report

September 30, 2018

1 Machine Learning Basics Nanodegree

2 Project : Creating Customer Segments

Welcome to the second project of the Machine Learning Basics 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.

Table of Contents

2.1 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("data.csv")
            data.drop(['Region', 'Channel'], axis = 1, inplace = True)
            print("Wholesale customers dataset has {} samples with {} features each.".format(*
        except:
            print("Dataset could not be loaded. Is the dataset missing?")
```

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

2.2 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
        data.describe()
```

Out[2]:

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

2.2.1 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 = [10, 150, 339]

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

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	3366	5403	12974	4400	5977	1744
1	16225	1825	1765	853	170	1067
2	2617	1188	5332	9584	573	1942

2.2.2 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: 7951.3
- Detergents_paper: 2881.4
- Delicatessen: 1524.8

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

Answer: - The first customer seems to be a **Grocery Shop or Small supermarket**. As we can see that it has good amounts of stock for almost all feautres.Fresh foods and Milk though are less than mean but its not that less to debate about. The most selling item are Grocery items and Detergent Papers as they well beyond the means of their respective features.

- The second customer likely seems to be a **Fresh Food Market / Retailer or Mass Producer.** It has large amounts of fresh foods which is well beyond the mean and remaining other features have a very less stock when compared with the means of their respective features.

- Third customer must be a **Frozen Superamrket or Mass producer of Frozen items. or Meat Producer/Shop** Very less stocks in fresh foods, milk, grocery and detergent pspers compared to their respective means. Way high up above the mean are frozen items and Delicatessen (meat) which clearly indicate that customer is a frozen seller specifically in meats.

2.2.3 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]: # TODO: Make a copy of the DataFrame, using the 'drop' function to drop the given feat
        new_data = data.drop(['Detergents_Paper'], axis = 1, inplace=False)


        # TODO: Split the data into training and testing sets(0.25) using the given feature as
        from sklearn.model_selection import train_test_split

        # Set a random state.
        X_train, X_test, y_train, y_test = train_test_split(new_data, data['Detergents_Paper']

        # TODO: Create a decision tree regressor and fit it to the training set
        from sklearn.tree import DecisionTreeRegressor

        regressor = DecisionTreeRegressor(random_state = 0)
        regressor.fit(X_train, y_train)

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

        print('Score is:', score)
```

```
Score is: 0.728655181254
```

2.2.4 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.

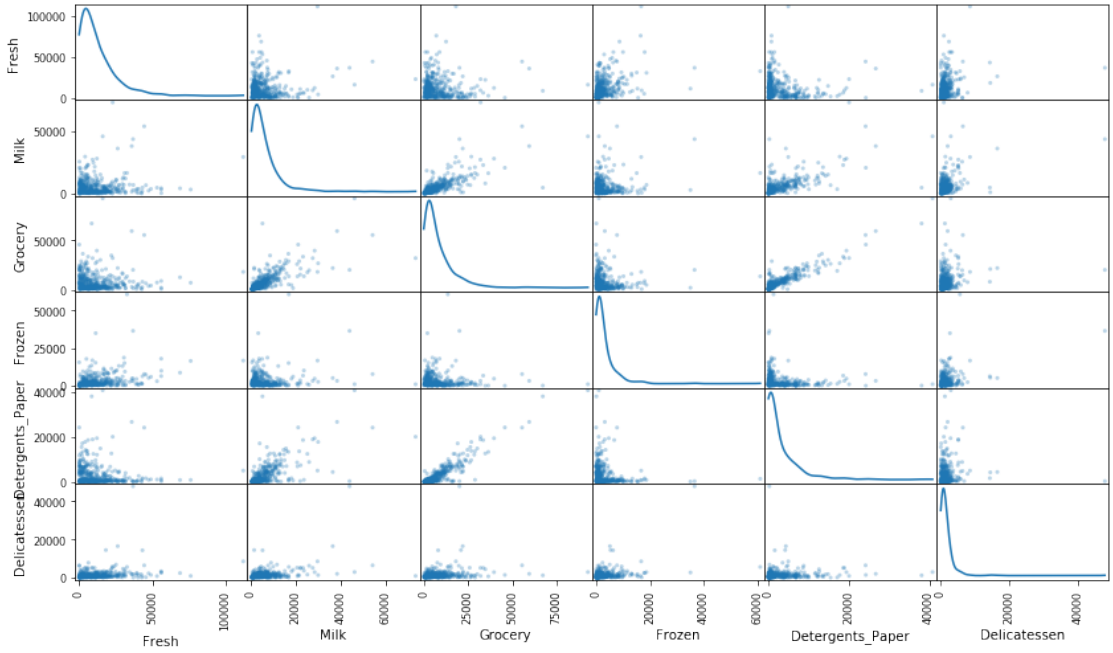
Answer: - First I looked the data given to us and thought which one seems the odd feature. I found `'Detergents_Paper'` is an odd feature since all the other feautres come in Food & Drinks category. - So I predicted its score to be 72.8% (approx 73%). I also than gave a shot to predict score of each feature but none were as high as `'Detergents_Paper'`. The lowest calculated score was of `'Delicatessen'` (meat) of -11. - The higher R^2 suggests that the feature `'Detergents Paper'` correlate well with other features, another way to think about it is that it does not provide much amount of Information gain, hence making the feature less necessary. On the other hand, feature such as `'Delicatessen'` with such a low R^2 score tells us that it doesn't correlate well with other feature, hence making the feature valuable and impactful.

2.2.5 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

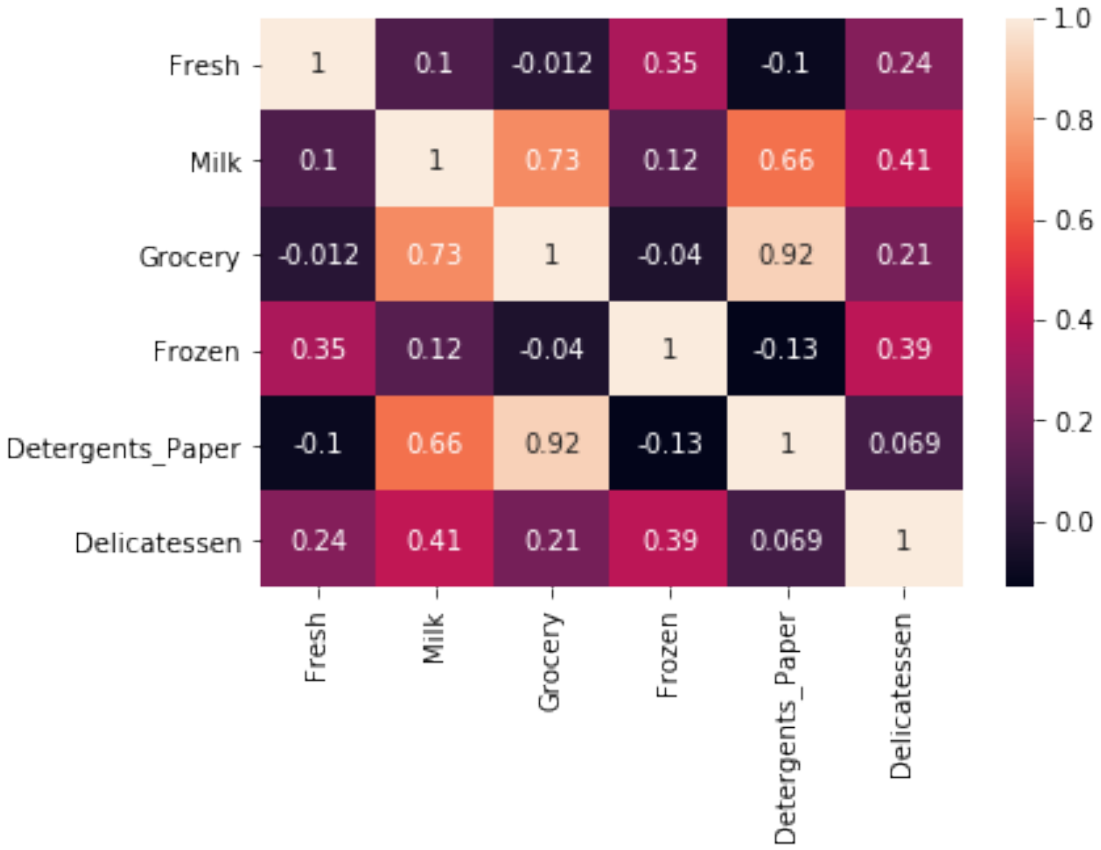
correlation between that feature and the others. Conversely, if you believe that feature is not relevant for identifying a specific customer, the scatter matrix might show a correlation between that feature and another feature in the data. Run the code block below to produce a scatter matrix.

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');
```



In [6]:
```
import seaborn as sns
sns.heatmap(data.corr(), annot=True)
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x121ff130>
```



2.2.6 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.

Answer: - `'Detergents Paper' and 'Grocery'`, as we can see in the above heatmap, has a strong linear correlation with the coefficient of 92%. - `'Detergents Paper' and 'Milk'`, with a coefficient of 66%, also having a strong linear correlation but not as strong as 'Detergent Papers'+'Grocery'. - `'Grocery' and 'Milk'`, another slighly strong linear correlation with a coefficient of 73%.

Hence by studying the above maps and its results we can conclude that 'Detergent Paper' does not give us much of imformation gain because the other two features, 'Milk' and 'Grocery', have a strong linear correlation between them.

Above mentioned feature tends to follow a `y = mx + b` relatioship.(Linear Relationship)

Most of the points for all the features are distributed near the origin in other words it's highly skewed towards the origin, hence it isn't normally distributed.

2.3 Data Preprocessing

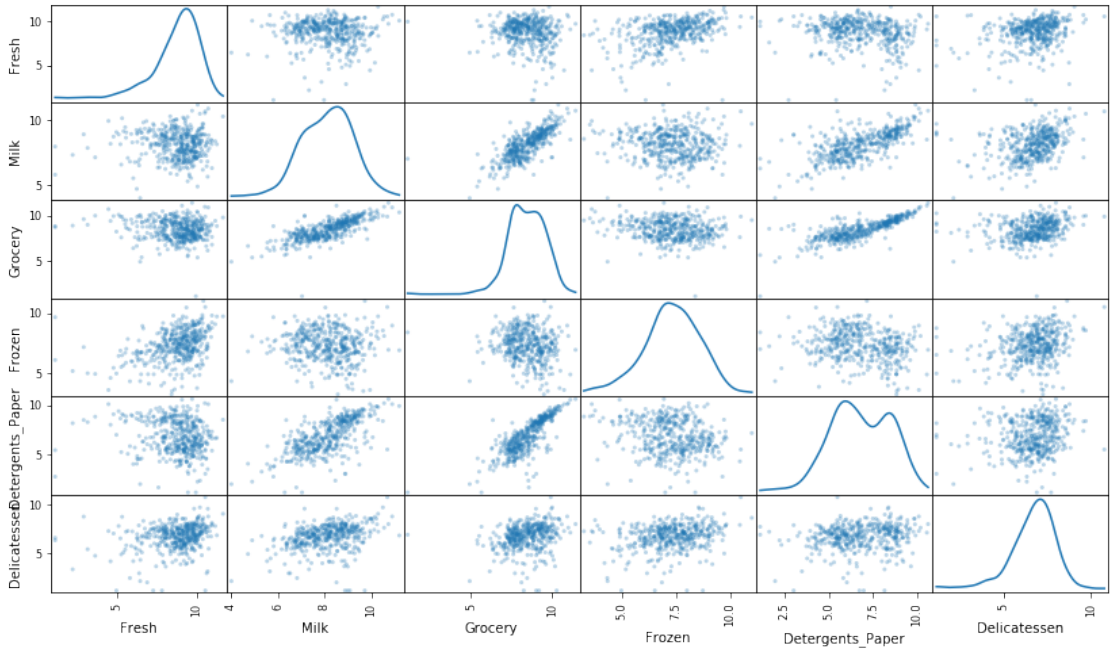
In this section, you will preprocess the data to create a better representation of customers by performing a scaling on the data and detecting (and optionally removing) outliers. Preprocessing data is often times a critical step in assuring that results you obtain from your analysis are significant and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

2.3.1 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');
```



2.3.2 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	8.121480	8.594710	9.470703	8.38936	8.695674	7.463937
1	9.694309	7.509335	7.475906	6.74876	5.135798	6.972606
2	7.869784	7.080026	8.581482	9.16785	6.350886	7.571474

2.3.3 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 in
            step = 1.5 * (Q3-Q1)
            print('Our IQR is', step)

            # 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 + st

        # OPTIONAL: Select the indices for data points you wish to remove
        outliers  = [65, 66, 75, 128, 154]

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

```
Our IQR is 2.53350786861
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

Our IQR is 2.31824827282
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

Our IQR is 2.3988562138
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

Our IQR is 2.34932750101
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

Our IQR is 4.08935876094
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


Our IQR is 2.24228065442
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
```

2.3.4 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: - **65:** Outliers in `Frozen` and `Fresh` features. - **66:** Outliers in `Delicatessen` and `Fresh` features. - **75:** Outliers in `Detergents Paper` and `Grocery` features. - **128:** Outliers in `Delicatessen` and `Fresh` features. - **154:** Outliers in `Delicatessen`, `Milk` and `Grocery` features.

These Five outliers above are seen for multiple features. In total there are 42 Outliers as we can see, leaving the five outliers mentioned above, rest of the ouliers appear in only one feature. This is all calculated by Tukey's Method for identifying outliers. I don't think we should remove all the outliers since the observations which have outliers in one feature might be having important information in the rest of the features. 42 Observations which are outliers are about 9% of the dataset and this could lead to loss in classifying customer behavior, hence giving misleading results. So I will only choose the five outliers that occur in mutiple features which I have already mentioned above.

2.4 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.

2.4.1 Implementation: PCA

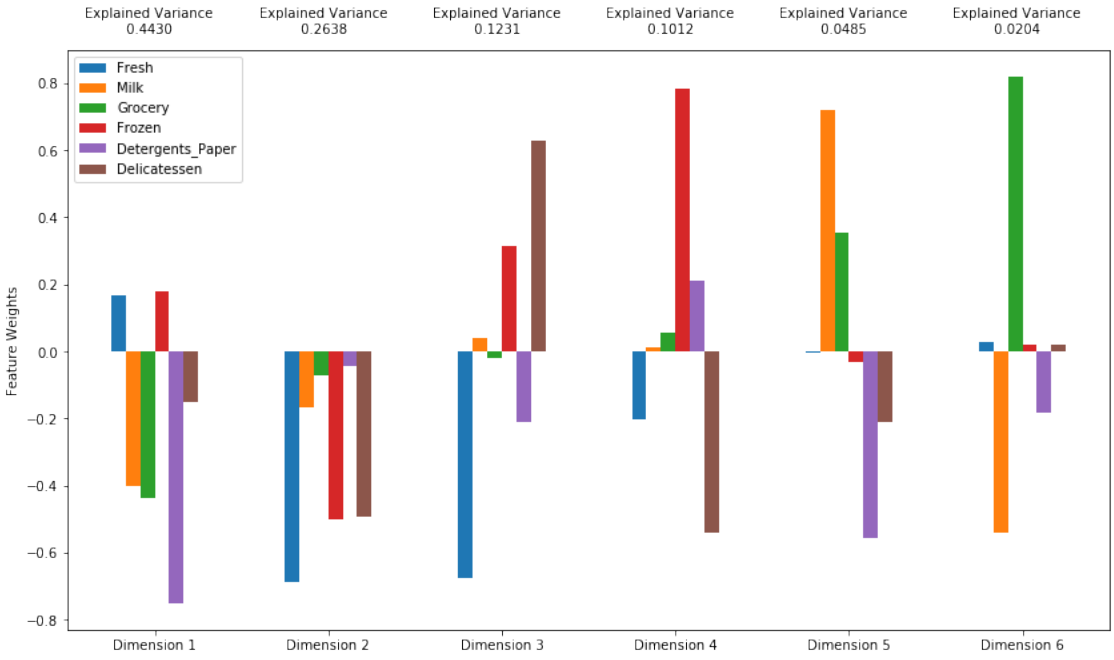
Now that the data has been scaled to a more normal distribution and has had any necessary outliers removed, we can now apply PCA to the `good_data` to discover which dimensions about the data best maximize the variance of features involved. In addition to finding these dimensions, PCA will also report 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 [10]: from sklearn.decomposition import PCA
         # TODO: Apply PCA by fitting the good data with the same number of dimensions as feat
         pca = PCA(n_components=6, random_state=0)
         pca.fit(good_data)

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

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

         # Calculating Total Variance
```

```
total_var = pca.explained_variance_ratio_

total_var2 = sum([total_var[i] for i in range(2)])
total_var4 = sum([total_var[i] for i in range(4)])

print('The total variance from first 2 components is ', total_var2)
print('The total variance from first 4 components is ', total_var4)
```

```
The total variance from first 2 components is  0.706817230807
The total variance from first 4 components is  0.931090109951
```



2.4.2 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: - The first 2 components explain 70.7% (approx) of variance in the data (as calc above). - And the first 4 components explain 93.2% (approx) of variance in the data.

- Thinking about in terms of spending, the 1st dimension shows us that customer with those values would spend much more on `Fresh` and equally on `Frozen`. A large negative emphasis is placed on `Detergents_Paper` with a smaller yet large negative emphasis on the `Grocery and Milk` This could represent **Supermarket Spending.**
- In 2nd dimension, we see all features has negative emphasis but a large negative emphasis is on`Fresh` followed by little less negative emphasis on`Frozen and Delicatessen`. This kind of spending is best represented by spending on **Fresh Frozen Meat or Fresh Vegetables**.
- In 3rd dimension, we see similar negative emphasis on `Fresh` followed by less negative emphasis om `Detergent Papers.Delicatessen` and `Frozen` items have a positive emphasis though. This could represent a **Restaurant.**
- In 4th dimention, we see negative emphasis on `Fresh` and `Delicatessen` , and very high positive emphasis on `Frozen`. This could represent bulk buyers of Frozen good such as Fish importers.

2.4.3 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      -2.0887      -0.7006       0.8537       1.0105      -0.5587
1       1.9406      -0.2418      -0.2884      -1.2041       0.0917
2       0.7513      -0.5551       1.7899       1.0547      -0.7029

   Dimension 6
0       0.2495
1      -0.1492
2       0.7766
```

2.4.4 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 [12]: # 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'])
```

2.4.5 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
```

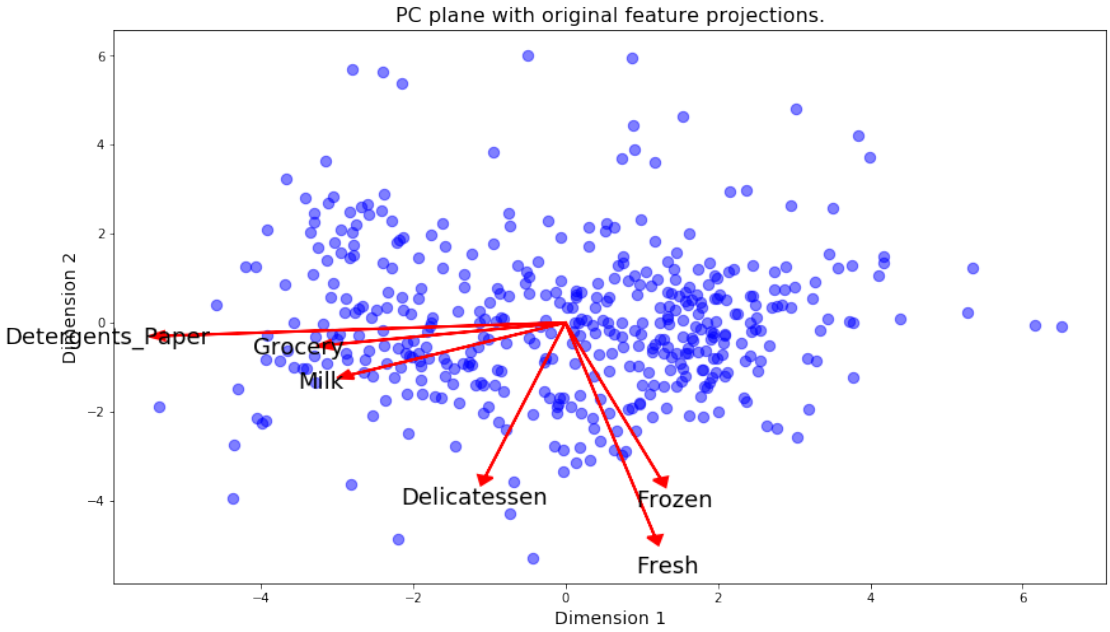
	Dimension 1	Dimension 2
0	-2.0887	-0.7006
1	1.9406	-0.2418
2	0.7513	-0.5551

2.5 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 [14]: # Create a biplot
         vs.biplot(good_data, reduced_data, pca)

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1063bc50>
```



2.5.1 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?

2.6 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.

2.6.1 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: - K-means is fast, simple and very easy to understand its theory of how it works. It also outperforms most of the other algorithms on the large dataset. It best perofrms on the data that are clearly defined and well separated. It has fewer parameters to be tuned and the algorithm strictly assigns to one cluster or the other (Hard Clustering). - Gaussian Mixture Model (GMM), here the data points aren't assigned strictly to one cluster, meaning if the one's which have lower probability could be assigned to multiple clusters at once (Soft Clustering) because it uses probabilities to predict event rather than rigidly assigning to only one cluster. This process is revised in each iterations making the algorithm more flexible when assigning points to a cluster hence performing well on an unclear defined dataset. - As we can see in above scatter plot, the data is almost uniform and lots of data points don't clearly belong to any one cluster. Using **GMM** would be logical to adopt here and would produce best outcome for the given problem.

2.6.2 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 [15]: from sklearn.mixture import GaussianMixture as GM
         from sklearn.metrics import silhouette_score as SS

         # TODO: Apply your clustering algorithm of choice to the reduced data
         clusterer = GM(n_components=2).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 = SS(reduced_data, preds)

         print(score)
```

```
0.421916846463
```

2.6.3 Question 7

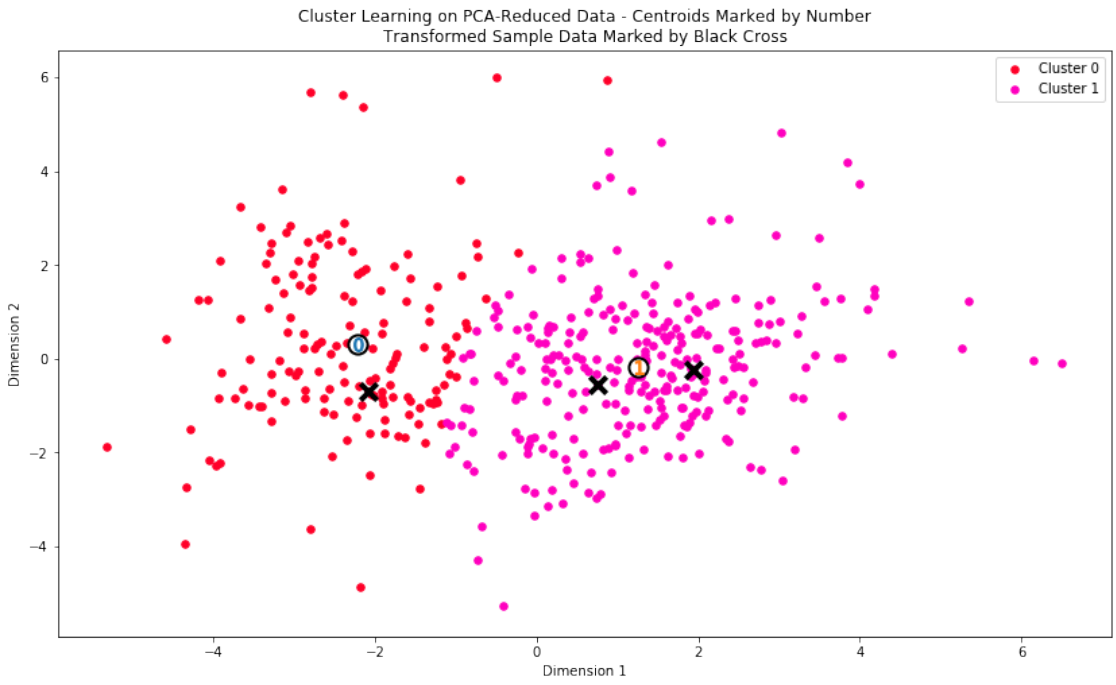
- Report the silhouette score for several cluster numbers you tried.
- Of these, which number of clusters has the best silhouette score?

Answer: By doing above calcultions we got the following silhouette score on different clusters.. - **2 Clusters - Silhouette Score - 0.422** - **3 Clusters - Silhouette Score - 0.414** - **4 Clusters - Silhouette Score - 0.346** - **5 Clusters - Silhouette Score - 0.264**

2.6.4 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 [16]:
```
# Display the results of the clustering from implementation
vs.cluster_results(reduced_data, preds, centers, pca_samples)
```



2.6.5 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 [17]: # 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	3552.0	7837.0	12219.0	870.0	4696.0	962.0
Segment 1	8953.0	2114.0	2765.0	2075.0	353.0	732.0

2.6.6 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.

Answer: - A customer who is assigned to `Cluster 0` seems most likely to represent a **Market Store,** as we see based on their consistent purchase cost in `Milk, Grocery, Frozen` and higher than average in `Milk` category. - A customer who is assigned to `Cluster 1` seems most likely to represent a **Restaurant,** as we see based on their consistent purchase cost in `Milk, Detergent Paper` and higher than average in `Grocery` category.

2.6.7 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 [18]: # Display the predictions
         for i, pred in enumerate(sample_preds):
             print("Sample point", i, "predicted to be in Cluster", pred)
```

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

Answer: Sample point 0 is best represented by a **Market Store** and Sample points 1 and 2 are best represented as some type of **Restaurant**. And its consistent with the predictions obtained from the clusters.

2.7 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.

2.7.1 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: - By Observing these two customer segments, i.e `Market Store (Cluster 0) and Restaurant (Cluster 1)`, the wholesaler could draw hypothesis that each will react different to reduction in number of deliveries a week. - Restaurants would have a negative affect as they're more concerned with `freshness` of an item to deliver to their guests or clients. Decreasing number of deliveries a week means buying more items and having more space to store (if they have) them and storing for longer time could spoil some of their goods. It would also have a negative affect in terms of quality of the food that they want to provide. - And talking about the `Market Store`, they on the other hand will react positively to the decrease in delivery a week since they have more inventory space to store the goods they want to sell to their customers. And they're also not concerned much with the `freshness` of the product. - All this said, company could run A/B

test and generalize and they can evaluate feedback seperately by picking a subset customers from each cluster. Then we can establish whether changing the delivery frequency is critical to each segment or the customers are happy with the change.

2.7.2 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?

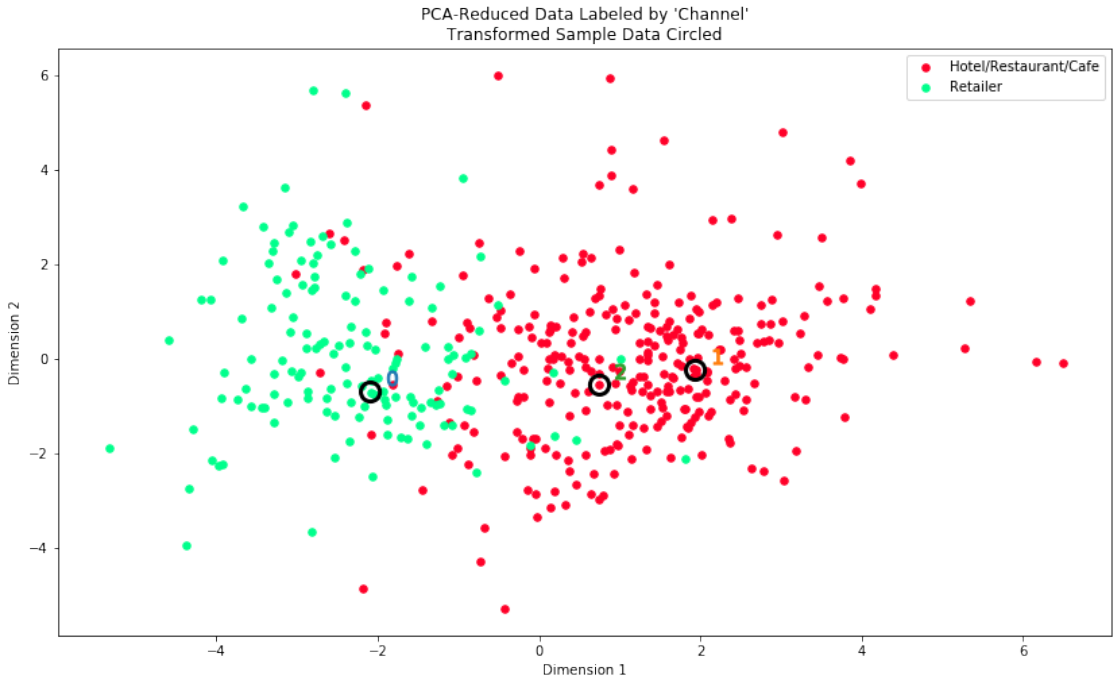
Answer: The wholesale distributor could train a supervised model, for example Decision Tree Classifier, input being a dataset of customer product spending and target variable being customer segments (as obtained by GMM). After training the classifier, it can be used to predict the customer segment for new customers which would then determine which delivery (3 or 5 times a week) would be appropriate.

2.7.3 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 [19]: # Display the clustering results based on 'Channel' data
         vs.channel_results(reduced_data, outliers, pca_samples)
```



2.7.4 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 GMM algorithm and the number of clusters chosen are highly comparable to the underlying distribution shown in the plot above. The customer segments as classified here closely match those I previously defined in Question 8 (i.e Green points: **Market Store**, Red Points: **Restaurants**.)