# customer_segments

January 4, 2018

## 1 Machine Learning Engineer Nanodegree

### 1.1 Unsupervised Learning

### 1.2 Project: Creating Customer Segments

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

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

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

### 1.3 Getting Started

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

The dataset for this project can be found on the UCI Machine Learning Repository. For the purposes of this project, the features `'Channel'` and `'Region'` will be excluded in the analysis — with focus instead on the six product categories recorded for customers.

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

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

# Import supplementary visualizations code visuals.py
import visuals as vs

# Pretty display for notebooks
%matplotlib inline

# Load the wholesale customers dataset
try:
    data = pd.read_csv("customers.csv")
    data.drop(['Region', 'Channel'], axis = 1, inplace = True)
    print "Wholesale customers dataset has {} samples with {} features each.".format(*
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 = [183, 289, 309]

        # 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 | 36847 | 43950 | 20170 | 36534 | 239 | 47943 |
| 1 | 42786 | 286 | 471 | 1388 | 32 | 22 |
| 2 | 918 | 20655 | 13567 | 1465 | 6846 | 806 |

### 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:** - Customer 1: with a total purchase cost of 185683 m.u. customer 1 has more expanditure than the average. Moreover expenses are globally balanced towards the 5 food categories of products (not the Detergents_Paper category). It could be a Hotel or a big restaurant. - Customer 2: with a total purchase cost of 44985 m.u. and a focus on the Fresh category (95% of the expenses), customer 2 could be a small restaurant. - Customer 3: with a total purchase cost of 44257 m.u. and the Milk, Grocery and Detergents_Paper covering more than 90% of the expenses, customer 3 could be a small convenience 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 feat
        explained_variable = data['Fresh']
        new_data = data.drop('Fresh', 1)

        # TODO: Split the data into training and testing sets using the given feature as the t
        from sklearn.cross_validation import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(new_data, explained_variable, test_

        # 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)
        # pred = regressor.fit(X_test)

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

```
-0.333070533605
```

```
C:\Users\Tron\Anaconda2\lib\site-packages\sklearn\cross_validation.py:44: DeprecationWarning: T
  "This module will be removed in 0.20.", DeprecationWarning)
```

### 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.

## 2 Answer:

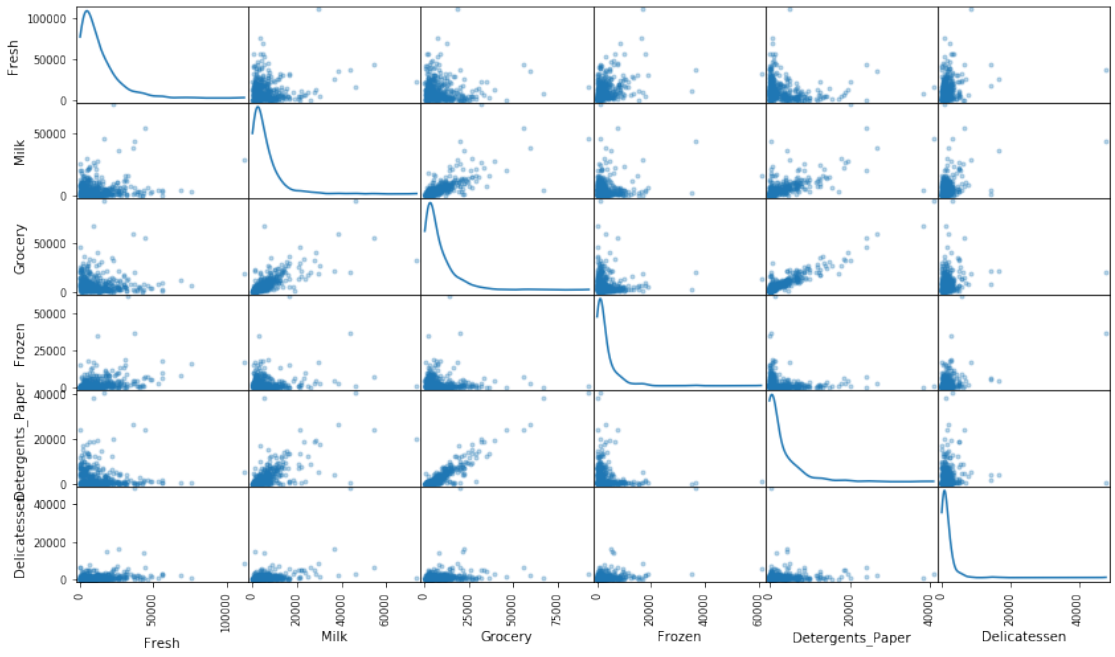
When trying to predict the Fresh feature and the obtained R2_score is -0.333070533605

This low score means that a great part of the information contained in this feature cannot be predicted by the 5 other features. So the Fresh feature is necessary for identifying customers' spending habits.

## 3 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');
```




### 3.0.1 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:** - Detergents_Paper and Grocery seems strongly correlated - Milk and Grocery seems relatively strongly correlated - Milk and Detergents_Paper seems relatively strongly correlated

This confirms that the Fresh feature cannot be easily obtained with the other features as it does not seem to be highly correlated with one of them.

All features are positively skewed. As expenses cannot be negative, the features are not normally distributed but seem to have a log-normal distribution.

Most of the data is concentrated on the lower part of the distribution as it is often the case with variables related to income or expenses.

## 3.1 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.

### 3.1.1 Implementation: Feature Scaling

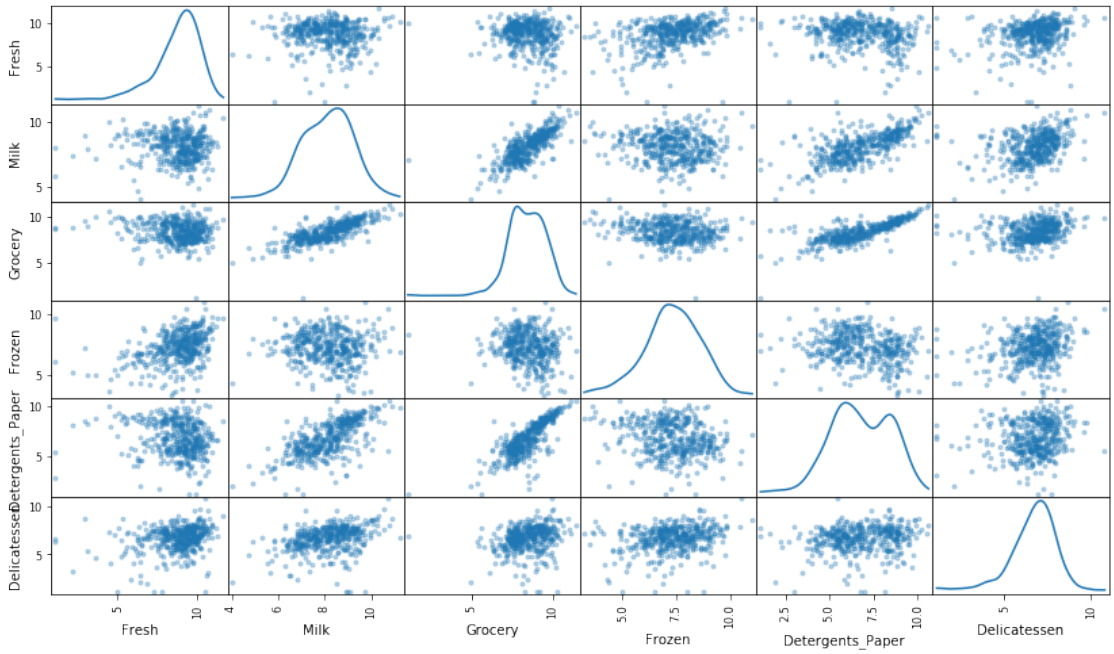
If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test, which calculates the best power 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.scatter_matrix(log_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
```




### 3.1.2 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 | 10.514529 | 10.690808 | 9.911952 | 10.505999 | 5.476464 | 10.777768 |
| 1 | 10.663966 | 5.655992 | 6.154858 | 7.235619 | 3.465736 | 3.091042 |
| 2 | 6.822197 | 9.935713 | 9.515396 | 7.289611 | 8.831420 | 6.692084 |

### 3.1.3 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)

            # 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)
```

```
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
```

#### 3.1.4 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:**

5 data points are considered outliers for more than 1 feature: # 65, # 66, # 75, # 128 & # 175

There are many reasons why a data point could be an outlier: - data errors - intentional or motivated mis-reporting - sampling error - standardization failure - faulty distributional assumptions - legitimate cases sampled from the correct population

Cf. https://www.researchgate.net/publication/242073851_The_Power_of_Outliers_and_Why_Researchers_

In this case it cannot be sampling error nor faulty distributional assumptions. The evidence of data errors, intentional or motivated mis-reporting or standardization failure, given domain knowledge, cannot clearly be proven. It would have been easier if it was possible to compare the expenses in m.u. to the cost of living or purchasing power in Portugal. It can be considered that

the outliers are legitimate cases sampled from the correct population.

However, considering that the customer population is composed by a few outliers and the majority with a simple underlying structure to be described, it is a coherent approach to remove those outliers inorder to best describe that underlying structure.

## 3.2 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.

### 3.2.1 Implementation: PCA

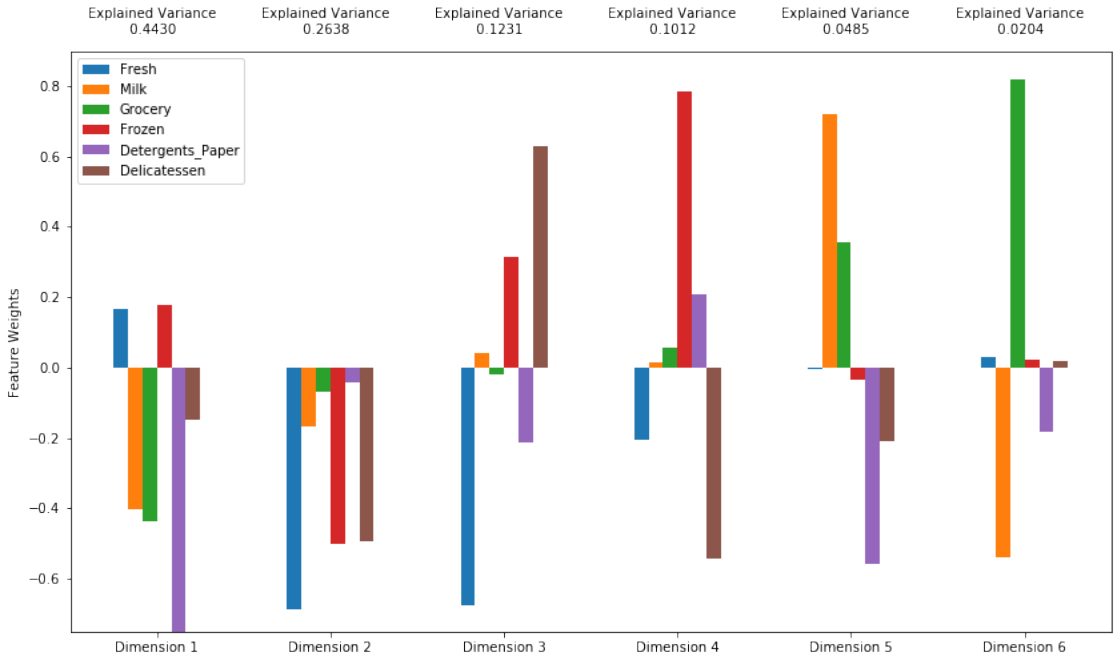
Now that the data has been scaled to a more normal distribution and has had any necessary outliers removed, we can now apply PCA to the `good_data` to discover which dimensions about the data best maximize the variance of features involved. In addition to finding these dimensions, PCA will also report 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()
         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)
```




### 3.2.2 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:** - variance explained by the first 2 principal components : 0.7190 - variance explained by the first 4 principal components : 0.9314

As shown earlier, Milk, Grocery and Detergent_Papers expenses are significatively correlated. They are thus well represented in the same axis (Dimension 1: 0.4430 of explained variance) with some residual variance in the low-explained variance axes (Dimension 5 & 6).

The first dimension may represent on the negative side the tendency to for the customer to spend money on Milk / Grocery / Detergent_Papers products, that is to say products that: - can be stocked - correspond more to retailers kind of product than to services establishment product (restaurant, hotel, cafe, etc...)

The other features are dispatched in the 3 next dimensions as they are not strongly correlated to each others: - Fresh: Dimension 2 (negative) & 3 (negative) - Delicatessen: Dimension 2 (negative), 3 (positive) & 4 (negative) - Frozen: Dimension 2 (negative), 3 (positive) & 4 (positive)

The dimension may represent on the negative side the tendency for a customer to spend money on products that can be cooked or served (restaurant, hotel, cafe, etc...)

Dimension 3 & 4 are more complex to interpret and may not represent a global tendency but fluctuations at a customer level, as their respective explained variances is around 0.1.

### 3.2.3 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      -0.4232      -5.2842       2.7396      -0.2364       2.1331
1       5.3491       1.2417      -2.9268       0.6380       0.0147
2      -3.0465       0.8893       0.9265       0.8789       0.5472

   Dimension 6
0       0.2452
1       0.0342
2      -0.5386
```

### 3.2.4 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)
         pca.fit(good_data)

         # TODO: Transform the good data using the PCA fit above
         reduced_data = pca.transform(good_data)

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

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

### 3.2.5 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      -0.4232      -5.2842
1       5.3491       1.2417
2      -3.0465       0.8893
```

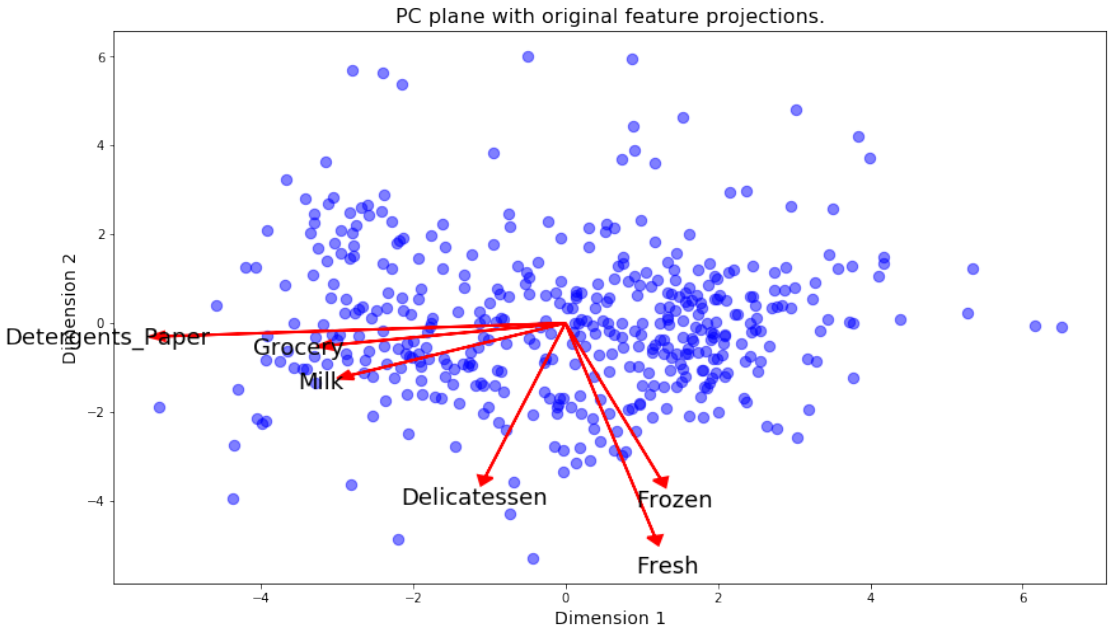
## 3.3 Visualizing a Biplot

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 0xa36cf30>
```



### 3.3.1 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?

## 3.4 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.

### 3.4.1 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:**

I K-Means:

```
I.1 advantages:
- easy to understand

I.2 disadvantages:
- need to specify K
- distance metrics can imply "round" clusters
- different results with non linear data transformation such as log transformation
- sensitive to noise and outliers

Cf. https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm
```

II Gaussian Mixture Model:

```
II.1 advantages:
- fast ()
- Can recognize "stretched" clusters

II.2 disadvantages:
- need to specify K
- need for a sufficiant number of data points in the different clusters

Cf. http://scikit-learn.org/stable/modules/mixture.html
```

III Which of the two algorithms to use?

Given the presence of outliers and the log transformation that has been applied, the GMM could be a good choice. But as the dataset is relatively small, it is quite possible to try the two algorithms (in fact I obtain comparable performance scores with the two algorithms).

### 3.4.2 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]: # TODO: Apply your clustering algorithm of choice to the reduced data
         n_clusters = 2
         from sklearn.mixture import GaussianMixture
         from sklearn.cluster import KMeans
         clusterer = GaussianMixture(n_components = n_clusters, random_state = 42)
         #clusterer = KMeans(n_clusters = n_clusters, random_state = 0)
         clusterer.fit(reduced_data)

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

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

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

```
silhouette score du clustering
0.421916846463
```

### 3.4.3 Question 7

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

**Answer:** Silhouette Score for: - 2 clusters: 0.421916846463 - 3 clusters: 0.404248738241 - 4 clusters: 0.293269564847 - 5 clusters: 0.300456388725 - 6 clusters: 0.326139450471 - 7 clusters: 0.324227205384 - 8 clusters: 0.296476656397 - 9 clusters: 0.307187479579 - 10 clusters: 0.310351081779

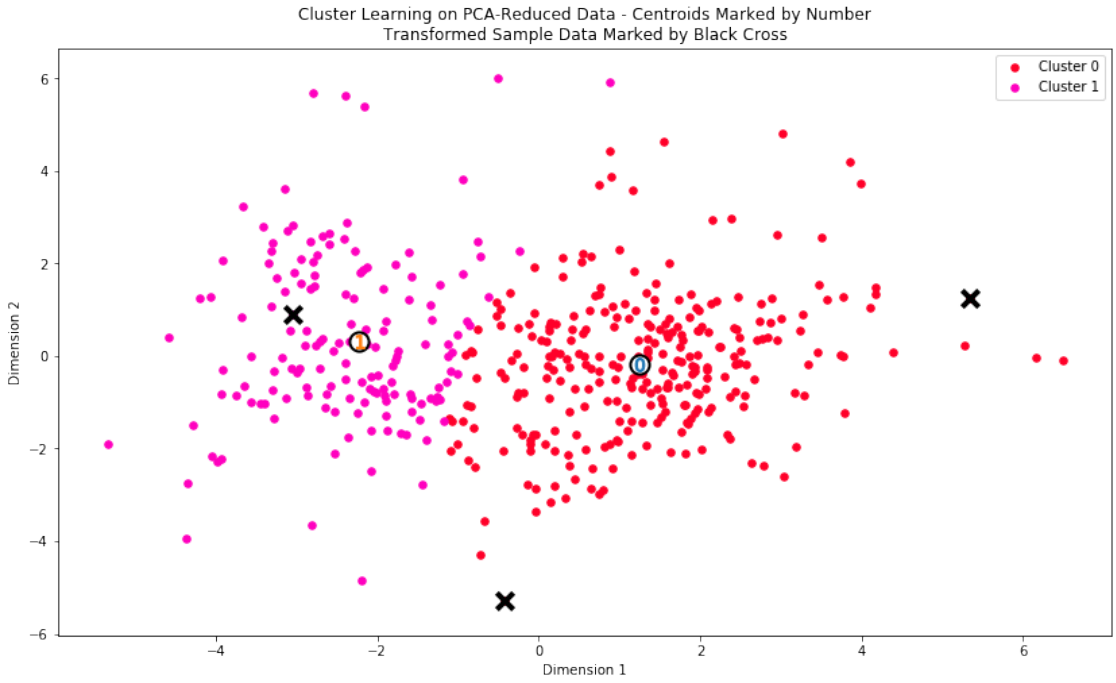
The 2 segments clustering and the 3 segments clustering obtain far better scores than the other clustering with the 2 segments clustering being slightly better than the 3 segments clustering (0.42 compared to 0.40)

The clusters are overlapping each other so the best score is relatively low.

### 3.4.4 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)
```



### 3.4.5 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)

         data_less_outliers = np.exp(good_data)
         data_less_outliers.describe()
```

| | Fresh | Milk | Grocery | Frozen | Detergents_Paper | Delicatessen |
|---|---|---|---|---|---|---|
| Segment 0 | 8953.0 | 2114.0 | 2765.0 | 2075.0 | 353.0 | 732.0 |
| Segment 1 | 3552.0 | 7837.0 | 12219.0 | 870.0 | 4696.0 | 962.0 |

Out[17]:

| | Fresh | Milk | Grocery | Frozen | \ |
|---|---|---|---|---|---|
| count | 435.000000 | 435.000000 | 435.000000 | 435.000000 | |
| mean | 12089.372414 | 5788.103448 | 7911.158621 | 3096.126437 | |
| std | 12662.796341 | 7374.172350 | 9365.740973 | 4873.769559 | |
| min | 3.000000 | 112.000000 | 218.000000 | 25.000000 | |
| 25% | 3208.000000 | 1579.500000 | 2156.000000 | 770.500000 | |
| 50% | 8565.000000 | 3634.000000 | 4757.000000 | 1541.000000 | |
| 75% | 16934.500000 | 7168.000000 | 10665.500000 | 3559.500000 | |
| max | 112151.000000 | 73498.000000 | 92780.000000 | 60869.000000 | |

| | Detergents_Paper | Delicatessen |
|---|---|---|
| count | 435.000000 | 435.000000 |
| mean | 2848.473563 | 1536.797701 |
| std | 4679.364623 | 2833.363881 |
| min | 3.000000 | 3.000000 |
| 25% | 260.000000 | 411.500000 |

```
50%          813.000000    967.000000
75%         3935.000000   1825.500000
max        40827.000000  47943.000000
```

### 3.4.6 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:**

Segment 0:

Globally low expenses in the categories of products that are best fitted for stores: - Milk: segment mean (2114.0) significantly lower than data mean without ouliers (5788.103448) - Grocery: segment mean (2765.0) significantly lower than data mean without ouliers (7911.158621) - Detergents_Paper: segment mean (353.0) significantly lower than data mean without ouliers (2848.473563)

Moreover Fresh expanditure is higher for segment 0 than segment 1 (8953.0 vs 3552.0)

=> this cluster could represent "food service" establishment such as restaurants

Segment 1:

Globally high expenses in the categories of products that are best fitted for stores: - Milk: segment mean (7837.0) higher than data mean without ouliers (5788.103448) - Grocery: segment mean (12219.0) substantially higher than data mean without ouliers (7911.158621) - Detergents_Paper: segment mean (4696.0) higher than data mean without ouliers (2848.473563)

This cluster could represent stores

### 3.4.7 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 [38]: # 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:**

Guess in question 1: - sample 0: big restaurant/hotel - sample 1: restaurant - sample 2: convinience store

Given my guess in question 1 based on the different expenses of products categories, I would predict that: - Sample point 0 belongs to segment 0 - Sample point 1 belongs to segment 0 - Sample point 2 belongs to segment 1

That is coherent with the algorithm's prediction: - 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

## 3.5 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.

### 3.5.1 Question 10

Companies will often run A/B tests when making small changes to their products or services to determine whether making that change will affect its customers positively or negatively. The wholesale distributor is considering changing its delivery service from currently 5 days a week to 3 days a week. However, the distributor will only make this change in delivery service for customers that react positively. *How can the wholesale distributor use the customer segments to determine which customers, if any, would react positively to the change in delivery service?*
**Hint:** Can we assume the change affects all customers equally? How can we determine which group of customers it affects the most?

**Answer:** The cluster 1, which represent customers which order more perishables could react negatively to a decrease of delivery frequency as it could improve the quality of their services.

What is A/B testing and how to we use it to determine good vs. bad changes? - A/B testing is a method to measure whether a change could be positive or not. - For instance, it can be used in eCommerce website to determine whether a new website look could increase the rate ofvisitors that buy products on the website or not - The principle is to split the population that would be affected by that change in 2 groups (control group and test group) and to apply a different treatment for each group in order to compare the results - The control group will have the regular treatment and the test group will have the tested treatment - Then by measuring a feedback variable for each group and comparing them, we can determine their significance (under the assumption of a given distribution of that variable) and decide whether to reject the null hypothesis or not

How does A/B testing benefit from our clustering algorithm?

To identify the group of customer that is the most affected by the A/B testing, we could use the clustering result by splitting each segment in half and propose A or B to each half i.e.: - Control group 1: 50% of segment 0 with delivery A (5 days a week) - Test group 1: 50% of segment 0 with delivery B (3 days a week) - Control group 2: 50% of segment 1 with delivery A (5 days a week) - Test group 2: 50% of segment 1 with delivery B (3 days a week)

It is necessary to define a feedback variable that would be measured (customer satisfaction, key performance indicators, etc...) to compare the effects on each half of each segment and to determine if those effects are desirable or not.

As each segment is heterogenous it can be useful to sub divise them into mutually exclusive sub-segment by adding some feature such as the total expenses or the size of the establishment and to do a stratified sampling for each A/B subgroup.

How would we use A/B testing on our new clusters to collect information and implement the new service schedules?

If it has been determined for one or the 2 segments, that it is better to distribute with delivery b than delivery A, then switch to delivery B for that segment.

### 3.5.2 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 can to train a supervised learner that could predict the customer segment from the expense features. The target variable (or labels) would be the customer segment predicted by the clustering.

### 3.5.3 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 [35]: # Display the clustering results based on 'Channel' data
         vs.channel_results(reduced_data, outliers, pca_samples)

         display(proba)
```

```
array([[  1.47029864e-01,   8.52970136e-01],
       [  1.99214066e-01,   8.00785934e-01],
       [  2.14579497e-01,   7.85420503e-01],
       [  9.94161109e-01,   5.83889082e-03],
       [  7.02092116e-01,   2.97907884e-01],
       [  4.51463918e-01,   5.48536082e-01],
       [  3.54058932e-01,   6.45941068e-01],
       [  2.83455888e-01,   7.16544112e-01],
       [  4.32448620e-01,   5.67551380e-01],
       [  3.14899097e-02,   9.68510090e-01],
       [  1.18215356e-01,   8.81784644e-01],
       [  9.89738736e-01,   1.02612645e-02],
       [  1.13737859e-01,   8.86262141e-01],
       [  1.98496130e-01,   8.01503870e-01],
       [  8.53209901e-02,   9.14679010e-01],
       [  9.35741819e-01,   6.42581812e-02],
```

```
[  3.48224690e-03,   9.96517753e-01],
[  9.46319182e-01,   5.36808177e-02],
[  4.15170980e-01,   5.84829020e-01],
[  3.89711412e-01,   6.10288588e-01],
[  6.45331313e-01,   3.54668687e-01],
[  9.98415523e-01,   1.58447721e-03],
[  8.90470185e-01,   1.09529815e-01],
[  6.66357828e-02,   9.33364217e-01],
[  1.85945970e-01,   8.14054030e-01],
[  1.42020789e-01,   8.57979211e-01],
[  9.99093661e-01,   9.06339422e-04],
[  9.99726567e-01,   2.73432617e-04],
[  1.24228240e-02,   9.87577176e-01],
[  9.76212747e-01,   2.37872530e-02],
[  5.25337386e-01,   4.74662614e-01],
[  8.78376798e-01,   1.21623202e-01],
[  9.61592973e-01,   3.84070267e-02],
[  9.86571819e-01,   1.34281811e-02],
[  9.11466378e-01,   8.85336216e-02],
[  1.56299539e-02,   9.84370046e-01],
[  9.03416971e-01,   9.65830293e-02],
[  1.17095297e-01,   8.82904703e-01],
[  1.19119104e-03,   9.98808809e-01],
[  9.99937673e-01,   6.23271429e-05],
[  9.08790784e-01,   9.12092163e-02],
[  6.83064450e-01,   3.16935550e-01],
[  1.48284971e-02,   9.85171503e-01],
[  7.97662169e-04,   9.99202338e-01],
[  8.64518546e-02,   9.13548145e-01],
[  2.28571194e-02,   9.77142881e-01],
[  2.13394854e-02,   9.78660515e-01],
[  1.00072308e-02,   9.89992769e-01],
[  1.07962921e-01,   8.92037079e-01],
[  8.92665202e-03,   9.91073348e-01],
[  9.93628985e-01,   6.37101503e-03],
[  2.08475439e-01,   7.91524561e-01],
[  6.15844585e-01,   3.84155415e-01],
[  3.33462670e-03,   9.96665373e-01],
[  9.99464164e-01,   5.35836492e-04],
[  5.42198502e-01,   4.57801498e-01],
[  6.51178127e-03,   9.93488219e-01],
[  3.62391234e-03,   9.96376088e-01],
[  9.98356551e-01,   1.64344938e-03],
[  7.15864385e-02,   9.28413562e-01],
[  6.08193990e-02,   9.39180601e-01],
[  8.83825721e-03,   9.91161743e-01],
[  3.84144548e-01,   6.15855452e-01],
[  7.09567468e-02,   9.29043253e-01],
```

```
[  9.55995395e-01,   4.40046053e-02],
[  1.77416253e-01,   8.22583747e-01],
[  8.54339203e-01,   1.45660797e-01],
[  9.84976314e-01,   1.50236865e-02],
[  9.99701628e-01,   2.98371945e-04],
[  4.03104175e-01,   5.96895825e-01],
[  9.36850088e-01,   6.31499117e-02],
[  9.47605914e-01,   5.23940860e-02],
[  4.07543122e-01,   5.92456878e-01],
[  8.89997689e-01,   1.10002311e-01],
[  1.63445213e-02,   9.83655479e-01],
[  9.99092540e-01,   9.07460081e-04],
[  5.89147106e-01,   4.10852894e-01],
[  9.78451737e-01,   2.15482631e-02],
[  1.87822111e-04,   9.99812178e-01],
[  1.38375187e-01,   8.61624813e-01],
[  9.96607041e-01,   3.39295934e-03],
[  5.09209474e-01,   4.90790526e-01],
[  2.02166842e-03,   9.97978332e-01],
[  5.81694483e-03,   9.94183055e-01],
[  7.01160529e-01,   2.98839471e-01],
[  9.99925733e-01,   7.42672892e-05],
[  9.76935188e-01,   2.30648116e-02],
[  9.99988511e-01,   1.14894364e-05],
[  9.91564019e-01,   8.43598061e-03],
[  1.06921019e-02,   9.89307898e-01],
[  9.99885346e-01,   1.14653564e-04],
[  1.65145904e-02,   9.83485410e-01],
[  2.35877305e-03,   9.97641227e-01],
[  3.04489955e-05,   9.99969551e-01],
[  9.99972969e-01,   2.70308688e-05],
[  9.99989576e-01,   1.04237917e-05],
[  9.99434725e-01,   5.65274521e-04],
[  1.09606177e-01,   8.90393823e-01],
[  1.66430379e-02,   9.83356962e-01],
[  1.48678091e-01,   8.51321909e-01],
[  8.96922812e-01,   1.03077188e-01],
[  9.18416748e-01,   8.15832518e-02],
[  9.99159617e-01,   8.40383191e-04],
[  4.36570623e-03,   9.95634294e-01],
[  5.72486872e-02,   9.42751313e-01],
[  3.69864541e-02,   9.63013546e-01],
[  2.38360953e-02,   9.76163905e-01],
[  9.99417189e-01,   5.82810891e-04],
[  5.10630439e-02,   9.48936956e-01],
[  9.93603926e-01,   6.39607362e-03],
[  9.93078656e-01,   6.92134351e-03],
[  9.99850456e-01,   1.49543751e-04],
```

```
[  9.99066510e-01,   9.33489882e-04],
[  9.95164889e-01,   4.83511089e-03],
[  9.76260755e-01,   2.37392448e-02],
[  9.97620213e-01,   2.37978719e-03],
[  9.98205245e-01,   1.79475540e-03],
[  9.99295312e-01,   7.04688325e-04],
[  9.97182495e-01,   2.81750529e-03],
[  9.99999968e-01,   3.17735645e-08],
[  5.00877008e-01,   4.99122992e-01],
[  9.95946950e-01,   4.05305045e-03],
[  9.80223422e-01,   1.97765779e-02],
[  9.99268913e-01,   7.31087010e-04],
[  3.55838723e-01,   6.44161277e-01],
[  9.98901655e-01,   1.09834526e-03],
[  9.98946821e-01,   1.05317887e-03],
[  9.99975971e-01,   2.40294533e-05],
[  9.60404343e-01,   3.95956565e-02],
[  9.98766689e-01,   1.23331140e-03],
[  9.99890659e-01,   1.09340927e-04],
[  9.98569785e-01,   1.43021465e-03],
[  4.20352705e-01,   5.79647295e-01],
[  4.79343263e-01,   5.20656737e-01],
[  9.49885439e-01,   5.01145610e-02],
[  8.96281935e-01,   1.03718065e-01],
[  8.67811944e-01,   1.32188056e-01],
[  9.97183610e-01,   2.81639000e-03],
[  9.99989995e-01,   1.00048623e-05],
[  9.99432270e-01,   5.67730367e-04],
[  9.57605280e-01,   4.23947198e-02],
[  4.62150359e-03,   9.95378496e-01],
[  9.99759986e-01,   2.40013960e-04],
[  9.25739314e-01,   7.42606860e-02],
[  9.99910318e-01,   8.96824952e-05],
[  9.99787161e-01,   2.12838872e-04],
[  9.99152068e-01,   8.47931820e-04],
[  9.61795447e-01,   3.82045533e-02],
[  9.98624757e-01,   1.37524291e-03],
[  9.46466220e-01,   5.35337799e-02],
[  2.43048245e-03,   9.97569518e-01],
[  2.12922261e-02,   9.78707774e-01],
[  9.93450044e-01,   6.54995609e-03],
[  8.02979120e-02,   9.19702088e-01],
[  7.45619433e-04,   9.99254381e-01],
[  5.13664239e-02,   9.48633576e-01],
[  9.99999999e-01,   1.48875300e-09],
[  9.98022247e-01,   1.97775278e-03],
[  1.27970646e-02,   9.87202935e-01],
[  2.34700546e-01,   7.65299454e-01],
```

```
[  9.07068159e-02,   9.09293184e-01],
[  9.23277108e-02,   9.07672289e-01],
[  4.54425621e-01,   5.45574379e-01],
[  9.99977477e-01,   2.25231521e-05],
[  9.99948034e-01,   5.19656372e-05],
[  1.84274085e-03,   9.98157259e-01],
[  6.66638838e-04,   9.99333361e-01],
[  3.13600276e-02,   9.68639972e-01],
[  2.15690716e-03,   9.97843093e-01],
[  9.83066769e-01,   1.69332311e-02],
[  5.00519640e-03,   9.94994804e-01],
[  9.16730479e-01,   8.32695214e-02],
[  9.99835875e-01,   1.64124756e-04],
[  9.95515436e-01,   4.48456403e-03],
[  9.89527955e-01,   1.04720455e-02],
[  4.45270527e-01,   5.54729473e-01],
[  1.08793239e-01,   8.91206761e-01],
[  2.40994970e-03,   9.97590050e-01],
[  6.01386742e-01,   3.98613258e-01],
[  1.45812108e-01,   8.54187892e-01],
[  9.99890133e-01,   1.09866705e-04],
[  9.96409239e-01,   3.59076071e-03],
[  1.73205231e-01,   8.26794769e-01],
[  1.19175966e-01,   8.80824034e-01],
[  6.39802872e-03,   9.93601971e-01],
[  9.99808800e-01,   1.91200467e-04],
[  9.99998481e-01,   1.51904006e-06],
[  9.97538749e-01,   2.46125127e-03],
[  2.22343877e-03,   9.97776561e-01],
[  9.93491626e-01,   6.50837398e-03],
[  9.88490050e-01,   1.15099504e-02],
[  9.51563446e-01,   4.84365544e-02],
[  6.49888965e-02,   9.35011103e-01],
[  9.76799398e-01,   2.32006017e-02],
[  9.89092863e-01,   1.09071374e-02],
[  2.14778324e-02,   9.78522168e-01],
[  1.62632649e-02,   9.83736735e-01],
[  9.41564837e-01,   5.84351630e-02],
[  6.32475757e-01,   3.67524243e-01],
[  9.99998308e-01,   1.69205111e-06],
[  3.68461267e-03,   9.96315387e-01],
[  9.99360926e-01,   6.39074144e-04],
[  1.62578837e-01,   8.37421163e-01],
[  1.97797568e-01,   8.02202432e-01],
[  1.23262165e-02,   9.87673783e-01],
[  9.98883085e-01,   1.11691521e-03],
[  1.22700453e-02,   9.87729955e-01],
[  9.98384834e-01,   1.61516605e-03],
```

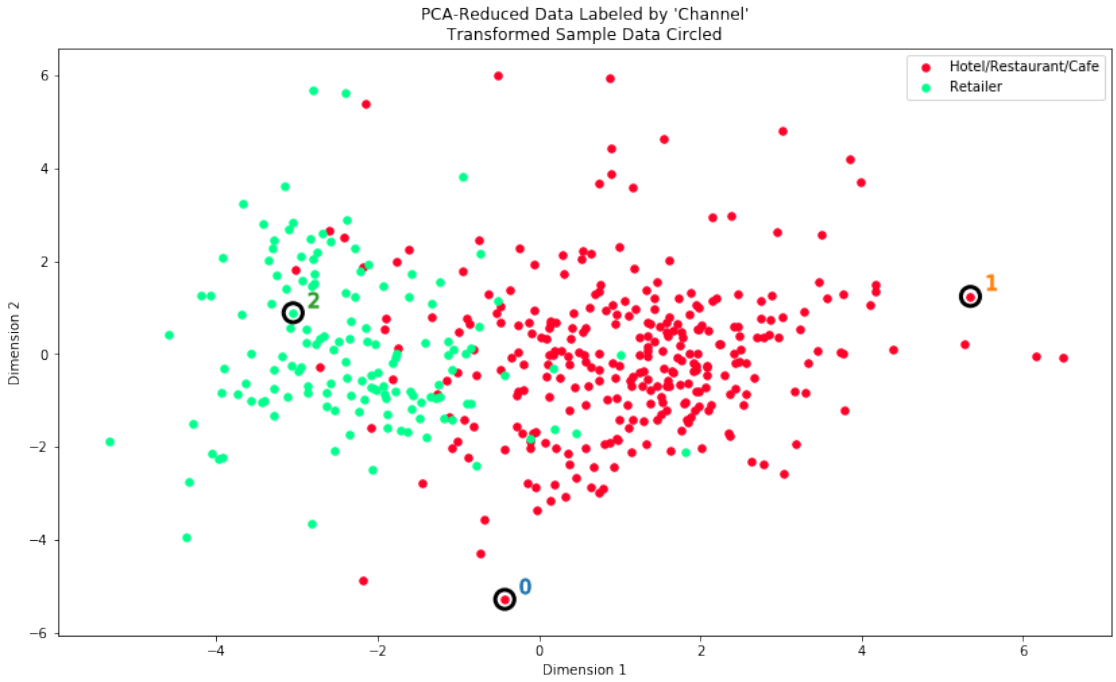
```
[  1.42660364e-01,   8.57339636e-01],
[  3.28628531e-02,   9.67137147e-01],
[  3.27437055e-02,   9.67256295e-01],
[  8.04988333e-04,   9.99195012e-01],
[  9.99743318e-01,   2.56681953e-04],
[  3.73608981e-04,   9.99626391e-01],
[  9.99984365e-01,   1.56346531e-05],
[  9.99984017e-01,   1.59828659e-05],
[  1.48141479e-02,   9.85185852e-01],
[  9.99172348e-01,   8.27651644e-04],
[  9.32448565e-01,   6.75514354e-02],
[  9.99950210e-01,   4.97895959e-05],
[  9.22892030e-01,   7.71079696e-02],
[  4.94839117e-01,   5.05160883e-01],
[  9.97478866e-01,   2.52113429e-03],
[  8.07009770e-01,   1.92990230e-01],
[  9.87406590e-01,   1.25934101e-02],
[  9.05280644e-01,   9.47193561e-02],
[  4.01403744e-01,   5.98596256e-01],
[  9.95376375e-01,   4.62362469e-03],
[  7.82429281e-01,   2.17570719e-01],
[  9.93364259e-01,   6.63574052e-03],
[  5.98944035e-01,   4.01055965e-01],
[  9.97751458e-01,   2.24854187e-03],
[  9.99995321e-01,   4.67863100e-06],
[  9.99994564e-01,   5.43552792e-06],
[  9.80132697e-01,   1.98673026e-02],
[  9.88574822e-01,   1.14251779e-02],
[  9.96610393e-01,   3.38960728e-03],
[  9.97894700e-01,   2.10530029e-03],
[  6.27880948e-01,   3.72119052e-01],
[  1.70858794e-01,   8.29141206e-01],
[  9.01452722e-03,   9.90985473e-01],
[  9.94258503e-01,   5.74149654e-03],
[  9.99999400e-01,   6.00152617e-07],
[  9.62476474e-01,   3.75235263e-02],
[  9.81835427e-01,   1.81645730e-02],
[  9.96957951e-01,   3.04204942e-03],
[  1.28723645e-02,   9.87127635e-01],
[  9.97703589e-01,   2.29641066e-03],
[  8.52836721e-01,   1.47163279e-01],
[  5.99188881e-01,   4.00811119e-01],
[  9.99981625e-01,   1.83750188e-05],
[  9.43152257e-01,   5.68477425e-02],
[  8.73072312e-01,   1.26927688e-01],
[  9.82915645e-01,   1.70843546e-02],
[  9.33092738e-01,   6.69072620e-02],
[  8.88700357e-01,   1.11299643e-01],
```

```
[  9.99995198e-01,   4.80185183e-06],
[  9.97562594e-01,   2.43740561e-03],
[  6.45922413e-01,   3.54077587e-01],
[  9.50150350e-04,   9.99049850e-01],
[  5.66001475e-01,   4.33998525e-01],
[  1.32317343e-02,   9.86768266e-01],
[  9.90668557e-01,   9.33144265e-03],
[  6.52755970e-02,   9.34724403e-01],
[  9.99925730e-01,   7.42704972e-05],
[  9.99914602e-01,   8.53983321e-05],
[  9.94411139e-01,   5.58886059e-03],
[  8.92652352e-01,   1.07347648e-01],
[  9.99252819e-01,   7.47181415e-04],
[  8.57208011e-01,   1.42791989e-01],
[  9.99850227e-01,   1.49772959e-04],
[  9.62497710e-01,   3.75022902e-02],
[  9.99785339e-01,   2.14661209e-04],
[  9.97931557e-01,   2.06844350e-03],
[  2.98794729e-01,   7.01205271e-01],
[  9.95047068e-01,   4.95293189e-03],
[  4.39300853e-01,   5.60699147e-01],
[  9.62639540e-01,   3.73604597e-02],
[  9.98017186e-01,   1.98281430e-03],
[  9.29875635e-01,   7.01243651e-02],
[  9.99255479e-01,   7.44520938e-04],
[  9.74780685e-01,   2.52193150e-02],
[  9.97267596e-01,   2.73240367e-03],
[  9.99133774e-01,   8.66225620e-04],
[  9.99999975e-01,   2.48093461e-08],
[  8.08413583e-01,   1.91586417e-01],
[  9.95770202e-01,   4.22979819e-03],
[  9.15782716e-01,   8.42172836e-02],
[  3.37473978e-02,   9.66252602e-01],
[  9.34664120e-01,   6.53358796e-02],
[  5.45727154e-01,   4.54272846e-01],
[  9.90522257e-01,   9.47774301e-03],
[  3.94089922e-01,   6.05910078e-01],
[  1.48929447e-01,   8.51070553e-01],
[  9.16374024e-01,   8.36259759e-02],
[  4.15810171e-01,   5.84189829e-01],
[  2.21843201e-02,   9.77815680e-01],
[  2.22080585e-02,   9.77791941e-01],
[  7.51309318e-03,   9.92486907e-01],
[  3.89315283e-04,   9.99610685e-01],
[  2.81270617e-03,   9.97187294e-01],
[  3.49466862e-02,   9.65053314e-01],
[  9.93833269e-01,   6.16673148e-03],
[  9.94954326e-01,   5.04567385e-03],
```

```
[  7.22585754e-03,   9.92774142e-01],
[  9.98830464e-01,   1.16953569e-03],
[  9.94741338e-01,   5.25866233e-03],
[  6.23809783e-03,   9.93761902e-01],
[  9.99768810e-01,   2.31190274e-04],
[  9.22407397e-01,   7.75926034e-02],
[  2.13378877e-02,   9.78662112e-01],
[  9.99848652e-01,   1.51347704e-04],
[  5.91334662e-01,   4.08665338e-01],
[  9.83703925e-01,   1.62960750e-02],
[  1.53195605e-02,   9.84680439e-01],
[  9.59153924e-01,   4.08460763e-02],
[  9.96053710e-01,   3.94628986e-03],
[  9.87387157e-01,   1.26128426e-02],
[  8.08253707e-01,   1.91746293e-01],
[  9.99167387e-01,   8.32612812e-04],
[  6.18468096e-01,   3.81531904e-01],
[  9.97299015e-01,   2.70098462e-03],
[  9.97607206e-01,   2.39279355e-03],
[  9.98696235e-01,   1.30376544e-03],
[  9.89805030e-01,   1.01949703e-02],
[  9.98056983e-01,   1.94301678e-03],
[  2.35809606e-02,   9.76419039e-01],
[  9.98979308e-01,   1.02069224e-03],
[  8.90696397e-04,   9.99109304e-01],
[  9.98720692e-01,   1.27930777e-03],
[  2.75473078e-01,   7.24526922e-01],
[  9.99580743e-01,   4.19256592e-04],
[  9.97584037e-01,   2.41596310e-03],
[  9.99031542e-01,   9.68458185e-04],
[  9.83101145e-01,   1.68988552e-02],
[  3.16999536e-03,   9.96830005e-01],
[  4.17890116e-02,   9.58210988e-01],
[  4.19767001e-03,   9.95802330e-01],
[  2.37562152e-03,   9.97624378e-01],
[  9.98894905e-01,   1.10509467e-03],
[  3.39402671e-02,   9.66059733e-01],
[  5.28387173e-02,   9.47161283e-01],
[  3.05009755e-01,   6.94990245e-01],
[  9.03280797e-01,   9.67192027e-02],
[  1.75976327e-02,   9.82402367e-01],
[  9.99029471e-01,   9.70528722e-04],
[  6.38069329e-03,   9.93619307e-01],
[  9.98786951e-01,   1.21304869e-03],
[  1.21094322e-04,   9.99878906e-01],
[  9.03518709e-01,   9.64812915e-02],
[  9.09389969e-01,   9.06100309e-02],
[  1.00000000e+00,   4.28204489e-10],
```

```
[  1.01761722e-05,   9.99989824e-01],
[  8.48899040e-01,   1.51100960e-01],
[  8.80408835e-01,   1.19591165e-01],
[  9.95590532e-01,   4.40946812e-03],
[  9.99875389e-01,   1.24610706e-04],
[  9.93404738e-01,   6.59526165e-03],
[  6.64842872e-01,   3.35157128e-01],
[  9.91746053e-01,   8.25394689e-03],
[  1.53055443e-01,   8.46944557e-01],
[  9.98508586e-01,   1.49141399e-03],
[  9.99689034e-01,   3.10965947e-04],
[  9.99891437e-01,   1.08562525e-04],
[  9.99998583e-01,   1.41687522e-06],
[  9.70438285e-01,   2.95617153e-02],
[  9.95780259e-01,   4.21974140e-03],
[  9.71783381e-01,   2.82166194e-02],
[  5.49788363e-01,   4.50211637e-01],
[  9.96229122e-01,   3.77087762e-03],
[  9.94365080e-01,   5.63491999e-03],
[  1.28844723e-01,   8.71155277e-01],
[  9.99515907e-01,   4.84093302e-04],
[  9.98284720e-01,   1.71527983e-03],
[  2.30286239e-01,   7.69713761e-01],
[  9.98630715e-01,   1.36928468e-03],
[  9.99071318e-01,   9.28681997e-04],
[  6.64855898e-01,   3.35144102e-01],
[  9.90431815e-01,   9.56818500e-03],
[  8.27552122e-01,   1.72447878e-01],
[  9.99694769e-01,   3.05231213e-04],
[  9.58404631e-01,   4.15953694e-02],
[  9.99214374e-01,   7.85625683e-04],
[  9.99789662e-01,   2.10338377e-04],
[  9.99434752e-01,   5.65247697e-04],
[  9.98902670e-01,   1.09732976e-03],
[  9.98015832e-01,   1.98416843e-03],
[  9.59682333e-01,   4.03176669e-02],
[  9.90855790e-01,   9.14420983e-03],
[  9.99605051e-01,   3.94949081e-04],
[  9.95402063e-01,   4.59793683e-03],
[  1.66326231e-01,   8.33673769e-01],
[  9.99973878e-01,   2.61216553e-05],
[  9.99995756e-01,   4.24355943e-06],
[  9.99343820e-01,   6.56180374e-04],
[  9.99811938e-01,   1.88061843e-04],
[  9.99836385e-01,   1.63614522e-04],
[  9.99976347e-01,   2.36529036e-05],
[  7.61998137e-01,   2.38001863e-01],
[  9.63971160e-01,   3.60288401e-02],
```

```
[  9.97231500e-01,   2.76850033e-03],
[  9.48564056e-01,   5.14359440e-02],
[  5.04815362e-02,   9.49518464e-01],
[  5.28037041e-01,   4.71962959e-01],
[  6.16730309e-01,   3.83269691e-01],
[  7.55784461e-01,   2.44215539e-01],
[  9.79801659e-01,   2.01983415e-02],
[  8.90701377e-05,   9.99910930e-01],
[  9.10039191e-01,   8.99608088e-02],
[  9.68331746e-01,   3.16682539e-02],
[  2.64216882e-01,   7.35783118e-01],
[  3.37612123e-02,   9.66238788e-01],
[  7.93502073e-02,   9.20649793e-01],
[  7.59520650e-04,   9.99240479e-01],
[  7.92388265e-01,   2.07611735e-01],
[  2.89003212e-03,   9.97109968e-01],
[  3.39693352e-01,   6.60306648e-01],
[  9.66242004e-01,   3.37579962e-02],
[  7.86299343e-01,   2.13700657e-01],
[  4.06771604e-01,   5.93228396e-01],
[  9.99719078e-01,   2.80922488e-04],
[  1.62481902e-01,   8.37518098e-01],
[  9.66861076e-01,   3.31389236e-02],
[  8.07899390e-01,   1.92100610e-01],
[  9.94194476e-01,   5.80552446e-03],
[  8.40548197e-01,   1.59451803e-01],
[  8.82896113e-01,   1.17103887e-01],
[  8.92942245e-01,   1.07057755e-01],
[  9.84053451e-01,   1.59465490e-02],
[  4.99881575e-01,   5.00118425e-01],
[  9.76690642e-01,   2.33093583e-02],
[  9.99967628e-01,   3.23718433e-05],
[  8.79112770e-03,   9.91208872e-01],
[  9.98290816e-01,   1.70918436e-03],
[  7.39475179e-01,   2.60524821e-01]])
```




### 3.5.4 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 silhouette score has permitted to select the same number of clusters as in the graph above (2 segments). The "service" establishment vs store "establishment" determined by the PCA and the clustering is coherent with the underlying distribution (Hotels/Restaurants/Cafes vs Retailers).

However the main difference between the clustering obtained segments andthe underlying distribution is that there is obviously some overlapping through the different segments (i.e. there are some sample points that are not purely in one segment or the other). Examining the 'proba' (probability of belonging to one of the other segment for each sample point) variable could help bring to light that.

Given the distribution with labeled data, there are reasons to think that GMM is a better fit for this dataset. Theoritically K-Means does not perform well with overlapping clusters. However obtained silhouette scores are comparable when applying those 2 different algorithms, even slightly better for K-Means.

To conclude, this classification is consistent with the previous definition of the customer segments, but an algorithm wouldn't be able to predict such a classification with this dataset.

> **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.