customer_segments

May 20, 2019

1 Machine Learning Engineer Nanodegree

1.1 Unsupervised Learning

1.2 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.

1.3 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. 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.".format(*da
        except:
            print("Dataset could not be loaded. Is the dataset missing?")
```

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

1.4 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  \
count     440.000000    440.000000    440.000000    440.000000
mean    12000.297727   5796.265909   7951.277273   3071.931818
std     12647.328865   7380.377175   9503.162829   4854.673333
min         3.000000     55.000000      3.000000     25.000000
25%      3127.750000   1533.000000   2153.000000    742.250000
50%      8504.000000   3627.000000   4755.500000   1526.000000
75%     16933.750000   7190.250000  10655.750000   3554.250000
max    112151.000000  73498.000000  92780.000000  60869.000000

       Detergents_Paper  Delicatessen
count        440.000000    440.000000
mean        2881.493182   1524.870455
```

```
std         4767.854448   2820.105937
min            3.000000      3.000000
25%          256.750000    408.250000
50%          816.500000    965.500000
75%         3922.000000   1820.250000
max        40827.000000  47943.000000
```

1.4.1 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 = [25,50,75]

        # 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	16165	4230	7595	201	4003	57
1	6269	1095	1980	3860	609	2162
2	20398	1137	3	4407	3	975

1.4.2 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: - The **First Customer(index 25)** spends heavily on Fresh, Average spending on Milk, Grocery in higher than average, Frozen items is minimal, amount on Detergents Paper is quite huge and Delicatessen is very small. From this the customer appears to be a **Fresh Retail Store** of some kind. - The **Second Customer(index 50)** spends more on the Delicatessen and Frozen items, spends less than average on Fresh items, Milk, Grocery and Detergents Paper. From this the customer appears to be **small retails store in a market**. - The **Third Customer(index 75)** spending is high for Fresh items,average spending on Frozen items, less than average spending for Milk and Delicatessen and exceptionally less spending on Grocery and Detergent Paper. From this the customer seems to be a **Large Sized Cafe**, providing for fresh items.

1.4.3 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.model_selection import train_test_split
        from sklearn.tree import DecisionTreeRegressor

        features = list(data.columns)

        # TODO: Make a copy of the DataFrame, using the 'drop' function to drop the given featur
        new_data = data.drop('Delicatessen', axis=1)

        # TODO: Split the data into training and testing sets(0.25) using the given feature as t
        # Set a random state.
        X_train, X_test, y_train, y_test = train_test_split(new_data, data['Delicatessen'], test

        # TODO: Create a decision tree regressor and fit it to the training set
        regressor = DecisionTreeRegressor(random_state=42).fit(X_train, y_train)

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

        print(score)
```

```
-2.2547115372
```

1.4.4 Question 2

- Which feature did you attempt to predict?
- What was the reported prediction score?
- Is this feature necessary for identifying customers' spending habits?

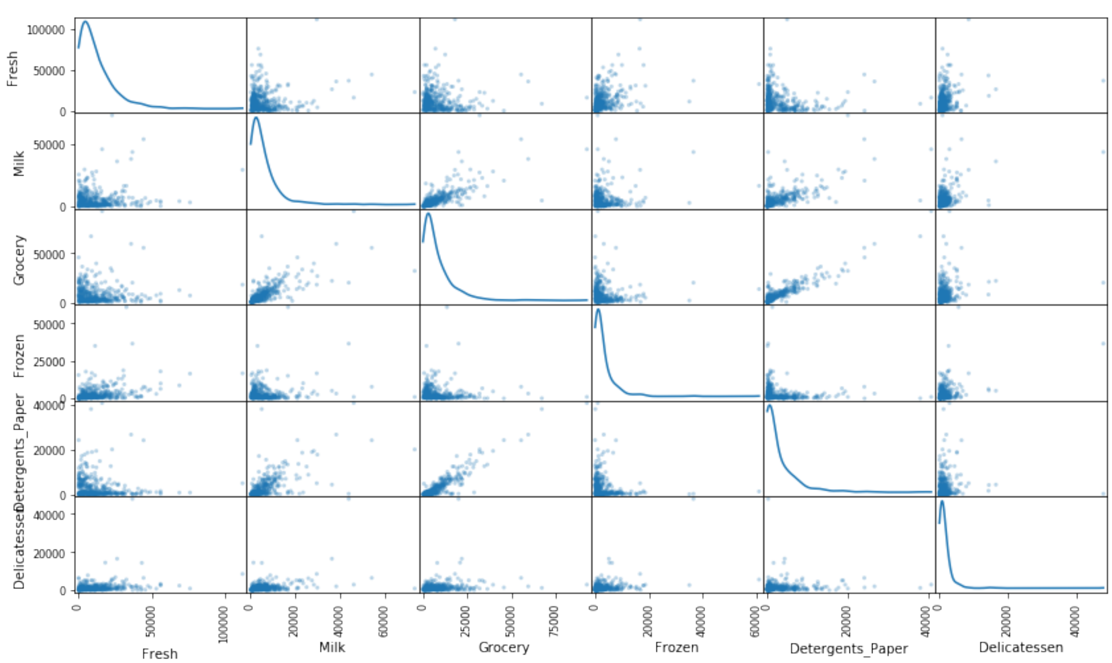
Hint: The coefficient of determination, `R^2`, is scored between 0 and 1, with 1 being a perfect fit. A negative `R^2` implies the model fails to fit the data. If you get a low score for a particular feature, that lends us to beleive that that feature point is hard to predict using the other features, thereby making it an important feature to consider when considering relevance.

Answer: - I attempted to predict the Delicatessen feature. - The reported prediction score value is **-2.2547115372**. - The negative value indicates that it this feature cannot be predicted by other features in the dataset. So it makes this feature very important for identifying the customer spending habits.

1.4.5 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');
```



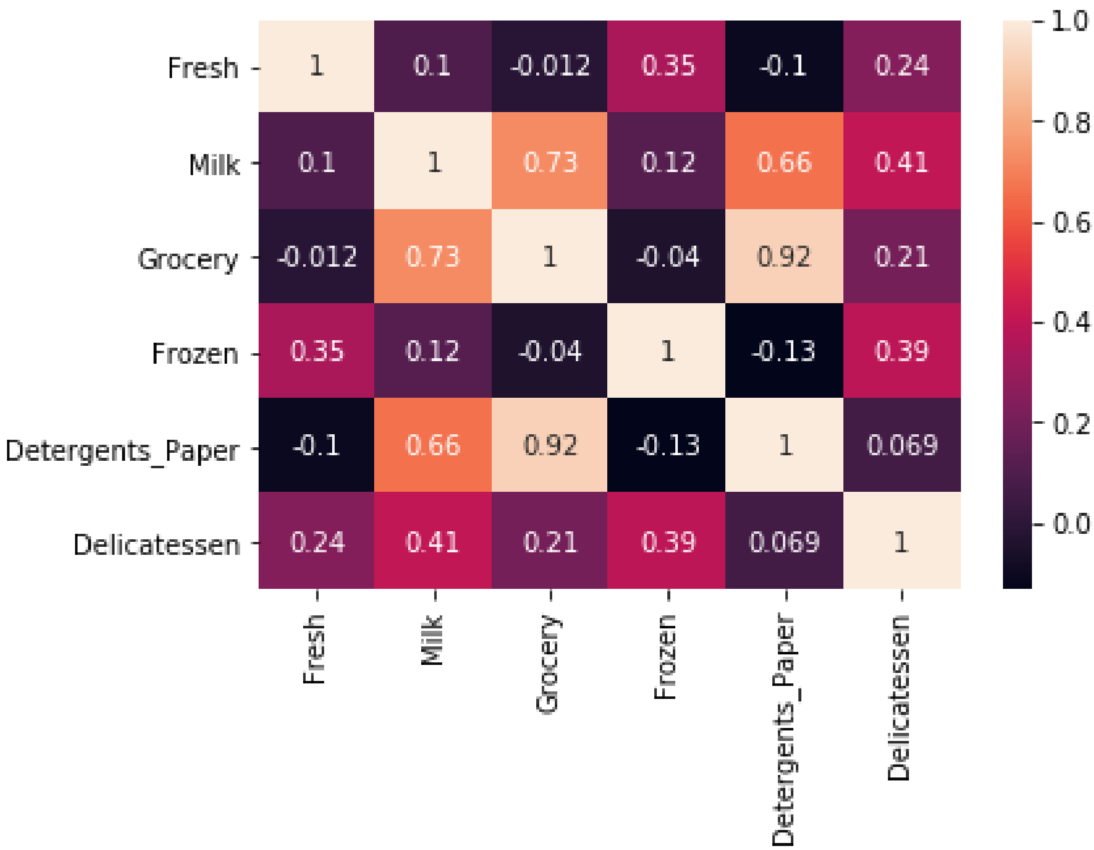
1.4.6 Question 3

- 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() to get the feature correlations and then visualize them using a heatmap (the data that would be fed into the heatmap would be the correlation values, for eg: `data.corr()`) to gain further insight.

In [6]:
```
from seaborn import heatmap

heatmap(data.corr(), annot=True)
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3adf0b5cc0>
```



Answer: 1. From the scatter matrix, we can see the data is **not distributed Normally** as there are many outliers. Majority of it is skewed to the left where most of the data points are. This indicates us to normalize the data, as required by the clustering algorithms. 2. Most Correlated Features: - **Detergents_Paper and Grocery:** We can see a linear correlation with a coefficient of 0.92. - **Grocery and Milk:** Linear correlation with a coefficient of 0.73. - **Detergents_Paper and Milk:** Linear correlation with a coefficient of 0.66. 3. The results does not fall in line about the relevance of Delicatessen. I had suspected that Delicatessen might be a strong indicator for general patterns. But as visible in the scatter plot its not a strong feature. 4. This might be due to Delicatessen having the smallest standard deviation compared to the other categories here.The data for Delicatessen (X-axis) is distributed near a vertical line, indicating that there is either high or low Delicatessen spending, there is no correlation with the Y-axis.

1.5 Data Preprocessing

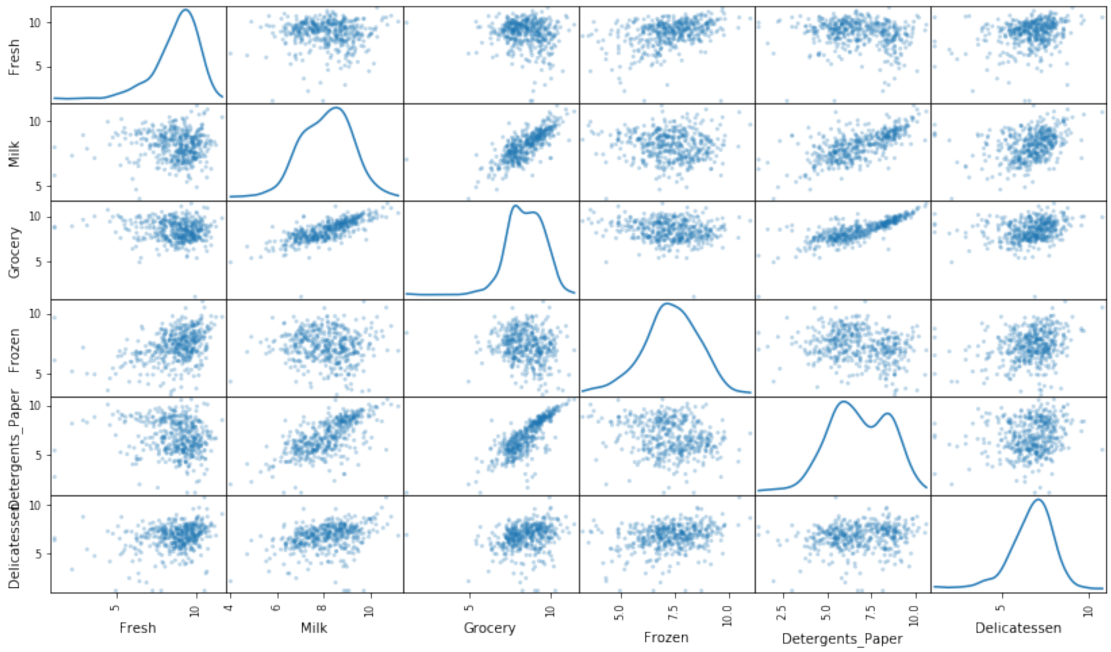
In this section, you will preprocess the data to create a better representation of customers by performing a scaling on the data and detecting (and optionally removing) outliers. Preprocessing data is often times a critical step in assuring that results you obtain from your analysis are significant and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

1.5.1 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 to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test, 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.

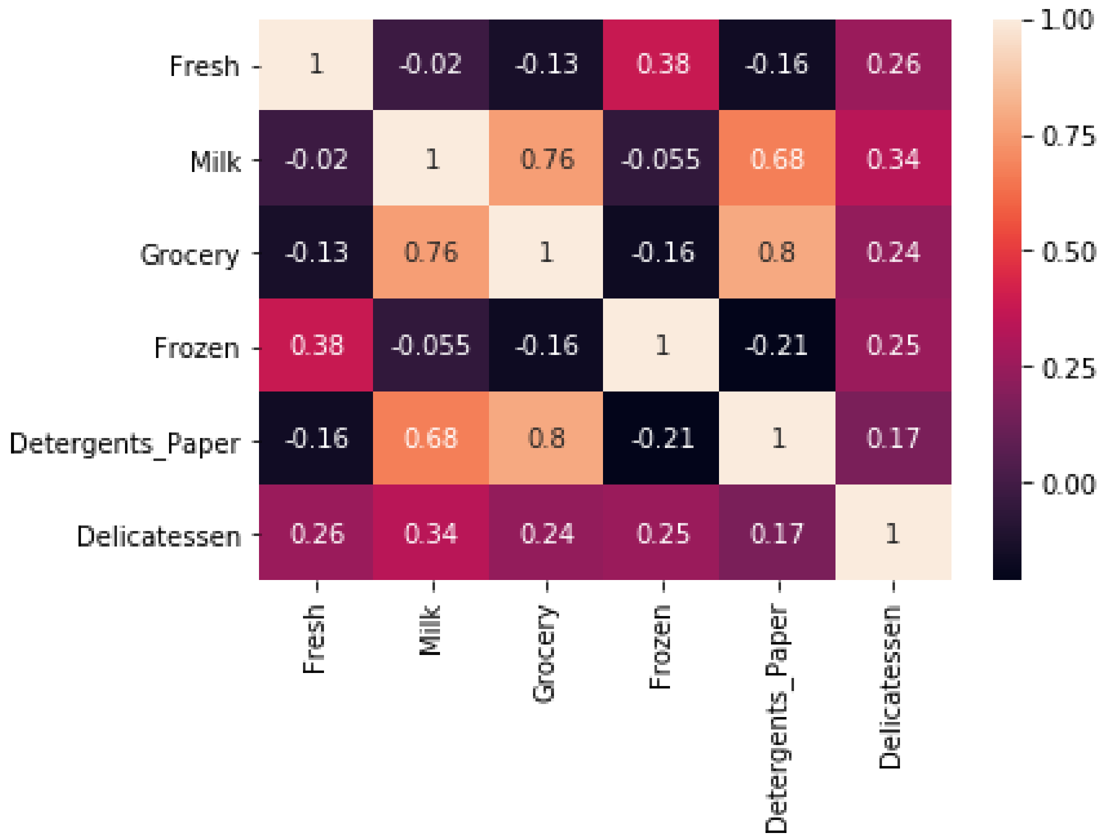
In the code block below, you will need to implement the following: - Assign a copy of the data to `log_data` after applying logarithmic scaling. Use the `np.log` function for this. - Assign a copy of the sample data to `log_samples` after applying logarithmic scaling. Again, use `np.log`.

```
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.plotting.scatter_matrix(log_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
```



In [8]: heatmap(log_data.corr(), annot=True)

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3ae622f320>



1.5.2 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
        print("Samples of the customers dataset: {}".format(indices))
        display(log_samples)
```

```
Samples of the customers dataset: [25, 50, 75]
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9.690604	8.349957	8.935245	5.303305	8.294799	4.043051
1	8.743372	6.998510	7.590852	8.258422	6.411818	7.678789
2	9.923192	7.036148	1.098612	8.390949	1.098612	6.882437

1.5.3 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: 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]:
```
outliers = []
# 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 int
    step = 1.5 * (Q3 - Q1)

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

# OPTIONAL: Select the indices for data points you wish to remove
outliers = list(set([x for x in outliers if outliers.count(x) > 1]))
print ("Outliers: {}".format(outliers))

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

print("There is {} good data out of {} data".format(len(good_data),len(log_data)))
```

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

	Delicatessen
66	3.295837
109	1.098612
128	1.098612
137	3.583519
142	1.098612
154	2.079442
183	10.777768
184	2.397895
187	1.098612
203	2.890372
233	1.945910
285	2.890372
289	3.091042
343	3.610918

```
Outliers: [128, 65, 66, 75, 154]
There is 435 good data out of 440 data
```

1.5.4 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: 1. Customer numbers **128, 65, 66, 75, 154** were considered outliers for multiple features. 2. The outliers should be removed as there are smaller number of data points(440), it can have an impact on out analysis. 3. The outliers impact the classification done by k-means and similar algorithms which do not perform well with the outliers. The data points outside the norm are likely to just get mis-classified instead of aiding the classification problem we are trying to solve.

1.6 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.

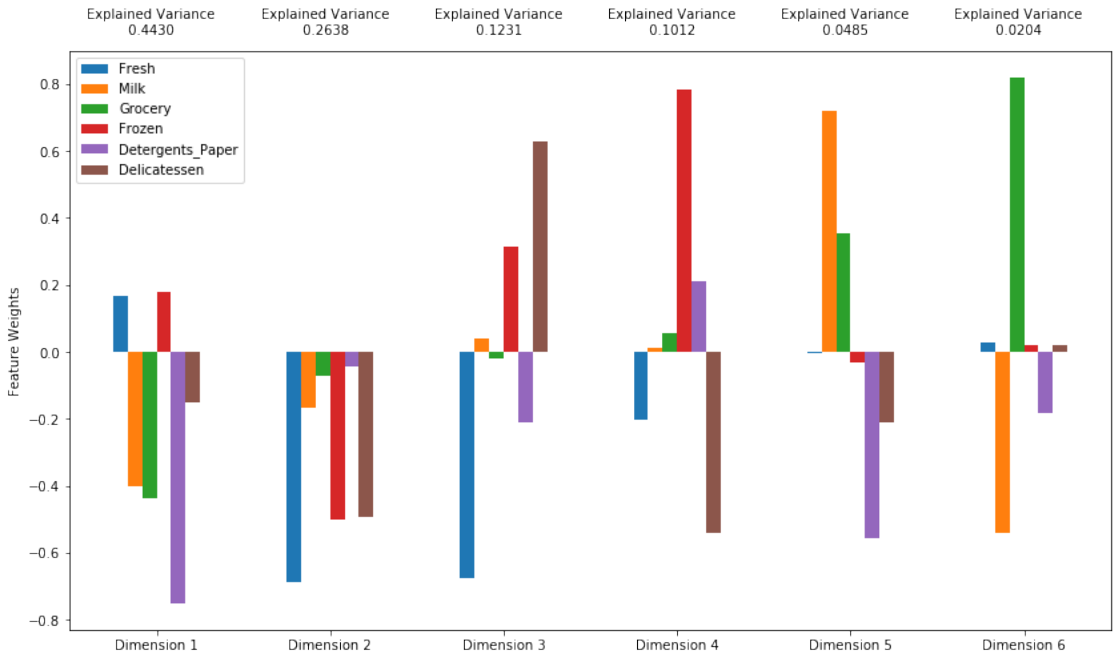
1.6.1 Implementation: PCA

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]: from sklearn.decomposition import PCA

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

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

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




```python
In [12]: display(pca_results)
         print(pca_results['Explained Variance'].cumsum())
```

	Explained Variance	Fresh	Milk	Grocery	Frozen
Dimension 1	0.4430	0.1675	-0.4014	-0.4381	0.1782
Dimension 2	0.2638	-0.6859	-0.1672	-0.0707	-0.5005
Dimension 3	0.1231	-0.6774	0.0402	-0.0195	0.3150
Dimension 4	0.1012	-0.2043	0.0128	0.0557	0.7854
Dimension 5	0.0485	-0.0026	0.7192	0.3554	-0.0331
Dimension 6	0.0204	0.0292	-0.5402	0.8205	0.0205

	Detergents_Paper	Delicatessen
Dimension 1	-0.7514	-0.1499
Dimension 2	-0.0424	-0.4941
Dimension 3	-0.2117	0.6286
Dimension 4	0.2096	-0.5423
Dimension 5	-0.5582	-0.2092
Dimension 6	-0.1824	0.0197

```
Dimension 1    0.4430
Dimension 2    0.7068
```

```
Dimension 3    0.8299
Dimension 4    0.9311
Dimension 5    0.9796
Dimension 6    1.0000
Name: Explained Variance, dtype: float64
```

1.6.2 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: 1. The first and second Pricipal Component contributes to 70.68% variance in the data. 2. The first four Pricipal Component contributes to 93.11% variance in the data. 3. - **Dimension 1:** Increase in Dimension 1 is associated with increase in Milk, Grocery and Detergents_Paper spending, these features dominate the definition of First Dimension. This is in line with our initial findings where the 3 features are highly correlated. - **Dimension 2:** Increase in Dimension 2 is associated with increases in "Fresh", "Frozen" and "Delicatessen" spending, these features best represent the Second Dimension. This makes sense as First Dimension represents different features, Dominating features in First Dimension have very small positive weights in the Second Dimension. - **Dimension 3:** Increase in Dimension 3 is associated with increase in "Delicatessen" and decrease in "Fresh" spending, these features best represent the Third Dimension. - **Dimension 4:** An increase in Dimension 4 is associated with increase in "Frozen" and decrease in "Delicatessen" spending, these features best represent the Fourth Dimension.

1.6.3 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  \
0      -1.2292       1.5540      -3.2462       0.0043       0.1124
1       1.1404      -0.6710       0.9823       0.0603      -1.1423
2       8.3026      -0.4747       0.9774      -1.1195      -0.2977

   Dimension 6
```

```
0       -0.0697
1        0.0055
2       -4.3514
```

1.6.4 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).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'])
```

1.6.5 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', 'Dimension 2']
```

```
   Dimension 1  Dimension 2
0      -1.2292       1.5540
1       1.1404      -0.6710
2       8.3026      -0.4747
```

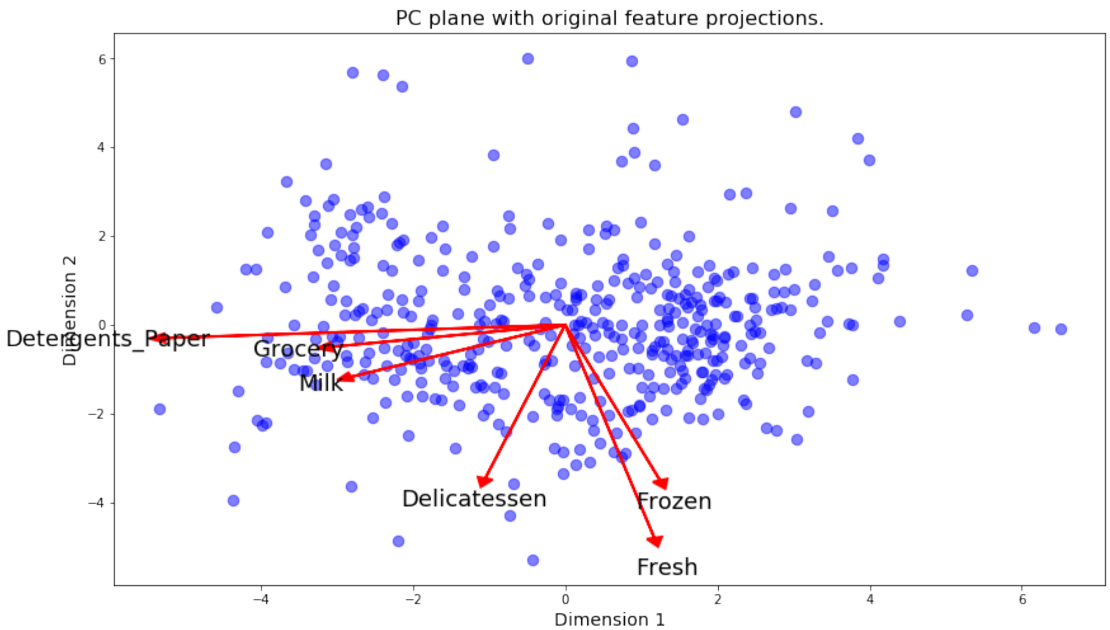
1.7 Visualizing a Biplot

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)

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3ae5f91b70>
```



1.7.1 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?

1.8 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.

1.8.1 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: 1. K-Means Clustering - **Advantages:** K-Means is a fast and efficient algorithm with easy implementation,it works well with data containing more dimensions. - **Disadvantages:** K-means highly depends on the starting points chosen. K-Means does a hard clustering, where each data point(even outliers) is determined to be within the cluster. K-Means is not aware of the density of the clusters.

2. Gaussian Mixture Modeling (GMM)

    - **Advantages:** GMM is good on clusters of any geometry and does not bias the clusters to be roughly-circular i.e. it uses soft classification or soft clustering, where each point is assigned a probability to which cluster it belongs to. GMM tends to perform better than K-Means where data point density is non-uniform.
    - **Disadvantages:** Could fail where the number of dimensions is high. Also, take more time compared to K-Means.

3. Choice of Algorithm:

    - I would go with **Gaussian Mixture Model** because of its ability to apply "soft" classification, also as the dimension of the problem is reduced to 2 using PCA, GMM should be able to do the job efficiently. If our dataset was larger, k-Means would have been the choice.

1.8.2 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 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 [17]: from sklearn.mixture import GaussianMixture
         from sklearn.metrics import silhouette_score
```

```
def exec_clusterer(k):
    # TODO: Apply your clustering algorithm of choice to the reduced data
    clusterer = GaussianMixture(n_components=k, random_state=0).fit(reduced_data)

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

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

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

    # TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
    score = silhouette_score(reduced_data,preds)
    return clusterer, preds, centers, sample_preds, score
```

1.8.3 Question 7

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

```
In [18]: best_cluster_score = 0
         best_cluster = 0

         for i in range(2, 21):
             clusterer, preds, centers, sample_preds, score = exec_clusterer(i)
             print("The score with {} clusters is: {}".format(i, score))

             if score > best_cluster_score:
                 best_cluster_score = score
                 best_cluster = i

         clusterer, preds, centers, sample_preds, score = exec_clusterer(best_cluster)

         print("\nThe best performing cluster count is: {}, with a score of: {}".format(best_clu
```

```
The score with 2 clusters is: 0.4219168464626149
The score with 3 clusters is: 0.37420184754032176
The score with 4 clusters is: 0.32908046670035607
The score with 5 clusters is: 0.30587455287316045
The score with 6 clusters is: 0.2248151815327168
The score with 7 clusters is: 0.27433381348916297
The score with 8 clusters is: 0.3443700570270379
The score with 9 clusters is: 0.3343238853627524
The score with 10 clusters is: 0.3406316553045381
The score with 11 clusters is: 0.3197157686178644
```

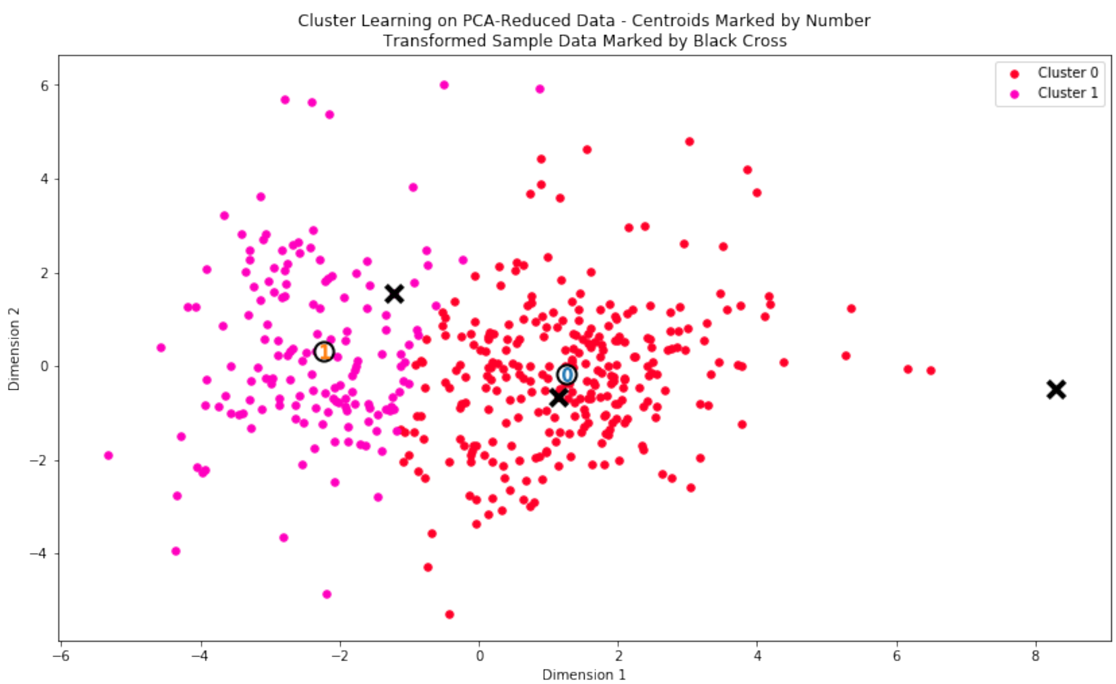
```
The score with 12 clusters is: 0.32155333382067597
The score with 13 clusters is: 0.3146131722471681
The score with 14 clusters is: 0.34109260319943646
The score with 15 clusters is: 0.35607218373039873
The score with 16 clusters is: 0.324463762903454
The score with 17 clusters is: 0.3118877656772356
The score with 18 clusters is: 0.3062853063704496
The score with 19 clusters is: 0.3166755356716159
The score with 20 clusters is: 0.31996784719174437

The best performing cluster count is: 2, with a score of: 0.4219168464626149
```

Answer: As seen with the above output; The best performing GMM model has a **cluster count of 2**, with a score of **0.4219168464626149**

1.8.4 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 [19]: # Display the results of the clustering from implementation
         vs.cluster_results(reduced_data, preds, centers, pca_samples)
```



1.8.5 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 [20]: # 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	8953.0	2114.0	2765.0	2075.0	353.0	732.0
Segment 1	3552.0	7837.0	12219.0	870.0	4696.0	962.0

1.8.6 Question 8

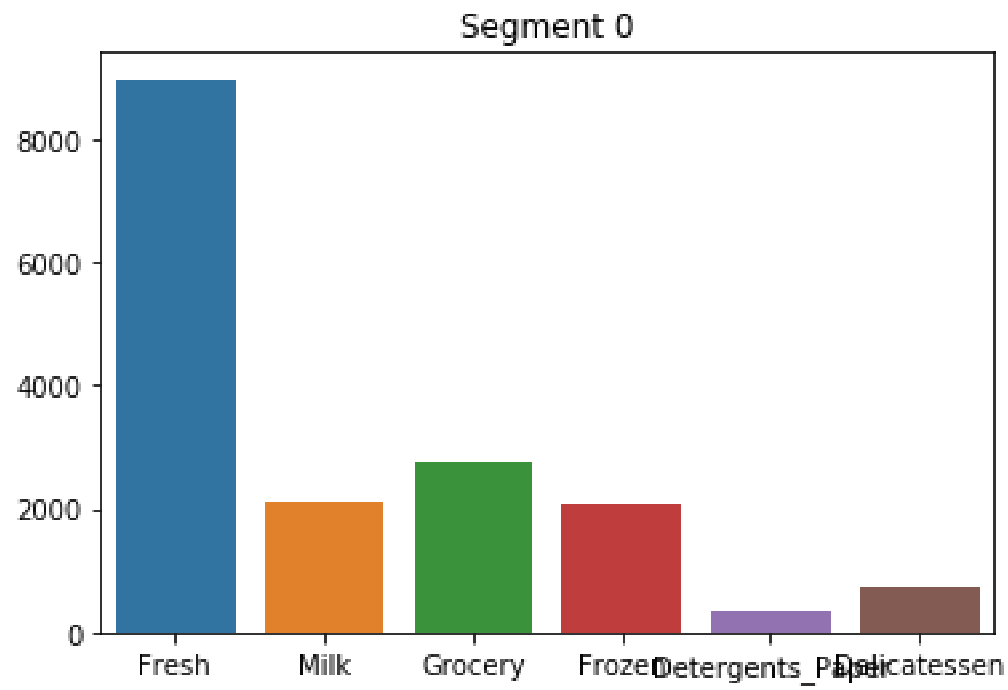
- Consider the total purchase cost of each product category for the representative data points above, and reference the statistical description of the dataset at the beginning of this project(specifically looking at the mean values for the various feature points). What set of establishments could each of the customer segments represent?

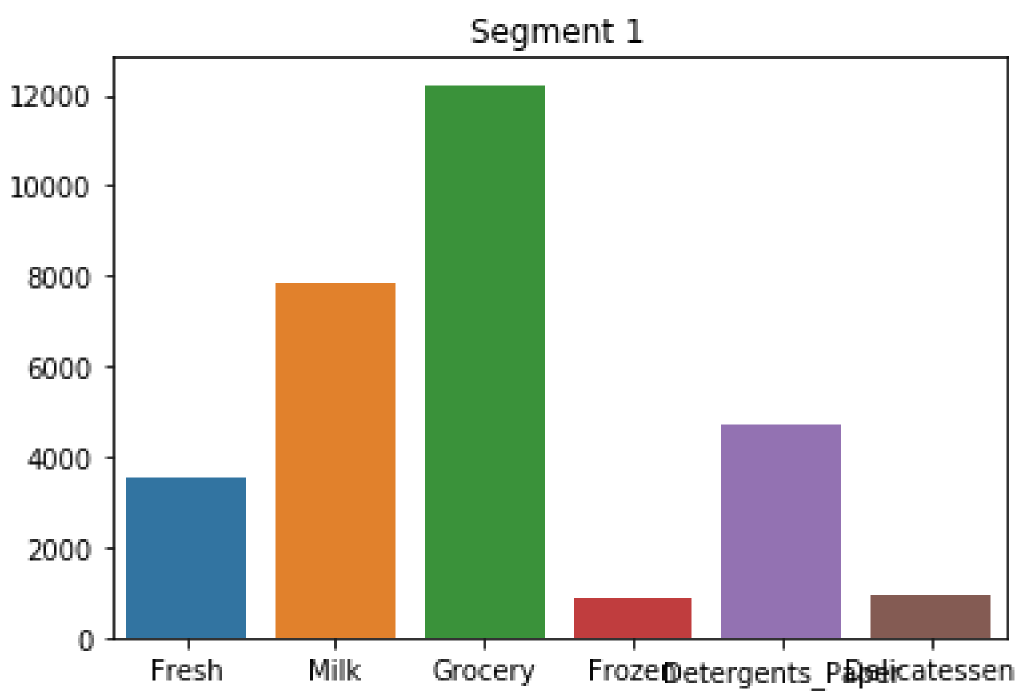
Hint: A customer who is assigned to `'Cluster X'` should best identify with the establishments represented by the feature set of `'Segment X'`. Think about what each segment represents in terms their values for the feature points chosen. Reference these values with the mean values to get some perspective into what kind of establishment they represent.

```
In [21]: import matplotlib.pyplot as plt
         import seaborn as sns

         plt.figure()
         plt.axes().set_title("Segment 0")
         sns.barplot(x=true_centers.columns.values,y=true_centers.iloc[0].values)
```

```
plt.figure()
plt.axes().set_title("Segment 1")
sns.barplot(x=true_centers.columns.values,y=true_centers.iloc[1].values)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3ae60cd860>





Answer: 1. **Cluster/Segment 0:** The strong weight on the Fresh category strongly suggests that this cluster segment represents **cafes serving fresh food**. Its also seen that the volume is below the overall mean, it falls in line with the original prediction for what a Cafe might look like in the Data Exploration done above.

2. **Cluster/Segment 1:** Here strong weights on Milk and Grocery suggests that his cluster segment represents a **Super-Market or a Whole-Sale retailer**. The values of the highlighted features is exceeding the overall mean observed in the Data Exploration section, further proves this cluster to be Super-Market or a Whole-Sale retailer.

1.8.7 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 [22]: # Display the predictions
         for i, pred in enumerate(sample_preds):
             print("Sample point", i, "predicted to be in Cluster", pred)
```

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

```
In [23]: display(samples)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	16165	4230	7595	201	4003	57
1	6269	1095	1980	3860	609	2162
2	20398	1137	3	4407	3	975

Answer: 1. First Customer(index 25) is classified in **Cluster 1(Super-Market or a Whole-Sale retailer)**. - Predicted as **Fresh Retail Store**. 2. Second Customer (index 50) is classified in **Cluster 0(cafes serving fresh food)**. - Predicted as **Small Retail Store**. 3. Third Customer (index 75) is also classified in **Cluster 0(cafes serving fresh food)**. - Predicted as **Large Size Cafe**.

The prediction done in Question 1 about the three Customers goes true for First and Third Customer. However, the prediction about the Second Customer goes wrong.

1.9 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.

1.9.1 Question 10

Companies will often run A/B tests 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 two classification cluster that we came across in question 8 are **Cluster 0(cafes serving fresh food)** & **Cluster 1(Super-Market or a Whole-Sale retailer)** The change that the company is trying to introduce(change in delivery time by making early deliveries) directly impacts for products who are to be served fresh and have less shelf-live.

In this case; the customers from cluster 0(cafes) can be considered for the A/B testing, while the customers in cluster 1(Whole-Sale retailers) would be flexible with the delivery, as they would already have lot of stock for clearing; they wont agree for the premium that they might have to pay for early delivery.

1.9.2 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: Any of the supervised learning algorithms taught in previous classes like support vector machines, naive bayes, logistic regression etc can be used to classify the new clients based on their features.

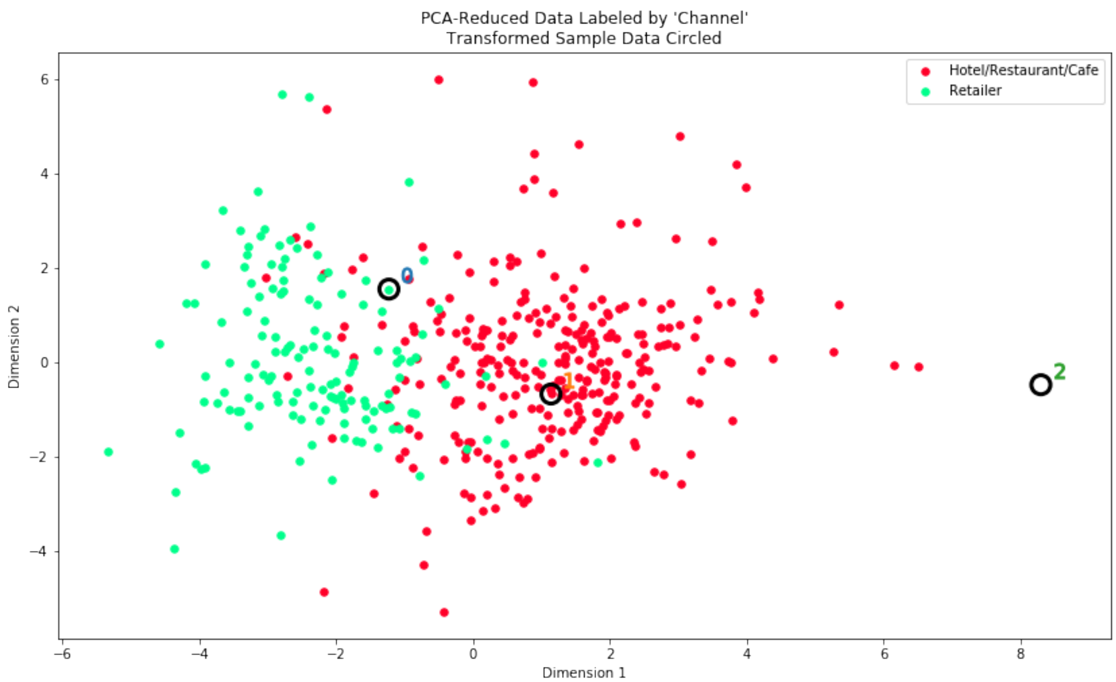
Target variable should be the cluster group. In this case. - 0: cluster 0. - 1: cluster 1.

1.9.3 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)
```



1.9.4 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: 1. The chosen clustering algorithm(Gaussian Mixture Model) and the number of classified clusters(2 in number) is consistent with the Distribution as seen in the plot. 2. There is no clear boundaries as such; but the customer segments in the extremes of horizontal axis will represent purely Hotels/Restaurants/Cafes(right extreme) or Retailers(extreme left). 3. The classifications (**Hotels/Restaurants/Cafes & Retailers**) looks consistent to the **Cluster 0(cafes serving fresh food) & Cluster 1(Super-Market or a Whole-Sale retailer)**.

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.