Customer Segmentation in Python

by Greg | August 25, 2015

In this post I'm going to talk about something that's relatively simple but fundamental to just about any business: Customer Segmentation. At the core of customer segmentation is being able to identify different types of customers and then figure out ways to find more of those individuals so you can... you guessed it, get more customers! In this post, I'll detail how you can use K-Means clustering to help with some of the exploratory aspects of customer segmentation.

Our Data

The data we're using comes from John Foreman's book Data Smart. The dataset contains both information on marketing newsletters/e-mail campaigns (e-mail offers sent) and transaction level data from customers (which offer customers responded to and what they bought).

```
import pandas as pd

df_offers = pd.read_excel("./WineKMC.xlsx", sheetname=0)
df_offers.columns = ["offer_id", "campaign", "varietal", "min_qty", "discount", "origin", "past_peak"]
df_offers.head()
```

	offer_id	campaign	varietal	min_qty	discount	origin	past_peak
0	1	January	Malbec	72	56	France	False
1	2	January	Pinot Noir	72	17	France	False
2	3	February	Espumante	144	32	Oregon	True
3	4	February	Champagne	72	48	France	True
4	5	February	Cabernet Sauvignon	144	44	New Zealand	True

And the transaction level data...

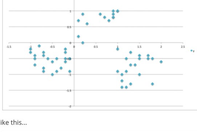
```
df_transactions = pd.read_excel("./WineKMC.xlsx", sheetname=1)
df_transactions.columns = ["customer_name", "offer_id"]
df_transactions['n'] = 1
df_transactions.head()
```

	customer_name	offer_id	n
0	Smith	2	1
1	Smith	24	1
2	Johnson	17	1
3	Johnson	24	1
4	Johnson	26	1

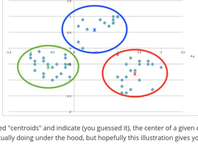
A quick K-Means primer

In order to segment our customers, we need a way to compare them. To do this we're going to use K-Means clustering. K-means is a way of taking a dataset and finding groups (or clusters) of points that have similar properties. K-means works by grouping the points together in such a way that the distance between all the points and the midpoint of the cluster they belong to is minimized.

Think of the simplest possible example. If I told you to create 3 groups for the points below and draw a star where the middle of each group would be, what would you do?



Probably (or hopefully) something like this...



In K-Means speak, the "x"s are called "centroids" and indicate (you guessed it), the center of a given cluster. I'm not going to go into the ins and outs of what K-Means is actually doing under the hood, but hopefully this illustration gives you a good idea.

Clustering our customers

Okay, so how does clustering apply to our customers? Well since we're trying to learn more about how our customers behave, we can use their behavior (whether or not they purchased something based on an offer) as a way to group similar minded customers together. We can then study those groups to look for patterns and trends which can help us formulate future offers.

The first thing we need is a way to compare customers. To do this, we're going to create a matrix that contains each customer and a 0/1 indicator for whether or not they responded to a given offer. This is easy enough to do in Python:

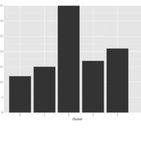
```
# join the offers and transactions table
df = pd.merge(df_offers, df_transactions)
# create a "pivot table" which will give us the number of times each customer responded to a given offer
matrix = df.pivot_table(index=['customer_name'], columns=['offer_id'], values='n')
# a little tidying up. fill NA values with 0 and make the index into a column
matrix = matrix.fillna(0).reset_index()
# save a list of the 0/1 columns. we'll use these a bit later
x_cols = matrix.columns[1:]
```

Now to create the clusters, we're going to use the `KMeans` functionality from `scikit-learn`. I arbitrarily chose 5 clusters. My general rule of thumb is to have **at least 7x** as many records as I do clusters.

```
from sklearn.cluster import KMeans

cluster = KMeans(n_clusters=5)
# slice matrix so we only include the 0/1 indicator columns in the clustering
matrix['cluster'] = cluster.fit_predict(matrix[matrix.columns[2:]])
matrix.cluster.value_counts()
```

```
2    32
1    22
4    20
0    15
3    11
dtype: int64
```



Visualizing the clusters

A really cool trick that the probably didn't teach you in school is Principal Component Analysis. There are lots of uses for it, but today we're going to use it to transform our multi-dimensional dataset into a 2 dimensional dataset. Why you ask? Well once it is in 2 dimensions (or simply put, it has 2 columns), it becomes much easier to plot!

Once again, `scikit-learn` comes to the rescue!

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
matrix['x'] = pca.fit_transform(matrix[x_cols])[:,0]
matrix['y'] = pca.fit_transform(matrix[x_cols])[:,1]
matrix = matrix.reset_index()


customer_clusters = matrix[['customer_name', 'cluster', 'x', 'y']]
customer_clusters.head()
```

	offer_id	customer_name	cluster	x	y
0	Adams	2	-1.007580	0.108215
1	Allen	4	0.287539	0.044715
2	Anderson	1	0.392032	1.038391
3	Bailey	2	-0.699477	-0.022542
4	Baker	3	-0.088183	-0.471695

What we've done is we've taken those `x_cols` columns of 0/1 indicator variables, and we've transformed them into a 2-D dataset. We took one column and arbitrarily called it `x` and then called the other `y`. Now we can throw each point into a scatterplot. We'll color code each point based on it's cluster so it's easier to see them.

```
df = pd.merge(df_clusters, customer_clusters)
df = pd.merge(df_offers, df)

from ggplot import *

ggplot(df, aes('x', 'y', color='cluster')) + \
    geom_point(size=75) + \
    ggtitle("Customers Grouped by Cluster")
```

If you want to get fancy, you can also plot the centers of the clusters as well. These are stored in the `KMeans` instance using the `cluster_centers_` variable. Make sure that you also transform the cluster centers into the 2-D projection.

```
cluster_centers = pca.transform(cluster.cluster_centers_)
cluster_centers = pd.DataFrame(cluster_centers, columns=['x', 'y'])
cluster_centers['cluster'] = range(0, len(cluster_centers))

ggplot(df, aes('x', 'y', color='cluster')) + \
    geom_point(size=75) + \
    geom_point(cluster_centers, size=500) +\
    ggtitle("Customers Grouped By Cluster")
```

Digging deeper into the clusters

Let's dig a little deeper into the clusters. Take cluster 4 for example. If we break out cluster 4 and compare it to the remaining customers, we can start to look for interesting facets that we might be able to exploit.

As a baseline, take a look at the `varietal` counts for cluster 4 vs. everyone else. It turns out that almost all of the Cabernet Sauvignon offers were purchased by members of cluster 4. In addition, none of the Espumante offers were purchased by members of cluster 4.

```
df['is_4'] = df.cluster==4
df.groupby("is_4").varietal.value_counts()
```

is_4	varietal	count
False	Champagne	45
	Espumante	40
	Prosecco	37
	Pinot Noir	37
	Malbec	17
	Pinot Grigio	16
	Merlot	8
	Cabernet Sauvignon	6
	Chardonnay	4
True	Champagne	36
	Cabernet Sauvignon	26
	Malbec	15
	Merlot	12
	Chardonnay	11
	Pinot Noir	7
	Prosecco	6
	Pinot Grigio	1

You can also segment out numerical features. For instance, look at how the mean of the `min_qty` field breaks out between 4 vs. non-4. It seems like members of cluster 4 like to by in bulk!

```
df.groupby("is_4")[['min_qty', 'discount']].mean()
```

is_4	min_qty	discount
False	47.685484	59.120968
True	93.394737	60.657895

Send a bulk Cab Sav offer Cluster 4's way!

Final Thoughts

While it's not going to magically tell you all the answers, clustering is a great exploratory exercise that can help you learn more about your customers. For more info on K-Means and customer segmentation, check out these resources:

- INSEAD Analytics Cluster Analysis and Segmentation Post
- Customer Segmentation at Bain & Company
- Customer Segmentation Wikipedia

Code for this post can be found here.