Machine Learning Engineer Nanodegree

Unsupervised Learning

Project 3: Creating Customer Segments

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

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

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

Getting Started

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

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

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

In [79]:
```
# Import libraries necessary for this project
import numpy as np
import pandas as pd
import renders as rs
from IPython.display import display # Allows the use of display() for DataFram
es
# Show matplotlib plots inline (nicely formatted in the notebook)
%matplotlib inline

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

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

Data Exploration

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

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

In [80]:
```
# Display a description of the dataset
display(data.describe())
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	D
count	440.000000	440.000000	440.000000	440.000000	440.000000	4
mean	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1
std	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2
min	3.000000	55.000000	3.000000	25.000000	3.000000	3
25%	3127.750000	1533.000000	2153.000000	742.250000	256.750000	4
50%	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	9
75%	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1
max	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	4

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 [81]: # TODO: Select three indices of your choice you wish to sample from the datase
         t
         indices = [125,290,435]

         # Create a DataFrame of the chosen samples
         samples = pd.DataFrame(data.loc[indices], columns = data.keys()).reset_index(d
         rop = True)
         print "Chosen samples of wholesale customers dataset:"
         display(samples)
         display(samples - np.round(data.mean()))
         display(samples - np.round(data.median()))
```

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

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	76237	3473	7102	16538	778	918
1	2708	2160	2642	502	965	1522
2	29703	12051	16027	13135	182	2204

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	64237	-2323	-849	13466	-2103	-607
1	-9292	-3636	-5309	-2570	-1916	-3
2	17703	6255	8076	10063	-2699	679

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	67733	-154	2346	15012	-38	-48
1	-5796	-1467	-2114	-1024	149	556
2	21199	8424	11271	11609	-634	1238

Question 1

Consider the total purchase cost of each product category and the statistical description of the dataset above for your sample customers.
What kind of establishment (customer) could each of the three samples you've chosen represent?
Hint: Examples of establishments include places like markets, cafes, and retailers, among many others. Avoid using names for establishments, such as saying *"McDonalds"* when describing a sample customer as a restaurant.

Answer:

Customer 0: is most likely a restaurant business.
The spending on Fresh and Frozen is well above both the mean as well as the median. Also spending on other categories is almost at the median. This is the biggest spender of all the selected samples.

Customer 1: is most likely a cafe business.
This customer is spending least of the 3 samples selected, with spend in all categories below the mean. It is below the median in all categories as well except Detergents_Paper and Delicatessen. Looking at these numbers it is likely to be a small cafe business.

Customer 2: is most likely a retail business.
Of the 3 samples, this customer is an intermediate spender with numbers over the mean and median for all categories except Detergent_paper.

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 [82]: # TODO: Make a copy of the DataFrame, using the 'drop' function to drop the gi
         ven feature

         #feature = 'Fresh'
         #feature = 'Milk'
         feature = 'Grocery'
         #feature = 'Frozen'
         #feature = 'Detergents_Paper'
         #feature = 'Delicatessen'

         new_data = data.drop(feature, axis = 1)

         # TODO: Split the data into training and testing sets using the given feature
          as the target
         from sklearn.cross_validation import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(new_data, data[feature], t
         est_size=0.25, random_state=10)

         # TODO: Create a decision tree regressor and fit it to the training set
         from sklearn.tree import DecisionTreeRegressor
         regressor = DecisionTreeRegressor(random_state = 10)
         regressor.fit(X_train, y_train)
         #y_predict= regressor.predict(X_test)

         # TODO: Report the score of the prediction using the testing set
         #score = y_predict - y_test
         score = regressor.score(X_test, y_test, sample_weight=None)
         print score
         #Fresh -0.379170072447
         #Milk -0.442085754637
         #Grocery 0.723800832652
         #Frozen 0.0548481103591
         #Detergents_Paper 0.494381677643
         #Delicatessen -10.5626610325
```

```
0.723800832652
```

Question 2

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

Answer: I attempted to predict the first feature 'Fresh' and its predicted score is approx -0.38. But then I went on to predict scores of each of the features and 'Grocery' turned out to have a score of 0.72, indicating it's highly predictable from rest of the features, hence it will likely be not necessary for identifying customers' habits.

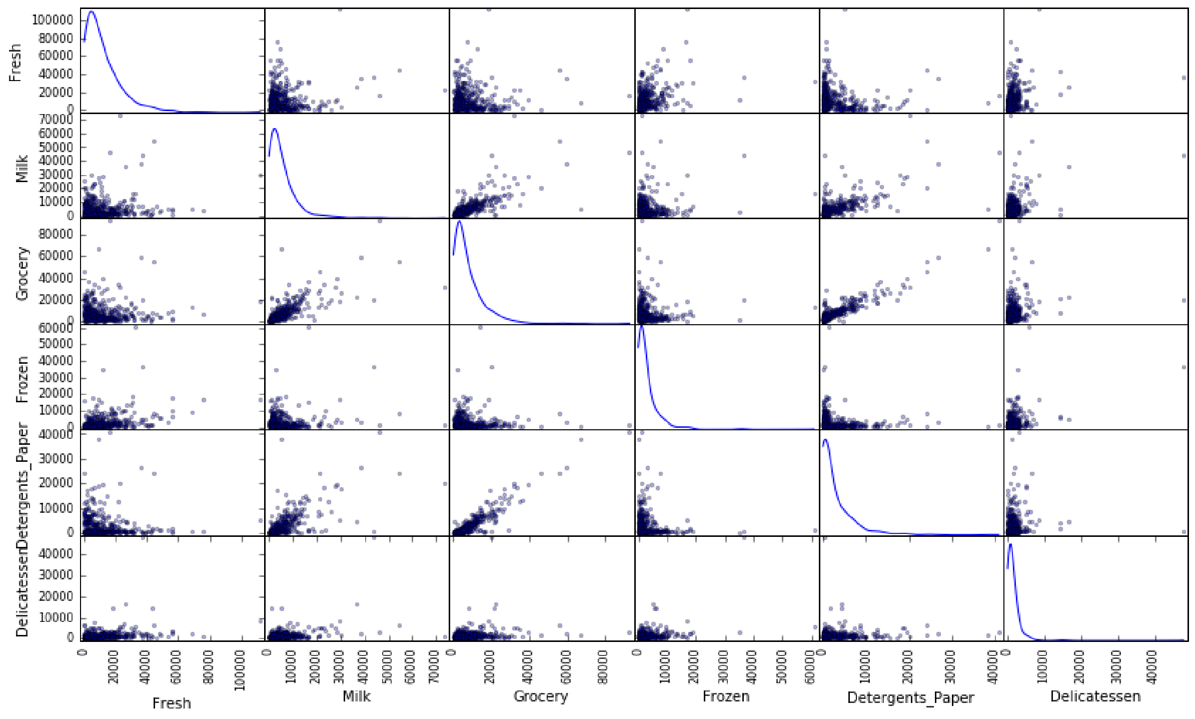
Visualize Feature Distributions

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 [83]: # Produce a scatter matrix for each pair of features in the data
         pd.scatter_matrix(data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');
         print "Skew:\n", data.skew()
         print "\nKurtosis:\n", data.kurtosis()
```

```
Skew:
Fresh                 2.561323
Milk                  4.053755
Grocery               3.587429
Frozen                5.907986
Detergents_Paper      3.631851
Delicatessen         11.151586
dtype: float64

Kurtosis:
Fresh                 11.536408
Milk                  24.669398
Grocery               20.914670
Frozen                54.689281
Detergents_Paper      19.009464
Delicatessen         170.694939
dtype: float64
```



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: I find some degree of correlation between Grocery/Detergents_Paper, Grocery/Milk and Milk/Detergents_Paper. This confirms the relevance of the feature attempted to predict. Thus the effect of Grocery in identifying customers' habits could be compensated by the rest of the features. Hence the feature Grocery might not be necessary for identifying customers' habits.

Also the data for all features is positively skewed with values greater than 2 for all, and Delicatessen being the highest. Also the kurtosis values are pretty high too with Delicatessen again being the highest. So the data are not very normally distributed for any of the features.

Data Preprocessing

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

Implementation: Feature Scaling

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

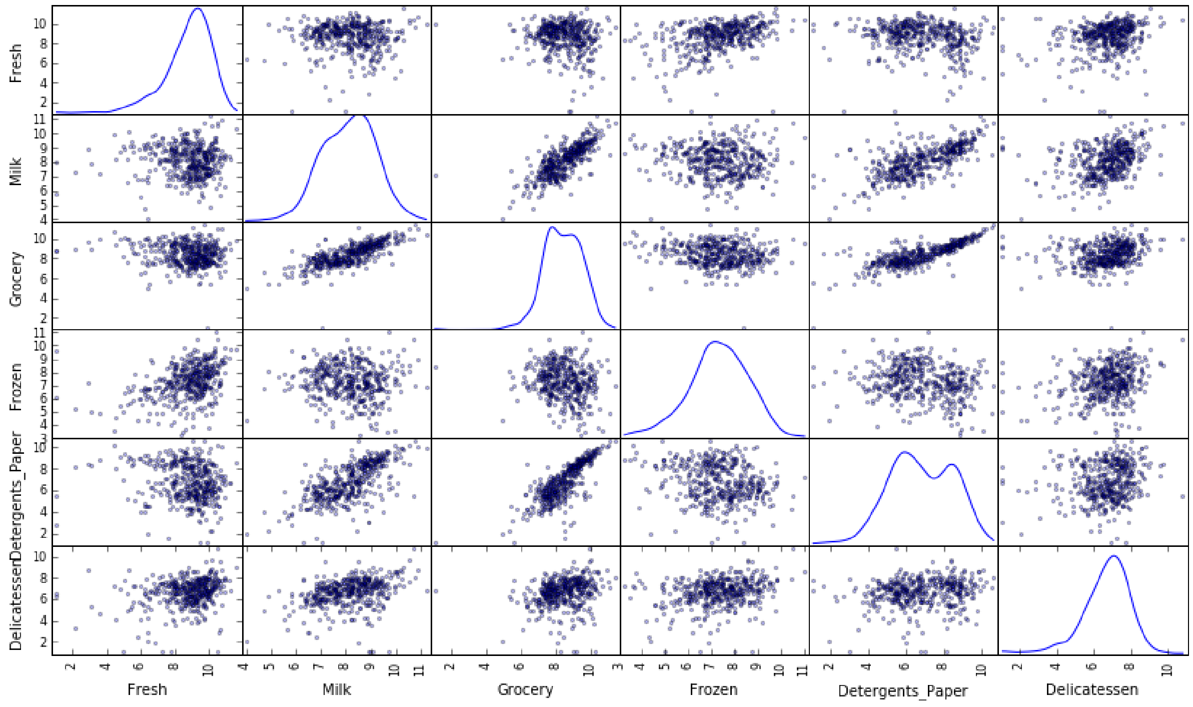
In the code block below, you will need to implement the following:

- Assign a copy of the data to `log_data` after applying a logarithm scaling. Use the `np.log` function for this.
- Assign a copy of the sample data to `log_samples` after applying a logrithm scaling. Again, use `np.log`.

```
In [84]: # TODO: Scale the data using the natural logarithm
         log_data = np.log(data)
         #print log_data
         # TODO: Scale the sample data using the natural logarithm
         log_samples = np.log(samples)
         print 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');
```

```
       Fresh      Milk   Grocery    Frozen  Detergents_Paper  Delicatessen
0  11.241602  8.152774  8.868132  9.713416          6.656727      6.822197
1   7.903966  7.677864  7.879291  6.218600          6.872128      7.327781
2  10.299003  9.396903  9.682030  9.483036          5.204007      7.698029
```



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 [85]:
```
# Display the log-transformed sample data
display(log_samples)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	11.241602	8.152774	8.868132	9.713416	6.656727	6.822197
1	7.903966	7.677864	7.879291	6.218600	6.872128	7.327781
2	10.299003	9.396903	9.682030	9.483036	5.204007	7.698029

Implementation: Outlier Detection

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

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

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

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

In [86]:
```
# For each feature find the data points with extreme high or low values
from collections import defaultdict
outliers = defaultdict(lambda: 0)

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 time
s the interquartile range)
    step = 1.5 * (Q3 - Q1)

    # Display the outliers
    print "Data points considered outliers for the feature '{}':".format(featu
re)
    outliers_df = log_data[~((log_data[feature] >= Q1 - step) & (log_data[feat
ure] <= Q3 + step))]
    for index in outliers_df.index.values:
        outliers[index] += 1
    display(outliers_df)

# OPTIONAL: Select the indices for data points you wish to remove
outliers_list = [index for (index, count) in outliers.iteritems() if count >
1]
print "Index of outliers for more than one feature: {} ".format(sorted(outlier
s_list))

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

```
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

```
Index of outliers for more than one feature: [65, 66, 75, 128, 154]
```

Question 4

Are there any data points considered outliers for more than one feature? 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: Yes, the items [65, 66, 75, 128, 154] are considered outliers for more than one feature and have been removed from the dataset.

An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism. Because an outlier is more like an exception than the norm, and can have an impact on the clustering results by chosing incorrect centroids of clusters, they are removed - there by making clustering results more reliable.

Feature Transformation

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

Implementation: PCA

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

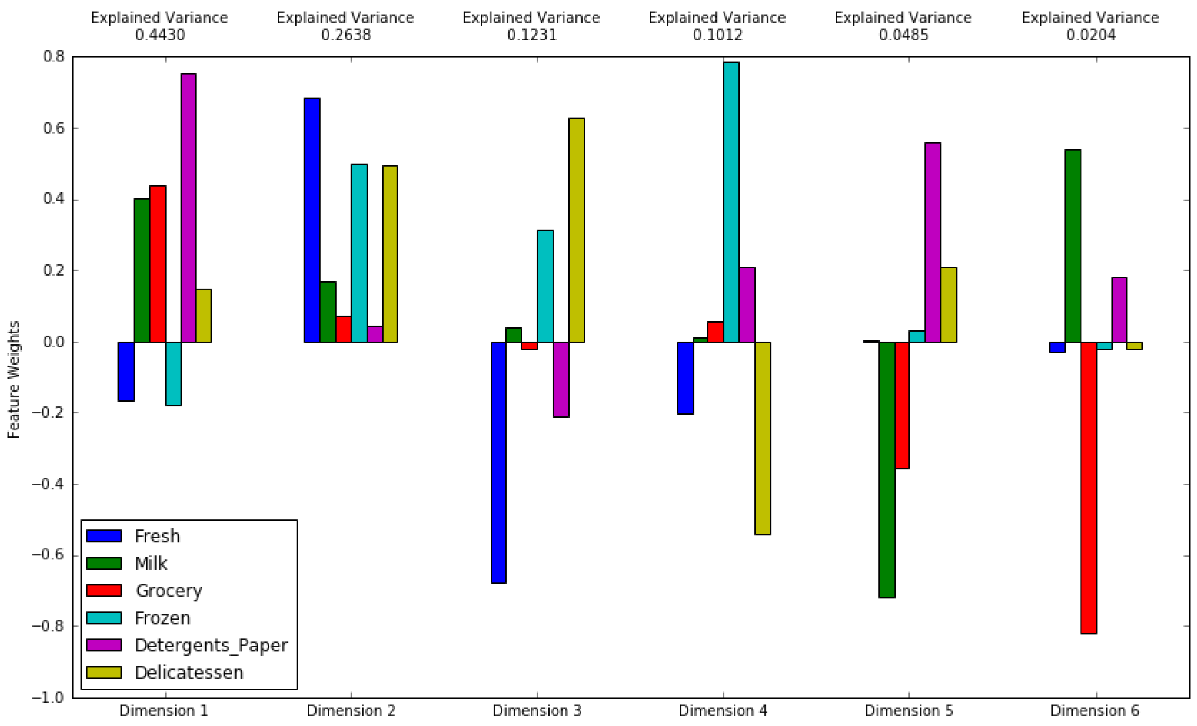
In the code block below, you will need to implement the following:

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

```
In [87]: from sklearn.decomposition import PCA
         # TODO: Apply PCA to the good data with the same number of dimensions as featu
         res
         pca = PCA(n_components=6)
         pca.fit(good_data)

         # TODO: Apply a PCA transformation to the sample log-data
         pca_samples = pca.transform(log_samples)

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



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 two principal components explain approximately (0.44+0.26=) 70% of the variance, and the first four approximately (0.44+0.26+0.12+0.1) 92%.

Dimension 1: This dimension mostly comprises of Milk, Grocery, and Detergents_paper. These features were earlier identified as being highly correlated and now are represented together as the 1st dimension of PCA. Together with Delicatessen, an increase in feature weights of these and decrease in Fresh & Frozen corresponds to a positive increase in this dimension.

Dimension 2: This dimension seems to complement the first. Concretely, it's roughly in equal-parts composed of spending that is not in Milk, Grocery and Detergents_paper.

Dimension 3: This dimension represents relationship of spending in Frozen/Delicatessen vs Fresh - as it decreases in Fresh, the other two viz Frozen & Delicatessen increase - which perhaps corresponds to how anyone would buy.

Dimension 4: This dimension counter balances the third, showing spending relationship in Frozen vs Delicatessen products.

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 [88]:
```
# Display sample log-data after having a PCA transformation applied
display(pd.DataFrame(np.round(pca_samples, 4), columns = pca_results.index.val
ues))
```

	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5	Dimension 6
0	-0.7394	2.9834	-0.8204	1.2945	-0.1297	-0.4712
1	0.0561	-0.9457	0.6118	-1.0588	0.6650	0.2820
2	-0.6448	2.8583	0.6377	0.5879	-1.9515	-0.7170

Implementation: Dimensionality Reduction

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

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

- Assign the results of fitting PCA in two dimensions with `good_data` to `pca`.
- Apply a PCA transformation of `good_data` using `pca.transform`, and assign the reuslts to `reduced_data`.
- Apply a PCA transformation of the sample log-data `log_samples` using `pca.transform`, and assign the results to `pca_samples`.

In [89]:
```
# TODO: Fit PCA to the good data using only two dimensions
pca = PCA(n_components=2)
pca.fit(good_data)

# TODO: Apply a PCA transformation the good data
reduced_data = pca.transform(good_data)

# TODO: Apply a PCA transformation to the sample log-data
pca_samples = pca.transform(log_samples)

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

Observation

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

In [90]:
```
# Display sample log-data after applying PCA transformation in two dimensions
display(pd.DataFrame(np.round(pca_samples, 4), columns = ['Dimension 1', 'Dime
nsion 2']))
```

	Dimension 1	Dimension 2
0	-0.7394	2.9834
1	0.0561	-0.9457
2	-0.6448	2.8583

Clustering

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

Question 6

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

Answer:

K-Means clustering is a general purpose, fast and highly scalable (in number of samples) algorithm. It is also scalable in the number of clusters, but to a lesser degree. K-Means could be considered a variation of Gaussian Mixture Model, but does hard assignments.

Gaussian Mixture Model can allow for clusters of different sizes & shapes (unlike K-Means, which assumes even cluster sizes) and is generally not as scalable compared to K-Means. Gaussian Mixture Model allows for soft assignments (unlike K-Means) and can be useful in certain circumstances.

I would first try K-Means, which is general purpose & does not require too much configuration. If the results seem to indicate that a different algorithm suitable for variable cluster size/shape outweighing speed and scalability could perform better, then I would try Gaussian Mixture Model.

Implementation: Creating Clusters

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

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

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

In [91]:
```
# TODO: Apply your clustering algorithm of choice to the reduced data
from sklearn.cluster import KMeans
clusterer = KMeans(n_clusters = 2).fit(reduced_data)

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

# TODO: Find the cluster centers
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 c
hosen
from sklearn.metrics import silhouette_score
score = silhouette_score(reduced_data, preds)
print score
```

```
0.426281015469
```

Question 7

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

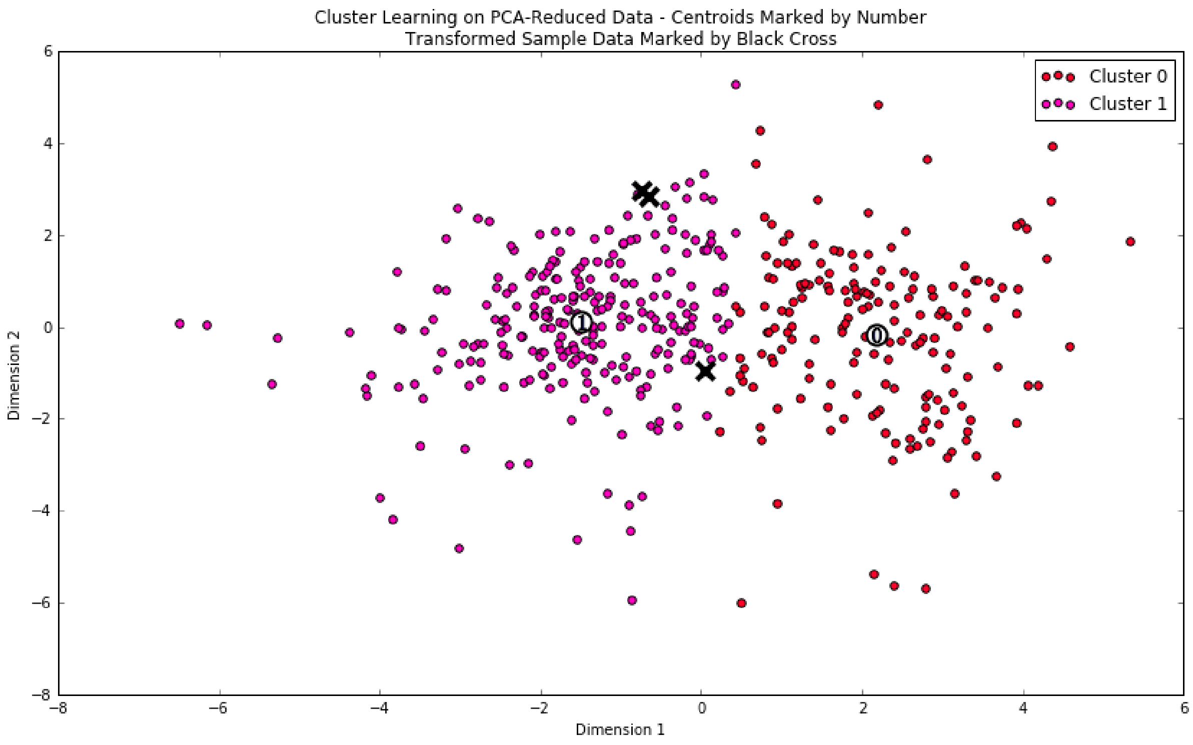
Answer: From the number of clusters I tried, cluster = 2 gives the best score.

2 clusters: 0.426281015469
3 clusters: 0.391151981946
4 clusters: 0.331245916061
5 clusters: 0.350990778931
6 clusters: 0.361868754295
7 clusters: 0.36487535911
8 clusters: 0.349740568222
9 clusters: 0.333386762372
10 clusters: 0.35283605157

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



Implementation: Data Recovery

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

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

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

In [93]:
```
# TODO: Inverse transform the centers
log_centers = pca.inverse_transform(centers)

# TODO: Exponentiate the centers
true_centers = np.exp(log_centers)

# Display the true centers
segments = ['Segment {}'.format(i) for i in range(0,len(centers))]
true_centers = pd.DataFrame(np.round(true_centers), columns = data.keys())
true_centers.index = segments
display(true_centers)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
Segment 0	4005	7900	12104	952	4561	1036
Segment 1	8867	1897	2477	2088	294	681

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:

Below are some of the quick numbers -

- An average customer in Segment 0 spends around 80% (of total spending) on Grocery, Milk and Detergent_Paper. Spending in these categories is well above the total mean (of the entire dataset)
- An average customer in Segment 1 spends over 50% on Fresh. While spending on Frozen is only about 13%, it's above the total median
- An average customer spending in Segment 1 is approximately 50% of Segment 0.

Looking at the above spending numbers, one could classify Segment 0 as "end consumers", i.e. individual people/families and Segment 1 to be perhaps "repackagers" i.e. some sort of businesses that produce other finished food products/goods.

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 [94]:
```
# 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 1
```

Answer:
Interestingly enough, the 3 samples I selected earlier have all been assigned to Cluster 1. I had assumed/assigned the samples to 3 different businesses (restaurant, small cafe and retail), given the dataset is of a wholesale distributor. Because Cluster 1 supposedly represents some repackagers/businesses, the segmentation for the samples is consistent with my earlier predictions - which is only coincidental, since my predictions were just initial guesses (I had no beforehand knowledge of these segments/segmentation).

The fact that all 3 samples belong to Cluster 1 can be seen from both the Cluster & Channel plots. Also it can be visually seen that the samples are closer to Centroid 1 (than Centroid 0). This is due to K-Means calculating that the Euclidean distance between each of the sample points and Centroid 1 is minimal, as compared to Centroid 0. In other words, the predicted cluster seems to be consistent with the category spendings for each of the samples.

Conclusion

Question 10

Companies often run A/B tests (https://en.wikipedia.org/wiki/A/B_testing) when making small changes to their products or services. If the wholesale distributor wanted to change its delivery service from 5 days a week to 3 days a week, how would you use the structure of the data to help them decide on a group of customers to test?
Hint: Would such a change in the delivery service affect all customers equally? How could the distributor identify who it affects the most?

Answer: The delivery service change from 5 to 3 days a week might negatively affect the segment(s) consuming Fresh products the most. The Segment 1 from our results would have the most impact since it consumes over 50% Fresh of the segment total. Also the change would impact those customers who place orders on as-needed basis and rely on timely/frequent delivery service.

Before making the delivery service change, I would select a representative sample set of small, medium and large customers from each of the segments and divide them into control and test groups. I would then communicate about the change to this group well in advance and make the change on the test group first and get their feedback along with running my own analysis of any changes in daily sales pattern for few weeks to see how it's impacting the business. Based on the feedback and analysis, I would assess the impact on all the important business parameters e.g. sales/revenue, brand impage/perception, customer LTV (Life Time Value) etc and then take an appropriate decision on change of delivery schedule.

Question 11

Assume the wholesale distributor wanted to predict a new feature for each customer based on the purchasing information available. How could the wholesale distributor use the structure of the data to assist a supervised learning analysis?
Hint: What other input feature could the supervised learner use besides the six product features to help make a prediction?

Answer:

We could use the results from our clustering/segmentation analysis, for any further supervised learning -

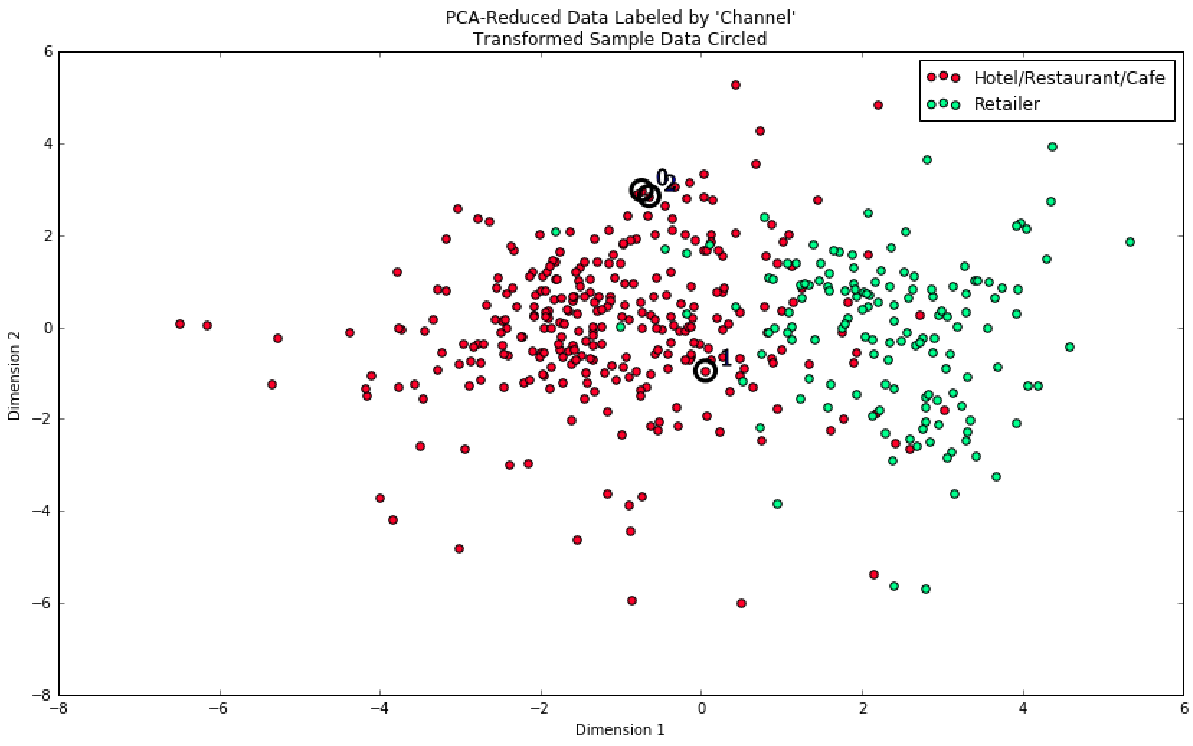
- The segmentation information/label obtained for each of the customer records as a new feature
- Some of the first few principal components (normalized), based on the amount of variance we would like to capture from the original data. Typically the number of components corresponding to about 80-85% of the original variation is used

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 on 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 [95]:
```
# Display the clustering results based on 'Channel' data
rs.channel_results(reduced_data, outliers_list, pca_samples)
```



Question 12

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

Answer: Our clustering algorithm seems to have identified the segments from the distributor data fairly accurately. Also my hypothesis of 2 clusters one each for businesses and end consumers is revealed to be correct. Although in my intuition I had used the term "retail" to mean a retail business, here in the Channel information it is used to mean end consumers as "retailers".

The graph above reveals that there are several HoReCa customers who extend into the Retailer cluster, whereas the left third of the graph consists entirely of HoReCa customers.

> **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.