Clustering to Determine Preferable Healthcare Clinic Locations in Ohio

Northwestern University IEMS 308

Jake Atlas

Executive Summary

In determining where to place a medical center, hospitals in the United States must consider the potential financial consequences of location choice, since they are non-governmentally affiliated institutions. In order to find a preferable location, potential health clinics should take advantage of the vast amount of healthcare data made public. By analyzing such data, medical centers can establish themselves in locations that are inherently conducive to higher demand for services and increased profit levels.

K-Means clustering is an analytical technique that serves to identify natural groups within data. Luckily for new health clinics, health data is readily available. One example of such a dataset is the 2015 Medicare Provider data, which contains information on each service performed by doctors across the United States. By paring down this enormous set of data to primarily contain information that is financially relevant, someone interested in starting a medical clinic can take advantage of clustering techniques to identify and exploit patterns that will lead to a more fiscally successful clinic.

After performing exploratory analysis of the data and determining a set of attributes that would affect the financial outlook of a burgeoning medical center, a subset of the dataset was taken such that all points referred to Ohio doctors. Clustering was then conducted to identify patterns.

The results proved to be favorable, in that there is no particular location that is more conducive to fiscal success than another. The primary factor that differentiates a doctor placed in one Ohio region versus another, on average, is the difference in location, not the number of patients seen or amount of money procedures cost. Therefore, regardless of the general location, any medical center can be successful provided there is a clear business plan indicating opportunity.

Problem Statement

The Medicare Provider dataset contains a wealth of information about doctors throughout the United States and the medical services they have conducted. My job was to identify whether certain regions within Ohio are financially preferable for a new health clinic that is wondering where to locate itself.

Methodology

The first step in conducting a k-means clustering analysis of the Medicare data is performing exploratory data analysis to determine which variables are suitable for clustering. After identifying the set of features on which to cluster, the dataset is trimmed to include only these features variables for all Ohio doctors. The remaining features were average_Medicare_allowed_amt, line_srvc_cnt, and zipcode, measures of procedural costs, number of procedures, and location, respectively.

In order to ensure meaningful results, four data preprocessing techniques were applied to the data. First, zip-code, which is categorical despite being represented with numbers, was one-hot encoded. All records were assigned to one of five regions within Ohio. Below, the regions are identified in Figure 1. The three digits appearing in each subsection of the larger regions denote the first three digits of the zip-codes for all locations within that subsection.

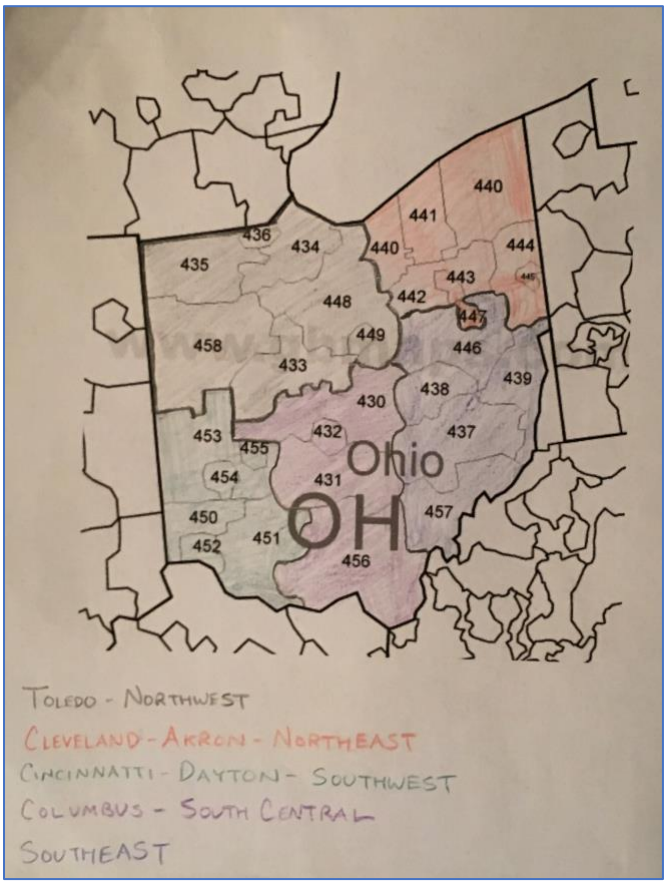

Figure 1

Second, outliers were removed such that they did not exert high leverage and cause errors in the clustering. Outliers were removed in both the average_Medicare_allowed_amt and line_srvc_cnt columns. Third, the data was standardized in order to ensure that values associated with any particular feature did not sway the clustering excessively. Last, the data was normalized using base-10 logarithmic transformation of all data-points in the

average_Medicare_allowed_amt and line_srvc_cnt columns. As you can see below in Figure 2 and Figure 3, applying the log transformation a single time made the average_Medicare_allowed_amt data approximately normal, but the line_srvc_cnt data had to be logged a second time to become approximately normal.

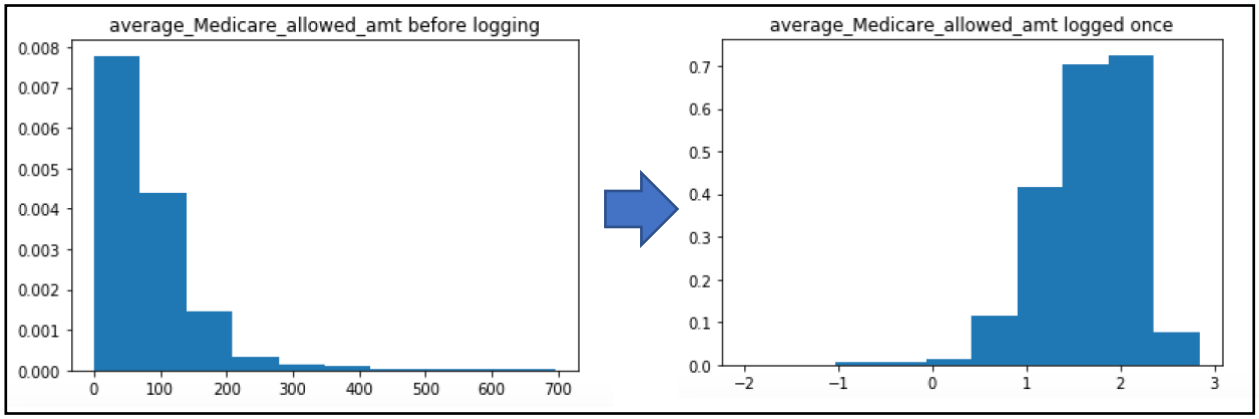

Figure 2

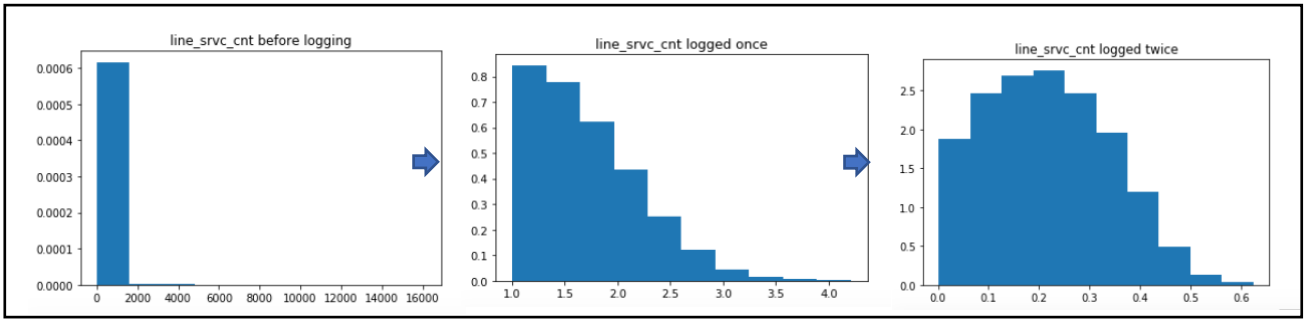

Figure 3

At this point, the data was now suitable for clustering. To assess how many clusters to use, a scree plot was created that indicated the total summed distance from points to the centroids of their clusters. As you can see in Figure 4 on the following page, there is a clear elbow point that occurs when the number of clusters is equal to five, so subsequent analysis uses five clusters.

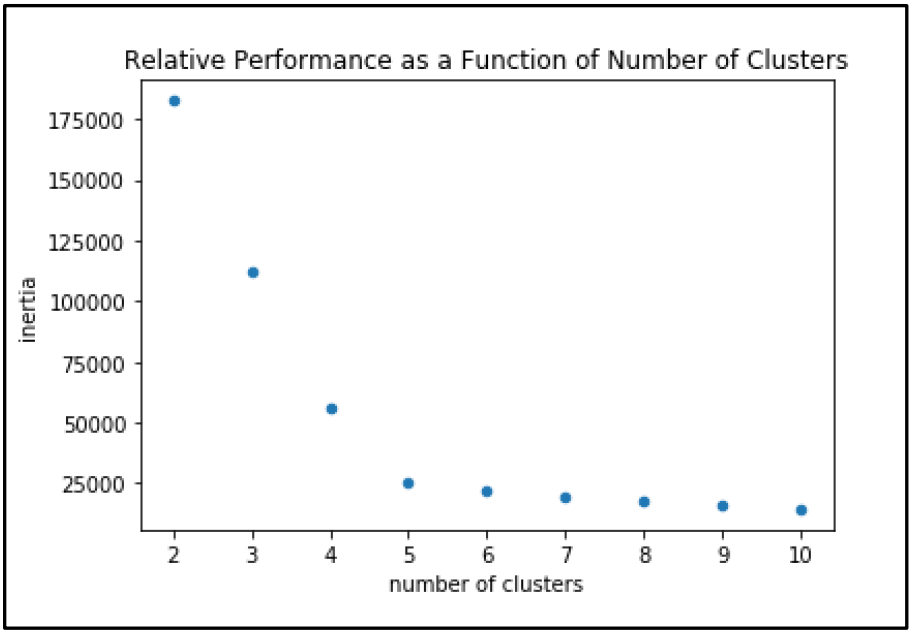

Figure 4

To verify that the clusters were in fact differentiated from each other, the average silhouette score was computed. At 0.773, this average silhouette score indicates that clusters are adequately differentiated from each other. Assessing the quality of the clustering is an important step in the process, because it ensures validity of subsequent analysis.

Analysis

The first and most important part of the analysis was to identify patterns in the data. The most important finding was the incredibly strong correlation between location and cluster placement. Almost without fail, location differentiated each point more so than monetary or procedural count measures.

To provide evidence to further elucidate this pattern, it was important to check that monetary measures did not, in fact, vary significantly from cluster to cluster. In order to provide a side by side comparison, a metric was constructed that represents the standardized amount of money doctors in a particular cluster should have earned. This metric was computed as:

$$Standardized\ Revenue_j = \frac{1}{n_j} \sum_{i \in C_j}^{n_j} average\_Medicare\_allowed\_amt_i * line\_srvc\_cnt_i$$

with $j = 1, 2, \ldots, 5$ representing the clusters, and $i = 1, 2, \ldots, n_j$ representing the records assigned to cluster j. Table 1, on the next page, shows the computed values of standardized revenue for a particular clustering with $k = 5$ clusters. Note that these values are unique to

each iteration of the clustering procedure, so running the code (see Appendix, on page 8) will produce similar but not necessarily identical values. Table 2, which appears below to the right of Table 1, indicates the association between the cluster and geographic region.

Standardized Revenue by Cluster	
Cluster 1	0.441032556462
Cluster 2	0.425462764885
Cluster 3	0.450618872944
Cluster 4	0.445547857851
Cluster 5	0.45595427763

Table 1

Approximate Location – Cluster Relationship	
Cleveland_Akron_Northeast	Cluster 1
Columbus_South-Central	Cluster 2
Toledo-Northwest	Cluster 3
Cincinnati_Dayton_Southwest	Cluster 4
Southeast	Cluster 5

Table 2

Note that the relationships in Figure 2 are approximate. Not every record will necessarily always be assigned to cluster based on location. This is only a pattern, not an invariant. Also important to note is that running the code in the Appendix will rarely assign locations to clusters exactly as above. For example, re-running the program may assign most points associated with Cleveland_Akron_Northeast to Cluster 3. There are actually $5! = 120$ different combinations, so there is under a 1% chance that running the code will produce these exact relationships.

It is, in fact, the case that standardized revenue is essentially the same from cluster to cluster. Therefore, it can be concluded that someone wishing to place a health clinic somewhere in Ohio should not expect any general region to be more lucrative than another.

Conclusions

The key insights from the clustering algorithms applied to the Medicare Provider data are:

Doctors in different areas are differentiated more by their location than the cost of their procedures. This is likely due to the fact that there are not significant differences in wealth between the areas. Each of the five areas was designed in order to contain both urban, suburban, and rural communities. If the analysis were to be redone, but the zones were drawn such that wealth disparities between zones was encouraged, this would likely not remain the case.

Doctors in different areas are differentiated more by their location than the number of procedures they do. This is likely due to the fact that number of procedures is determined based on medical necessity of the nearby population. It is to be expected that the set of people living in one area in Ohio have the same general medical needs as the set of people living in

another area. That is, large subsections of Ohio are not sicker or healthier than other large subsections, on average. This could vary if the regions were redrawn.

Location itself is not a determinant of financial viability for healthcare providers. For someone wishing to start a health clinic, this directs the focus towards determining locations that need additional healthcare services.

These key insights are validated both by the clustering model as well as by the broader analysis conducted to verify the patterns spotted using clustering. I am therefore confident in my conclusion that health clinics will succeed based on demand as opposed to based on location.

Next Steps

Since general location does not provide a new health clinic with a basis for financial success, it is important to do further analysis to determine methods that would give a healthcare entrepreneur a "head start," financially speaking. A first step would be to do a similar analysis, but after adjusting the regions. Comparing rural, suburban, and urban areas will likely result in more positive results.

As stated above, it is important to consider demand, so an important analysis to conduct is one centered around predicting demand. Medicare data is not limited to the 2015 dataset used in this analysis. By incorporating the data from years past, a time series analysis can be done that identifies areas of growth. A further project that stems from this is the creation of a predictive model that attempts to identify areas that are likely to grow significantly. In such a location, the need for medical services will also grow rapidly. Having an understanding of which areas have potential for demand enables a clinic to establish itself before potential competitors.

Appendix – Source Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

#Read in the text file, extract the columns of interest
file =
pd.read_csv('/Users/jakeatlas/Downloads/Medicare_Provider_Util_Payment_PUF_CY2015/Medicare_Provider_Uti
l_Payment_PUF_CY2015.txt',sep='\t')

df_USA = file[['average_Medicare_allowed_amt','nppes_provider_zip','line_srvc_cnt']].copy()

#Initialize a column with the first 2 digits of zipcodes and eliminate records
# that aren't associated with Ohio
df_USA['zipcode_first2_digits'] = df_USA['nppes_provider_zip'].astype(str).str[0:2]
df_OH = df_USA[(df_USA['zipcode_first2_digits']=='43') | (df_USA['zipcode_first2_digits']=='44') |
(df_USA['zipcode_first2_digits']=='45')]

del df_OH['zipcode_first2_digits']
df_OH = df_OH.reset_index(drop=True)

#Initialize a column with the first 3 digits of zipcodes and 1-hot encode all
# zipcodes to be associated with bins corresponding to 5 different geographical
# regions within Ohio
df_OH['zipcode_first3_digits'] = df_OH['nppes_provider_zip'].astype(str).str[0:3]
cleanup_nums = {'zipcode_first3_digits':    {'435':'toledo_northwest',
                                    '436':'toledo_northwest',
                                    '434':'toledo_northwest',
                                    '458':'toledo_northwest',
                                    '433':'toledo_northwest',
                                    '448':'toledo_northwest',
                                    '449':'toledo_northwest',
                                    '440':'cleveland_akron_northeast',
                                    '441':'cleveland_akron_northeast',
                                    '442':'cleveland_akron_northeast',
                                    '443':'cleveland_akron_northeast',
                                    '445':'cleveland_akron_northeast',
                                    '444':'cleveland_akron_northeast',
                                    '447':'cleveland_akron_northeast',
                                    '453':'cincinnatti_dayton_southwest',
                                    '454':'cincinnatti_dayton_southwest',
                                    '455':'cincinnatti_dayton_southwest',
                                    '450':'cincinnatti_dayton_southwest',
                                    '452':'cincinnatti_dayton_southwest',
                                    '451':'cincinnatti_dayton_southwest',
                                    '430':'columbus_southcentral',
                                    '432':'columbus_southcentral',
                                    '431':'columbus_southcentral',
```

```
                                '456':'columbus_southcentral',
                                '446':'southeast',
                                '438':'southeast',
                                '439':'southeast',
                                '437':'southeast',
                                '457':'southeast'}}
df_OH.replace(cleanup_nums,inplace=True)
df_OH.rename(columns={'zipcode_first3_digits':'zone'},inplace=True)
df_OH_cat = pd.get_dummies(df_OH,columns=['zone'])
del df_OH_cat['nppes_provider_zip']

#Remove outliers
df_OH_no_outlier_int = df_OH_cat[np.abs(df_OH_cat['average_Medicare_allowed_amt']) -
(df_OH_cat['average_Medicare_allowed_amt'].mean()) <=
(3*df_OH_cat['average_Medicare_allowed_amt'].std())]

df_OH_no_outlier_int = df_OH_no_outlier_int.reset_index(drop=True)

df_OH_no_outlier = df_OH_no_outlier_int[np.abs(df_OH_no_outlier_int['line_srvc_cnt']) -
(df_OH_no_outlier_int['line_srvc_cnt'].mean()) <= (3*df_OH_no_outlier_int['line_srvc_cnt'].std())]

df_OH_no_outlier = df_OH_no_outlier.reset_index(drop=True)

#Check normality, then log, then check normality again
for element in ['average_Medicare_allowed_amt','line_srvc_cnt']:
   plt.hist(x=df_OH_no_outlier[element],normed=1)
   plt.title(element + ' before logging')
   plt.show()
   df_OH_no_outlier[element] = np.log10(df_OH_no_outlier[element])
   plt.hist(x=df_OH_no_outlier[element],normed=1)
   plt.title(element + ' logged once')
   plt.show()

#line_srvc_cnt needs to be logged a second time because it's still not normal
df_OH_no_outlier['line_srvc_cnt'] = np.log10(df_OH_no_outlier['line_srvc_cnt'])
plt.hist(x=df_OH_no_outlier['line_srvc_cnt'],normed=1)
plt.title('line_srvc_cnt logged twice')
plt.show()

#Standardize the data by subtracting the min and dividing by the max
average_Medicare_allowed_amt_min = np.min(df_OH_no_outlier['average_Medicare_allowed_amt'])
average_Medicare_allowed_amt_max = np.max(df_OH_no_outlier['average_Medicare_allowed_amt'])
line_srvc_cnt_min = np.min(df_OH_no_outlier['line_srvc_cnt'])
line_srvc_cnt_max = np.max(df_OH_no_outlier['line_srvc_cnt'])

df_OH_no_outlier['average_Medicare_allowed_amt'] = (df_OH_no_outlier['average_Medicare_allowed_amt'] -
average_Medicare_allowed_amt_min) / average_Medicare_allowed_amt_max

df_OH_no_outlier['line_srvc_cnt'] = (df_OH_no_outlier['line_srvc_cnt'] - line_srvc_cnt_min) / line_srvc_cnt_max
```

```
#Determining number of clusters --> Turns out we should have 5
cluster_quality = pd.DataFrame(columns = ['inertia', 'number of clusters'])
for i in range(2,11):
   kmeans = KMeans(n_clusters = i)
   kmeans = kmeans.fit(df_OH_no_outlier)
   cluster_quality.loc[i-2] = [kmeans.inertia_,i]

cluster_quality.plot.scatter(x='number of clusters',y='inertia',title='Relative Performance as a Function of Number
of Clusters')

#Recluster with 5 clusters
kmeans = KMeans(n_clusters = 5)
kmeans = kmeans.fit(df_OH_no_outlier)
labels = kmeans.predict(df_OH_no_outlier)
centroids = kmeans.cluster_centers_


#Compute silhouette scores
df_with_labels = df_OH_no_outlier.copy()
df_with_labels['labels'] = labels
sampled_data = df_with_labels.sample(n=10000)
silhouette_avg = silhouette_score(sampled_data.drop('labels',axis=1),sampled_data['labels'])
print('The average silhouette score is ' + str(silhouette_avg))

#Add cluster column to df_OH_no_outlier
df_OH_no_outlier['cluster'] = labels

#Examining the data, it is clear that clustering is determined solely by location. Get the location for each cluster.
for row in range(0,np.shape(df_OH_no_outlier)[0]):
   if df_OH_no_outlier['cluster'][row]==0:
      if df_OH_no_outlier['zone_cincinnatti_dayton_southwest'][row]==1:
         cluster0 = 'zone_cincinnatti_dayton_southwest'
      elif df_OH_no_outlier['zone_cleveland_akron_northeast'][row]==1:
         cluster0 = 'zone_cleveland_akron_northeast'
      elif df_OH_no_outlier['zone_columbus_southcentral'][row]==1:
         cluster0 = 'zone_columbus_southcentral'
      elif df_OH_no_outlier['zone_southeast'][row]==1:
         cluster0 = 'zone_southeast'
      else:
         cluster0 = 'zone_toledo_northwest'
   elif df_OH_no_outlier['cluster'][row]==1:
      if df_OH_no_outlier['zone_cincinnatti_dayton_southwest'][row]==1:
         cluster1 = 'zone_cincinnatti_dayton_southwest'
      elif df_OH_no_outlier['zone_cleveland_akron_northeast'][row]==1:
         cluster1 = 'zone_cleveland_akron_northeast'
      elif df_OH_no_outlier['zone_columbus_southcentral'][row]==1:
         cluster1 = 'zone_columbus_southcentral'
      elif df_OH_no_outlier['zone_southeast'][row]==1:
         cluster1 = 'zone_southeast'
      else:
         cluster1 = 'zone_toledo_northwest'
   elif df_OH_no_outlier['cluster'][row]==2:
```

```
        if df_OH_no_outlier['zone_cincinnatti_dayton_southwest'][row]==1:
            cluster2 = 'zone_cincinnatti_dayton_southwest'
        elif df_OH_no_outlier['zone_cleveland_akron_northeast'][row]==1:
            cluster2 = 'zone_cleveland_akron_northeast'
        elif df_OH_no_outlier['zone_columbus_southcentral'][row]==1:
            cluster2 = 'zone_columbus_southcentral'
        elif df_OH_no_outlier['zone_southeast'][row]==1:
            cluster2 = 'zone_southeast'
        else:
            cluster2 = 'zone_toledo_northwest'
    elif df_OH_no_outlier['cluster'][row]==3:
        if df_OH_no_outlier['zone_cincinnatti_dayton_southwest'][row]==1:
            cluster3 = 'zone_cincinnatti_dayton_southwest'
        elif df_OH_no_outlier['zone_cleveland_akron_northeast'][row]==1:
            cluster3 = 'zone_cleveland_akron_northeast'
        elif df_OH_no_outlier['zone_columbus_southcentral'][row]==1:
            cluster3 = 'zone_columbus_southcentral'
        elif df_OH_no_outlier['zone_southeast'][row]==1:
            cluster3 = 'zone_southeast'
        else:
            cluster3 = 'zone_toledo_northwest'
    elif df_OH_no_outlier['cluster'][row]==4:
        if df_OH_no_outlier['zone_cincinnatti_dayton_southwest'][row]==1:
            cluster4 = 'zone_cincinnatti_dayton_southwest'
        elif df_OH_no_outlier['zone_cleveland_akron_northeast'][row]==1:
            cluster4 = 'zone_cleveland_akron_northeast'
        elif df_OH_no_outlier['zone_columbus_southcentral'][row]==1:
            cluster4 = 'zone_columbus_southcentral'
        elif df_OH_no_outlier['zone_southeast'][row]==1:
            cluster4 = 'zone_southeast'
        else:
            cluster4 = 'zone_toledo_northwest'

print('The clusters are associated with the following regions:' + '\n'
    + 'Cluster 1: ' + cluster0 + '\n'
    + 'Cluster 2: ' + cluster1 + '\n'
    + 'Cluster 3: ' + cluster2 + '\n'
    + 'Cluster 4: ' + cluster3 + '\n'
    + 'Cluster 5: ' + cluster4)

#Checking the total amount of money in each cluster to ensure they're similar
df_money = df_OH_no_outlier[['cluster','average_Medicare_allowed_amt','line_srvc_cnt']].copy()
df_money['money'] = df_money['average_Medicare_allowed_amt'] * df_money['line_srvc_cnt']

total_money_cluster0 = df_money.loc[df_money['cluster']==0, 'money'].sum()
total_money_cluster1 = df_money.loc[df_money['cluster']==1, 'money'].sum()
total_money_cluster2 = df_money.loc[df_money['cluster']==2, 'money'].sum()
total_money_cluster3 = df_money.loc[df_money['cluster']==3, 'money'].sum()
total_money_cluster4 = df_money.loc[df_money['cluster']==4, 'money'].sum()

#They're quite similar - location does not really affect profitability
standardized_money_cluster0 = total_money_cluster0 / df_OH_no_outlier[cluster0].sum()
standardized_money_cluster1 = total_money_cluster1 / df_OH_no_outlier[cluster1].sum()
```

```
standardized_money_cluster2 = total_money_cluster2 / df_OH_no_outlier[cluster2].sum()
standardized_money_cluster3 = total_money_cluster3 / df_OH_no_outlier[cluster3].sum()
standardized_money_cluster4 = total_money_cluster4 / df_OH_no_outlier[cluster4].sum()

print('The standardized amount of money per cluster is: ' +
    '\n' + 'Cluster 1: ' + str(standardized_money_cluster0) +
    '\n' + 'Cluster 2: ' + str(standardized_money_cluster1) +
    '\n' + 'Cluster 3: ' + str(standardized_money_cluster2) +
    '\n' + 'Cluster 4: ' + str(standardized_money_cluster3) +
    '\n' + 'Cluster 5: ' + str(standardized_money_cluster4))
```