customer_segments

July 8, 2016

1 Machine Learning Engineer Nanodegree

1.1 Unsupervised Learning

1.2 Project 3: Creating Customer Segments

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

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

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

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

        # Show matplotlib plots inline (nicely formatted in the notebook)
        %matplotlib inline
        # Load the wholesale customers dataset
        try:
            data = pd.read_csv("customers.csv")
            data.drop(['Region', 'Channel'], axis = 1, inplace = True)
            print "Wholesale customers dataset has {} samples with {} features each.".format(*data.shap
        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 [3]: # 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 [4]: # TODO: Select three indices of your choice you wish to sample from the dataset
        indices = [12,56,390]
        # 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  31714  12319    11757     287              3881          2931
1   4098  29892    26866    2616             17740          1340
2   3352   1181     1328    5502               311          1000
```

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, and retailers, among many others. Avoid using names for establishments, such as saying "McDonalds" when describing a sample customer as a restaurant.

Answer: For the first random example, (index 12) of the original dataset, we can see that the all of the features are above the mean except frozen. This clearly indicate a large sized retail store.

In the second random example (index 56) the Milk, Grocery and Detergent Paper features are quite higher than the mean, but all the others are below their corresponding means. This is a relatively confusing example that would arguably resemble something like a baking store, which sells milk products or a retailler specialized in milk products.

The values of the features of the third example (index 390) are well below their corresponding means, except for the frozen feature. This record could potentially point to a small sized store.

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

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

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

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

```
-0.210135890125
```

1.4.4 Question 2

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

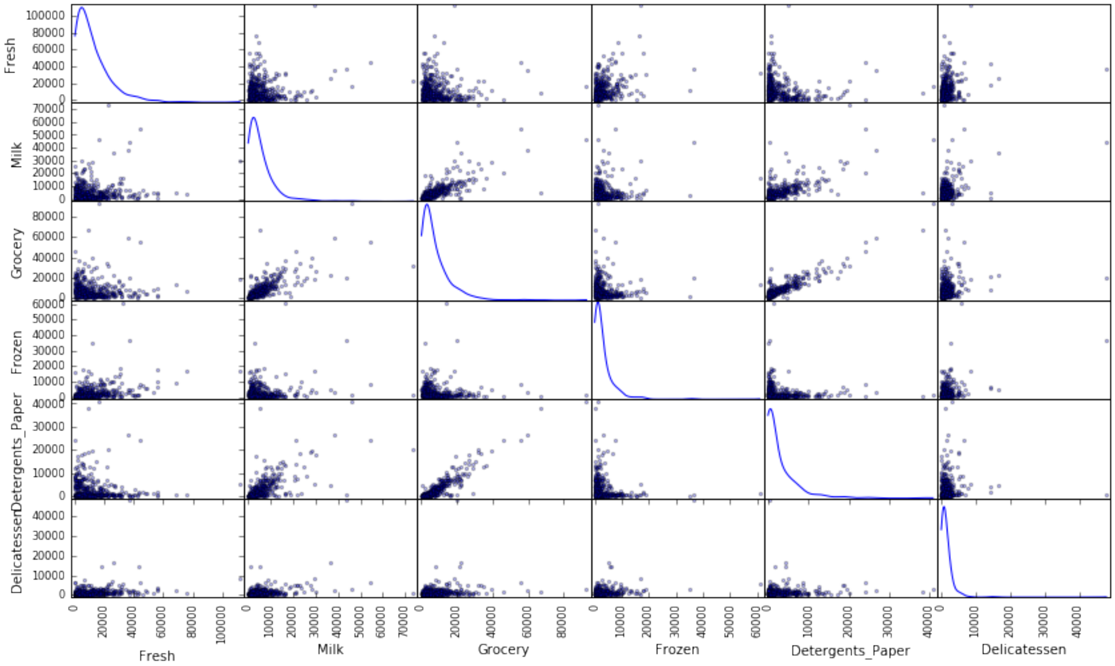
Answer: I attempted predicting Frozen feature and the reported prediction score was -0.21.

The negative R^2 score in this case suggests that the remaining features do not already contain the information found in the selected feature (i.e. the target feature is telling something new about the sample) and therefore this feature is necessary for identifying customers' spending habit.

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 [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');
```

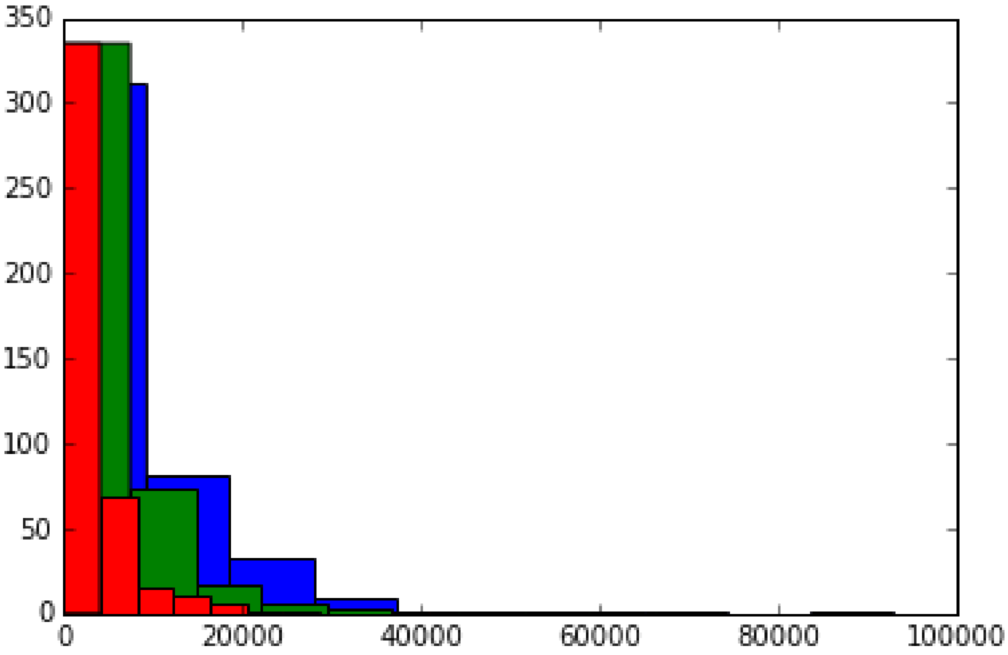


1.4.6 Question 3

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

```
In [7]: import matplotlib.pyplot as plt
        plt.hist(new_data['Grocery'])
        plt.hist(new_data['Milk'])
        plt.hist(new_data['Detergents_Paper'])
```

```
Out[7]: (array([ 335.,   68.,   16.,   10.,    6.,    2.,    1.,    0.,    0.,    2.]),
         array([  3.00000000e+00,   4.08540000e+03,   8.16780000e+03,
                  1.22502000e+04,   1.63326000e+04,   2.04150000e+04,
                  2.44974000e+04,   2.85798000e+04,   3.26622000e+04,
                  3.67446000e+04,   4.08270000e+04]),
         <a list of 10 Patch objects>)
```



Answer: From the scatter matrix above we can see a clear correlation between grocery and detergents paper, while there is a level of correlation between milk and grocery as well as milk and detergents paper also. One the other hand, the 'Frozen' feature we chose as target for the previous question, has clearly no correlation with any of the other features, which confirms the finding that this feature cannot be used for predicting it.

The histogram for the three features clearly depicts a similar distribution for all of them which is right skewed.

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

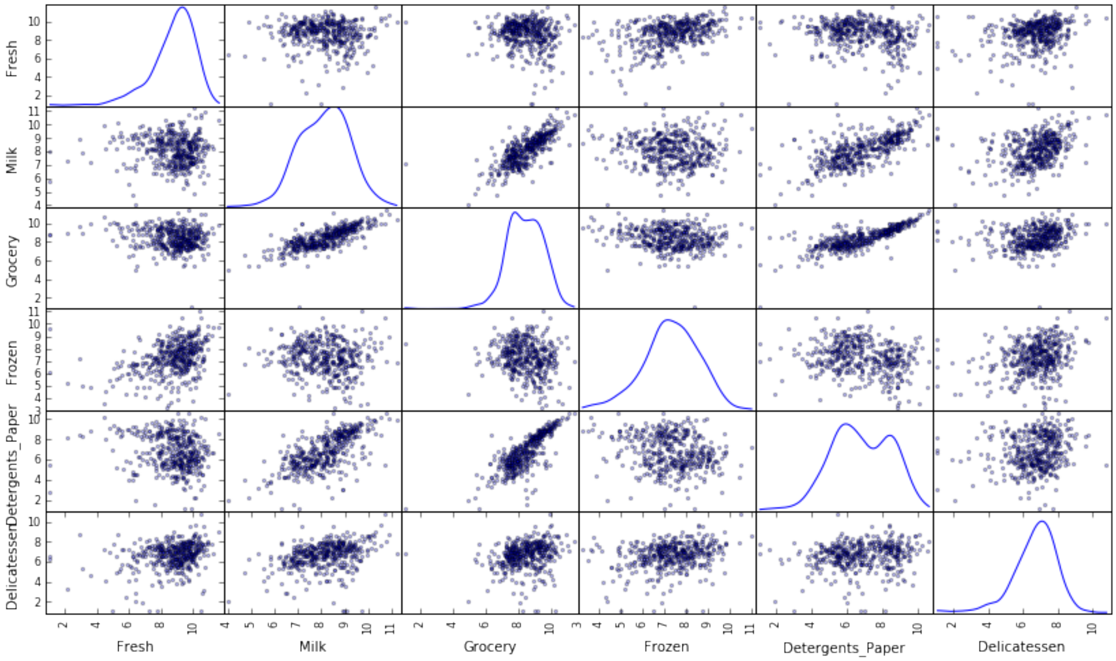
If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test, which calculates the best power 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 a logarithm scaling. Use the `np.log` function for this. - Assign a copy of the sample data to `log_samples` after applying a logrithm scaling. Again, use `np.log`.

```
In [6]: # 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');
```



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 [8]: # Display the log-transformed sample data
        display(log_samples)
        Fresh       Milk    Grocery    Frozen  Detergents_Paper  Delicatessen
0  10.364514   9.418898   9.372204  5.659482          8.263848      7.983099
1   8.318254  10.305346  10.198617  7.869402          9.783577      7.200425
2   8.117312   7.074117   7.191429  8.612867          5.739793      6.907755
```

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 [9]: # 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 interquarti
            step = 1.5*(Q3-Q1)

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

        # OPTIONAL: Select the indices for data points you wish to remove
        outliers  = [66,75,128,154,86,95,183,325,338]

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

```
Data points considered outliers for the feature 'Fresh':

        Fresh       Milk    Grocery    Frozen  Detergents_Paper  Delicatessen
65   4.442651   9.950323  10.732651  3.583519         10.095388      7.260523
66   2.197225   7.335634   8.911530  5.164786          8.151333      3.295837
81   5.389072   9.163249   9.575192  5.645447          8.964184      5.049856
95   1.098612   7.979339   8.740657  6.086775          5.407172      6.563856
96   3.135494   7.869402   9.001839  4.976734          8.262043      5.379897
128  4.941642   9.087834   8.248791  4.955827          6.967909      1.098612
171  5.298317  10.160530   9.894245  6.478510          9.079434      8.740337
193  5.192957   8.156223   9.917982  6.865891          8.633731      6.501290
218  2.890372   8.923191   9.629380  7.158514          8.475746      8.759669
```

```
304  5.081404   8.917311  10.117510  6.424869          9.374413      7.787382
305  5.493061   9.468001   9.088399  6.683361          8.271037      5.351858
338  1.098612   5.808142   8.856661  9.655090          2.708050      6.309918
353  4.762174   8.742574   9.961898  5.429346          9.069007      7.013016
355  5.247024   6.588926   7.606885  5.501258          5.214936      4.844187
357  3.610918   7.150701  10.011086  4.919981          8.816853      4.700480
412  4.574711   8.190077   9.425452  4.584967          7.996317      4.127134


Data points considered outliers for the feature ’Milk’:

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


Data points considered outliers for the feature ’Grocery’:

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


Data points considered outliers for the feature ’Frozen’:

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


Data points considered outliers for the feature ’Detergents_Paper’:

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


Data points considered outliers for the feature ’Delicatessen’:

         Fresh       Milk    Grocery    Frozen  Detergents_Paper  \
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
```

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.

Answer: Instances 65 66, 75, 128 and 154 are considered outliers for more than one feature and therefore they should be removed as they will affect any algorithm applied to the data.Additionally we can consider removing outliers for each features individually.

For the fresh feature we can remove outlier 95 and 338 for beting too small.There are no big outliers that stand out.

For the feature milk we can remove outlier 86 for being too big.

For the grocery and detergents paper we can keep both the outliers.

For the frozen feature we can remove outlier 325 for being too large.

For the feature Delicatessen we can remove outlier 183 for being too big.

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

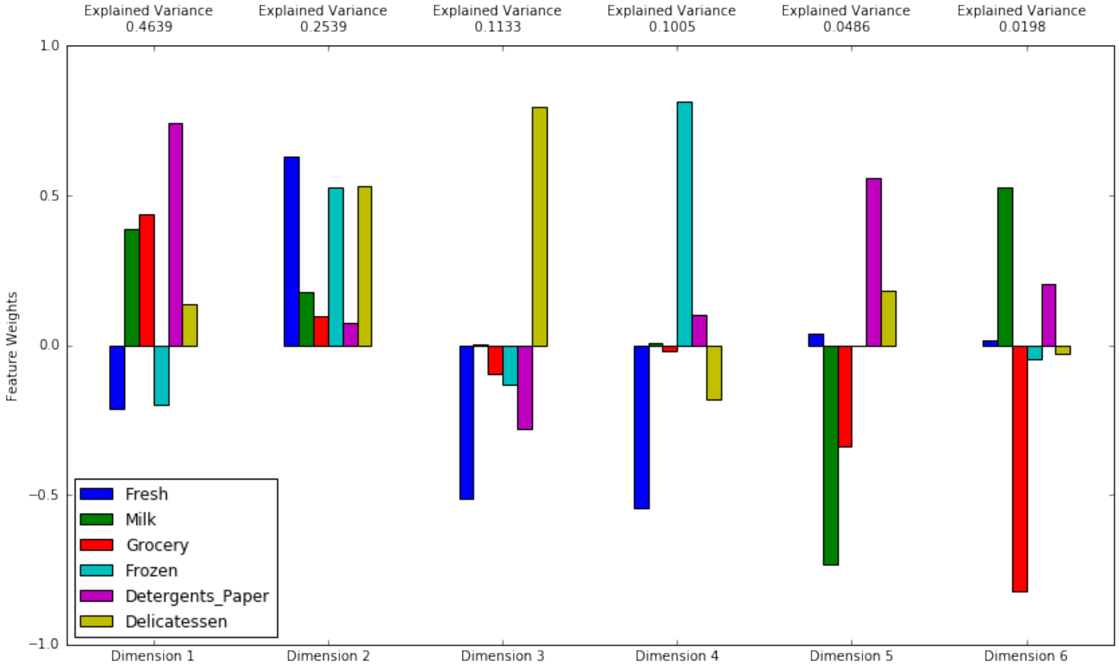
Now that the data has been scaled to a more normal distribution and has had any necessary outliers removed, we can now apply PCA to the `good_data` to discover which dimensions about the data best maximize the variance of features involved. In addition to finding these dimensions, PCA will also report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

```
In [10]: from sklearn.decomposition import PCA
         # TODO: Apply PCA by fitting the good data with the same number of dimensions as features
         pca = PCA(n_components=6)
         pca.fit(good_data)

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

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



1.6.2 Question 5

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

Answer: A total of 0.7178 variance is explained in the data by the first and second principal component whereas the total variance explained by the first four principal component is 0.9316 .

In terms of customer spending the first principal components measures customers spending on Detergents paper and the second component on Fresh, Frozen and Delicatessen at the same time. Since the third principal component includes a strong negative correlation, it measures customer spending on Delicatessen and at the same time lack of spending on Fresh. The same applies to the fourth component, which represents high spending on Frozen and low spending on Fresh.

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 [11]: # 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       2.1525       1.2444      -0.0433      -2.2852      -0.1482
1       3.8784       1.0568      -0.4046       0.9119      -0.4577
2      -1.8473       0.0098       0.7575       1.2940       0.6062

   Dimension 6
0       0.2891
1       0.2739
2       0.1906
```

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

```
In [12]: # 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 the sample log-data using the PCA fit above
         pca_samples = pca.transform(log_samples)

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

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 [13]: # 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       2.1525       1.2444
1       3.8784       1.0568
2      -1.8473       0.0098
```

1.7 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.

1.7.1 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?

Answer: Some of the advantages of K-Means are a) it is simple to understand and implement. b) it works well with relatively large datasets both on instances and features.

As per the scikit learn documentation, some of the advantages of GMM are: a) it is the fastest algorithm for learning mixture models b) as this algorithm maximizes only the likelihood, it will not bias the means towards zero, or bias the cluster sizes to have specific structures that might or might not apply.

By observing the distributions of the features we can see that they are somewhat normal, which means that each customer may belong to more than one cluster and therefore since GMM is a soft clustering algorithm, it could be more appropriate for this dataset.

1.7.2 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 [14]: # TODO: Apply your clustering algorithm of choice to the reduced data
         from sklearn import mixture
         clusterer = mixture.GMM(n_components=2)
         clusterer.fit(reduced_data)

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

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

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

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

```
0.421998672995
```

1.7.3 Question 7

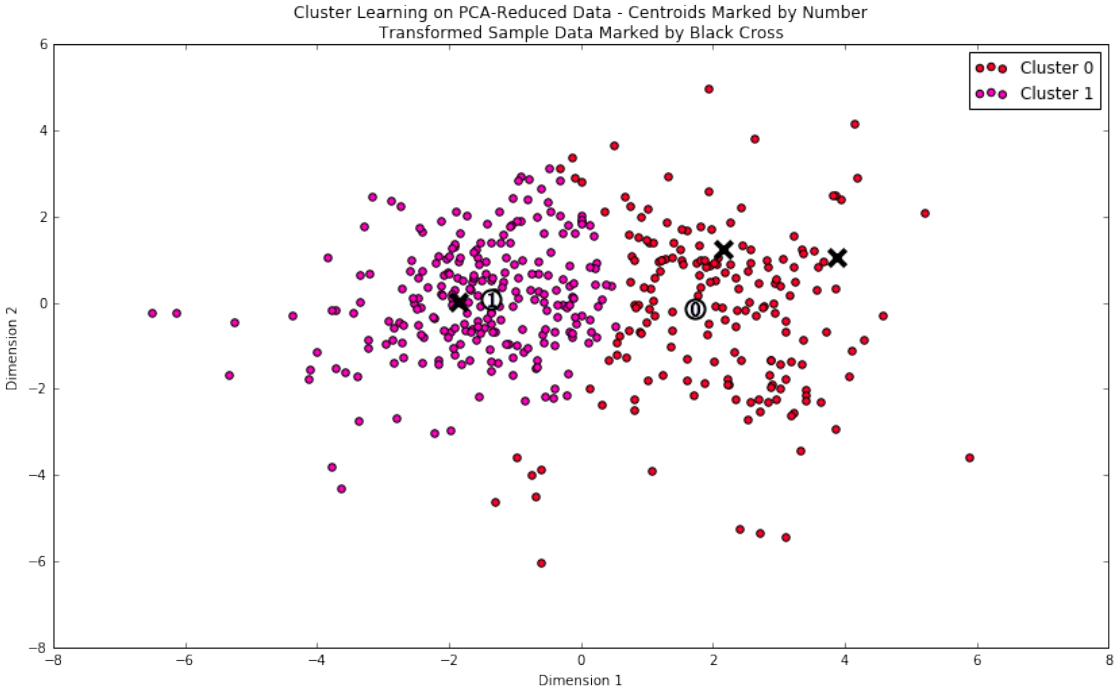
Report the silhouette score for several cluster numbers you tried. Of these, which number of clusters has the best silhouette score?

Answer: I tried finding silhouette score for 5 different number of clusters and i found out that as the number of cluster increases the score decreases. Therefore, the silhouette score is best for 2 clusters(0.42199).

1.7.4 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 [15]: # Display the results of the clustering from implementation
         rs.cluster_results(reduced_data, preds, centers, pca_samples)
```



1.7.5 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 [16]: # 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  4161.0  6463.0   9867.0   981.0            3268.0         949.0
Segment 1  9255.0  2018.0   2619.0  2039.0             334.0         697.0
```

1.7.6 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. What set of establishments could each of the customer segments represent?
Hint: A customer who is assigned to `Cluster X` should best identify with the establishments represented by the feature set of `Segment X`.

Answer: When comparing the total cost for each category for the representative data points of each customer segment with the initial statistical description of the dataset, we can see that a) for segment 0, fresh, frozen,detergents_paper and delicatessen features are below their corresponding means, and the 50% of the cases (2nd Quartile), while three of them lie near the 1st Quartile. This indicates that the segment lies close to the higher values of the instances.

b) for segment 1, all of the features are below their corresponding means and at the same time below the 2nd Quartile (50% of the cases) except for fresh feature. This indicates that the whole segment represents the lower values of the instances.

Therefore we can generally assume that on the first segment lie the highest spending customers, while on the second the lowest spending customers.

1.7.7 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 [17]: # Display the predictions
         for i, pred in enumerate(sample_preds):
             print "Sample point", i, "predicted to be in Cluster", pred
```

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

Answer: For sample point 0, all the features are above the mean and near the 3rd Quartile (75% of the cases) except for frozen feature so according to our previous theory this clearly belongs to the higher values of the instances and thus belongs to Cluster 0.

For sample point 1, milk, grocery and detergents are above the mean and way above the 3rd Quartile (75% of the cases) therefore it clearly indicates that this sample point belongs to the higher values of the instances and thus belongs to cluster 0.

For sample point 2, all of the features except frozen are less than the mean and lie near the 1st Quartile(25% of the cases) so this clearly indicate that our data point belongs to the lower values of the instances and thus belongs to cluster 1.

Our predictions for each example are indeed consistent with the above theory.

1.8 Conclusion

1.8.1 Question 10

Companies often run A/B tests when making small changes to their products or services to determine whether that change affects its customers positively or negatively. The wholesale distributor wants to consider changing its delivery service from 5 days a week to 3 days a week, but will only do so if it affects their customers positively. How would you use the customer segments you found above to perform an A/B Test for this change?
Hint: Can we assume the change affects all customers equally? How can we determine which group of customers it affects the most?

Answer: Using the structure of the data we can find clusters of customers with similar business needs and buying patterns. The distributor would want to ensure that their A/B test used sufficient customers in all clusters. We would want to randomly divide each grouping in two, such that A/B has the same number of customers in each of the two groupings.

That is, cluster zero would be evenly split into A/B and cluster one would be evenly split into A/B grouping. There would then be four groups, making identifying the affected customers easier. They can also be directly compared within their cluster types as we will treat all members of a cluster as equivalent.

Differences in the products being purchased by a customer could have a big effect on their preference for delivery service. Customers that have large fresh purchases would be more severely impacted by delivery changes than customers with large frozen purchases.

1.8.2 Question 11

Additional structure is derived from originally unlabelled data when using clustering techniques. Since each customer has a 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 has made estimates for each customer's annual spending of the six product categories. Knowing these estimates, the wholesale distributor wants to classify each new customer to one of the customer segments to determine the most appropriate delivery service.
Describe a supervised learning strategy you could use to make classification predictions for the ten new customers.
Hint: What other input feature could the supervised learner use besides the six product features to help make a prediction?

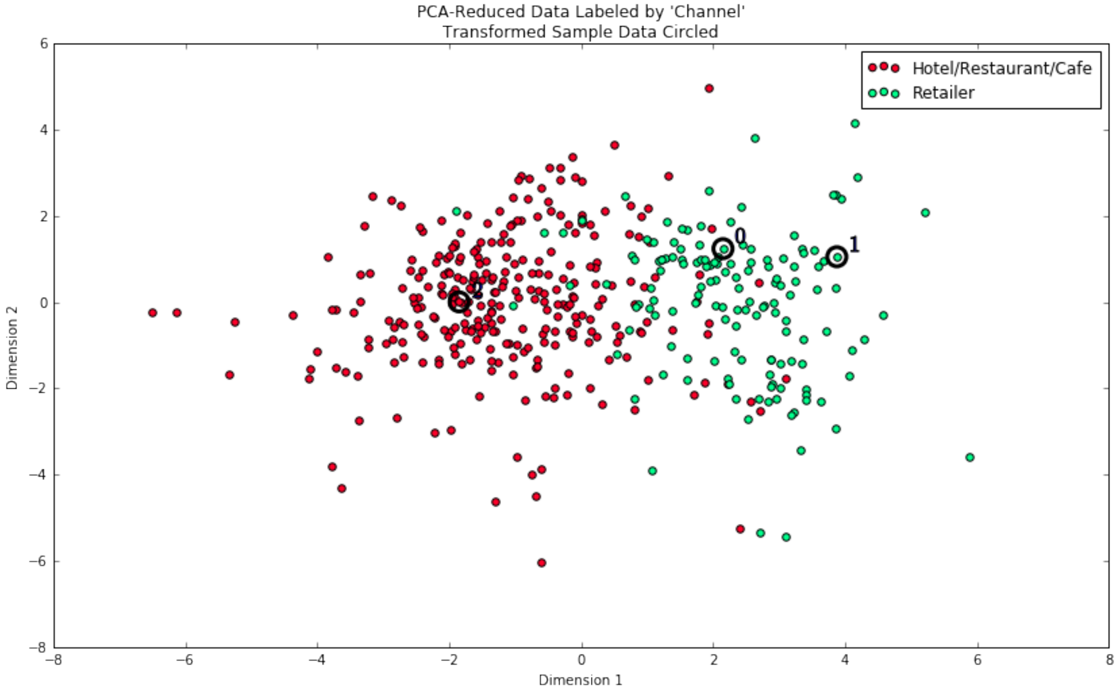
Answer: A supervised learning analysis could be assisted by a new feature depicting the cluster to which the new customer belongs. Therefore any predictive task could use this information in order to evaluate the other features in respect to the customer segment to which they belongs.

1.8.3 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 [18]: # Display the clustering results based on 'Channel' data
         rs.channel_results(reduced_data, outliers, pca_samples)
```



1.8.4 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: The visualization above depicts a relatively clear distinction between lower spending customers (Hotels Restaurants and Cafes) and higher spending customers (Retailers). While there are a few retailers falling in the first segment, a higher number of hotels restaurants and cafes fall in the second which highlights a portion where the two categories are mixed and not easily distinguished.

> **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.