Machine Learning Engineer Nanodegree

Unsupervised Learning

Project 3: Creating Customer Segments ¶ (//htmlpreview.github.io/?https://github.com/robertodias/machine_learning/blob/master/py 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.

Getting Started

In this project, you will analyze a dataset containing data on various customers' annual spending amounts (reported in *monetary units*) of diverse product categories for internal structure. One goal of this project is to best describe the variation in the different types of customers that a wholesale distributor interacts with. Doing so would equip the distributor with insight into how to best structure their delivery service to meet the needs of each customer.

The dataset for this project can be found on the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Wholesale+customers). For the purposes of this project, the features `'Channel'` and `'Region'` will be excluded in the analysis — with focus instead on the six product categories recorded for customers.

Run the code block below to load the wholesale customers dataset, along with a few of the necessary Python libraries required for this project. You will know the dataset loaded successfully if the size of the dataset is reported.

```
In [75]: # 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.shape)
except:
    print "Dataset could not be loaded. Is the dataset missing?"
```

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

Data Exploration

In this section, you will begin exploring the data through visualizations and code to understand how each feature is related to the others. You will observe a statistical description of the dataset, consider the relevance of each feature, and select a few sample data points from the dataset which you will track through the course of this project.

Run the code block below to observe a statistical description of the dataset. Note that the dataset is composed of six important product categories: **'Fresh'**, **'Milk'**, **'Grocery'**, **'Frozen'**, **'Detergents_Paper'**, and **'Delicatessen'**. Consider what each category represents in terms of products you could purchase.

In [2]:

```
# Display a description of the dataset
display(data.describe())
```

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

Implementation: Selecting Samples

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

```
In [48]: # TODO: Select three indices of your choice you wish to sample from th
         e dataset
         indices = [0, 222, 342]

         # 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	12669	9656	7561	214	2674	1338
1	5041	1115	2856	7496	256	375
2	255	5758	5923	349	4595	1328

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:

- **Customer 0**: This customer spents with Fresh and Milk products exceeds the third quartile of all other customers. Grocery is also considerable, but we can't ignore it also uses Detergent a lot as Delicatessen. I this this establishment represents a Restaurant, probably an **Italian Restaurant** with lots of pasta, tomatoes, cheese and wine.

- **Customer 222**: This customer spents with Frozen products represents way more than the third quartile of all other customers, we can also identify a very small amount of Detergents. I think this customer is a very busy **Family**. That eats Frozen food most of the time.

- **Customer 342**: This customer based on its cost of Delicatessen, I think it is a **small cafe**. Where you have Milk items, some grecery and delicatesen. I don't think it is a family because of the high cost of the Detergents during the year. It is certainly a business.

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 [175]:

```
# TODO: Make a copy of the DataFrame, using the 'drop' function to drop the given feature
new_data = data.copy()

# TODO: Split the data into training and testing sets using the given feature as the target
y = new_data['Fresh'].copy()
X = new_data.drop('Fresh', 1)
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=72)

# TODO: Create a decision tree regressor and fit it to the training set
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=72)
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
```

```
-1.21236771823
```

Question 2

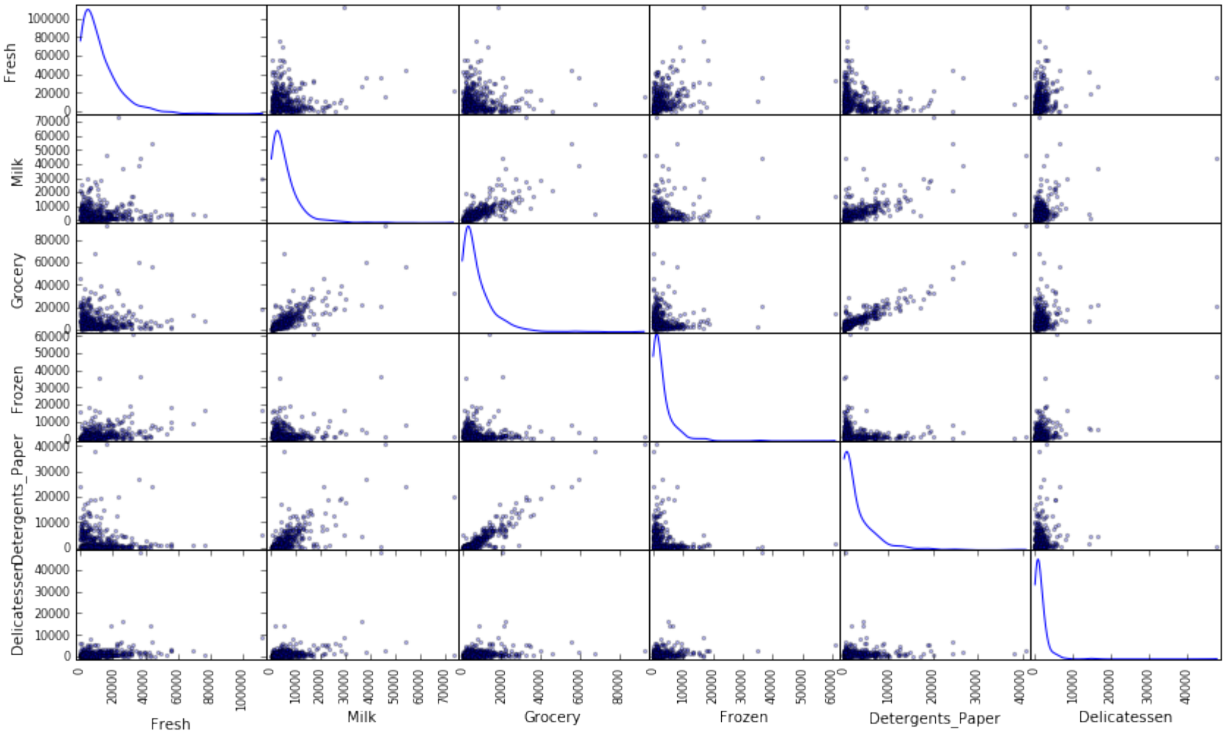
Which feature did you attempt to predict? What was the reported prediction score? Is this feature is necessary for identifying customers' spending habits?
Hint: The coefficient of determination, `R^2`, is scored between 0 and 1, with 1 being a perfect fit. A negative `R^2` implies the model fails to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

Answer: The feature I was attempting to predict was the **Fresh**. The prediction score I've found, using the Decision Tree Regressor -1.21236771823l. As my R^2 was negative I am assuming that the model failed to fit the data, that way, this feature looked to not have a strong relation with any of the other features. The final conclusion, about the role of the feature **Fresh** in model, is that it have a high Relevance to model as it was not possible to predict its value only by using the information shared with the other features from the model.

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



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?

Answer:

Most of the points lie close to zero, showing that most of the features are independent. But there is pair the show a good correlation, the pair **Detergents_Paper x Grocery**. We can also see some correlation, not strong as the other but considerable between the pair **Detergents_Paper x Milk** and between **Milk x Grocery** After my analysis I assumed that **Fresh** is an important feature, but I would probably remove **Detergents_Paper and Milk** from the data as **Grocery** will be there to representing them.

Data Preprocessing

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

Implementation: Feature Scaling

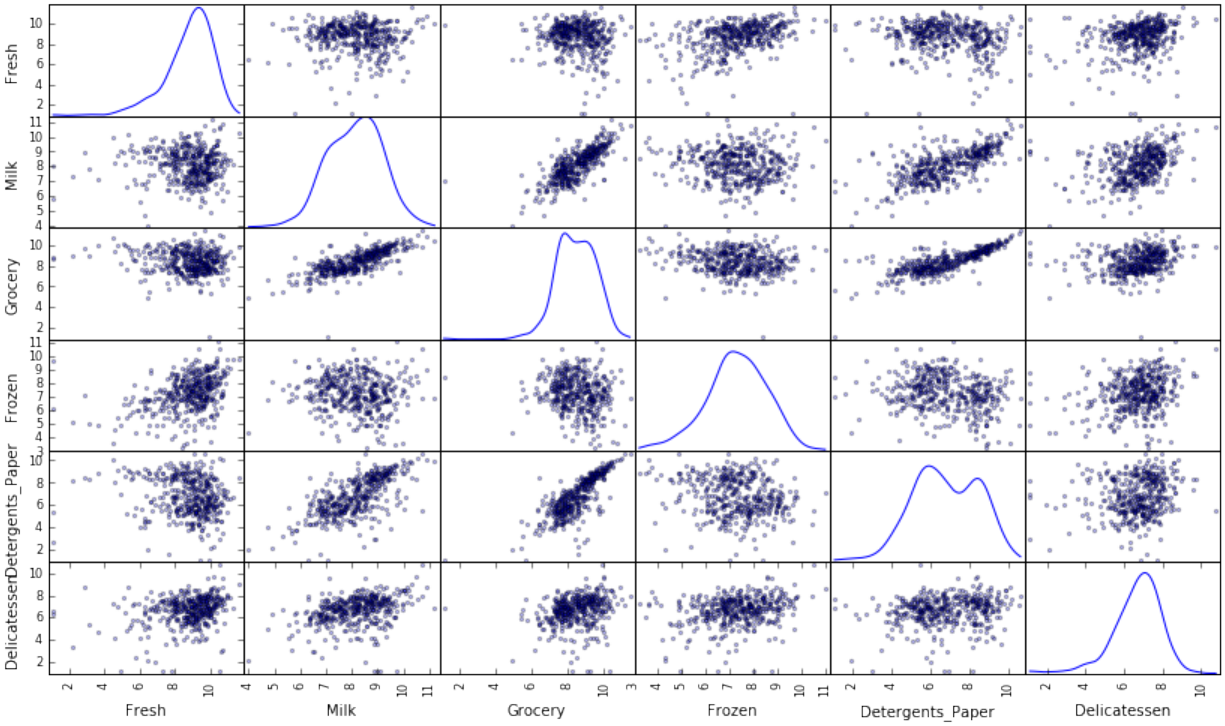
If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate (http://econbrowser.com/archives/2014/02/use-of-logarithms-in-economics) to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test (http://scipy.github.io/devdocs/generated/scipy.stats.boxcox.html), which calculates the best power transformation of the data that reduces skewness. A simpler approach which can work in most cases would be applying the natural logarithm.

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

- Assign a copy of the data to `log_data` after applying 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 [108]:

```
# 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');
```



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 [109]:

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	8.525360	7.016610	7.957177	8.922125	5.545177	5.926926
2	5.541264	8.658345	8.686598	5.855072	8.432724	7.191429

Implementation: Outlier Detection

Detecting outliers in the data is extremely important in the data preprocessing step of any analysis. The presence of outliers can often skew results which take into consideration these data points. There are many "rules of thumb" for what constitutes an outlier in a dataset. Here, we will use Tukey's Method for identfying outliers (http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/): An *outlier step* is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal.

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

- Assign the value of the 25th percentile for the given feature to `Q1`. Use `np.percentile` for this.
- Assign the value of the 75th percentile for the given feature to `Q3`. Again, use `np.percentile`.
- Assign the calculation of an outlier step for the given feature to `step`.
- Optionally remove data points from the dataset by adding indices to the `outliers` list.

NOTE: If you choose to remove any outliers, ensure that the sample data does not contain any of these points!
Once you have performed this implementation, the dataset will be stored in the variable `good_data`.

In [221]:

```
#Inicialize uma lista vazia de indices_outliers
outliers_dict = dict()

# For each feature find the data points with extreme high or low values
for feature in log_data.keys():

    # TODO: Calculate Q1 (25th percentile of the data) for the given feature
    Q1 = np.percentile(log_data[feature], 25)

    # TODO: Calculate Q3 (75th percentile of the data) for the given feature
    Q3 = np.percentile(log_data[feature], 75)

    # TODO: Use the interquartile range to calculate an outlier step (1.5 times the interquartile range)
    step = 1.5 * (Q3 - Q1)

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

    #Iterate over outliers from the current feature and add them to the outliers_dict
    for key in outlier.index.values:
        if key in outliers_dict:
            outliers_dict[key] += 1
        else:
            outliers_dict[key] = 1

# OPTIONAL: Select the indices for data points you wish to remove
outliers = list()
for key, value in outliers_dict.items():
    if value > 1:
        outliers.append(key)

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

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
65	4.442651	9.950323	10.732651	3.583519	10.095388	7.260523
66	2.197225	7.335634	8.911530	5.164786	8.151333	3.295837
81	5.389072	9.163249	9.575192	5.645447	8.964184	5.049856
95	1.098612	7.979339	8.740657	6.086775	5.407172	6.563856
96	3.135494	7.869402	9.001839	4.976734	8.262043	5.379897
128	4.941642	9.087834	8.248791	4.955827	6.967909	1.098612
171	5.298317	10.160530	9.894245	6.478510	9.079434	8.740337
193	5.192957	8.156223	9.917982	6.865891	8.633731	6.501290
218	2.890372	8.923191	9.629380	7.158514	8.475746	8.759669
304	5.081404	8.917311	10.117510	6.424869	9.374413	7.787382
305	5.493061	9.468001	9.088399	6.683361	8.271037	5.351858
338	1.098612	5.808142	8.856661	9.655090	2.708050	6.309918
353	4.762174	8.742574	9.961898	5.429346	9.069007	7.013016
355	5.247024	6.588926	7.606885	5.501258	5.214936	4.844187
357	3.610918	7.150701	10.011086	4.919981	8.816853	4.700480
412	4.574711	8.190077	9.425452	4.584967	7.996317	4.127134

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

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

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

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

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

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

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

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

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
66	2.197225	7.335634	8.911530	5.164786	8.151333	3.295837
109	7.248504	9.724899	10.274568	6.511745	6.728629	1.098612
128	4.941642	9.087834	8.248791	4.955827	6.967909	1.098612
137	8.034955	8.997147	9.021840	6.493754	6.580639	3.583519
142	10.519646	8.875147	9.018332	8.004700	2.995732	1.098612
154	6.432940	4.007333	4.919981	4.317488	1.945910	2.079442
183	10.514529	10.690808	9.911952	10.505999	5.476464	10.777768
184	5.789960	6.822197	8.457443	4.304065	5.811141	2.397895
187	7.798933	8.987447	9.192075	8.743372	8.148735	1.098612
203	6.368187	6.529419	7.703459	6.150603	6.860664	2.890372
233	6.871091	8.513988	8.106515	6.842683	6.013715	1.945910
285	10.602965	6.461468	8.188689	6.948897	6.077642	2.890372
289	10.663966	5.655992	6.154858	7.235619	3.465736	3.091042
343	7.431892	8.848509	10.177932	7.283448	9.646593	3.610918

Question 4

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

Answer:

I've found some data points considered **outliers** in our dataset. By being an outlier, before considering the possible elimination of these points from the data, I've reviewed the data distrubtion seeking the understanding of why they appeared in the dataset. After assuming they were really outliers, and confirming that a **few of them were present in more than one feature** I've considered those as bad data points for our data analysis and I decided to **remove** them from the dateaset.

Feature Transformation

In this section you will use principal component analysis (PCA) to draw conclusions about the underlying structure of the wholesale customer data. Since using PCA on a dataset calculates the dimensions which best maximize variance, we will find which compound combinations of features best describe customers.

Implementation: PCA

Now that the data has been scaled to a more normal distribution and has had any necessary outliers removed, we can now apply PCA to the `good_data` to discover which dimensions about the data best maximize the variance of features involved. In addition to finding these dimensions, PCA will also report the *explained variance ratio* of each dimension — how much variance within the data is explained by that dimension alone. Note that a component (dimension) from PCA can be considered a new "feature" of the space, however it is a composition of the original features present in the data.

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

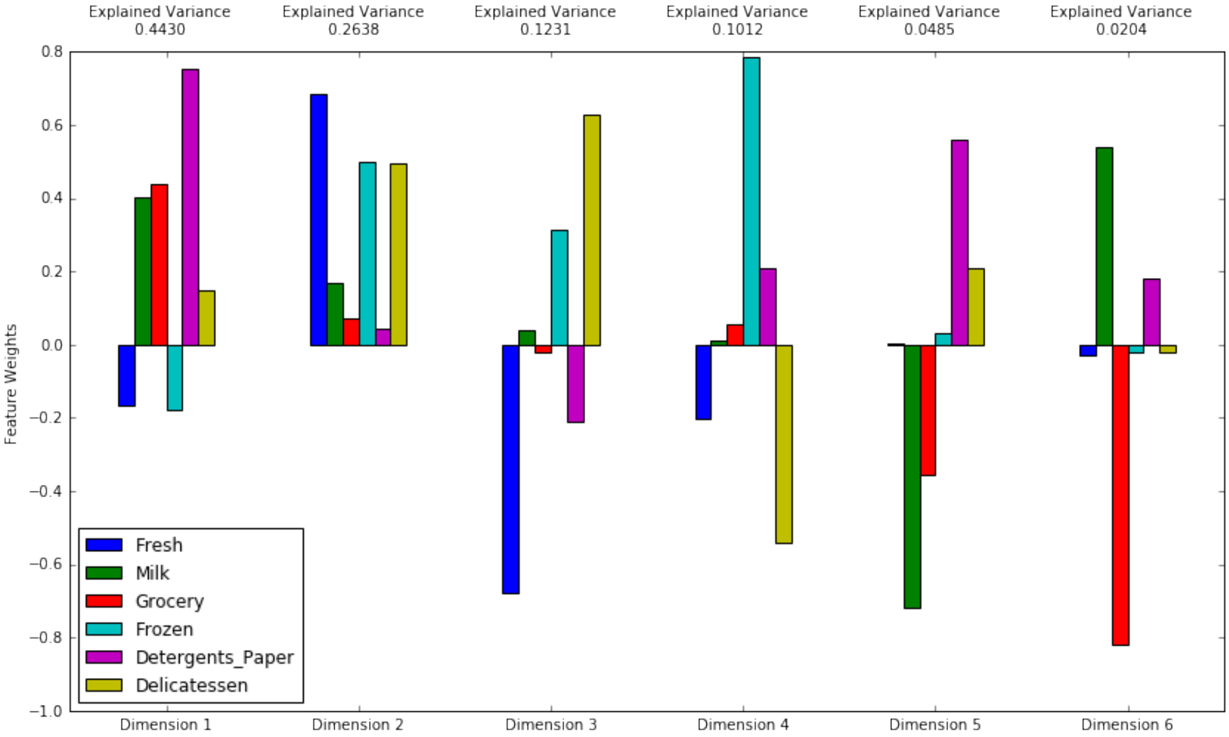
- Import `sklearn.decomposition.PCA` and assign the results of fitting PCA in six dimensions with `good_data` to `pca`.
- Apply a PCA transformation of the sample log-data `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

In [223]:

```
from sklearn.decomposition import PCA

# TODO: Apply PCA by fitting the good data with the same number of dim
ensions as features
pca = PCA(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)

np.cumsum(pca.explained_variance_ratio_)
```

Out[223]:

```
array([ 0.44302505,  0.70681723,  0.82988103,  0.93109011,  0.979592
07,  1.        ])
```



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:

The **variance** for the first and second components is equals to 0.7068 or around **70%**. If we count with the first four principal components the **variance** becomes 0.9311 or around **90%**.

Considering that an increase in a specific dimension corresponds with an increase of the positive-weighted features and a decrease of the negative-weighted features the fisrt four dimensions can be analyzed as:

Dimension 1 Seems to be an area primarly focused in Detergents_Paper (weight close to 0.8) follwed by Grocery and Milk (around 0.4). There is negative weights on Fresh and Frozen. So I think it describes a **Retailer**.

Dimension 2 Seems to be an area well distributed in consume, but we can notice Fresh and top weight (around 0.7) follwed by Frozen and Delicatessen equaly important (around 0.5). Because of the occurence of all features at this dimension and the preseted weights I think it describes a **Market**.

Dimension 3 Seems to be an area extremely focused in Delicatessen (around 0.7) and Frozen (around 0.4), it is also important to notice that there is a significant negative weight on Fresh (around -0.8). Our Customer 342 (from question 1) fits here. This dimension describes something like a **Small Cofe** or an **Ice Cream Shop**.

Dimension 4 Seems to be an area where Frozen goods is very important (around 0.8) the other positive feature is Detergents_Paper (but is not that significant). Our Customer 222 (from question 1) fits here our **very busy modern family**.

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 [222]:

```
# Display sample log-data after having a PCA transformation applied
display(pd.DataFrame(np.round(pca_samples, 4), columns = pca_results.index.values))
```

	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5	Dimension 6
0	1.7557	0.0102	-1.5998	-1.0694	-0.1526	0.3871
1	-1.9731	-0.0694	0.9407	1.1893	-0.0614	-0.4796
2	2.4647	-2.6924	1.1413	-0.6281	0.5964	0.5811

Out[222]:

```
array([ 0.47635078,  0.72903135,  0.84662795,  0.93495637,  0.97489841,  1.        ])
```

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 [234]:

```
from sklearn.decomposition import PCA

# TODO: Apply PCA by fitting the good data with only two dimensions
pca = PCA(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'])
```

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 [122]:

```
# 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	1.7580	-0.0097
1	-1.9682	-0.0198
2	2.4161	-2.5284

Clustering

In this section, you will choose to use either a K-Means clustering algorithm or a Gaussian Mixture Model clustering algorithm to identify the various customer segments hidden in the data. You will then recover specific data points from the clusters to understand their significance by transforming them back into their original dimension and scale.

Question 6

What are the advantages to using a K-Means clustering algorithm? What are the advantages to using a Gaussian Mixture Model clustering algorithm? Given your observations about the wholesale customer data so far, which of the two algorithms will you use and why?

Answer:

K-Means Clustering Algorithm Simpliest, easy to implement, the number of clusters is defined by the user and most of time it's fast and efficient than the other clustering algorithms (considering you have a reduced number of Ks).

Gaussian Mixture Model Clustering Algorithm better to find hidden parameters in data, more flexible to identify clusters by allowing a point to be classified by more than one cluster (mixture). As the K-means, this alorigthm have the number of clusters defined by the user, what do not happen in other algorithms like the hierachical clustering.

I am still not confortable with the classification of the customers, so I think Gaussian Mixture Model will help me to see if I have any hidden cluster or, at least, confirm how the data its being classified.

Implementation: Creating Clusters

Depending on the problem, the number of clusters that you expect to be in the data may already be known. When the number of clusters is not known *a priori*, there is no guarantee that a given number of clusters best segments the data, since it is unclear what structure exists in the data — if any. However, we can quantify the "goodness" of a clustering by calculating each data point's *silhouette coefficient*. The silhouette coefficient (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html) for a data point measures how similar it is to its assigned cluster from -1 (dissimilar) to 1 (similar). Calculating the *mean* silhouette coefficient provides for a simple scoring method of a given clustering.

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

- Fit a clustering algorithm to the `reduced_data` and assign it to `clusterer`.
- Predict the cluster for each data point in `reduced_data` using `clusterer.predict` and assign them to `preds`.
- Find the cluster centers using the algorithm's respective attribute and assign them to `centers`.
- Predict the cluster for each sample data point in `pca_samples` and assign them `sample_preds`.
- Import sklearn.metrics.silhouette_score and calculate the silhouette score of `reduced_data` against `preds`.
    - Assign the silhouette score to `score` and print the result.

In [243]:

```
# TODO: Apply your clustering algorithm of choice to the reduced data
from sklearn.mixture import GMM
from sklearn import metrics

def silhouette_by_cluster(cluster):
    clusterer = GMM(cluster).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 = metrics.silhouette_score(reduced_data, preds)
    return score

for cluster in range(2, 9):
    print("{} Clusters: {}".format(cluster, silhouette_by_cluster(cluster)))
```

```
2 Clusters: 0.411818864386
3 Clusters: 0.372313708076
4 Clusters: 0.337969395886
5 Clusters: 0.280672691612
6 Clusters: 0.278739044291
7 Clusters: 0.321289330242
8 Clusters: 0.304106857662
```

Question 7

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

Answer:

2 Clusters: 0.411818864386

3 Clusters: 0.372313708076

4 Clusters: 0.337969395886

5 Clusters: 0.280672691612

6 Clusters: 0.278739044291

7 Clusters: 0.321289330242

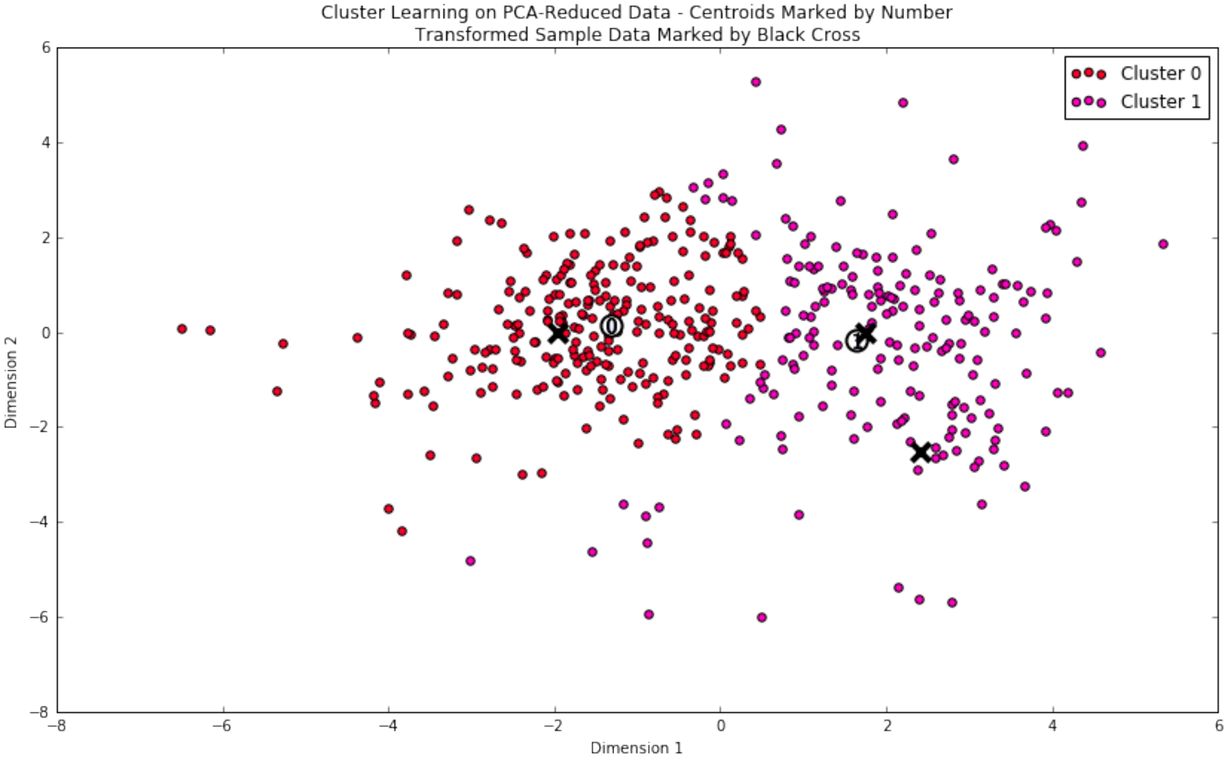
8 Clusters: 0.304106857662

The best silhouette Score was found using **2 Clusters**.

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 [162]:
```
# Display the results of the clustering from implementation
rs.cluster_results(reduced_data, preds, centers, pca_samples)
```



Implementation: Data Recovery

Each cluster present in the visualization above has a central point. These centers (or means) are not specifically data points from the data, but rather the *averages* of all the data points predicted in the respective clusters. For the problem of creating customer segments, a cluster's center point corresponds to *the average customer of that segment*. Since the data is currently reduced in dimension and scaled by a logarithm, we can recover the representative customer spending from these data points by applying the inverse transformations.

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

- Apply the inverse transform to `centers` using `pca.inverse_transform` and assign the new centers to `log_centers`.
- Apply the inverse function of `np.log` to `log_centers` using `np.exp` and assign the true centers to `true_centers`.

```
In [163]: # 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.key
          s())
          true_centers.index = segments
          display(true_centers)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Segment 0	8812.0	2052.0	2689.0	2058.0	337.0	712.0
Segment 1	4316.0	6347.0	9555.0	1036.0	3046.0	945.0

Question 8

Consider the total purchase cost of each product category for the representative data points above, and reference the statistical description of the dataset at the beginning of this project. *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:

A customer who is assigned to the **Cluster 0**, by considering segment 0, is customer that represents a **Restaurant** as it have high values associeted with Grocery and is well distributed among Fresh, Milk and Detergents.

A customer who is assinged to the **Cluster 1**, by considering segment 1, is a customer highly associated with Fresh food, and have a low cost of Detergents_paper. Looks to be a **Fresh Market**.

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 [164]:

```
# 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 1
```

Answer:

Sample 0: Was assigned to Cluster 0, Restaurant, I've originally assigned this customer to an Italian Restaurant. I belive this classification was done right, pretty consistent.

Sample 1: Was assigned to Cluster 1, Fresh Market, I've originally assigned this customer to a very busy familiy, but now it looks to be Fresh Market. It's center showed more Frozen costs associated than with the other segment, but not as much as the sample point presented. This lead me to consider the sample was like Busy Families living by Frozen foods. I think my observation was wrong (probably an outlier).

Sample 2: Was assigned to Cluster 0, Restaurant, I've originally assigned this customer to a Small Cafe. I think it was pretty close, I assume it was consistent enough.

Conclusion

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

Question 10

Companies will often run A/B tests (https://en.wikipedia.org/wiki/A/B_testing) when making small changes to their products or services to determine whether making that change will affect its customers positively or negatively. The wholesale distributor is considering changing its delivery service from currently 5 days a week to 3 days a week. However, the distributor will only make this change in delivery service for customers that react positively. *How can the wholesale distributor use the customer segments to determine which customers, if any, would reach 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 wholesale distributor should use the segments produced by our PCA, dividing its customers in 2 segments. As they behave differently, they should not be considered as one.

To better apply this test, the distributor should select a few customers from each segment and create a "Pilot", only customers from the Pilot will have their delivery service changed. The distributor needs to be aware of the average amount spent by those customers, and also consider the values from same period of the year (in the past).

That way, after a pre-determined period of analysis and test of this new delivery model on its Pilot customers, the distributor can compare the amounts and values and check if they were lower (negative reaction) equal or greather (positive reaction).

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:

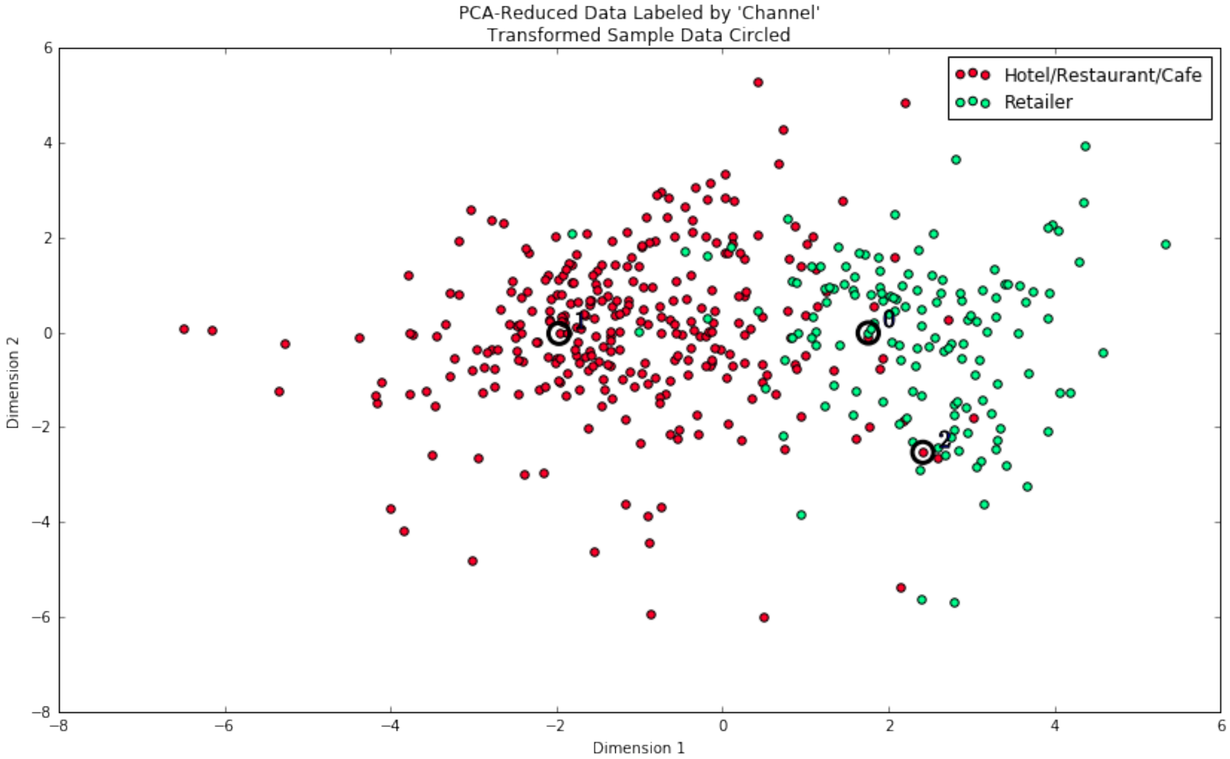
This looks to be a classification problem. In our case, we would need to classify the new customers in one of the two segments we have defined (wich segment will be our target variable).

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



Question 12

How well does the clustering algorithm and number of clusters you've chosen compare to this underlying distribution of Hotel/Restaurant/Cafe customers to Retailer customers? Are there customer segments that would be classified as purely 'Retailers' or 'Hotels/Restaurants/Cafes' by this distribution? Would you consider these classifications as consistent with your previous definition of the customer segments?

Answer:

The clustering algorithm (plus the number of clusters) applied into my research looks pretty similar to the underlying distribution of Hotel/Restaurant/Cafe customers to Retailer customers, they are both following the best silhouette score produced by the chosen 2 Clusters only.

For my first Cluster (0): I've considered this cluster a Restaurant like, and in this example it was classified as Hotel/Restaurant/Cafe. I think it was very aligned with my classifcation, however it still looks to be to much generic and presenting some points that are presenting a mixed behavior.

For my second Cluser (1): I've considered this cluster a Market like, and in this example it was classified as Retailer. I think it was also aligned with my original classification, even not being directly set as retailer by me, I think a Market like classification means pretty much the same.

After this project I saw that it is better to use the segments identified by Clustering Algorithms instead of labels we used assign based on superficial analysis. The use of Clustering helped us to identify segments that are really similar like Hotels/Restaurants/Cafes and that would probably being considerated differents by the common Labeling approach. Lessons Learnt: always classify based on Data analysis than simply trust in pre-defined labels.

> **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.