

# Machine Learning Engineer Nanodegree

## Unsupervised Learning

### Project: 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) (<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
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.".format(*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 [3]: # Display a description of the dataset
display(data.describe())
```

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

## Implementation: Selecting Samples

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

```
In [4]: # TODO: Select three indices of your choice you wish to sample from the dataset
indices = [75, 86, 183]

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

Chosen samples of wholesale customers dataset:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	20398	1137	3	4407	3	975
1	22925	73498	32114	987	20070	903
2	36847	43950	20170	36534	239	47943

## 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: 3071.9
- 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:**

Sample 0 might be traditional market due to high fresh purchase cost and very low grocery and detergents\_paper purchase cost.

Sample 1 might be super market due to high purchase cost of fresh, milk, grocery and detergents\_paper.

Sample 2 might be warehouse due to almost every products (except Detegents\_paper) are very higher than mean values.

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

# TODO: Split the data into training and testing sets(0.25) using the given feature as the target
# Set a random state.
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(new_data, data['Milk'], test_size=0.25, random_state=3)

# TODO: Create a decision tree regressor and fit it to the training set
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=3)
regressor.fit(X_train, y_train)

# TODO: Report the score of the prediction using the testing set
score = regressor.score(X_test, y_test)
display(score)
```

/opt/conda/lib/python3.6/site-packages/sklearn/cross\_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model\_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.

"This module will be removed in 0.20.", DeprecationWarning)

0.39504003544603272

## 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 'Milk' and the prediction score was 0.395. Because the score is low, it seems necessary for identifying customer's 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.

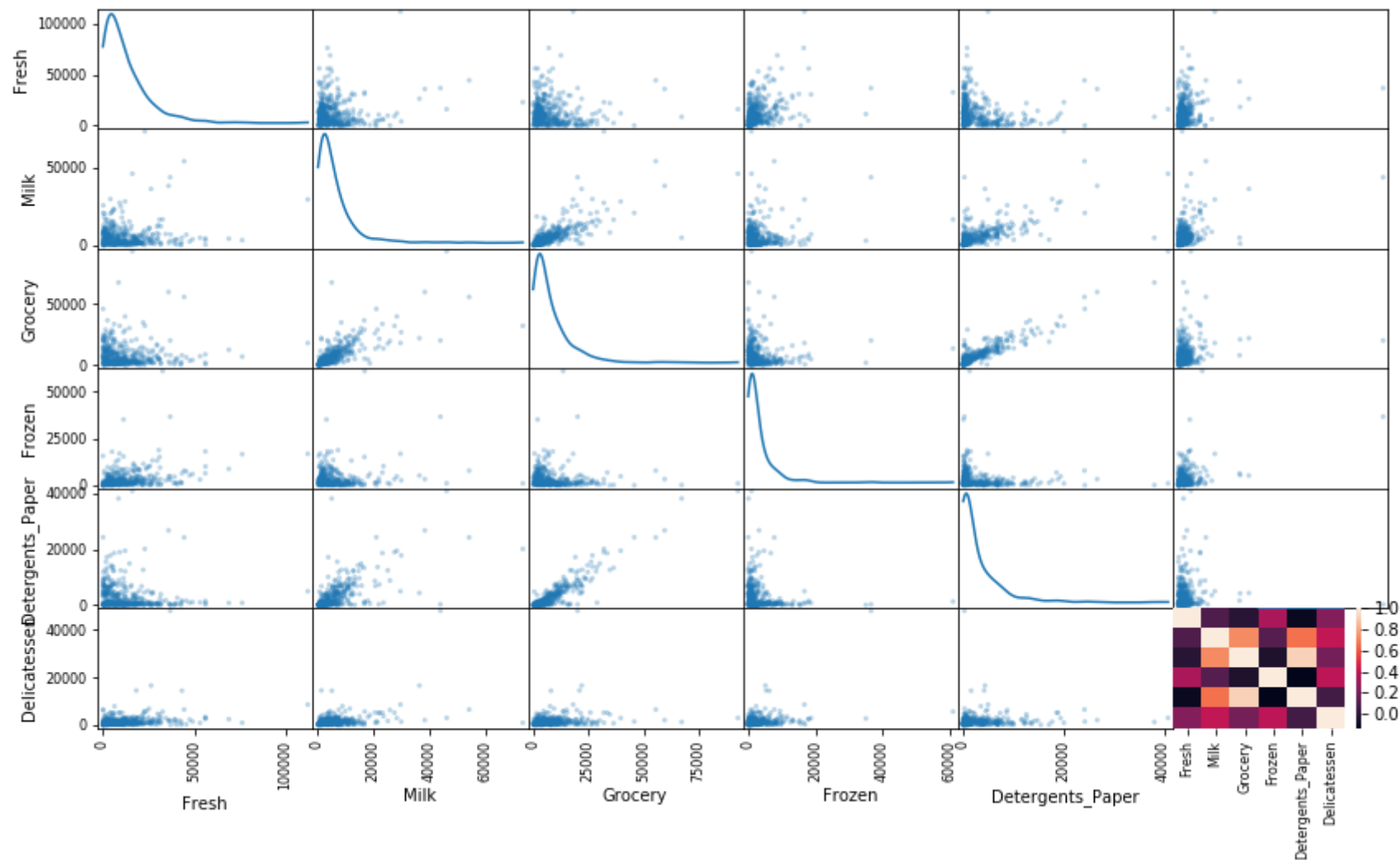
```
In [6]: # Produce a scatter matrix for each pair of features in the data
pd.scatter_matrix(data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');

import seaborn as sns
sns.heatmap(data.corr())
```



```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning: pandas.scatter_matrix is deprecated. Use pandas.plotting.scatter_matrix instead
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f38c4422da0>
```



### 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 separate 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 dataset are not normally distributed. Data is skewed.

And each of plot has few points far from the most of dataset which are outliers.

Moreover delicatenessen has large number of data points near 0.

There is a high correlation between Grocery and Detergents\_paper.

There are correlations between Milk, Grocery and Detergents\_paper.

Yes, this confirm my suspicions about the releance of Milk. The prediction score of Milk was 0.395. It is low but not extremely low score.

Compare to `corr()`, if the two features have correlations, the data set distributions look like a slant line, conversely data set is discrete.

## 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:

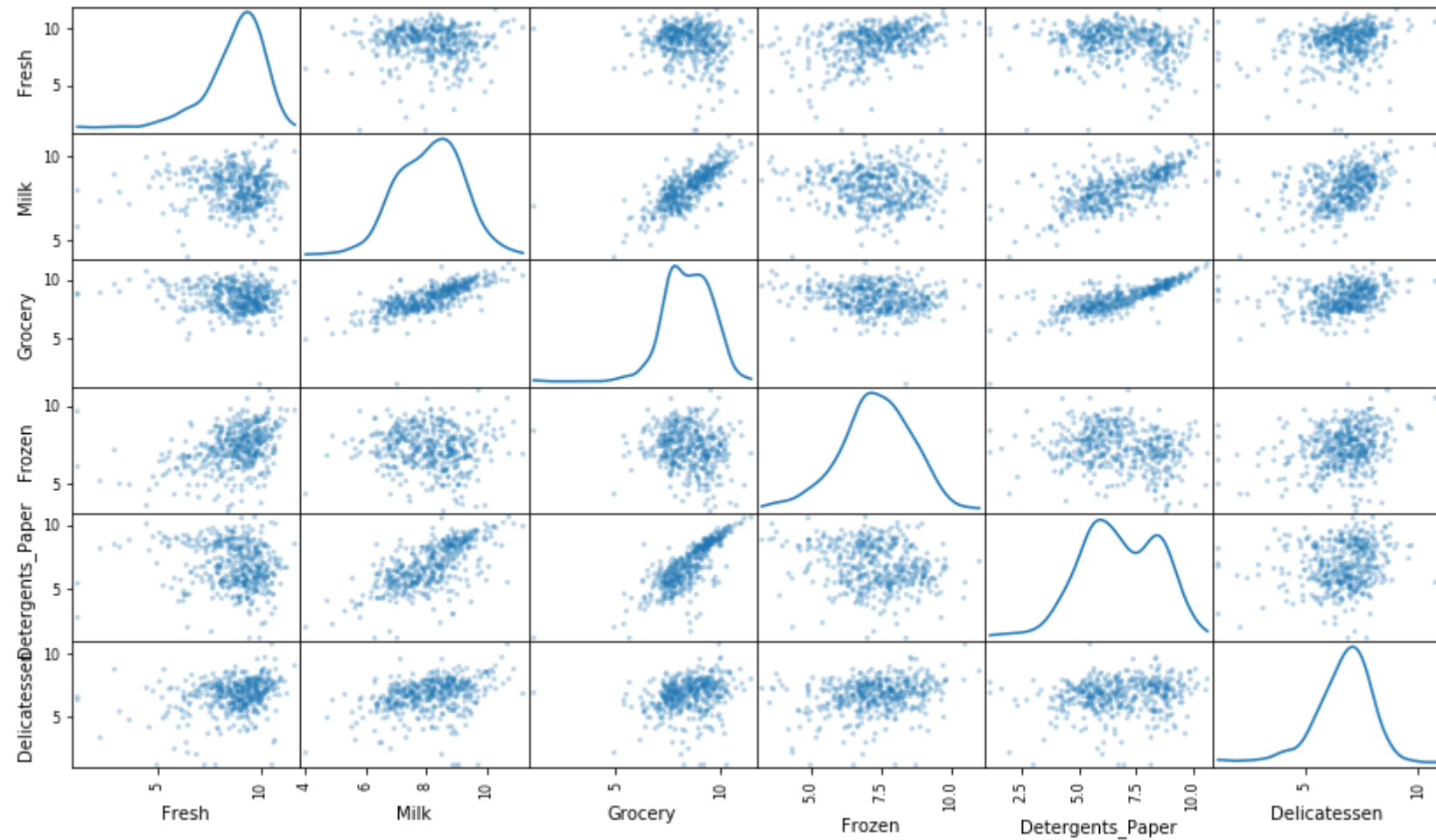
- Assign a copy of the data to `log_data` after applying logarithmic scaling. Use the `np.log` function for this.
- Assign a copy of the sample data to `log_samples` after applying logarithmic scaling. Again, use `np.log`.

```
In [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)

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

/opt/conda/lib/python3.6/site-packages/ipykernel\_launcher.py:8: FutureWarning: pandas.scatter\_matrix is deprecated. Use pandas.plotting.scatter\_matrix instead



## 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	9.923192	7.036148	1.098612	8.390949	1.098612	6.882437
1	10.039983	11.205013	10.377047	6.894670	9.906981	6.805723
2	10.514529	10.690808	9.911952	10.505999	5.476464	10.777768

## 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 identifying 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 [10]: # 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 = (Q3 - Q1) * 1.5

            # 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, 66, 75, 128, 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':

	<b>Fresh</b>	<b>Milk</b>	<b>Grocery</b>	<b>Frozen</b>	<b>Detergents_Paper</b>	<b>Delicatessen</b>
<b>65</b>	4.442651	9.950323	10.732651	3.583519	10.095388	7.260523
<b>66</b>	2.197225	7.335634	8.911530	5.164786	8.151333	3.295837
<b>81</b>	5.389072	9.163249	9.575192	5.645447	8.964184	5.049856
<b>95</b>	1.098612	7.979339	8.740657	6.086775	5.407172	6.563856
<b>96</b>	3.135494	7.869402	9.001839	4.976734	8.262043	5.379897
<b>128</b>	4.941642	9.087834	8.248791	4.955827	6.967909	1.098612
<b>171</b>	5.298317	10.160530	9.894245	6.478510	9.079434	8.740337
<b>193</b>	5.192957	8.156223	9.917982	6.865891	8.633731	6.501290
<b>218</b>	2.890372	8.923191	9.629380	7.158514	8.475746	8.759669
<b>304</b>	5.081404	8.917311	10.117510	6.424869	9.374413	7.787382
<b>305</b>	5.493061	9.468001	9.088399	6.683361	8.271037	5.351858
<b>338</b>	1.098612	5.808142	8.856661	9.655090	2.708050	6.309918
<b>353</b>	4.762174	8.742574	9.961898	5.429346	9.069007	7.013016
<b>355</b>	5.247024	6.588926	7.606885	5.501258	5.214936	4.844187
<b>357</b>	3.610918	7.150701	10.011086	4.919981	8.816853	4.700480
<b>412</b>	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
<b>86</b>	10.039983	11.205013	10.377047	6.894670	9.906981	6.805723
<b>98</b>	6.220590	4.718499	6.656727	6.796824	4.025352	4.882802
<b>154</b>	6.432940	4.007333	4.919981	4.317488	1.945910	2.079442
<b>356</b>	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
<b>75</b>	9.923192	7.036148	1.098612	8.390949	1.098612	6.882437
<b>154</b>	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
<b>38</b>	8.431853	9.663261	9.723703	3.496508	8.847360	6.070738
<b>57</b>	8.597297	9.203618	9.257892	3.637586	8.932213	7.156177
<b>65</b>	4.442651	9.950323	10.732651	3.583519	10.095388	7.260523
<b>145</b>	10.000569	9.034080	10.457143	3.737670	9.440738	8.396155
<b>175</b>	7.759187	8.967632	9.382106	3.951244	8.341887	7.436617
<b>264</b>	6.978214	9.177714	9.645041	4.110874	8.696176	7.142827
<b>325</b>	10.395650	9.728181	9.519735	11.016479	7.148346	8.632128
<b>420</b>	8.402007	8.569026	9.490015	3.218876	8.827321	7.239215
<b>429</b>	9.060331	7.467371	8.183118	3.850148	4.430817	7.824446
<b>439</b>	7.932721	7.437206	7.828038	4.174387	6.167516	3.951244

Data points considered outliers for the feature 'Detergents\_Paper':

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

Data points considered outliers for the feature 'Delicatessen':

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

Yes, index 65, 66, 75, 128 are considered outliers for 2 features and index 154 for 3 features.

Yes, these data points should remove from the dataset.

I add datapoints which are outliers in multiple categories to outlier list to be removed. Since k-means algorithm calculate new centroids by taking the mean value of all of the samples assigned to each previous centroid, the datapoints which are outliers in multiple categories could calculate new centroids to an unexpected place and affect the analysis.

## 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.

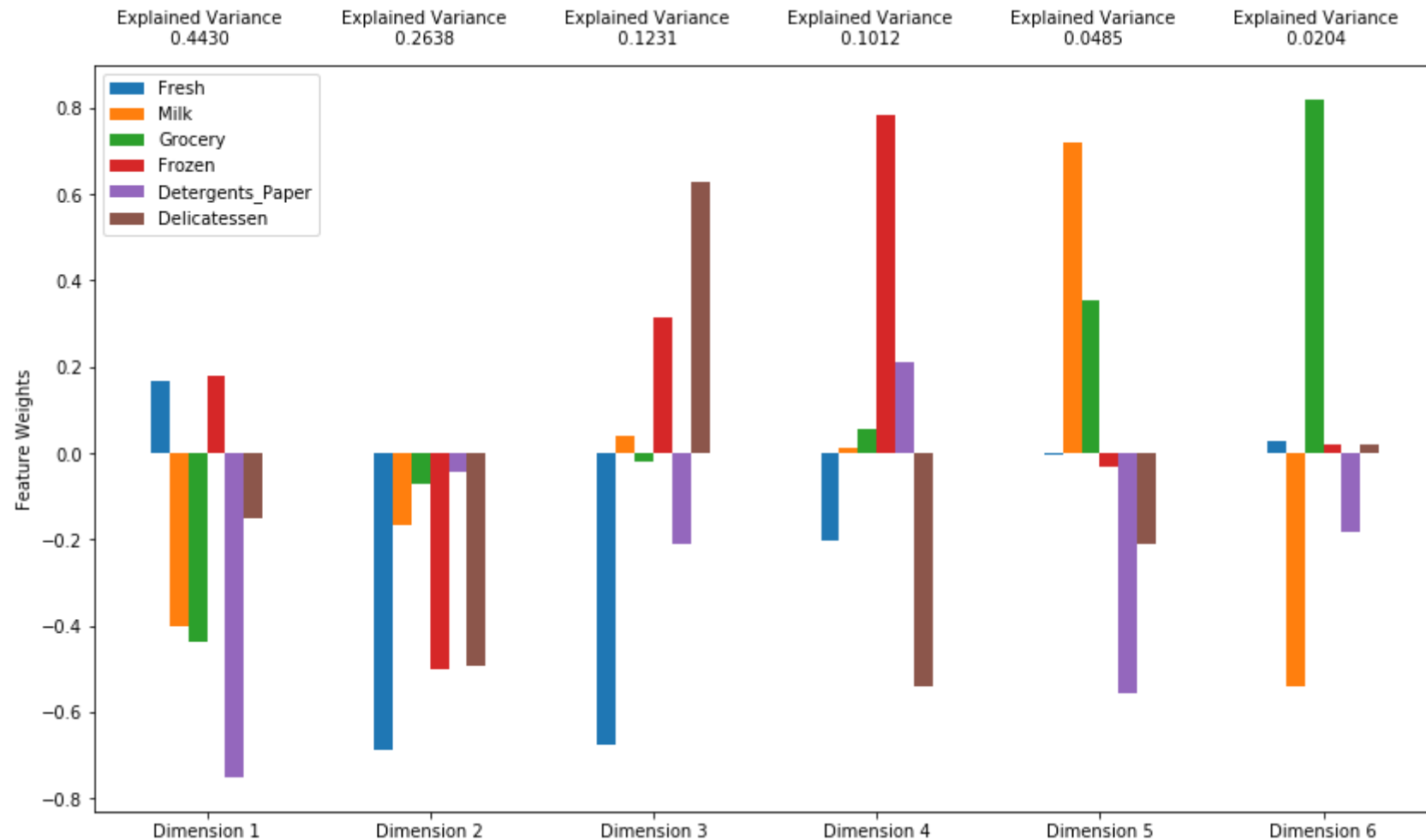
In the code block below, you will need to implement the following:

- 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 [11]: # TODO: Apply PCA by fitting the good data with the same number of dimensions as features
from sklearn.decomposition import PCA
pca = PCA(n_components=6)
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:

The variance in the data is explained in total by the first and second principal component is 0.719.

The variance in the data is explained in total by the first four principal component is 0.931.

The first dimensions: Detergents\_Paper, Grocery and Milk are well represented. It could be super market's customer spending.

The second dimensions: Fresh, Delicatessen and Frozen are well represented. It could be from restaurant.

The third dimensions: negative Fresh and positive Delicatessen are well represented. It could be from market.

The fourth dimensions: positive Frozen and negative Delicatessen are well represented. It could be from convenient store.

## 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 [12]: # 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	8.3026	-0.4747	0.9774	-1.1195	-0.2977	-4.3514
1	-4.2903	-1.4952	-1.4997	0.1394	1.1469	-0.6255
2	-0.4232	-5.2842	2.7396	-0.2364	2.1331	0.2452

## 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 significant 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 [13]: # 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 [14]: # Display sample log-data after applying PCA transformation in two dimensions
display(pd.DataFrame(np.round(pca_samples, 4), columns = ['Dimension 1', 'Dimension 2']))
```

	Dimension 1	Dimension 2
0	8.3026	-0.4747
1	-4.2903	-1.4952
2	-0.4232	-5.2842

## Visualizing a Biplot

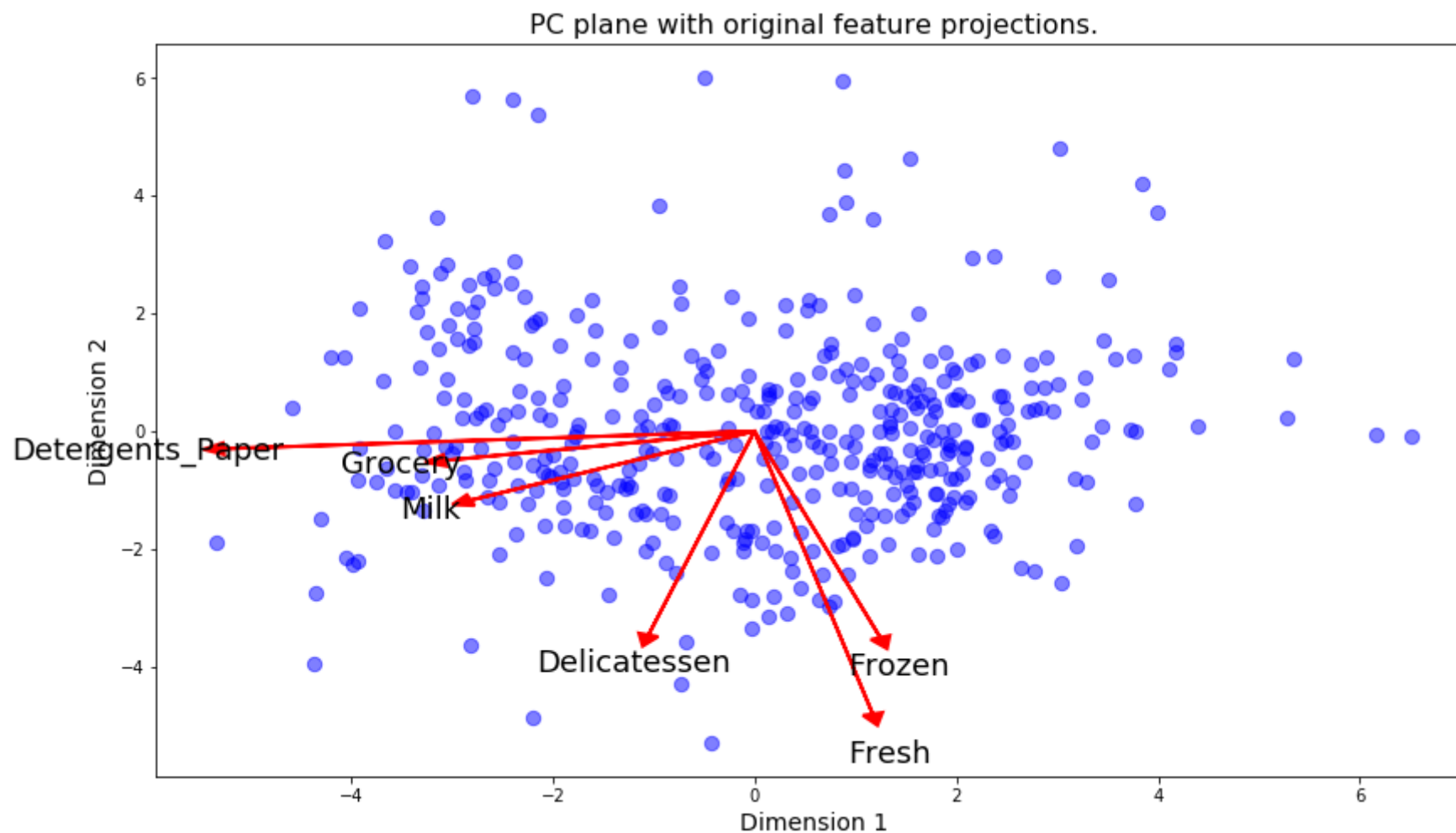
A biplot is a scatterplot where each data point is represented by its scores along the principal components. The axes are the principal components (in this case `Dimension 1` and `Dimension 2`). In addition, the biplot shows the projection of the original features along the components. A biplot can help 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 [15]: # Create a biplot  
vs.biplot(good_data, reduced_data, pca)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f38bb8df080>
```



## 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:**

Advantages to using a K-Means clustering algorithm:

1. simple, easy to implement
2. easy to interpret the clustering results
3. fast and efficient to compute

Advantages to using a Gaussian Mixture Model clustering algorithm:

1. does not assume clusters to be of any geometry
2. does not bias the cluster sizes

In my opinion, Gaussian Mixture Model clustering algorithm is soft clustering and which would be appropriate for our dataset.

## 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) ([http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette\\_score.html](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 [29]: # TODO: Apply your clustering algorithm of choice to the reduced data
from sklearn.mixture import GaussianMixture
clusterer = GaussianMixture(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 chosen
from sklearn.metrics import silhouette_score
score = silhouette_score(reduced_data, preds)
display(score)

0.42232468264593881
```

## Question 7

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

## Answer:

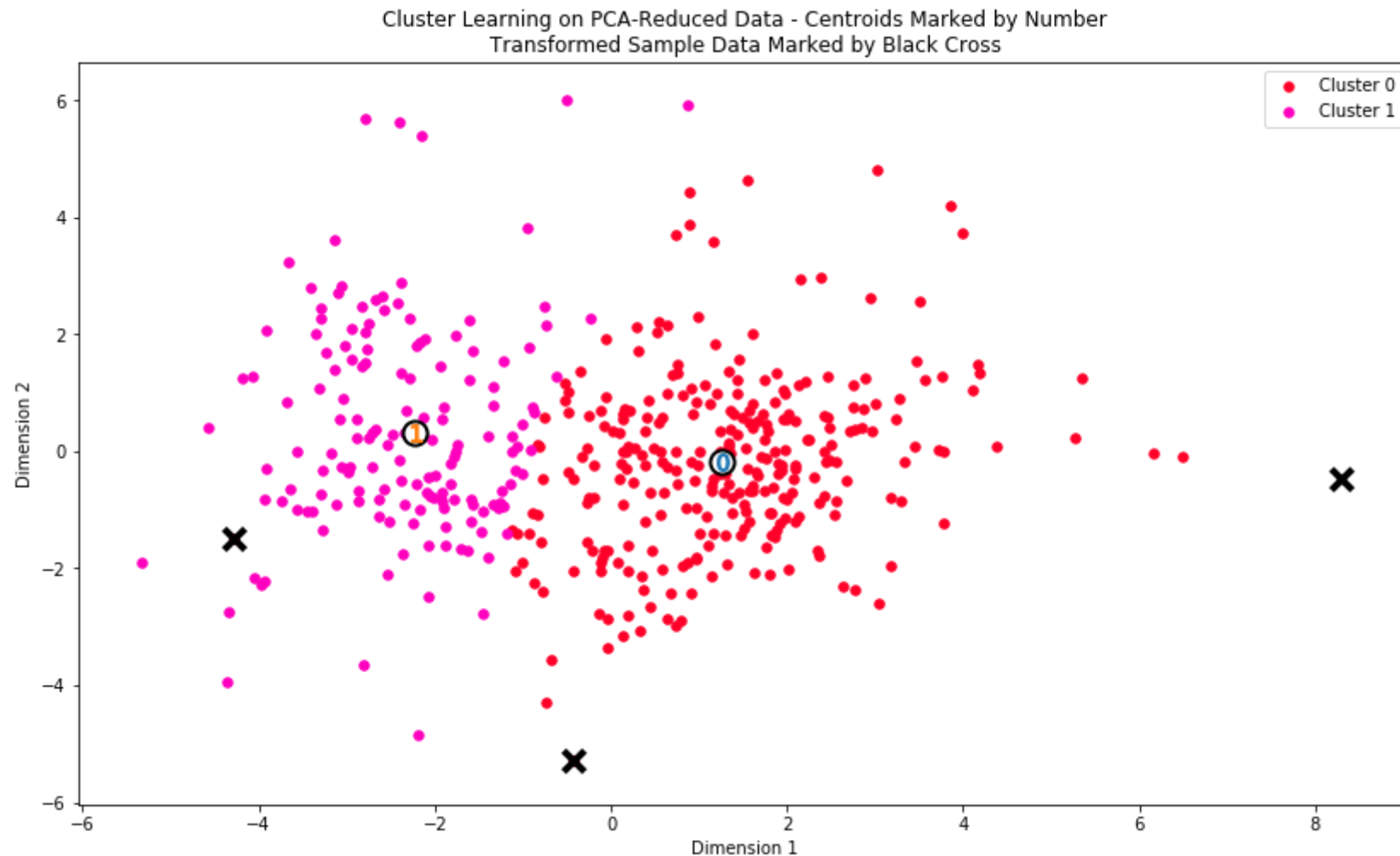
2: 0.422  
3: 0.404  
4: 0.330  
5: 0.333  
6: 0.281

When cluster number equals to 2 has the best silhouette score.

## 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 [30]: # 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 [31]: # TODO: Inverse transform the centers
log_centers = pca.inverse_transform(centers)

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

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Segment 0	8939.0	2108.0	2758.0	2073.0	352.0	730.0
Segment 1	3567.0	7860.0	12249.0	873.0	4713.0	966.0

## 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:

cluster0 could be restaurant. The cost of 'Fresh' and 'Frozen' are fairly high and close to the mean values.

cluster1 could be wholesale retailers due to the cost of 'Milk', 'Grocery' and 'Detergents\_paper' are all over their mean values.

## 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 [33]: # 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 0
```

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

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

**Answer:**

Sample point 0 predicted to be in Cluster 0

Sample point 1 predicted to be in Cluster 1

Sample point 2 predicted to be in Cluster 0

Sample point 0 is consistent that with high fresh purchase cost and could be from a restaurant.

Sample point 1 is consistent, since high purchase cost of fresh, milk, grocery and detergents\_paper could be from a wholesale.

Sample point 2 is inconsistent due to almost every products are very higher than their mean values, it is ambiguous to be in Cluster 0 and Cluster 1.

## 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](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:**

Due to cluster0 with high cost of 'Fresh' and 'Fresh' should keep fresh and can't wait too long for delivery, cluster0 will be affected more than cluster1. Running A/B tests, the wholesale distributor can randomly select some customers from cluster0 as experimental and control groups. Experimental groups deliver 3 days a week and control groups keep delivery 5 days a week to determine whether making that change will affect its customers.

**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:**

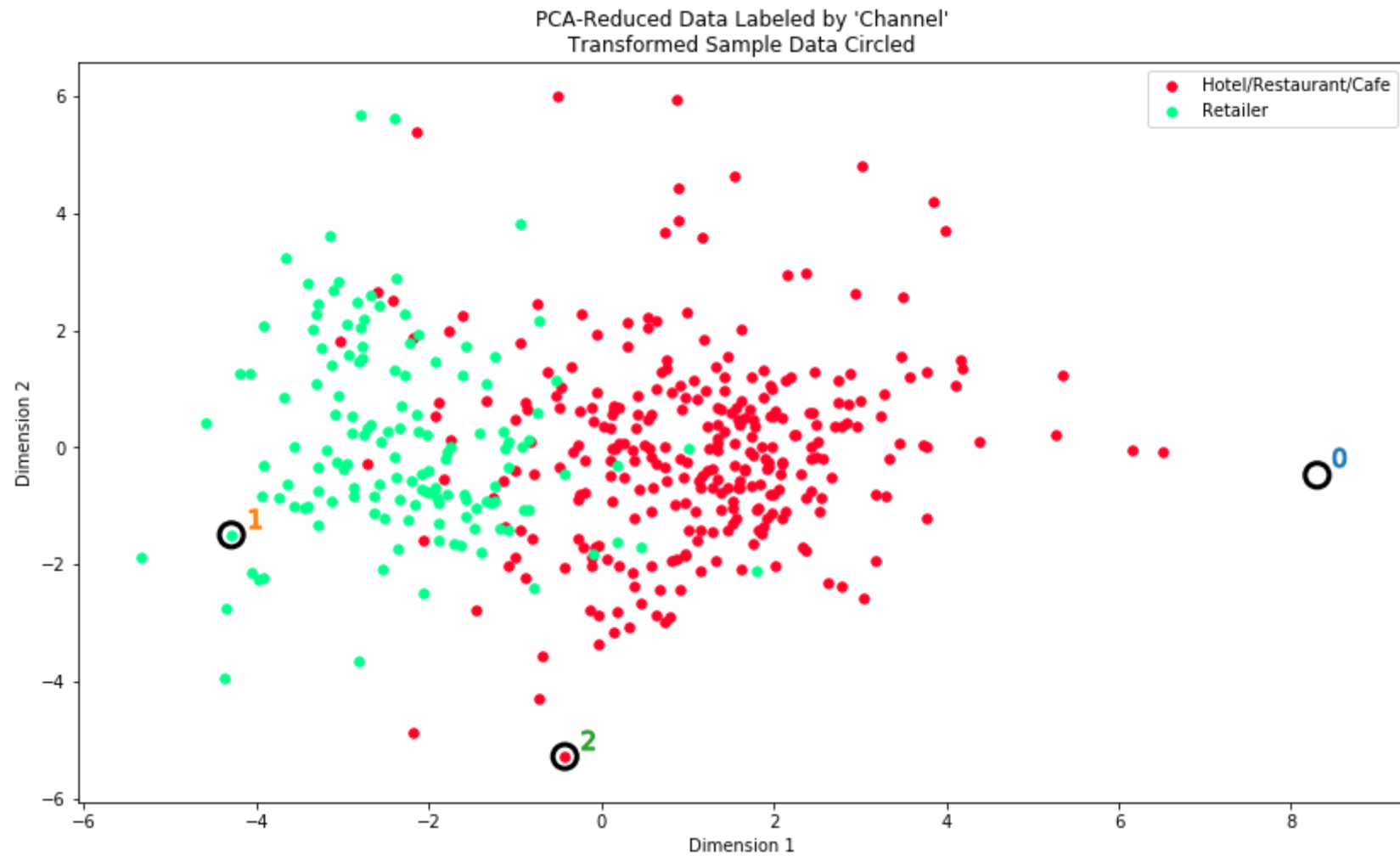
The 'customer segment' can be treated as target variable and product categories as input to train the original customers. Then according to supervised learning to predict the new customers' segment.

**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 [32]: # 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:

The silhouette score of cluster number is equals to 2 and compare to this underlying distribution, the Gaussian Mixture Model clustering algorithm does better job.

No, there are many overlaps.

No, I won't consider these classifications as consistent with my previous definition of the customer segments.

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