Report

Data processing (1)

- Data after combining:

AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	Remaining_Balance1	Remaining_Balance2	Remaining_Balance3	Remaining_Balance4	Remaining_Balance5	Remaining_Balance6
4000	4000	4000	6100	96191	95732	100705	101478	103725	103894
1000	200	265	500	-401	-431	401	1303	569	431
1396	0	0	0	1909	-5076	4543	1390	-5	-5
0	446	1729	0	549	-987	1166	344	-923	1729
7100	5300	5000	5000	146927	152848	153554	149759	136570	129143
...	...	...	...	...	...	...	...	...	...
1212	1000	1020	800	42483	43076	27766	16008	12760	13793
1000	400	0	0	14527	19571	19434	19474	19546	0
545	726	353	99	16841	18004	18801	15028	15502	14912
2000	2000	3000	2000	61508	50382	47271	48449	48636	51666
1500	2000	1000	6443	-23699	42944	38953	38951	39709	29126

24000 rows × 18 columns

Data processing (2)

Using PCA to reduce the dimensionality of features

In [7]:
```
pca = PCA(n_components=3)
pca.fit(table)
x_re = pca.transform(table)
x_re
```

Out[7]:
```
array([[ 205332.16354691,   41400.11553396,  -12146.26823875],
       [-145481.77366702,    1586.65804841,    -951.27514133],
       [-143227.8718299 ,    1330.17748985,    7213.00437983],
       ...,
       [ -88870.56892055,     801.22186823,   -1196.21800443],
       [  33900.98477725,    5948.14332954,  -12840.74895319],
       [ -37720.33459165,   39087.89105548,    3807.26816733]])
```

In [8]:
```
x_norm = normalize(x_re)
x_norm.shape
```

Out[8]:
```
(24000, 3)
```

K-means Implement (1)

In [12]:
```python
def rand_center(data,k):
    """
    >>> Function you need to write
    >>> Select "k" random points from "data" as the initial centroids.
    """
    centers = sample(data, k)
    return centers
```

In [13]:
```python
def distance(point1, point2):
    # Calculate Euclidean distance
    return math.sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2 + (point1[2]-point2[2])**2)
```

In [14]:
```python


def converged(centroids1, centroids2):
    """
    >>> Function you need to write
    >>> check whether centroids1==centroids2
    >>> add proper code to handle infinite loop if it never converges
    """
    # In order to fairly compare with sklearn.cluster.KMeans I used same sigma as in sklearn.cluster.KMeans
    sigma = 0.01 * np.mean(np.var(temp, axis=0))
    total_dis = 0
    for i in range(len(centroids1)):
        total_dis += distance(centroids1[i], centroids2[i])**2
    if total_dis > sigma:
        return False
    return True
```

K-means Implement (2)

In [15]:
```python
def average(points):
    array = np.array(points)
    means = np.mean(array, axis = 0)
    return means.tolist()
```

In [16]:
```python
def update_centroids(data, centroids, k):
    """
    >>> Function you need to write
    >>> Assign each data point to its nearest centroid based on the Euclidean dis
    tance
    >>> Update the cluster centroid to the mean of all the points assigned to tha
    t cluster
    """
    new_centroids_points = [[] for i in range(k)]
    label = []
    for point in data:
        mem = 0
        best = float('inf')
        for i in range(len(centroids)):
            dis = distance(np.array(point), centroids[i])
            if dis < best:
                best = dis
                mem = i
        new_centroids_points[mem].append(point)
        label.append(mem)
    # updata centroids
    new_centroids = []
    for i in range(len(centroids)):
        new_centroids.append(average(new_centroids_points[i]))
    return np.array(new_centroids), label
```

Evaluation Metrics Implement

```
In [17]: def SSE(data, centroids, label):
             sum = 0
             for i in range(len(label)):
                 sum += distance(data[i],centroids[label[i]])**2
             return sum
```

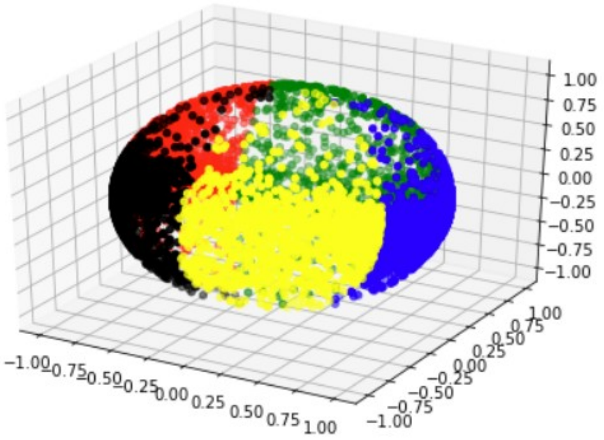
Results and Follow-up Discussion

Problem 1

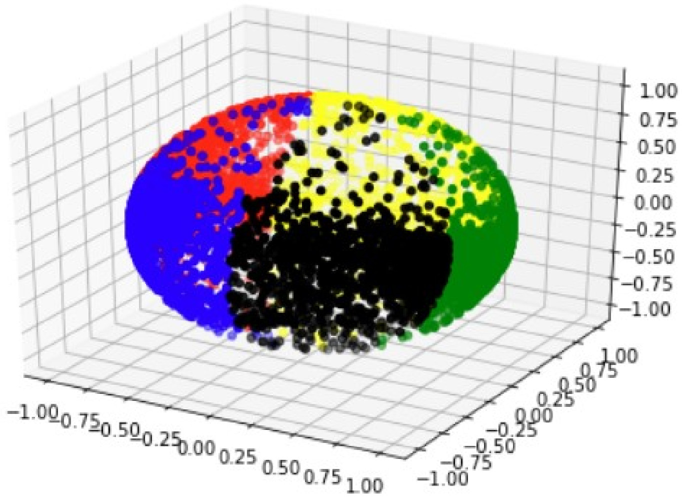
In [22]:
```
mini = float('inf')
index = 0
for i in range(20):
    centroids, label = result[i]
    loss = SSE(temp, centroids, label)

    print('Round {}, loss {}'.format(i,loss))
    if loss < mini:
        mini = loss
        index = i
print('Best Round: {}'.format(index))
```

```
Round 0, loss 2195.0811397830944
Round 1, loss 2213.914073166793
Round 2, loss 2260.256607363182
Round 3, loss 2215.320217505542
Round 4, loss 2017.9024393642492
Round 5, loss 2225.8271886956472
Round 6, loss 2020.6892742243065
Round 7, loss 2025.3657581544173
Round 8, loss 2234.1409036770833
Round 9, loss 2013.390764008825
Round 10, loss 2195.6686506647757
Round 11, loss 2019.3848016940874
Round 12, loss 2018.325603870741
Round 13, loss 2022.1620048861037
Round 14, loss 2196.8523228071867
Round 15, loss 2207.1849687444637
Round 16, loss 2220.136502247267
Round 17, loss 2189.531204864428
Round 18, loss 2215.0931373472463
Round 19, loss 2017.9822293505276
Best Round: 9
```

I run 20 rounds of K-means and get the Best K-means as my model to do following experiments



Clustering results of my best model



Clustering results of Sklearn’s K-means model

Picture below shows the SSE loss of sklearn K-means. I order to fairly compared with my model , I set tol = 1e-2.

```
In [24]: kmeans = KMeans(n_clusters=5,tol=1e-2).fit(temp)

In [25]: kmeans.cluster_centers_

Out[25]: array([[ 0.95409025,  0.0731525 , -0.04111057],
                [-0.97688861,  0.03394283, -0.01458933],
                [ 0.03165921,  0.77713444, -0.20813076],
                [-0.56992415, -0.69010576,  0.12528131],
                [ 0.52460191, -0.70539153,  0.15847814]])

In [26]: kmeans.labels_

Out[26]: array([0, 1, 1, ..., 1, 0, 1], dtype=int32)

In [27]: kmeans.n_iter_

Out[27]: 2

In [28]: loss = SSE(temp, kmeans.cluster_centers_, kmeans.labels_)
         loss

Out[28]: 2014.180866223673
```

Problem 2

The SSE loss of my best model is 2013.39
The SSE loss of Sklearn K-means is 2010.28

According to SSE loss, Sklearn K-means clustering implementation is better

why it is important to choose proper initial centroids?

By choosing proper initial centroids, not only can we improve the clustering proformance, but also we can speed up our algorithm (The n_iter of Sklearn is only 2).

Problem 3

The Impurity (Gini Index) and percentage of defaults for different data types.

For each data type, I report three outputs.

1, Impurity (Gini Index)

2, Label distribution within each cluster. E.g., if cluster 2 has 400 samples and 100 of them are label 1. Cluster2: label1(local) = 25%

3, Label distribution in the whole data set. E.g., if cluster 2 has 100 sample with label 1, the whole dataset has 1000 samples with label 1. Cluster2 : label1(global) = 10%

Data type SEX

Out[47]:

	cluster0	cluster1	cluster2	cluster3	cluster4
Impurity (Gini Index)	0.47	0.48	0.48	0.49	0.50
label1(local)	38.379%	40.659%	39.635%	42.672%	46.061%
label2(local)	61.621%	59.341%	60.365%	57.328%	53.939%
label1(global)	58.01%	5.07%	23.36%	5.55%	8.01%
label2(global)	60.89%	4.84%	23.26%	4.88%	6.13%

```
Data type EDUCATION
```

Out[48]:

	cluster0	cluster1	cluster2	cluster3	cluster4
Impurity (Gini Index)	0.63	0.62	0.62	0.64	0.61
label0(local)	0.070%	0.085%	0	0	0
label1(local)	37.919%	35.249%	33.196%	27.773%	24.000%
label2(local)	44.244%	47.675%	50.098%	50.121%	54.424%
label3(local)	16.533%	15.723%	14.935%	18.785%	20.242%
label4(local)	0.460%	0.085%	0.411%	0.648%	0.242%
label5(local)	0.669%	1.183%	1.180%	2.024%	0.970%
label6(local)	0.105%	0	0.179%	0.648%	0.121%
label0(global)	90.91%	9.09%	0	0	0
label1(global)	64.36%	4.93%	21.96%	4.06%	4.69%
label2(global)	56.52%	5.02%	24.95%	5.51%	8.00%
label3(global)	59.90%	4.70%	21.10%	5.86%	8.44%
label4(global)	64.71%	0.98%	22.55%	7.84%	3.92%
label5(global)	44.24%	6.45%	30.41%	11.52%	7.37%
label6(global)	42.86%	0	28.57%	22.86%	5.71%

```
Data type MARRIAGE
```

Out[49]:

	cluster0	cluster1	cluster2	cluster3	cluster4
Impurity (Gini Index)	0.51	0.50	0.51	0.51	0.52
label0(local)	0.209%	0.254%	0.089%	0.162%	0.303%
label1(local)	45.039%	41.589%	47.058%	45.020%	43.515%
label2(local)	53.581%	57.143%	52.102%	53.684%	54.303%
label3(local)	1.171%	1.014%	0.751%	1.134%	1.879%
label0(global)	66.67%	6.67%	11.11%	4.44%	11.11%
label1(global)	59.50%	4.53%	24.24%	5.12%	6.61%
label2(global)	59.88%	5.27%	22.70%	5.17%	6.98%
label3(global)	62.92%	4.49%	15.73%	5.24%	11.61%

Out[52]:

	cluster0	cluster1	cluster2	cluster3	cluster4
Impurity (Gini Index)	0.70	0.68	0.70	0.71	0.71
label1(local)	33.373%	35.503%	28.278%	32.794%	34.485%
label2(local)	36.371%	38.039%	41.656%	34.899%	32.970%
label3(local)	21.498%	20.626%	21.123%	23.239%	22.364%
label4(local)	7.698%	4.987%	7.619%	8.178%	9.333%
label5(local)	1.060%	0.845%	1.324%	0.891%	0.848%
label1(global)	61.67%	5.41%	20.37%	5.22%	7.33%
label2(global)	58.15%	5.02%	25.96%	4.80%	6.06%
label3(global)	59.70%	4.73%	22.87%	5.56%	7.15%
label4(global)	59.87%	3.20%	23.10%	5.48%	8.35%
label5(global)	58.24%	3.83%	28.35%	4.21%	5.36%

Label1	$age \leq 30$
Label2	$30 < age \leq 40$
Label3	$40 < age \leq 50$
Label4	$50 < age \leq 60$
Label5	$60 < age$

The mean of 6 balance records for each cluster.

```
In [51]: for i in range(5):
             print('The mean balance of cluster {} is {}'.format(i,np.mean(my_data_points[i])))

The mean balance of cluster 0 is 7233.300409083514
The mean balance of cluster 1 is 39374.09227951535
The mean balance of cluster 2 is 122123.14439873605
The mean balance of cluster 3 is 60925.024561403516
The mean balance of cluster 4 is 25606.757474747472
```

According to picture above, we can see that cluster1 has the lease mean balance, the cluster 2 has the most mean balance.

In order to find out how data on previous table affect means balance. I compared cluster 1 and 2 on sex, education, marriage, and age. For sex and marriage, cluster 1 and 2 roughly the same.
The difference between cluster 1 and cluster 2 appears on age and education. Cluster 2 have more percentage of sample with age between 30~ 40 and less percentage of sample with age less than 30.
For education, compared with cluster1, cluster 2 have more percentage of label2 and less percentage of label1.
In order to find out how age and education level affect means balance, I calculate means balance with different age range and education level.

```
data type Age
mean balance of label 0 if 0
mean balance of label 1 if 34417.70070437661
mean balance of label 2 if 42495.08829431427
mean balance of label 3 if 40723.74599793434
mean balance of label 4 if 42624.85556760655
mean balance of label 5 if 51348.63090676882
data type Education
mean balance of label 0 if 0
mean balance of label 1 if 34417.70070437661
mean balance of label 2 if 42495.08829431427
mean balance of label 3 if 40723.74599793434
mean balance of label 4 if 42624.85556760655
mean balance of label 5 if 51348.63090676882
```

After we calculate the mean balance over whole dataset.
Mean balance will increase with age and education level.

data type Age

Out[60]:

	cluster0	cluster1	cluster2	cluster3	cluster4
label0	0.000000	0.000000	0.000000	0.000000	0.000000
label1	9133.666736	41067.355952	106610.549441	54755.353086	27111.704745
label2	5962.887078	39419.242222	125684.279519	63238.163186	22730.751225
label3	6420.190831	36309.348361	129643.878634	65207.375145	26613.977416
label4	7438.636926	39979.646893	134500.510172	64597.671617	27322.059524
label5	5992.826754	37432.250000	150187.792793	51996.136364	30779.023810

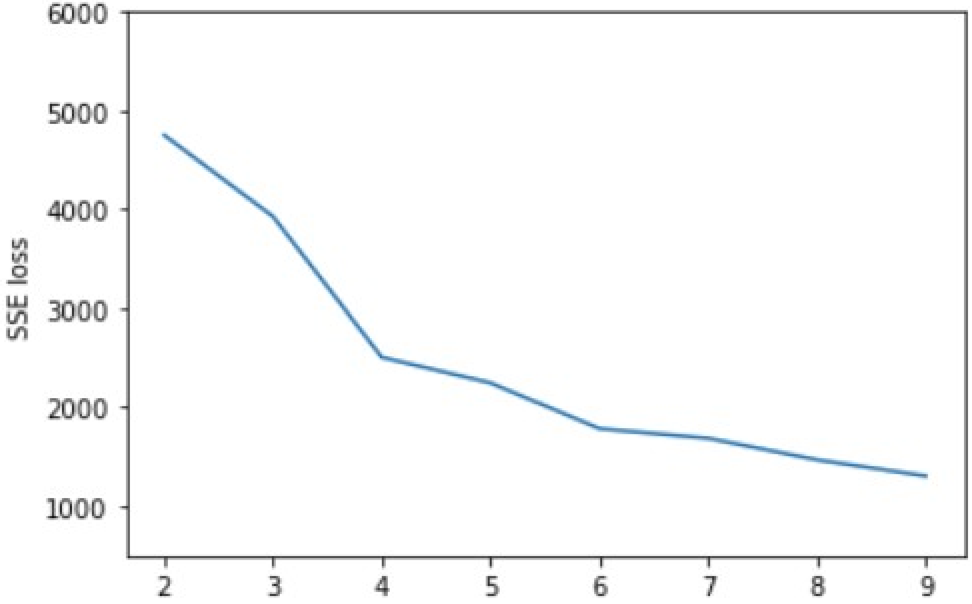
data type Education

Out[61]:

	cluster0	cluster1	cluster2	cluster3	cluster4
label0	1562.066667	41394.000000	0.000000	0.000000	0.000000
label1	4564.758428	36330.366107	131751.151491	61609.755588	21177.859428
label2	8907.524008	40827.699468	118123.478698	60543.488153	26837.472346
label3	9016.021510	41682.622760	112435.844311	59176.012213	27536.893713
label4	2524.063131	79859.166667	121337.789855	66047.770833	23021.291667
label5	8254.645833	37767.500000	140977.085859	71705.093333	25678.354167
label6	2648.211111	0.000000	141731.250000	72999.458333	32203.000000

Interestingly, K-means has grouped these samples with different mean balance level.

Problem 4



Picture above is SSE loss with different K value.
According ot the elbow method, the best K shoud be 6

```
Data type SEX
```

Out[95]:

	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5
Impurity (Gini Index)	0.49	0.47	0.50	0.48	0.48	0.48
label1(local)	42.438%	38.393%	46.378%	39.825%	41.194%	39.427%
label2(local)	57.562%	61.607%	53.622%	60.175%	58.806%	60.573%
label1(global)	6.09%	57.75%	7.89%	1.92%	4.44%	21.91%
label2(global)	5.40%	60.58%	5.97%	1.89%	4.14%	22.01%

Data type EDUCATION

Out[96]:

	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5
Impurity (Gini Index)	0.63	0.63	0.60	0.62	0.62	0.62
label1(local)	28.267%	37.706%	23.096%	43.982%	33.855%	33.453%
label2(local)	50.954%	44.412%	54.737%	40.700%	48.532%	49.858%
label3(local)	17.401%	16.600%	20.743%	13.129%	16.438%	14.963%
label4(local)	0.514%	0.469%	0.248%	0	0.098%	0.436%
label5(local)	2.203%	0.652%	1.053%	1.751%	0.978%	1.119%
label6(local)	0.661%	0.105%	0.124%	0	0	0.171%
label1(global)	4.56%	63.68%	4.41%	2.38%	4.09%	20.88%
label2(global)	6.18%	56.45%	7.87%	1.66%	4.42%	23.42%
label3(global)	5.99%	59.85%	8.46%	1.52%	4.24%	19.93%
label4(global)	6.86%	65.69%	3.92%	0	0.98%	22.55%
label5(global)	13.82%	42.86%	7.83%	3.69%	4.61%	27.19%
label6(global)	25.71%	42.86%	5.71%	0	0	25.71%
label0(local)	0	0.056%	0	0.438%	0.098%	0
label0(global)	0	72.73%	0	18.18%	9.09%	0

Data type MARRIAGE

Out[97]:

	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5
Impurity (Gini Index)	0.51	0.51	0.51	0.50	0.51	0.51
label0(local)	0.147%	0.217%	0.248%	0	0.391%	0.076%
label1(local)	45.595%	44.937%	42.848%	44.639%	42.857%	47.184%
label2(local)	53.010%	53.668%	54.923%	54.486%	55.675%	52.077%
label3(local)	1.248%	1.177%	1.981%	0.875%	1.076%	0.664%
label0(global)	4.44%	68.89%	8.89%	0	8.89%	8.89%
label1(global)	5.72%	59.07%	6.37%	1.88%	4.03%	22.92%
label2(global)	5.63%	59.69%	6.91%	1.94%	4.43%	21.40%
label3(global)	6.37%	62.92%	11.99%	1.50%	4.12%	13.11%

Data type Age

Out[98]:

	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5
Impurity (Gini Index)	0.71	0.70	0.71	0.69	0.68	0.70
label1(local)	30.984%	33.551%	35.046%	27.352%	36.106%	28.276%
label2(local)	35.609%	36.269%	32.632%	42.232%	37.867%	41.760%
label3(local)	24.156%	21.435%	22.105%	23.632%	19.961%	20.994%
label4(local)	8.297%	7.687%	9.350%	6.346%	5.088%	7.624%
label5(local)	0.954%	1.058%	0.867%	0.438%	0.978%	1.346%
label1(global)	5.44%	61.69%	7.29%	1.61%	4.75%	19.21%
label2(global)	5.41%	57.70%	5.88%	2.15%	4.31%	24.55%
label3(global)	6.37%	59.24%	6.91%	2.09%	3.95%	21.44%
label4(global)	6.13%	59.49%	8.19%	1.57%	2.82%	21.80%
label5(global)	4.98%	57.85%	5.36%	0.77%	3.83%	27.20%

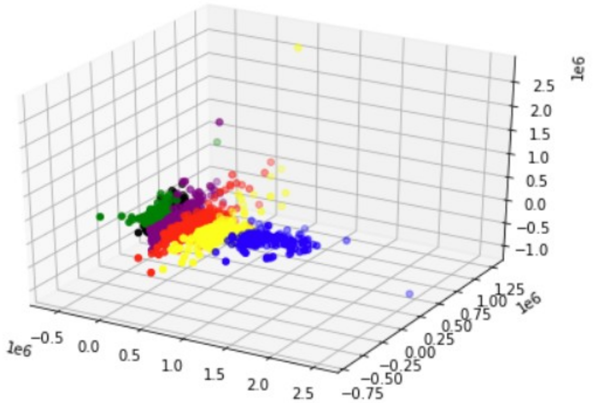
The mean of 6 balance records for each cluster.

```
In [100]: for i in range(6):
              print('The mean balance of cluster {} is {}'.format(i,np.mean(my_data_points_p4[i])))

The mean balance of cluster 0 is 69938.33724914343
The mean balance of cluster 1 is 7313.208943545184
The mean balance of cluster 2 is 27974.34385964912
The mean balance of cluster 3 is 15782.622173595915
The mean balance of cluster 4 is 42530.03832354859
The mean balance of cluster 5 is 124239.83023579238
```

According to picture above, we can see that cluster1 has the lease mean balance, the cluster 5 has the most mean balance.

Problem 5

In [119]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure()
ax=Axes3D(fig)
ax.scatter(my_data_points_p5[0][:,0], my_data_points_p5[0][:,1], my_data_points_p5[0][:,2],color='red')
ax.scatter(my_data_points_p5[1][:,0], my_data_points_p5[1][:,1], my_data_points_p5[1][:,2],color='green')
ax.scatter(my_data_points_p5[2][:,0], my_data_points_p5[2][:,1], my_data_points_p5[2][:,2],color='blue')
ax.scatter(my_data_points_p5[3][:,0], my_data_points_p5[3][:,1], my_data_points_p5[3][:,2],color='yellow')
ax.scatter(my_data_points_p5[4][:,0], my_data_points_p5[4][:,1], my_data_points_p5[4][:,2],color='black')
ax.scatter(my_data_points_p5[5][:,0], my_data_points_p5[5][:,1], my_data_points_p5[5][:,2],color='purple')
plt.show()
```




```
Round 2, loss 479088484861377.56
Round 3, loss 304044268046601.9
Round 4, loss 226151277677335.0
Round 5, loss 210264773140709.78
Round 6, loss 166643734517858.0
Round 7, loss 151112397221642.4
Round 8, loss 138655450020978.77
Round 9, loss 125858335870665.77
Best Round: 9
```

Picture above is the K-means result without normalization. We can see that the data point is more sparse (Note the axis number). As a result, SSE loss is much larger. Which means normalization can improve the performance of K-means

```
data type Age
```

Out[107]:

	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5
label0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
label1	62099.819905	9281.482212	29144.960836	18219.141333	43635.879855	108380.170467
label2	74082.536426	5945.133984	25761.201455	18586.907599	41858.487941	127664.049576
label3	73269.605876	6546.756674	28609.327264	9325.089506	41797.097222	132265.535682
label4	73570.011799	7484.560164	29549.193157	11526.982759	42867.993590	136942.570896
label5	53903.807692	6079.227373	30779.023810	3300.166667	40908.116667	154037.915493

Similarity, K-means has grouped these samples with different mean balance level. This time the total cluster is 6.

```
data type Education
```

Out[108]:

	cluster0	cluster1	cluster2	cluster3	cluster4	cluster5
label0	0.000000	1909.729167	0.000000	171.416667	41394.000000	0.000000
label1	71381.360173	4562.894753	24918.077301	11844.288557	40951.474952	133867.641251
label2	69249.839577	9007.442963	28612.119721	20435.758065	43184.355511	120320.741030
label3	67343.255274	9157.528141	29615.469154	14593.961111	43988.447421	114193.444233
label4	70054.452381	2880.181592	25925.541667	0.000000	79859.166667	121337.789855
label5	87502.938889	8414.379928	29512.843137	19365.604167	36573.616667	142358.556497
label6	70998.018519	2648.211111	32203.000000	0.000000	0.000000	151369.555556