customer_segments

October 26, 2017

1 Machine Learning Engineer Nanodegree

1.1 Unsupervised Learning

1.2 Project: Creating Customer Segments

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

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

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

1.3 Getting Started

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

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

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

In [1]:
```
# Import libraries necessary for this project
import numpy as np
import pandas as pd
from IPython.display import display # Allows the use of display() for DataFrames

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

# Pretty display for notebooks
%matplotlib inline

# Load the wholesale customers dataset
try:
    data = pd.read_csv("customers.csv")
    imported_data = data.drop(['Region', 'Channel'], axis = 1, inplace = None)
    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 8 features each.
```

1.4 Data Exploration

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

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

In [2]:
```
# Display a description of the dataset
display(data.describe())
```

	Channel	Region	Fresh	Milk	Grocery	\
count	440.000000	440.000000	440.000000	440.000000	440.000000	
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	
min	1.000000	1.000000	3.000000	55.000000	3.000000	
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	

	Frozen	Detergents_Paper	Delicatessen
count	440.000000	440.000000	440.000000
mean	3071.931818	2881.493182	1524.870455

```
std     4854.673333       4767.854448   2820.105937
min       25.000000          3.000000      3.000000
25%      742.250000        256.750000    408.250000
50%     1526.000000        816.500000    965.500000
75%     3554.250000       3922.000000   1820.250000
max    60869.000000      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 [5]: # TODO: Select three indices of your choice you wish to sample from the dataset
        indices = [1, 2, 3]

        # 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:
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	\
0	2	3	7057	9810	9568	1762	3293	
1	2	3	6353	8808	7684	2405	3516	
2	1	3	13265	1196	4221	6404	507	

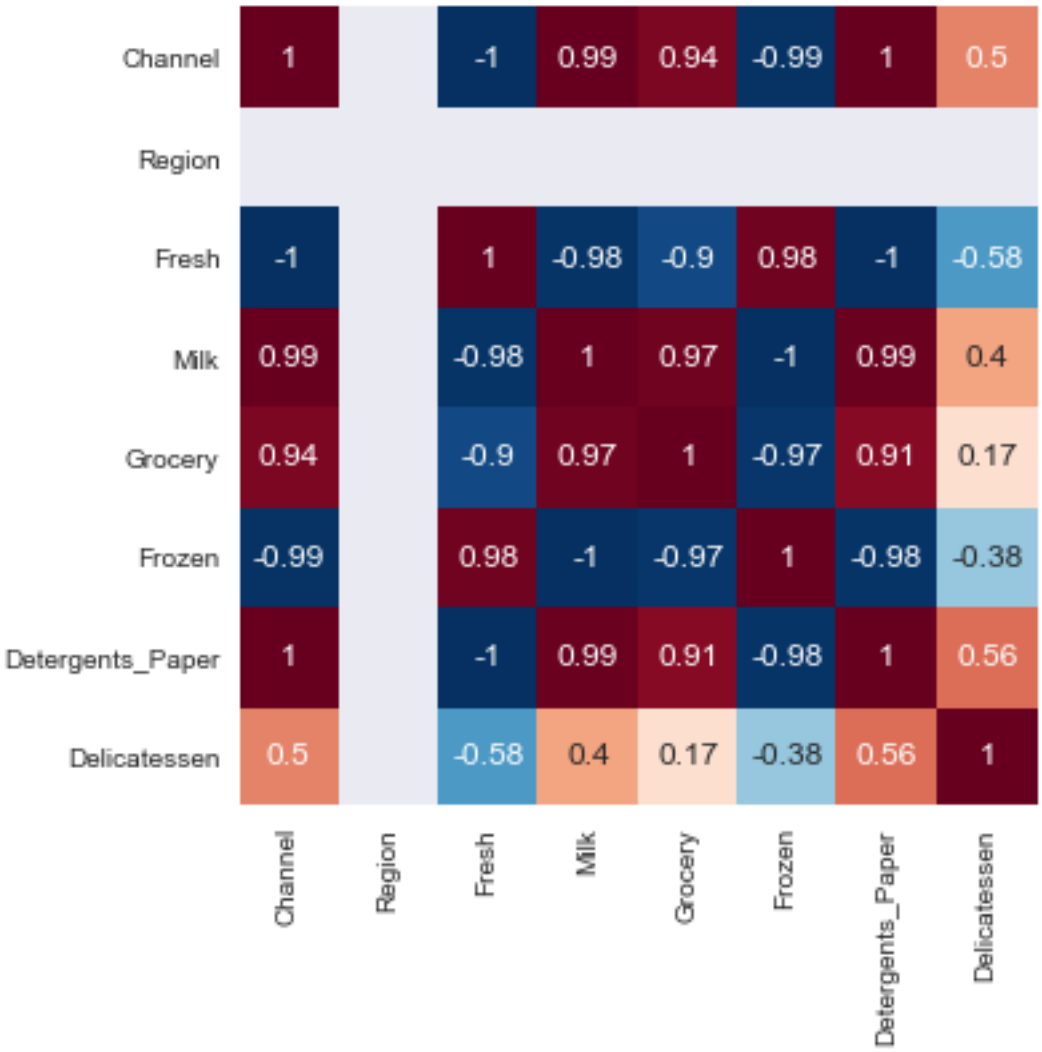
	Delicatessen
0	1776
1	7844
2	1788

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.

```
In [7]: import seaborn as sns
        sns.heatmap(samples.corr(), annot=True, cbar=False, square=True)
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x17638e92390>
```



Answer:

The first sample indicates that the milk and grocery items have the customer spending values greater than the overall mean of the dataset. Detergents_Paper and Delicatessen product categories are also above the mean values. The distance of the milk products is the closest value to the standard deviation shown in the statistical description table compared to Grocery and Detergents_Paper products indicating that there are greater chances that customers will buy milk products from that segment. This indicates the relative share of the sample is slightly greater for Milk, Grocery and detergents_paper products compared with the dataset.

The second sample shows a greater rise in usage of Delicatessen products. The mean values for Milk, Grocery, Detergents_Paper and Delicatessen are higher than the overall mean whereas the distance of Delicatessen category from the mean is higher than the standard deviation. This indicates that the Delicatessen (Ready to Eat) is the most probable product category that is bought from that segment.

*** The third sample indicates a large increase in the usage of fresh products. The distance value is about 0.6 of the overall standard deviation of the Frozen category.***

The first sample should focus on delivery of milk products rather than Fresh or Frozen products to the Hotel/Restaurant/Cafe Customer Establishment Area.

The second sample should focus on delivery or production of Delicatessen products to the Hotel/Restaurant/Cafe Customer Establishment Area.

The third sample should focus on delivery of fresh products to the Retail customer establishment area to meet the needs of the Customer.

The first sample is likely to be a customer for a restaurant using light weight products for delivery. The second sample is likely to be a customer for a cafe using light weight products for delivery. The third sample is likely to be a large retail store using heavy weight products for delivery.

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 [8]: from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.metrics import r2_score

        # TODO: Make a copy of the DataFrame, using the 'drop' function to drop the given feat
        new_data = data.drop(['Detergents_Paper'], axis = 1, inplace = None)

        y_data = pd.DataFrame(imported_data, columns = ['Detergents_Paper'])

        # TODO: Split the data into training and testing sets using the given feature as the t
        X_train, X_test, y_train, y_test = train_test_split(new_data, y_data, test_size=0.25, r

        # TODO: Create a decision tree regressor and fit it to the training set
        regressor = DecisionTreeRegressor(criterion='mse', splitter='best', max_leaf_nodes=15)

        # fitting the regressor
        regressor.fit(new_data, y_data)

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

```
        y_pred = regressor.predict(new_data)

        R_2 = r2_score(y_pred, y_data)
```

```
In [9]: print(score)
        print(R_2)
```

```
0.964033750244
0.939072336171
```

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.

```
In [15]: from sklearn.cross_validation import train_test_split
         from sklearn.tree import DecisionTreeRegressor

         def calculate_r_2_for_feature(data,feature):
             new_data = data.drop(feature, axis=1)
             X_train, X_test, y_train, y_test = train_test_split(
             new_data,data[feature],test_size=0.25
             )
             regressor = DecisionTreeRegressor()
             regressor.fit(X_train,y_train)
             score = regressor.score(X_test,y_test)
             return score

         def r_2_mean(data,feature,runs=200):
             return np.array([calculate_r_2_for_feature(data,feature) for _ in range(200) ]).me

         print("{0:17} {1}".format("Fresh: ", r_2_mean(data,'Fresh')))
         print("{0:17} {1}".format("Milk: ", r_2_mean(data,'Milk')))
         print("{0:17} {1}".format("Grocery: ", r_2_mean(data,'Grocery')))
         print("{0:17} {1}".format("Frozen: ", r_2_mean(data,'Frozen')))
         print("{0:17} {1}".format("Detergents_Paper: ", r_2_mean(data,'Detergents_Paper')))
         print("{0:17} {1}".format("Delicatessen: ", r_2_mean(data,'Delicatessen')))
```

```
Fresh:            -0.7583
Milk:             0.0728
Grocery:          0.6908
Frozen:           -1.2237
Detergents_Paper:  0.7097
Delicatessen:     -3.0488
```

Answer:

The feature attempted to predict is Detergents_Paper within the 440 region and channel combinations. The prediction considers the relation between "Detergents & Paper" and the rest of the features which are food related products.

The prediction score I got is 0.97 taking into consideration max_leaf_nodes = 15, which is as close to 1. The feature Detergents and Paper compared to the other food products is the least dependent parameter that affects the customer spending habits. A decision tree classifier is capable to achieve the prediction score of 1, but since we do not know the segmentation behaviour of the customers' spending habits, we have to try with a selected parameter from the parameter space.

The feature (Detergents_Paper) will matter when the priority for the feature is important, but I would say the customer spending habits will depend on this feature because 1st 25% of the dataset is having a relatively low mean value compared to the 1st 50% of the dataset and triples the values each time as going towards 75% of the dataset.*

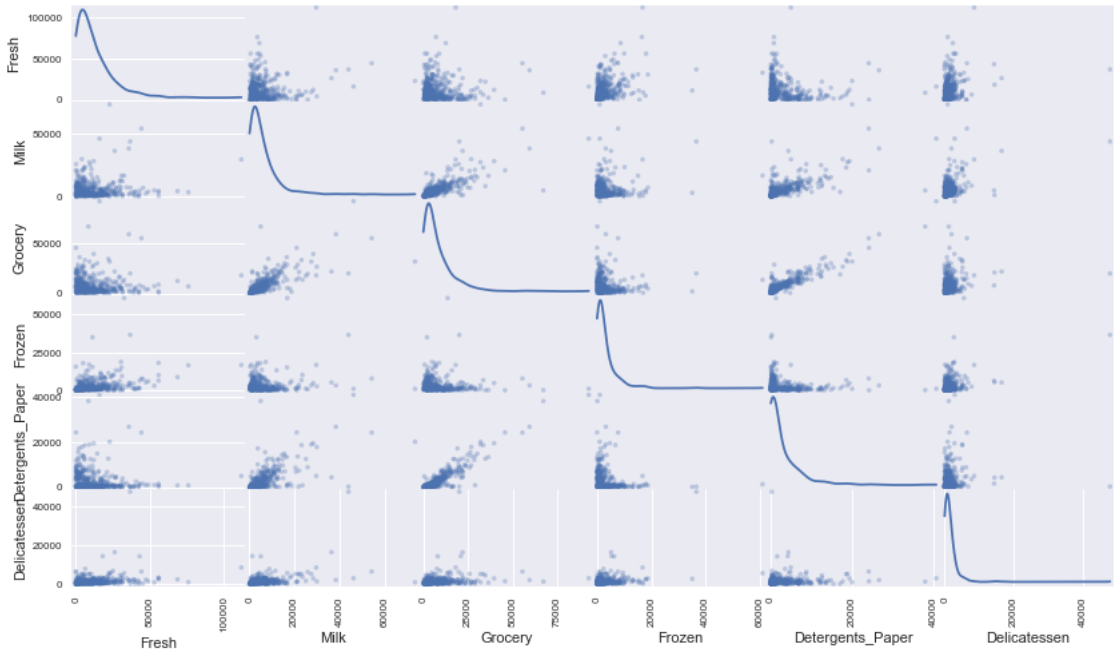
The coefficient of determination R_2 score I got is around 0.9322, which is a reasonable value to fit our world view of what we have obtained.

1.4.5 Visualize Feature Distributions

To get a better understanding of the dataset, we can construct a scatter matrix of each of the six product features present in the data. If you found that the feature you attempted to predict above is relevant for identifying a specific customer, then the scatter matrix below may not show any correlation between that feature and the others. Conversely, if you believe that feature is not relevant for identifying a specific customer, the scatter matrix might show a correlation between that feature and another feature in the data. Run the code block below to produce a scatter matrix.

```
In [13]: # Produce a scatter matrix for each pair of features in the data
         pd.scatter_matrix(imported_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
```

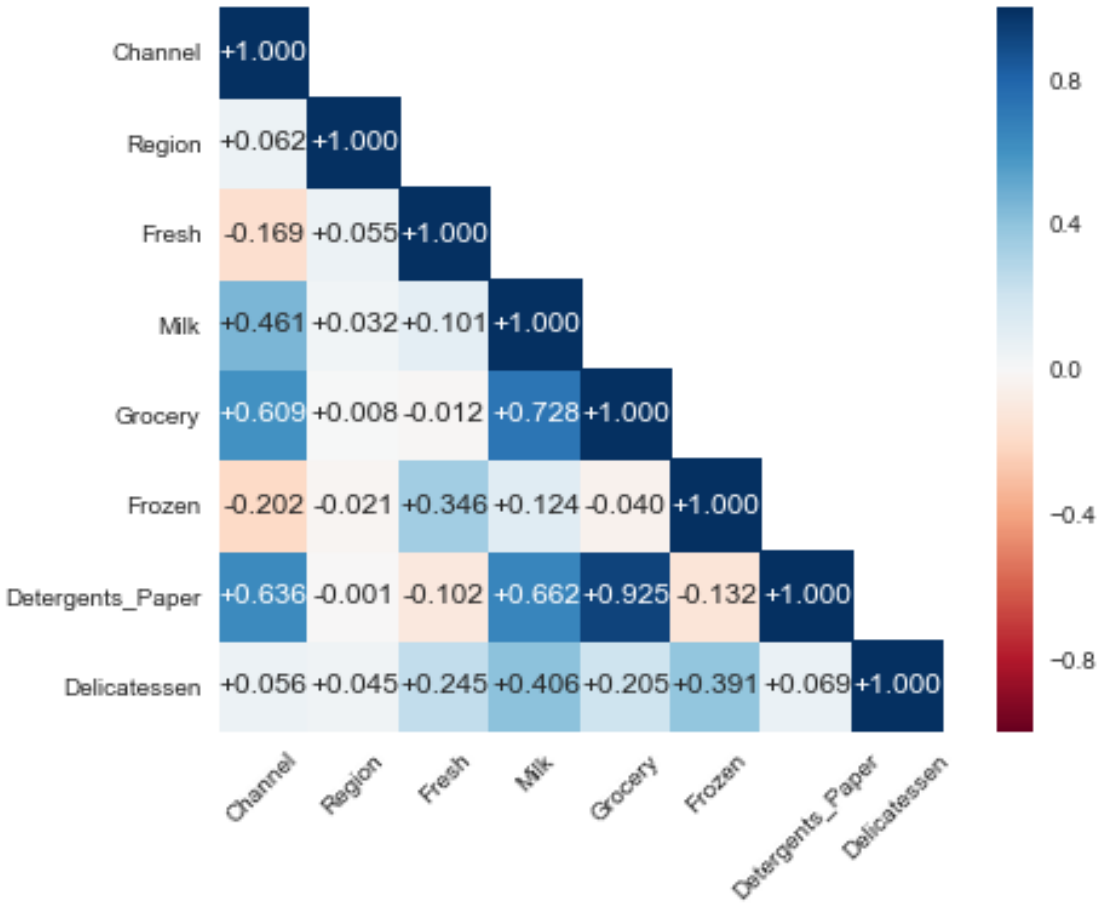
```
C:\Users\Aswin\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: pandas.scat
```



1.4.6 Question 3

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

```
In [17]: import matplotlib.pyplot as plt
         corr = data.corr()
         mask = np.zeros_like(corr)
         mask[np.triu_indices_from(mask, 1)] = True

         with sns.axes_style("white"):
             ax = sns.heatmap(corr, mask=mask, square=True, annot=True, cmap='RdBu', fmt='+.3f
             plt.xticks(rotation=45, ha='center')
```



Answer:

Out of the 6 features, the Milk and Frozen features are highly correlated as their standard deviation is low. Reflecting upon the variation that Frozen products show, it shows a high count at a very low price value and then the count decreases while Milk products show a gradual decrease of the count value.

Frozen products are mainly used for storage and usage, which implies stock management gets easier. In order to consider the customer behaviour, we need to take similar categories that show similar buying behaviour. The product pricing of the Frozen and Milk products do not have much variations due to their low standard deviation value. Frozen products are bought well in advance so that they get retained at the customer's place minimising the storage lifecycle of the customer establishment area, this is comparable to Milk products which gets bought on a frequent manner minimising the usage lifecycle of the product.

The distributions of data for Milk products, Frozen products and Delicatessen products are skewed towards the left with high concentration of product count.

A distribution with higher standard deviation will have lesser probability to find the desired product within a range and shows greater amount of outliers. Fresh products have comparatively wider distribution provides a probability distribution to deviate from the true value to a larger extent compared to the Delicatessen products which provides a probability distribution formed from hypothesis to have lesser deviation from the true value,. This can be shown from Jenson's

inequality.

1.5 Data Preprocessing

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

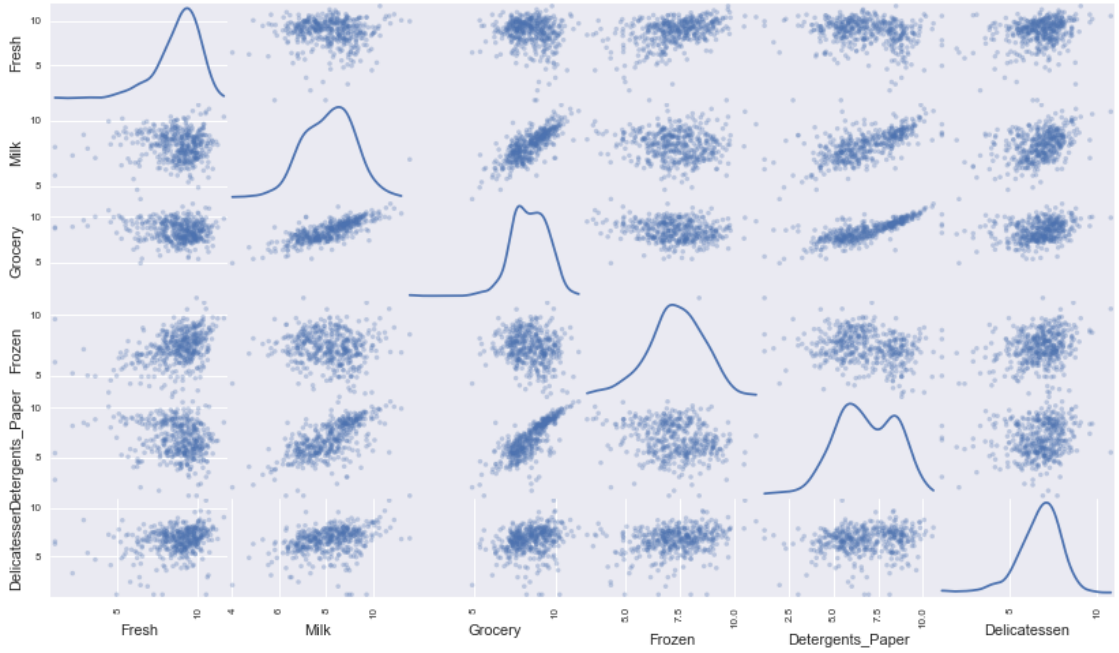
1.5.1 Implementation: Feature Scaling

If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test, which calculates the best power transformation of the data that reduces skewness. A simpler approach which can work in most cases would be applying the natural logarithm.

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

```
In [18]: # TODO: Scale the data using the natural logarithm
         log_data = np.log(imported_data)

         # TODO: Scale the sample data using the natural logarithm
         sample_data = pd.DataFrame(imported_data.loc[indices], columns = imported_data.keys()
         log_samples = np.log(sample_data)

         # Produce a scatter matrix for each pair of newly-transformed features
         pd.scatter_matrix(log_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
```

```
C:\Users\Aswin\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: FutureWarning: pandas.scatt
  if __name__ == '__main__':
```



1.5.2 Observation

After applying a natural logarithm scaling to the data, the distribution of each feature should appear much more normal. For any pairs of features you may have identified earlier as being correlated, observe here whether that correlation is still present (and whether it is now stronger or weaker than before).

Run the code below to see how the sample data has changed after having the natural logarithm applied to it.

```
In [20]: # Display the log-transformed sample data
         display(log_samples)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
1	8.756682	9.083416	8.946896	7.785305	8.165079	8.967504
2	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853

1.5.3 Implementation: Outlier Detection

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

In the code block below, you will need to implement the following: - Assign the value of the 25th percentile for the given feature to `Q1`. Use `np.percentile` for this. - Assign the value of the 75th percentile for the given feature to `Q3`. Again, use `np.percentile`. - Assign the calculation of an outlier step for the given feature to `step`. - Optionally remove data points from the dataset by adding indices to the `outliers` list.

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

```
In [35]: # 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 i
             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 + s

         # OPTIONAL: Select the indices for data points you wish to remove
         outliers  = [65, 154, 304, 86, 183, 325, 86, 338, 95]

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

         display(good_data)
```

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
65	4.442651	9.950323	10.732651	3.583519	10.095388	7.260523
66	2.197225	7.335634	8.911530	5.164786	8.151333	3.295837
81	5.389072	9.163249	9.575192	5.645447	8.964184	5.049856
95	1.098612	7.979339	8.740657	6.086775	5.407172	6.563856
96	3.135494	7.869402	9.001839	4.976734	8.262043	5.379897
128	4.941642	9.087834	8.248791	4.955827	6.967909	1.098612
171	5.298317	10.160530	9.894245	6.478510	9.079434	8.740337
193	5.192957	8.156223	9.917982	6.865891	8.633731	6.501290
218	2.890372	8.923191	9.629380	7.158514	8.475746	8.759669
304	5.081404	8.917311	10.117510	6.424869	9.374413	7.787382

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

Data points considered outliers for the feature 'Milk':

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

Data points considered outliers for the feature 'Grocery':

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

Data points considered outliers for the feature 'Frozen':

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

Data points considered outliers for the feature 'Detergents_Paper':

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

Data points considered outliers for the feature 'Delicatessen':

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
2	8.756682	9.083416	8.946896	7.785305	8.165079	8.967504
3	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853
4	10.026369	8.596004	8.881558	8.272571	7.482682	8.553525
5	9.149847	9.019059	8.542081	6.501290	7.492760	7.280008
6	9.403107	8.070594	8.850088	6.173786	8.051978	6.300786
7	8.933137	8.508354	9.151227	7.419980	8.108021	7.850104
8	8.693329	8.201934	8.731013	6.052089	7.447751	6.620073
9	8.700514	9.314070	9.845911	7.055313	8.912608	7.648740
10	8.121480	8.594710	9.470703	8.389360	8.695674	7.463937

11	9.483873	7.024649	8.416931	7.258412	6.308098	6.208590
12	10.364514	9.418898	9.372204	5.659482	8.263848	7.983099
13	9.962558	8.733594	9.614605	8.037543	8.810907	6.400257
14	10.112654	9.155356	9.400217	5.683580	8.528726	7.681560
15	9.235326	7.015712	8.248267	5.983936	6.871091	6.021023
16	6.927558	9.084324	9.402695	4.897840	8.413609	6.984716
17	8.678632	8.725345	7.983781	6.732211	5.913503	8.406932
18	9.830971	8.752581	9.220192	7.698483	7.925519	8.064951
19	8.959312	7.822044	9.155250	6.505784	7.831220	6.216606
20	9.772581	8.416046	8.434246	6.971669	7.722678	7.661056
21	8.624612	6.769642	7.605890	8.126518	5.926926	6.343880
22	10.350606	7.558517	8.404920	9.149316	7.775276	8.374246
23	10.180096	10.502956	9.999661	8.547528	8.374938	9.712509
24	10.027783	9.187686	9.531844	7.977625	8.407825	8.661813
25	9.690604	8.349957	8.935245	5.303305	8.294799	4.043051
26	9.200088	6.867974	7.958926	8.055475	5.488938	6.725034
27	9.566335	6.688355	8.021256	6.184149	4.605170	6.249975
28	8.321908	9.927399	10.164197	7.054450	9.059982	8.557567
29	10.671000	7.649693	7.866722	7.090077	7.009409	6.712956
..	...	...	...	...	...	...
402	8.799812	7.647786	8.425736	7.236339	7.528332	7.545390
403	7.661998	8.098339	8.095904	7.336286	5.459586	8.381373
404	4.574711	8.190077	9.425452	4.584967	7.996317	4.127134
405	8.513787	8.488588	8.799812	9.790655	6.815640	7.797702
406	8.694335	7.595890	8.136518	8.644530	7.034388	5.669881
407	8.967249	8.707152	9.053920	7.433075	8.171882	7.535830
408	8.386857	9.300181	9.297252	6.742881	8.814033	6.900731
409	8.530109	8.612322	9.310638	5.897154	8.156223	6.968850
410	6.492240	9.047115	9.832099	4.890349	8.815815	6.654153
411	9.089415	8.238273	7.706613	6.450470	7.365180	7.327123
412	8.402007	8.569026	9.490015	3.218876	8.827321	7.239215
413	9.744668	8.486115	9.110851	6.938284	8.135933	7.486613
414	10.181119	7.227662	8.336151	6.721426	6.854355	7.104965
415	9.773664	8.212297	8.446127	6.965080	7.497207	6.504288
416	9.739791	7.966933	9.411811	6.773080	8.074960	5.517453
417	9.327501	7.786552	7.860571	9.638740	4.682131	7.542213
418	9.482960	9.142811	9.569133	8.052296	8.532870	7.546446
419	10.342130	9.722385	8.599510	9.621257	6.084499	7.058758
420	8.021913	8.694502	8.499029	7.695303	6.745236	5.758902
421	9.060331	7.467371	8.183118	3.850148	4.430817	7.824446
422	8.038189	8.349957	9.710085	6.354370	5.484797	7.640123
423	9.051696	8.613594	8.548692	9.509407	7.227662	7.311886
424	9.957834	7.057898	8.466742	5.594711	7.191429	5.978886
425	7.591862	8.076515	7.308543	7.340187	5.874931	7.278629
426	9.725019	8.274357	8.986447	6.533789	7.771067	6.731018
427	10.299003	9.396903	9.682030	9.483036	5.204007	7.698029
428	10.577146	7.266129	6.638568	8.414052	4.532599	7.760467
429	9.584040	9.647821	10.317020	6.079933	9.605149	7.532088

```
430   9.238928   7.591357   7.710653  6.945051          5.123964      7.661527
431   7.932721   7.437206   7.828038  4.174387          6.167516      3.951244

[432 rows x 6 columns]
```

1.5.4 Question 4

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

Answer:

The data point with row number 154, is considered as outliers for features like Milk, Grocery and Delicatessen. This data point must be removed from the dataset as compared with other rows in the dataset because it has got low customer spending for each of the features. The frozen data point with row number 65 is containing a high value of Detergents_Paper and low values for Fresh and Frozen products making it an extreme case for data analysis. The outlier values must not be included within the dataset which is used for predicting, because if the same region and same channel has a low customer spending value that will affect the mean and standard deviation of the distribution.

The data point 95 has been considered because the statistical description has revealed that the values are below in comparison with the other values.

The data point 338 has been removed because it shows variation in the values for different components and will nto satisfy the requirements for PCA.

The data point 86 show marginally high values in the log distribution, hence has been removed, which is the same case for 325. The data point 304 shows distinctive values for each components, hence renmoved to best fit the PCA model.

1.6 Feature Transformation

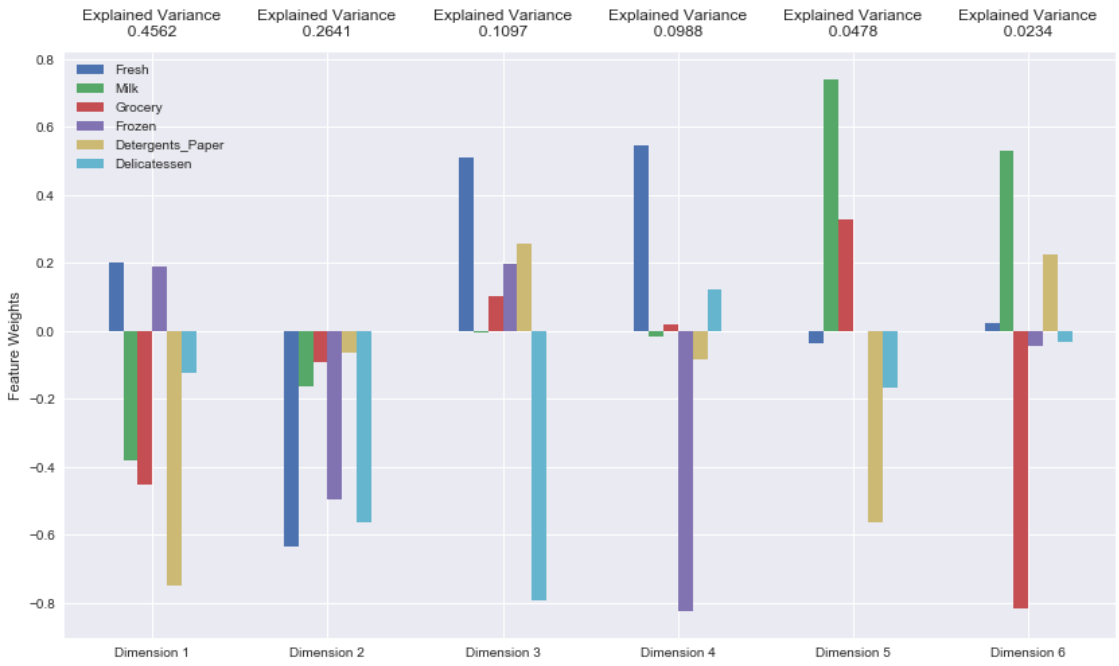
In this section you will use principal component analysis (PCA) to draw conclusions about the underlying structure of the wholesale customer data. Since using PCA on a dataset calculates the dimensions which best maximize variance, we will find which compound combinations of features best describe customers.

1.6.1 Implementation: PCA

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

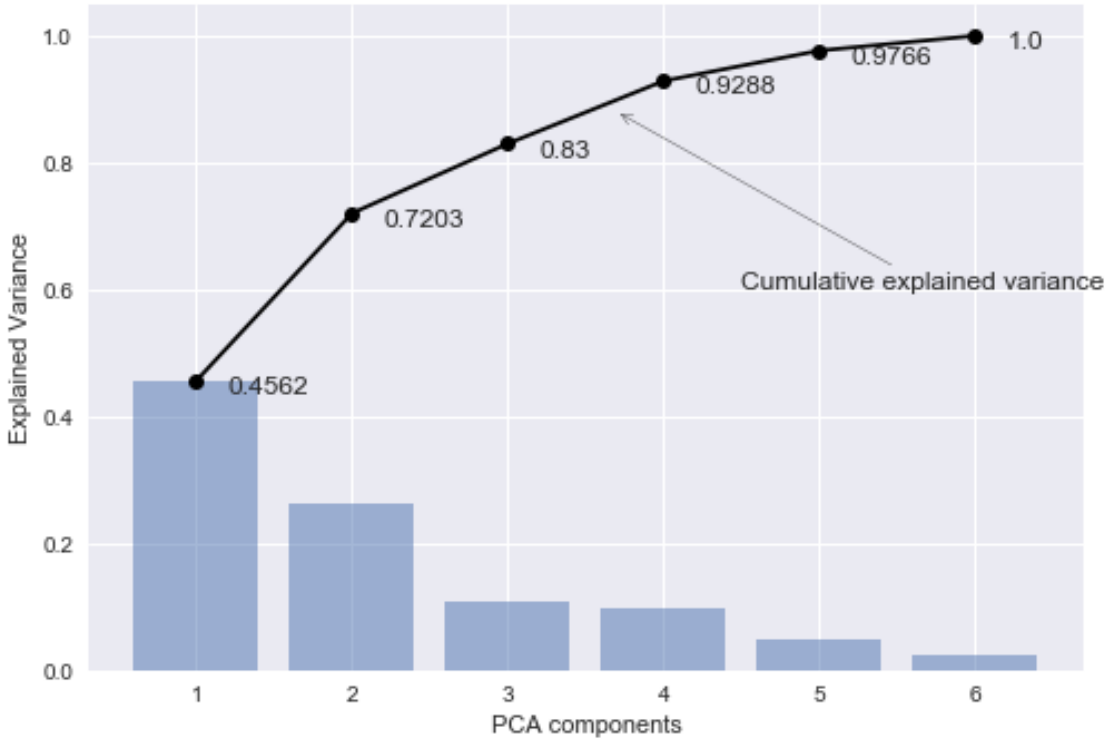
In the code block below, you will need to implement the following: - Import `sklearn.decomposition.PCA` and assign the results of fitting PCA in six dimensions with `good_data` to `pca`. - Apply a PCA transformation of `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

In [36]:
```
from sklearn.decomposition import PCA

# TODO: Apply PCA by fitting the good data with the same number of dimensions as feat
pca = PCA(n_components=6)
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)
```



In [37]:
```
# create an x-axis variable for each pca component
x = np.arange(1,7)

# plot the cumulative variance
plt.plot(x, np.cumsum(pca.explained_variance_ratio_), '-o', color='black')

# plot the components' variance
plt.bar(x, pca.explained_variance_ratio_, align='center', alpha=0.5)

# plot styling
plt.ylim(0, 1.05)

plt.annotate('Cumulative explained variance', xy=(3.7, .88), arrowprops=dict(arrowsty
```

```
for i,j in zip(x, np.cumsum(pca.explained_variance_ratio_)):
    plt.annotate(str(j.round(4)),xy=(i+.2,j-.02))

plt.xticks(range(1,7))
plt.xlabel('PCA components')
plt.ylabel('Explained Variance')
plt.show()
```



1.6.2 Question 5

*How much variance in the data is explained **in total** by the first and second principal component? What about the first four principal components? Using the visualization provided above, discuss what the first four dimensions best represent in terms of customer spending.*

Hint: A positive increase in a specific dimension corresponds with an *increase* of the *positive-weighted* features and a *decrease* of the *negative-weighted* features. The rate of increase or decrease is based on the indivdual feature weights.

Answer:

The first and second principal components show a large variance in data. The good data that have been deduced for Fresh products, indicates that fresh and frozen products are commonly used products. The customer spending for positive feature weights such as Detergents where Fresh products are taken into consideration, shows a very high concentration but very low dependency between the features. Access to customer establishment to such areas must be optmised for usage of Grocery, Detergents_Paper and Milk products.

Dimension 1: Since the logarithmic scaling with Fresh products as the principal component shows a strong scatter matrix of Frozen products within the valid range, a positive increase of Fresh products in 1st principal component will improve the weights of Fresh and Frozen products. Whereas a positive increase of Fresh products will show a drastic negative increase in customer spending of detergents & paper products as the ratio of weights is higher.

Dimension 2: A positive increase of Milk products in 2nd principal component will show a negative increase in the customer spending of all the products. Since the graph on the logarithmic scaling with Milk as the principal component shows a satisfying range of concentration of products, the greater the milk products are bought for that customer establishment, the lesser should be the growth rate of stocking other products.

Dimension 3: The logarithmic scaling of Fresh as the principal component and Delicatessen as the Principal component shows similar graphs. The actual distribution of Fresh and Delicatessen products show a relatively narrow deviation. The pricing of the Fresh and Delicatessen are proportional as the feature weights are equivalent. It can be deduced that increase of the feature will result in drastic increase in negative weighted feature of Fresh products from a proper Support Vector Machine.

Dimension 4: The fourth dimension of PCA component analysis shows Frozen as the most weighted feature. While decreasing the most weighted feature, the customer spending rate will decrease.

1.6.3 Observation

Run the code below to see how the log-transformed sample data has changed after having a PCA transformation applied to it in six dimensions. Observe the numerical value for the first four dimensions of the sample points. Consider if this is consistent with your initial interpretation of the sample points.

```
In [38]: # Display sample log-data after having a PCA transformation applied
         display(pd.DataFrame(np.round(pca_samples, 4), columns = pca_results.index.values))
```

```
   Dimension 1  Dimension 2  Dimension 3  Dimension 4  Dimension 5  \
0      -1.7750      -0.9284      -0.1696      -0.1114       0.1658
1      -1.8270      -1.8173      -1.3416      -0.2496      -0.2673
2       1.1665      -1.4310      -0.1513      -0.6482      -0.6355

   Dimension 6
0       0.2289
1       0.3002
2      -0.6804
```

1.6.4 Implementation: Dimensionality Reduction

When using principal component analysis, one of the main goals is to reduce the dimensionality of the data — in effect, reducing the complexity of the problem. Dimensionality reduction comes at a cost: Fewer dimensions used implies less of the total variance in the data is being explained. Because of this, the *cumulative explained variance ratio* is extremely important for knowing how many dimensions are necessary for the problem. Additionally, if a signifiant amount of variance is explained by only two or three dimensions, the reduced data can be visualized afterwards.

In the code block below, you will need to implement the following: - Assign the results of fitting PCA in two dimensions with `good_data` to `pca`. - Apply a PCA transformation of `good_data` using `pca.transform`, and assign the results to `reduced_data`. - Apply a PCA transformation of `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

```
In [41]: # 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'])
```

1.6.5 Observation

Run the code below to see how the log-transformed sample data has changed after having a PCA transformation applied to it using only two dimensions. Observe how the values for the first two dimensions remains unchanged when compared to a PCA transformation in six dimensions.

```
In [42]: # 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	-1.7750	-0.9284
1	-1.8270	-1.8173
2	1.1665	-1.4310

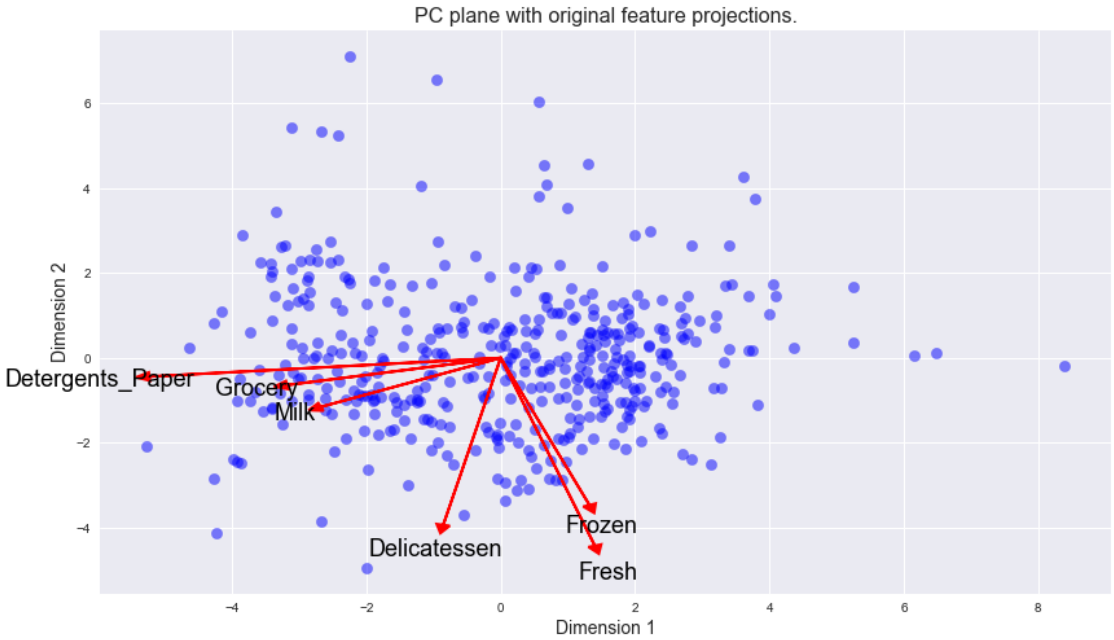
1.7 Visualizing a Biplot

A biplot is a scatterplot where each data point is represented by its scores along the principal components. The axes are the principal components (in this case `Dimension 1` and `Dimension 2`). In addition, the biplot shows the projection of the original features along the components. A biplot can help us interpret the reduced dimensions of the data, and discover relationships between the principal components and original features.

Run the code cell below to produce a biplot of the reduced-dimension data.

```
In [43]: # Create a biplot
         vs.biplot(good_data, reduced_data, pca)
```

```
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1763a9edc50>
```



1.7.1 Observation

Once we have the original feature projections (in red), it is easier to interpret the relative position of each data point in the scatterplot. For instance, a point the lower right corner of the figure will likely correspond to a customer that spends a lot on `'Milk'`, `'Grocery'` and `'Detergents_Paper'`, but not so much on the other product categories.

From the biplot, which of the original features are most strongly correlated with the first component? What about those that are associated with the second component? Do these observations agree with the pca_results plot you obtained earlier?

1.8 Clustering

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

1.8.1 Question 6

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

Answer:

K-Means clustering algorithm places the centroids at the mean of each cluster. It provides a better alternative to Support Vector Machines by providing distortion to the cluster based on the centroid values.

Gaussian Mixture Model provides the maximum likelihood estimate by considering the probabilities using the parameter space and weights assigned to the feature data.

K-Means clustering algorithm will be beneficial to consider the observations based on the Region and Channel data.

Gaussian Mixture Model will be beneficial to consider the PCA component analysis, based on each features. The parameters that can vary here will be explained variance and the feature weights.

1.8.2 Implementation: Creating Clusters

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

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

In [44]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# TODO: Apply your clustering algorithm of choice to the reduced data
clusterer_1 = KMeans(n_clusters=2)
clusterer_1.fit(reduced_data)

# TODO: Predict the cluster for each data point
preds_1 = clusterer_1.predict(reduced_data)

# TODO: Find the cluster centers
centers_1 = clusterer_1.cluster_centers_

# TODO: Predict the cluster for each transformed sample data point
sample_preds = clusterer_1.predict(pca_samples)

# TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
score_1 = silhouette_score(reduced_data, preds_1)

## __________

clusterer_2 = KMeans(n_clusters=3)
clusterer_2.fit(reduced_data)
```

```
        # TODO: Predict the cluster for each data point
        preds_2 = clusterer_2.predict(reduced_data)

        # TODO: Find the cluster centers
        centers_2 = clusterer_2.cluster_centers_

        # TODO: Predict the cluster for each transformed sample data point
        sample_preds_2 = clusterer_2.predict(pca_samples)

        # TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
        score_2 = silhouette_score(reduced_data, preds_2)

        ## ____________________

        clusterer_3 = KMeans(n_clusters=4)
        clusterer_3.fit(reduced_data)

        # TODO: Predict the cluster for each data point
        preds_3 = clusterer_3.predict(reduced_data)

        # TODO: Find the cluster centers
        centers_3 = clusterer_3.cluster_centers_

        # TODO: Predict the cluster for each transformed sample data point
        sample_preds_3 = clusterer_3.predict(pca_samples)

        # TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
        score_3 = silhouette_score(reduced_data, preds_3)

        ## _____________

        clusterer_4 = KMeans(n_clusters=5)
        clusterer_4.fit(reduced_data)

        # TODO: Predict the cluster for each data point
        preds_4 = clusterer_4.predict(reduced_data)

        # TODO: Find the cluster centers
        centers_4 = clusterer_4.cluster_centers_

        # TODO: Predict the cluster for each transformed sample data point
        sample_preds_4 = clusterer_4.predict(pca_samples)

        # TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
        score_4 = silhouette_score(reduced_data, preds_4)
```

```
In [45]: print(centers_1)
```

```
        print(score_1)

        print("__________")

        print(centers_2)

        print(score_2)

        print("__________")

        print(centers_3)

        print(score_3)

        print("__________")

        print(centers_4)

        print(score_4)
```

```
[[ 1.48619949 -0.08634174]
 [-2.2036751   0.12802396]]
0.427931848896
----------
[[-1.81755592  2.174526  ]
 [ 1.73593146 -0.02814806]
 [-1.71554581 -1.16906067]]
0.40050514804
----------
[[ 0.8741492  -1.0463113 ]
 [-2.15501995 -0.82242389]
 [-2.12015852  2.40694604]
 [ 2.22275954  0.83831123]]
0.335060119397
----------
[[ 1.14927076 -1.47165338]
 [-2.21719593 -0.88882133]
 [ 3.30215266  0.87517999]
 [ 0.79780223  0.61529195]
 [-2.31401408  2.51396947]]
0.353520218629
```

1.8.3 Question 7

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

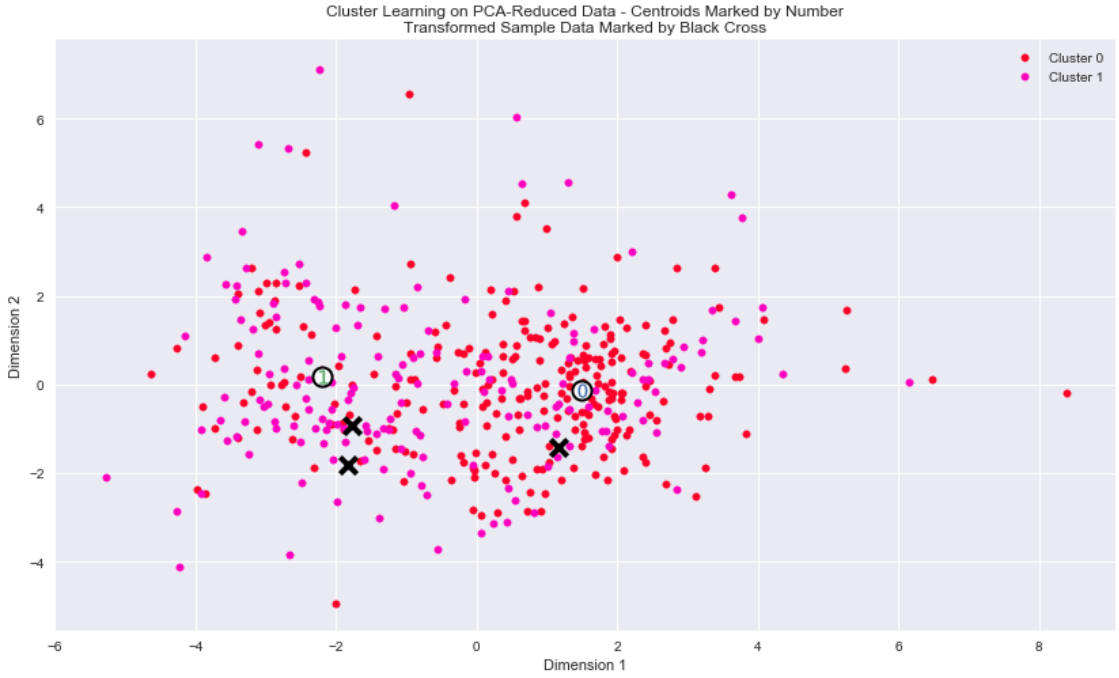
Answer:

Out of these several cluster numbers, the KMeans with number of clusters = 2 has the best silhouette score and the score I obtained for 2 clusters is 0.427931848896.

1.8.4 Cluster Visualization

Once you've chosen the optimal number of clusters for your clustering algorithm using the scoring metric above, you can now visualize the results by executing the code block below. Note that, for experimentation purposes, you are welcome to adjust the number of clusters for your clustering algorithm to see various visualizations. The final visualization provided should, however, correspond with the optimal number of clusters.

```
In [46]: # Display the results of the clustering from implementation
         vs.cluster_results(reduced_data, preds, centers, pca_samples)
```



1.8.5 Implementation: Data Recovery

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

In the code block below, you will need to implement the following: - Apply the inverse transform to `centers` using `pca.inverse_transform` and assign the new centers to `log_centers`. -

Apply the inverse function of `np.log` to `log_centers` using `np.exp` and assign the true centers to `true_centers`.

```
In [51]: # 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 = imported_data.keys())
         true_centers.index = segments
         display(true_centers)

         print(reduced_data.describe())

         data.describe()
```

```
              Fresh    Milk  Grocery  Frozen  Detergents_Paper  Delicatessen
Segment 0  9476.0  1928.0   2352.0  2090.0             295.0         697.0
Segment 1  3719.0  7504.0  12195.0   893.0            4545.0         918.0


        Dimension 1   Dimension 2
count  4.320000e+02  4.320000e+02
mean   5.181041e-16  1.536837e-15
std    2.157210e+00  1.641240e+00
min   -5.276173e+00 -4.940825e+00
25%   -1.829163e+00 -1.010392e+00
50%    2.399447e-01 -8.081434e-02
75%    1.583926e+00  8.692281e-01
max    8.394764e+00  7.106103e+00
```

Out[51]:

	Channel	Region	Fresh	Milk	Grocery
count	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829
min	1.000000	1.000000	3.000000	55.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000

	Frozen	Detergents_Paper	Delicatessen
count	440.000000	440.000000	440.000000
mean	3071.931818	2881.493182	1524.870455
std	4854.673333	4767.854448	2820.105937

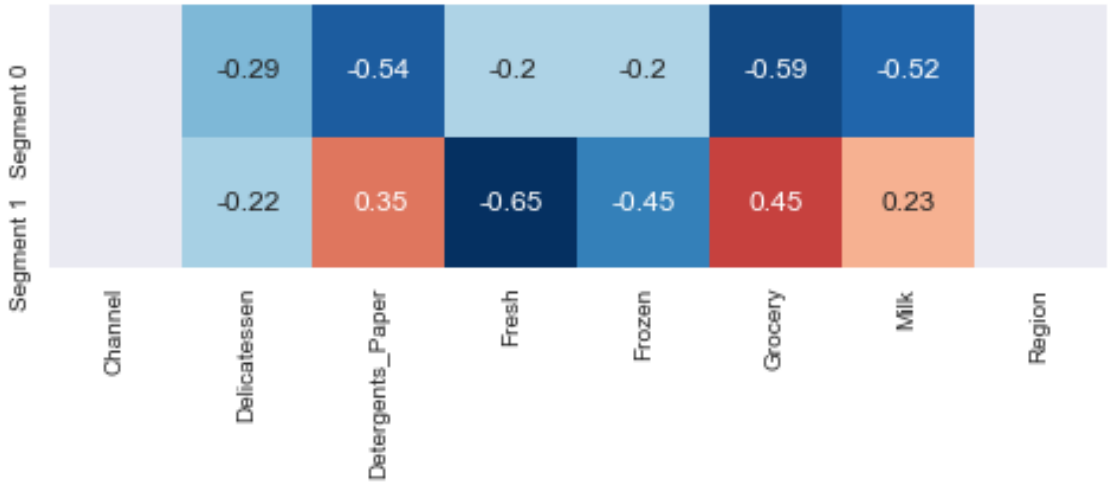
min	25.000000	3.000000	3.000000
25%	742.250000	256.750000	408.250000
50%	1526.000000	816.500000	965.500000
75%	3554.250000	3922.000000	1820.250000
max	60869.000000	40827.000000	47943.000000

1.8.6 Question 8

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

```
In [52]: sns.heatmap((true_centers-data.mean())/data.std(ddof=1), annot=True, cbar=False, squa
```

```
Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1763ac81ba8>
```



Answer:

The values of Segment 0 are within one Standard Deviation from the Mean and all the values aree at the left hand side of the distribution. For the Segment 1, the values of Groceery and Detergents_Paper are above the mean value.

It is interesting to note that, none of the segments are within the 1st Quartile. But Milk products, frozen, dtergents and paper and delicatessen for the Segment 0 are within the 2nd quartile.

The Tukey's method of outliers has enabled the removal of extreme cases which are having below 1st quartile and above 3rd quartile based on the step rule.

The Segment 1 characteristics are related to Milk, Grocery and detergents and Paper as thety are above the 3rd quartile of the original data.

Segment 0 indicates that Grocery products have a higher expectation for the customer in the customer establishment area. The Frozen products have relatively low selling potential in this segment. This segment has given importance to Detergents_Paper, Fresh and Milk products, and

Grocery indicating that delivery service has to be planned for large stores, industrial sectors and markets.

Segment 1 indicates that Fresh products gets sold more often, with very low margin for Delicatessen products. This segment has segmented based on the acceptance of fresh products but have kept Detergents_Paper category as a low potential category keeping the other products in a similar profile.

1.8.7 Question 9

For each sample point, which customer segment from ***Question 8*** *best represents it? Are the predictions for each sample point consistent with this?*

Run the code block below to find which cluster each sample point is predicted to be.

```
In [54]: # 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 1
Sample point 2 predicted to be in Cluster 0
```

Answer:

Each of the points which pca_samples contains are mapping exactly with Clusters obtained using the K-Means prediction algorithm.

1.9 Conclusion

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

1.9.1 Question 10

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

Hint: Can we assume the change affects all customers equally? How can we determine which group of customers it affects the most?

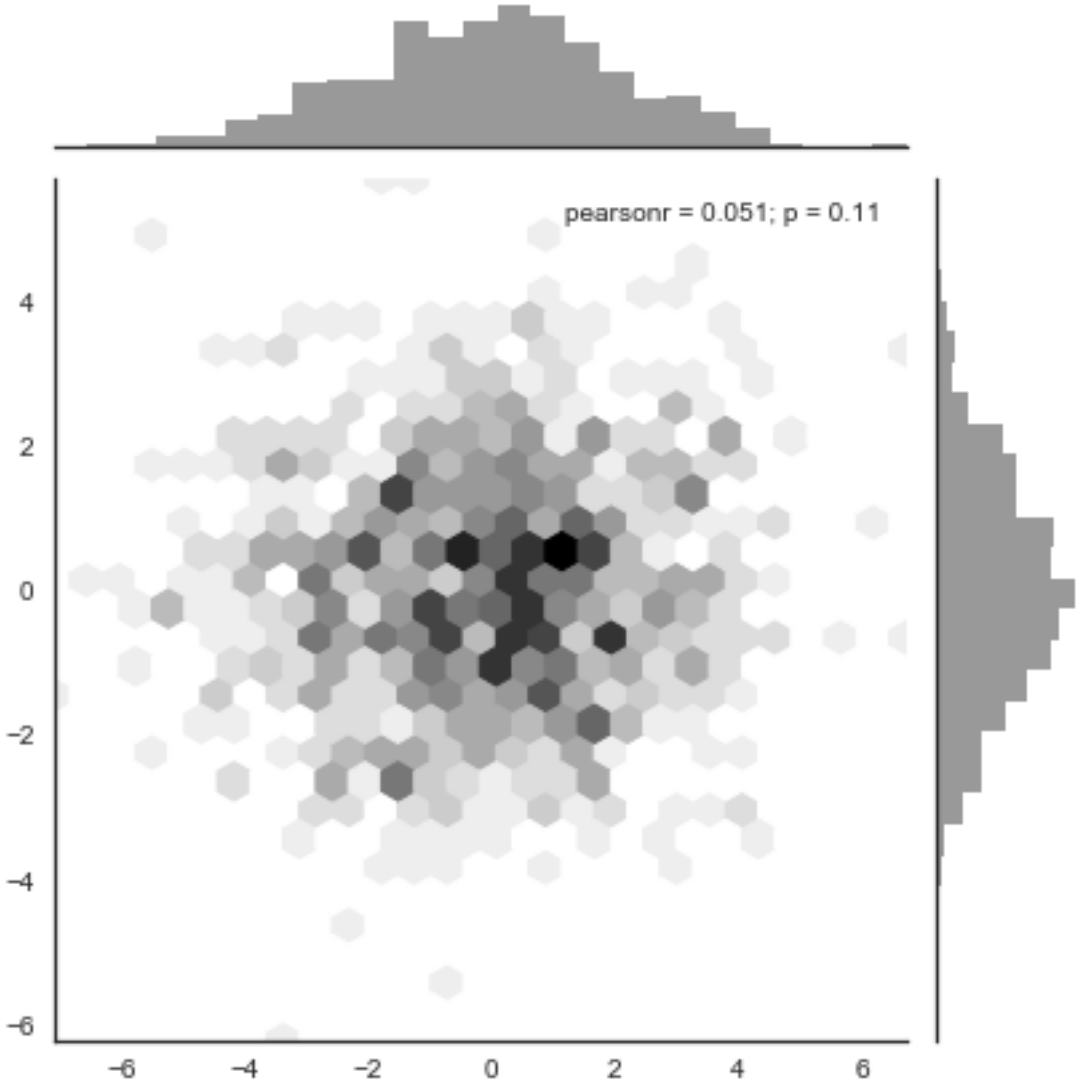
```
In [58]: display(reduced_data.describe())

         display(pd.DataFrame(np.round(reduced_data, 4), columns = ['Dimension 1', 'Dimension 2
```

```
        x, y = np.random.multivariate_normal(reduced_data.mean(), reduced_data.cov(), 1000).T
        with sns.axes_style("white"):
            sns.jointplot(x=x, y=y, kind="hex", color="k")
```

```
        Dimension 1   Dimension 2
count  4.320000e+02  4.320000e+02
mean   5.181041e-16  1.536837e-15
std    2.157210e+00  1.641240e+00
min   -5.276173e+00 -4.940825e+00
25%   -1.829163e+00 -1.010392e+00
50%    2.399447e-01 -8.081434e-02
75%    1.583926e+00  8.692281e-01
max    8.394764e+00  7.106103e+00
```

```
     Dimension 1  Dimension 2
0        -1.7528      -0.0580
1        -1.7750      -0.9284
2        -1.8270      -1.8173
3         1.1665      -1.4310
4        -0.7050      -2.4992
5        -1.0743      -0.3917
6        -1.1607       0.2539
7        -1.5545      -1.0401
8        -0.9096       0.6104
9        -2.8692      -0.8461
10       -2.1037      -0.8703
11        0.9698       0.0391
12       -2.1810      -1.3279
13       -1.8761      -1.3050
14       -2.2999      -0.9869
15        0.3620       0.9142
16       -2.8885       1.8256
17        0.2822      -0.6432
18       -1.3369      -1.9013
19       -1.0552       0.4443
20       -0.7998      -1.1376
21        1.6951       0.2162
22       -0.0586      -2.8413
23       -2.6661      -3.8563
24       -1.9855      -2.6299
25       -1.3184       1.7033
26        1.8797      -0.3475
27        2.3582       0.6940
28       -3.5449      -1.2566
29        0.6011      -1.0082
..           ...          ...
```

```
402      -0.4878      -0.4492
403       0.7236      -0.1638
404      -2.4264       5.2453
405      -0.0527      -1.8030
406       0.5057       0.0404
407      -1.5864      -0.9194
408      -2.5717      -0.0141
409      -1.9629       0.4293
410      -3.4175       2.2319
411      -0.3291      -0.1419
412      -3.0929       1.6330
413      -1.4327      -1.1046
414       0.4497      -0.7000
415      -0.4187      -0.4381
416      -1.1170       0.1492
417       2.4002      -1.7631
418      -2.0374      -1.6994
419       0.5368      -2.5984
420      -0.1872       0.6894
421       1.3834       1.1522
422      -0.1431       0.3109
423      -0.1803      -1.7541
424       0.0839       0.6269
425       0.9003       0.5478
426      -1.0108      -0.3980
427       0.7149      -2.8519
428       3.2574      -1.8625
429      -3.7205      -0.9974
430       1.6723      -0.4223
431       0.5676       3.8099

[432 rows x 2 columns]
```



Answer:

The above plot determines the dimensionality reduction values obtained from PCA. The highly correlated positive values of PCA determines those values of 'Dimension 1' and negative highly correlated values of PCA determines that of 'Dimension 2'.

Out of the 6 product categories, the change in delivery service affects all the customers depending on the customer establishment areas that are belonging to the particular region and channel. Each of the customer establishment areas must be segmented based on the customer spending. Factor analysis is a detailed method that can be applied alongside PCA analysis.

The 'Dimension 1' which consists of: Fresh and Frozen products are fixed candidates for A/B Testing implying the other candidates can be varied in this region

The 'Dimension 2' which consists of: Fresh and Frozen products are fixed candidates fo A/B Testing.

Covariance is used for A/B Tests for measuring the right variations of products or services. The joint plot provided here correlates the quantities for Dimension 1 and Dimension 2. From this plot

it can be determined that the candidates for A/B Testing could be Delicatessen, Detergent & Paper, Grocery and Milk such that varying them would not affect the buying behaviour much and can be used for testing purposes.

A/B Testing is a process which considers several outcomes by altering the parameters by affecting the buying behaviour and runs hypotheses on these outcomes, in order to obtain a model which is put to test to measure the buying behaviour. The measurements of changes in specifications of the product such as including larger portions of Milk and larger portions of Grocery in one section would identify potential sources of improving the buying behaviour.

A/B Testing depends on channels of distribution of products distributed to the customers at the front facing site and they are variables relevant for creating an optimised model.

1.9.2 Question 11

Additional structure is derived from originally unlabeled data when using clustering techniques. Since each customer has a ***customer segment*** it best identifies with (depending on the clustering algorithm applied), we can consider *'customer segment'* as an **engineered feature** for the data. Assume the wholesale distributor recently acquired ten new customers and each provided estimates for anticipated annual spending of each product category. Knowing these estimates, the wholesale distributor wants to classify each new customer to a ***customer segment*** to determine the most appropriate delivery service.
How can the wholesale distributor label the new customers using only their estimated product spending and the ***customer segment*** *data?*
Hint: A supervised learner could be used to train on the original customers. What would be the target variable?

```
In [60]: print("Sample Data: ")

         display(sample_data)

         print("Log Samples: ")

         display(log_samples)

         print("Sample Predicted Clusters")

         display(sample_preds)
```

```
Sample Data:
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	7057	9810	9568	1762	3293	1776
1	6353	8808	7684	2405	3516	7844
2	13265	1196	4221	6404	507	1788

```
Log Samples:
```

```
      Fresh      Milk   Grocery    Frozen  Detergents_Paper  Delicatessen
0  8.861775  9.191158  9.166179  7.474205          8.099554      7.482119
1  8.756682  9.083416  8.946896  7.785305          8.165079      8.967504
2  9.492884  7.086738  8.347827  8.764678          6.228511      7.488853


Sample Predicted Clusters


array([1, 1, 0])
```

Answer:

The Fresh and Frozen products are marginally higher in the 3rd predicted sample which is in cluster 0. The segment 0 is appropriate for customer spendingn in Fresh and Frozen. The segment 1 is appropriate for Milk, Grocery, Detergents & Paper and Fresh.

In a K-Means clustering technique, sample points are selected after deciding the number of clusters.

When new customers are added to the set, they are placed based on their product spending and customer segment data. The earlier formed centroids are then passed through a distortion parameter, which then results in a new centroid formation.

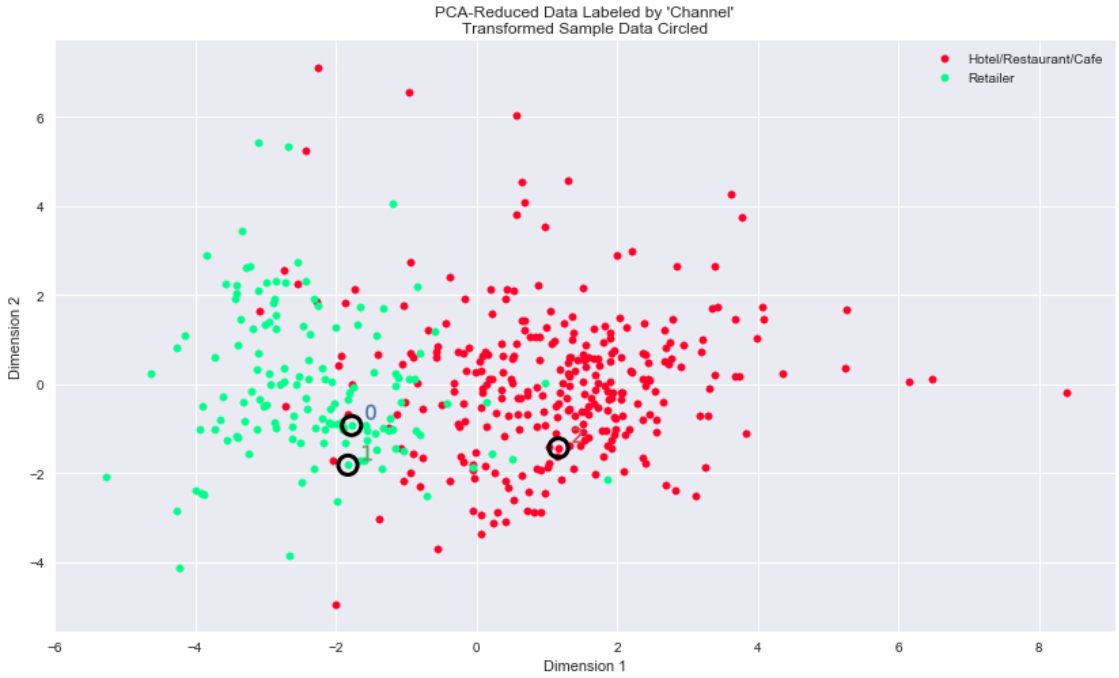
For a supervised learner, the PCA components deliver the dimension specific Feature Weights. The componments of PCA are the right target variables for the supervised learner.

1.9.3 Visualizing Underlying Distributions

At the beginning of this project, it was discussed that the `'Channel'` and `'Region'` features would be excluded from the dataset so that the customer product categories were emphasized in the analysis. By reintroducing the `'Channel'` feature to the dataset, an interesting structure emerges when considering the same PCA dimensionality reduction applied earlier to the original dataset.

Run the code block below to see how each data point is labeled either `'HoReCa'` (Hotel/Restaurant/Cafe) or `'Retail'` the reduced space. In addition, you will find the sample points are circled in the plot, which will identify their labeling.

```
In [61]: # Display the clustering results based on 'Channel' data
         vs.channel_results(reduced_data, outliers, pca_samples)
```



1.9.4 Question 12

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

Answer:

The clustering algorithm do not completely separate the clusters, as seen from the plot. The distribution does not distinguish between Hotel/restaurants/cafes and Retailers, and have several outliers. The classifications show consistency as the PCA analysis data has been obtained from transformed data as well as by removing the outliers.

Customer segments have been classified based on relative customer spending for optimum delivery of products to the customer establishment areas. Customer channels are divided into service based areas where these customer personas are suitable.

Previous definitions of cluster segments are obtained from cluster centers of the reduced data, but the channel data are additionally considered and their outliers are removed.

> **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.