## which we will run GMM.
number_of_clusters = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

## Graph plotting method
def makePlot(number_of_clusters, silhouette_scores):
    # Plot the each value of 'number of clusters' vs. the silhouette score at th
at value
    fig, ax = plt.subplots(figsize=(16, 6))
    ax.set_xlabel('GMM - number of clusters')
    ax.set_ylabel('Silhouette Score (higher is better)')
    ax.plot(number_of_clusters, silhouette_scores)

    # Ticks and grid
    xticks = np.arange(min(number_of_clusters), max(number_of_clusters)+1, 1.0)
    ax.set_xticks(xticks, minor=False)
    ax.set_xticks(xticks, minor=True)
    ax.xaxis.grid(True, which='both')
    yticks = np.arange(round(min(silhouette_scores), 2), max(silhouette_scores),
.02)
    ax.set_yticks(yticks, minor=False)
```

```
    ax.set_yticks(yticks, minor=True)

    ax.yaxis.grid(True, which='both')

## Graph the mean silhouette score of each cluster amount.
## Print out the number of clusters that results in the highest
## silhouette score for GMM.
def findBestClusterer(number_of_clusters):
    silhouette_scores = []
    for i in number_of_clusters:
        clusterer = mixture.GMM(n_components=i)
        clusterer.fit(reduced_data)
        preds = clusterer.predict(reduced_data)
        score = silhouette_score(reduced_data, preds)
        silhouette_scores.append(score)

    ## Print a table of all the silhouette scores
    print("")
    print("| Number of clusters | Silhouette score |")
    print("| ------------------ | ---------------- |")
    for i in range(len(number_of_clusters)):
        ## Ensure printed table is properly formatted, taking into account
        ## amount of digits (either one or two) in the value for number of clust
ers.
        if number_of_clusters[i] <= 9:
            print("| {number}                   | {score:.4f}           |".format
(number=number_of_clusters[i],
                                                                    score=ro
und(silhouette_scores[i], 4)))
        else:
            print("| {number}                  | {score:.4f}           |".format(
number=number_of_clusters[i],
                                                                    score=ro
und(silhouette_scores[i], 4)))


    ## Graph the plot of silhoutte scores for each amount of clusters
    makePlot(number_of_clusters, silhouette_scores)

    ## Find and print out the cluster amount that gives the highest
    ## silhouette score.
    best_silhouette_score = max(silhouette_scores)
    index_of_best_score = silhouette_scores.index(best_silhouette_score)
    ideal_number_of_clusters = number_of_clusters[index_of_best_score]
    print("")
    print("Having {} clusters gives the highest silhouette score of {}.".format(
ideal_number_of_clusters,

round(best_silhouette_score, 4)))

findBestClusterer(number_of_clusters)
```

Number of clusters	Silhouette score
2	0.4066
3	0.3844
4	0.3094
5	0.2575
6	0.2699
7	0.3179
8	0.2873
9	0.3032
10	0.2843
11	0.2906
12	0.2445
13	0.2693
14	0.2155
15	0.2353
16	0.2199
17	0.2561
18	0.2152
19	0.2133
20	0.2013

Having 2 clusters gives the highest silhouette score of 0.4066.



In [20]:

```
# TODO: Apply your clustering algorithm of choice to the reduced data
clusterer = mixture.GMM(n_components=2)
clusterer.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 cho
sen
score = silhouette_score(reduced_data, preds)
```

Question 7

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

Answer:

Number of clusters	Silhouette score
2	0.4066
3	0.3844
4	0.3094
5	0.2575
6	0.2699
7	0.3179
8	0.2873
9	0.3032
10	0.2843
11	0.2906
12	0.2445
13	0.2693
14	0.2155
15	0.2353
16	0.2199
17	0.2561
18	0.2152
19	0.2133
20	0.2013

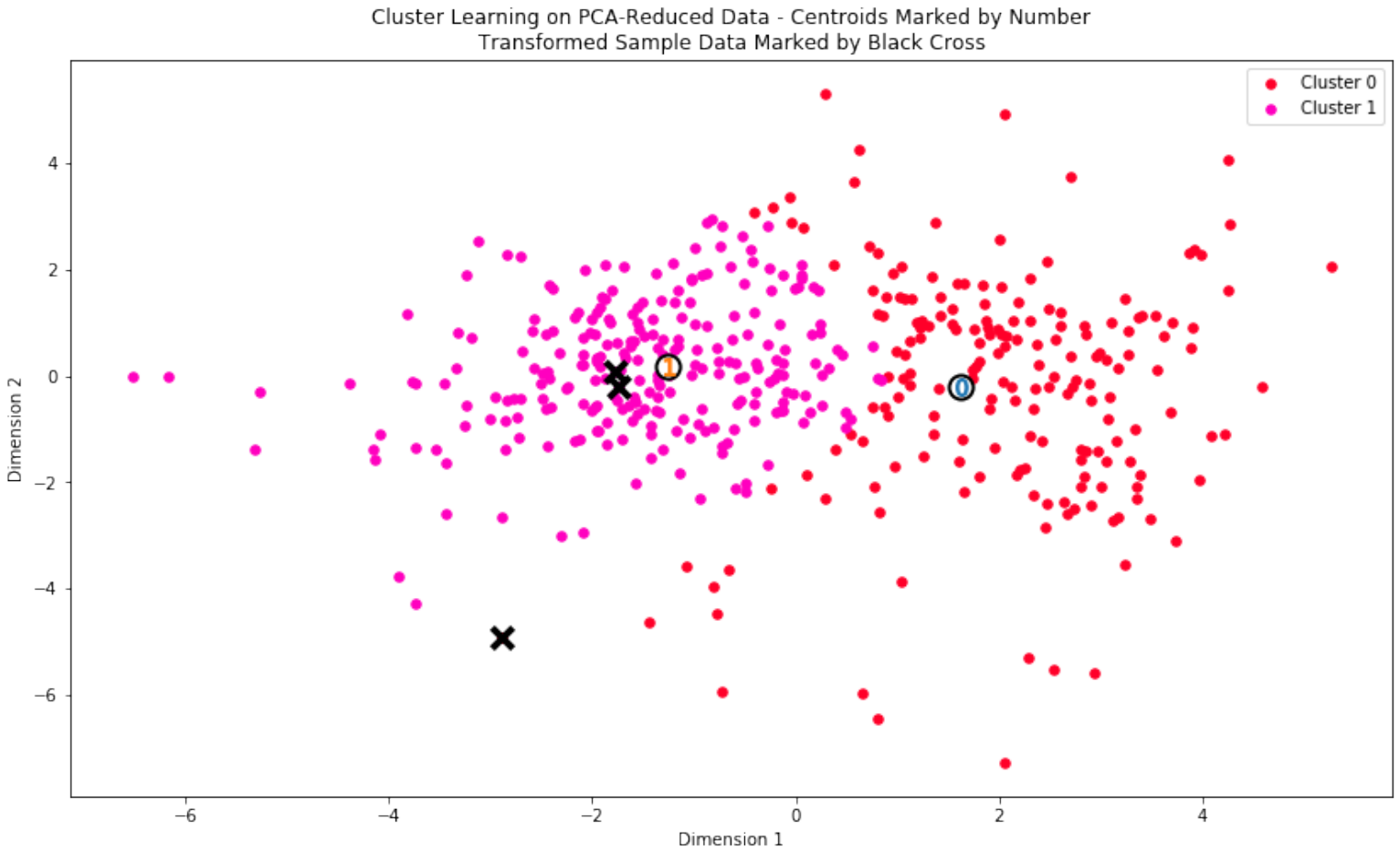
- Having 2 clusters gives the highest silhouette score of 0.4066.

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 [21]:

```
# Display the results of the clustering from implementation
vs.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 [22]:

```
# TODO: Inverse transform the centers
log_centers = [pca.inverse_transform(i) for i in centers]

# TODO: Exponentiate the centers
true_centers = [np.exp(i) for i in 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	3999.0	6222.0	9412.0	1001.0	2994.0	882.0
Segment 1	8881.0	2116.0	2768.0	2057.0	354.0	727.0

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(specifically looking at the mean values for the various feature points). 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'`. Think about what each segment represents in terms their values for the feature points chosen. Reference these values with the mean values to get some perspective into what kind of establishment they represent.

Answer:

- Segment 0 represents establishments who spend above average amounts on the `Milk`, `Grocery`, and `Detergents_Paper` categories (in between the 50th and 75th percentiles for each category). In contrast, these establishments spend below average amounts on the `Fresh`, `Frozen`, and `Delicatessen` categories (in between the 25th and 50th percentiles for each category). Furthermore, although the amount spent on `Fresh` is lower than average, it still appears to encompass roughly 20% of this segment's average spend.

  This spending pattern could well fit the needs of run-of-the-mill supermarkets that sell plenty of groceries, dairy, and non-food items like soap, shampoo, detergent, paper, etc. These kinds of stores also offer an undifferentiated selection of fresh food, though not extensive in quantity or variety. Hypermarkets such as Target or Wal-Mart might also have a similar purchase profile for their foodstuffs.

  We can safely infer that restaurants of any sort likely do not fall inside this segment, owing to the amount that average customer spends on `Detergents_Paper`, which would seem to be far too great for businesses who do not sell these items themselves.

- Segment 1, on the other hand, represents businesses who have an above average spend on the `Fresh` and `Frozen` categories, with well below average spends on all other categories. In constrast to the supermarkets and hypermarkets that are encapsulated inside segment 0, segment 1 would appear to represent customers who run restaurants. We would expect restaurants to spend above average amounts on categories like `Fresh` (ingredients that differentiate appetizers and entrees) and `Frozen` (longer lasting, staple food items that would comprise the basic components of dishes).

  This intuition is further reinforced by the fact that the average customer in segment 1 spends more than half of their money on fresh food, while at the same time virtually ignoring the `Detergents_Paper` category, which is a spending pattern that we'd expect a restaurant to follow. Although below average amounts are spent on `Milk` and `Grocery`, they nonetheless comprise over 25% of this segment's average customer's spend. This is a pattern we could also reasonably expect to apply to restaurants, as they would conceivably have needs for some amount of dairy and other sundry grocery ingredients.

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 [23]:

```
# 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 0
```

Answer:

- I had originally predicted that sample point 0 could belong to mid-sized, independent restaurant. It is reassuring to see that my GMM clustering predicts that this sample point would be inside cluster 1, which most likely encapsulates businesses that are restaurants, as I explained above in my answer to Question 8.

- I had originally predicted that sample point 1 could belong to a mid-sized grocery market that specializes in selling fresh and healthy meats and vegetables. I can understand why it was predicted to be part of cluster 1, which I identify as mostly representing restaurants. Although my initial hypothesis (not a restaurant) is inconsistent with this categorization (likely a restaurant) I can understand why the contradiction exists -- a market selling lots of fresh, healthy food may well have buying patterns much more similar to those of a restaurant, as opposed to those of a larger, blander supermarket like Safeway or Target.

- I had originally predicted that sample point 2 could belong to a fastfood restaurant (owing the the massive amount spent on frozen food). This hunch also ended up being inconsistent with my clustering, which places sample point 2 in the cluster that I associate with mass-market supermarkets. However, similar to what happened with sample point 1 above, I can understand why this is the case -- spending patterns for a fast food restaurant may well be more similar to those of a Safeway supermarket than those of a fine French restaurant.

Conclusion

In this final section, you will investigate ways that you can make use of the clustered data. First, you will consider how the different groups of customers, the ***customer segments***, may be affected differently by a specific delivery scheme. Next, you will consider how giving a label to each customer (which *segment* that customer belongs to) can provide for additional features about the customer data. Finally, you will compare the ***customer segments*** to a hidden variable present in the data, to see whether the clustering identified certain relationships.

Question 10

Companies will often run A/B tests (https://en.wikipedia.org/wiki/A/B_testing) when making small changes to their products or services to determine whether making that change will affect its customers positively or negatively. The wholesale distributor is considering changing its delivery service from currently 5 days a week to 3 days a week. However, the distributor will only make this change in delivery service for customers that react positively.

- How can the wholesale distributor use the customer segments to determine which customers, if any, would react positively to the change in delivery service?

Hint: Can we assume the change affects all customers equally? How can we determine which group of customers it affects the most?

Answer:

- The average spending patterns of our two customer segments are quite disparate, and we have already inferred that the segments represent two very different kinds of businesses. Therefore, we cannot assume that reducing delivery from 5 to only 3 days a week would affect all customers equally.
- In order to determine which customer types are affected, and to what extent, the wholesale distributor could run two A/B tests, each A/B test being conducted on a different customer segment.
- For each A/B test, we would set aside a small amount of customers inside each segment. This small group of customers would be the A/B test's "test" group, on which the distributor could test reducing delivery service from 5 to 3 days. The amount of customers from each segment put into these two test groups need only be large enough such that if we were to see a change of behavior from customers who began receiving the reduced delivery service, we could be sure that this change was indeed caused by the reduced delivery schedule, as opposed to merely being a random effect.
- The measured behavior inside each segment's test group would be compared to measurements taken from each segment's control group -- the rest of the customers inside each segment who would continue getting 5 day a week delivery.
- If the customers in a segment's test group exhibit a statistically significant measured positive effect, subsequent to being exposed to the 3 day a week delivery schedule, we can confirm that the majority of customers inside that segment would likely react positively to the reduced delivery schedule.

Question 11

Additional structure is derived from originally unlabeled data when using clustering techniques. Since each customer has a ***customer segment*** it best identifies with (depending on the clustering algorithm applied), we can consider '*customer segment*' as an **engineered feature** for the data. Assume the wholesale distributor recently acquired ten new customers and each provided estimates for anticipated annual spending of each product category. Knowing these estimates, the wholesale distributor wants to classify each new customer to a ***customer segment*** to determine the most appropriate delivery service.

- How can the wholesale distributor label the new customers using only their estimated product spending and the **customer segment** data?

Hint: A supervised learner could be used to train on the original customers. What would be the target variable?

Answer:

- The wholesale distributor could use a supervised learner that works well for classification problems, such as decision trees, naive bayes, or support vector machines. The target variable would be the customer segment, as determined for each original customer by our use of GMM clustering above.
- The distributor could split the original customers into training, validation, and test sets. Once the supervised learner had been adequately tuned using the training and validation sets (comprising 80% of the original customers), if the distributor was pleased with its accuracy in predicting the segments of original customers inside the test set (the remaining 20% of original customers), the classifier and its chosen parameters could then be used to predict the segment to which each of the ten new customers belonged, according to their anticipated annual spending habits.

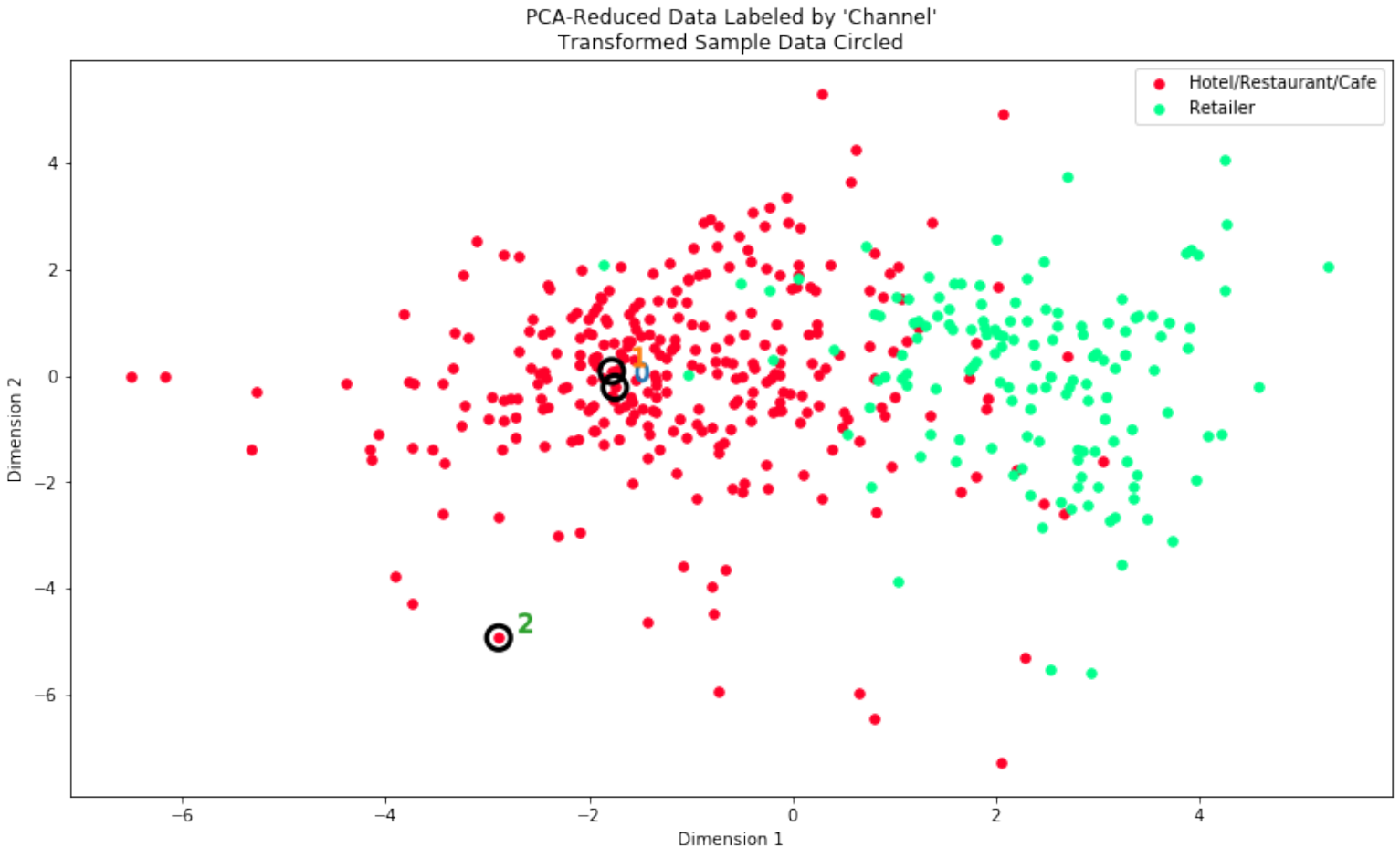
Visualizing Underlying Distributions

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 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 [24]:

```
# Display the clustering results based on 'Channel' data
vs.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:

- My clustering algorithm's results and number of clusters I've chosen are nearly identical to the distribution of 'Hotel/Restaurant/Cafe' customers to 'Retailers' customers.
- Looking at the scatterplot above of customer channels plotted in relation to the two dimensions of PCA-reduced data, we can observe a markedly clean delineation between the customers belonging to the 'Hotels/Restaurants/Cafes' channel and those customers belonging to the 'Retailers' channel. This, along with the fact that my implementation of GMM classified the customers into two clusters with a demarcation boundary nearly identical to what we see in the plot above, is sufficient justification for allowing us to classify segments as purely 'Retailers' or purely 'Hotels/Restaurants/Cafes'.
- Indeed, the definitions of these channel classifications are virtually identical to my previous definition of each segment: I had guessed that segment 0 was likely to represent supermarkets or hypermarkets, which are indeed retailers. I had guessed that segment 1 belonged to generally to restaurants, which is essentially hotels/restaurants/cafes.
- Sample points 0 and 1 were clustered in segment 1 by GMM, and in the graph above we see they are both safely inside the 'Hotels/Restaurants/Cafes' channel, which is what I'd expect given that this channel directly corresponds to segment 1.
- The only discrepancy between my segments and these channels is that my GMM clustering had predicted that a greater portion of points located in the lower left quadrant would belong to segment 0, or the 'Retailer' channel. Sample point 2 is one of these points. Above, we see that sample 2 is actually part of the 'Hotels/Restaurants/Cafes' channel, and is not, in fact, a retailer as my GMM clustering had predicted.
- Sample point 2 is a good example of rare occasions when my GMM clustering would be inaccurate. Nonetheless, at least based on visual inspection, it appears that my GMM clustering was spot-on accurate for the majority of customers. I am pleased with its performance.

> **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.