Name: SUN RUI
Student ID: 18083229g

Answer 1:

a)

sort the column Age: [17, 18, 18, 22, 22, 27, 29, 34, 35, 38, 39, 39, 46, 54, 59]

sort the column Monthly Income: [3000, 4000, 7500, 7800, 7800, 7900, 8500, 14000, 18000, 21000, 24700, 30000, 31000, 31110, 40500]

sort the column Service Plan: [100, 100, 100, 200, 200, 400, 600, 600, 600, 600, 800, 1000, 1600, 1600, 1600]

sort the column Extra Usage: [0, 0, 0, 7, 25, 31, 31, 50, 54, 64, 211, 254, 290, 303, 311]

Equal-width:

Age: (59-17)/3=14 → a=[17, 31), b=[31, 45), c=[45, 59]

Monthly Income: (40500-3000)/3=12500 → x=[3000, 15500), y=[15500, 28000), z=[28000, 40500]

Service Plan: (1600-100)/3=500 → A=[100, 600), B=[600, 1100), C=[1100, 1600]

Extra Usage: (311-0)/3=103.67 →X=[0, 103.67), Y=[103.67, 207.34), Z=[207.34, 311.01]

Change the data into below table according to above ranges:

(0)	(i)	(ii)	(iii)	(iv)	(v)	(vi)
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**
1	54 c	FEMALE	3000 x	YES	100 A	0 X
2	59 c	FEMALE	4000 x	NO	600 B	54 X
3	38 b	MALE	7800 x	NO	200 A	31 X
4	18 a	FEMALE	8500 x	NO	600 B	311 Z
5	27 a	MALE	14000 x	YES	100 A	211 Z
6	29 a	FEMALE	31000 z	YES	1600 C	25 X
7	17 a	MALE	7500 x	NO	600 B	254 Z
8	22 a	FEMALE	7900 x	NO	200 A	31 X
9	34 b	MALE	24700 y	NO	100 A	7 X
10	46 c	FEMALE	31110 z	YES	600 B	0 X
11	39 b	FEMALE	21000 y	YES	800 B	64 X
12	35 b	FEMALE	30000 z	NO	1600 C	0 X
13	39 b	MALE	40500 z	YES	1600 C	50 X
14	18 a	MALE	7800 x	NO	1000 C	290 Z
15	22 a	MALE	18000 y	YES	400 A	303 Z

initial cluster centers:

	Locus					
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**
1 (1)	54 c	FEMALE	3000 x	YES	100 A	0 X
2 (8)	22 a	FEMALE	7900 x	NO	200 A	31 X
3 (15)	22 a	MALE	18000 y	YES	400 A	303 Z

cluster distances of first step:

(0)	(i)	(ii)	(iii)	(iv)	(v)	(vi)	distance		
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**	**1**	**2**	**3**
1	54 c	FEMALE	3000 x	YES	100 A	0 X	**0**	2	4
2	59 c	FEMALE	4000 x	NO	600 B	54 X	**2**	2	6
3	38 b	MALE	7800 x	NO	200 A	31 X	3	**2**	4
4	18 a	FEMALE	8500 x	NO	600 B	311 Z	4	**2**	4
5	27 a	MALE	14000 x	YES	100 A	211 Z	3	3	**1**
6	29 a	FEMALE	31000 z	YES	1600 C	25 X	**3**	3	4
7	17 a	MALE	7500 x	NO	600 B	254 Z	5	**3**	3
8	22 a	FEMALE	7900 x	NO	200 A	31 X	2	**0**	4
9	34 b	MALE	24700 y	NO	100 A	7 X	4	**3**	3
10	46 c	FEMALE	31110 z	YES	600 B	0 X	**2**	4	5
11	39 b	FEMALE	21000 y	YES	800 B	64 X	**3**	4	4
12	35 b	FEMALE	30000 z	NO	1600 C	0 X	4	**3**	6
13	39 b	MALE	40500 z	YES	1600 C	50 X	**4**	5	4
14	18 a	MALE	7800 x	NO	1000 C	290 Z	5	**3**	3
15	22 a	MALE	18000 y	YES	400 A	303 Z	4	4	**0**

update cluster centers:

	Locus					
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**
1 (1)	54 c	FEMALE	**z**	YES	**B**	0 X
2 (8)	22 a	**MALE**	7900 x	NO	200 A	31 X
3 (15)	22 a	MALE	18000 y	YES	400 A	303 Z

Equal-depth:

Age: a={17,18,18,22,22}, b={27,29,34,35,38}, c={39,39,46,54,59}

Monthly Income: x={3000,4000,7500,7800,7800}, y={7900,8500,14000,18000,21000}, z={24700,30000,31000,31110,40500}

Service Plan: A={100,100,100,200,200}, B={400,600,600,600,600}, C={800,1000,1600,1600,1600}

Extra Usage: X={0,0,0,7,25}, Y={31,31,50,54,64}, Z={211,254,290,303,311}

Change the data into below table according to above rules:

(0)	(i)	(ii)	(iii)	(iv)	(v)	(vi)
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**
1	54 c	FEMALE	3000 x	YES	100 A	0 X
2	59 c	FEMALE	4000 x	NO	600 B	54 Y
3	38 b	MALE	7800 x	NO	200 A	31 Y
4	18 a	FEMALE	8500 y	NO	600 B	311 Z
5	27 b	MALE	14000 y	YES	100 A	211 Z
6	29 b	FEMALE	31000 z	YES	1600 C	25 X
7	17 a	MALE	7500 x	NO	600 B	254 Z
8	22 a	FEMALE	7900 y	NO	200 A	31 Y
9	34 b	MALE	24700 z	NO	100 A	7 X
10	46 c	FEMALE	31110 z	YES	600 B	0 X
11	39 c	FEMALE	21000 y	YES	800 C	64 Y
12	35 b	FEMALE	30000 z	NO	1600 C	0 X
13	39 c	MALE	40500 z	YES	1600 C	50 Y
14	18 a	MALE	7800 x	NO	1000 C	290 Z
15	22 a	MALE	18000 y	YES	400 B	303 Z

initial cluster centers:

	Locus					
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**
1(1)	54 c	FEMALE	3000 x	YES	100 A	0 X
2(8)	22 a	FEMALE	7900 y	NO	200 A	31 Y
3(15)	22 a	MALE	18000 y	YES	400 B	303 Z

cluster distances of first step:

(0)	(i)	(ii)	(iii)	(iv)	(v)	(vi)	distance		
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**	**1**	**2**	**3**
1	54 c	FEMALE	3000 x	YES	100 A	0 X	**0**	4	5
2	59 c	FEMALE	4000 x	NO	600 B	54 Y	**3**	3	5
3	38 b	MALE	7800 x	NO	200 A	31 Y	4	**3**	5
4	18 a	FEMALE	8500 y	NO	600 B	311 Z	5	**2**	2
5	27 b	MALE	14000 y	YES	100 A	211 Z	4	4	**2**
6	29 b	FEMALE	31000 z	YES	1600 C	25 X	**3**	5	5
7	17 a	MALE	7500 x	NO	600 B	254 Z	5	4	**2**
8	22 a	FEMALE	7900 y	NO	200 A	31 Y	4	**0**	4
9	34 b	MALE	24700 z	NO	100 A	7 X	**4**	4	5
10	46 c	FEMALE	31110 z	YES	600 B	0 X	**2**	5	4
11	39 c	FEMALE	21000 y	YES	800 C	64 Y	**3**	3	4
12	35 b	FEMALE	30000 z	NO	1600 C	0 X	**4**	4	6
13	39 c	MALE	40500 z	YES	1600 C	50 Y	**4**	5	4
14	18 a	MALE	7800 x	NO	1000 C	290 Z	5	4	**3**
15	22 a	MALE	18000 y	YES	400 B	303 Z	5	4	**0**

Update cluster centers:

	Locus					
Ref	**Age**	**Sex**	**Monthly Income**	**Marital Status**	**Service Plan**	**Extra Usage**
1 (1)	54 c	FEMALE	**z**	YES	**C**	0 X
2 (8)	22 a	FEMALE	7900 y	NO	200 A	31 Y
3 (15)	22 a	MALE	18000 y	YES	400 B	303 Z

b)

the clustering result of k-means by Python with package sklearn (**initial centers are 1, 8 and 15**):

samples of 1, 2, 10, 11 are cluster 1

samples of 3, 4, 6, 8, 9, 12 are cluster 2

samples of 5, 7, 13, 14, 15 are cluster 3

the program:

```
# -*- encoding:utf-8 -*-

# Name: SUN RUI    ID:18083229g

import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.cluster import KMeans


def load_data(file_path):
    df_data = pd.read_csv(file_path)
    df_data.columns = ["Age", "Sex", "MonthlyIncome", "MaritalStatus",
"ServicePlan", "ExtraUsage"]
    scaled_data = preprocessing.scale(df_data)
    return scaled_data

def do_k_means(data):
    centers = np.vstack((data[0], data[7], data[14]))
    k_means = KMeans(n_clusters=3, init=centers, n_init=1, max_iter=1000)
    return k_means.fit_predict(data), k_means.fit(data).cluster_centers_

def print_clusters(cluster_index):
    cluster_dict = {0:[],1:[],2:[]}
    for i in range(len(cluster_index)):
        cluster_dict[cluster_index[i]].append(i+1)
    print("cluster result: {}".format(cluster_dict))

if __name__ == '__main__':
    df_scaled_data = load_data("data.csv")
    cluster_index, cluster_center = do_k_means(df_scaled_data)
    print("cluster index: {}".format(cluster_index))
    print_clusters(cluster_index)
    print("cluster centers:\n{}".format(cluster_center))
```

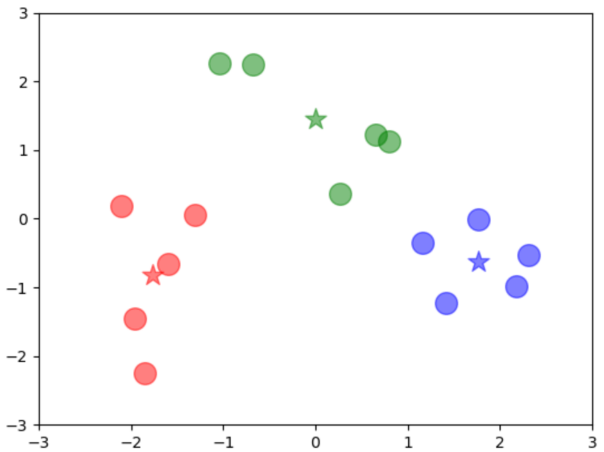
c)

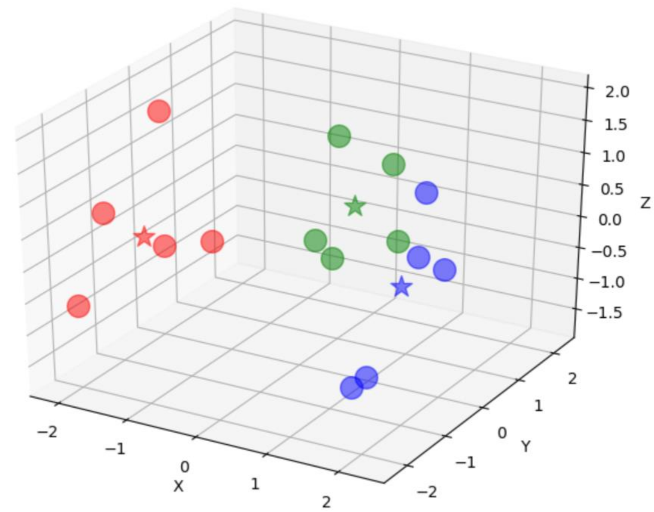
to compare the performances between k-means and k-modes, I plan to use scatter picture to show the differences. Because the data has six attributes, I have to reduce dimensions from 6 to 2 or from 6 to 3 by PCA, so the information of origin data will be lossy and the results are a little different from part b). Let us see the results of k-means firstly: (**the code "*ref1*" is at the end of this assignment**)

The cluster result: (**initial centers are 1, 8 and 15**)

samples of 6, 10, 11, 12, 13 are cluster 1

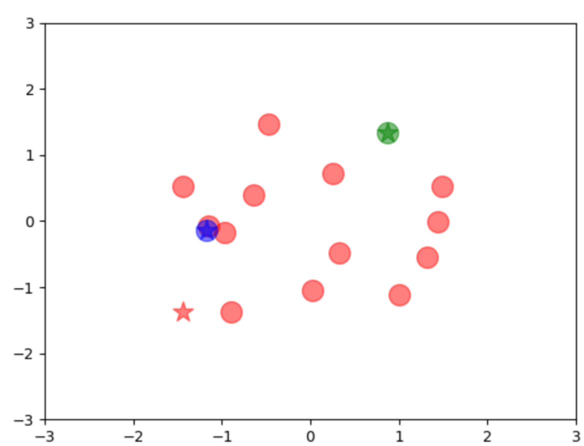
samples of 1, 2, 3, 8, 9 are cluster 2
samples of 4, 5, 7, 14, 15 are cluster 3
the scatter pictures of k-means:

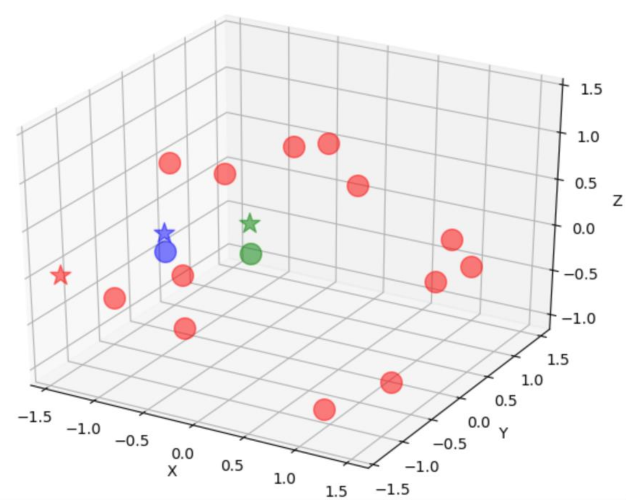




Let us see the results of k-modes: (**initial centers are 1, 8 and 15, the code "*ref2*" is at the end of this assignment**)
① Equal-width without reducing dimension:
samples of 1, 2, 6, 10, 11, 12, 13 are cluster 1
samples of 3, 4, 7, 8, 9, 14 are cluster 2
samples of 5, 15 are cluster 3
The result of cluster seems not good, because the cluster 3 only includes 2 samples.

reducing dimension:

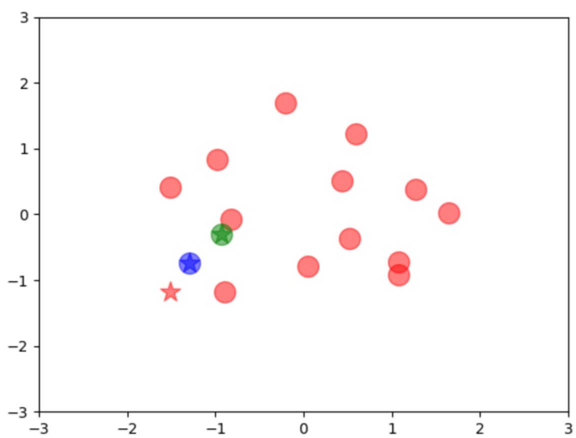


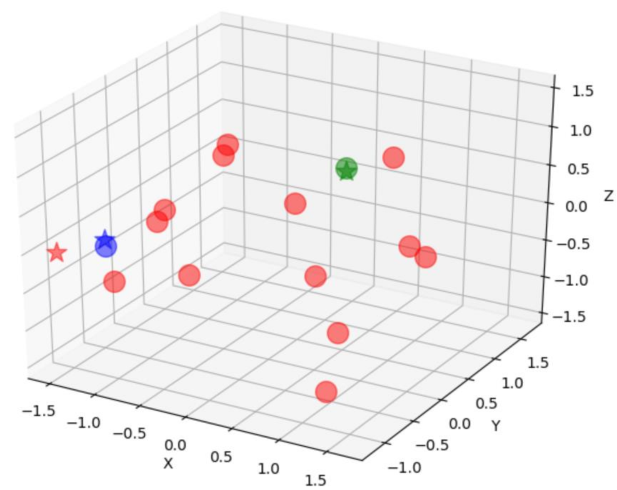


The clusters are different between 2-dimensions and 3-dimensions:
2-dimension: 0: [1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15], 1: [4], 2: [11]
3-dimension: 0: [2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15], 1: [1], 2: [11]

In conclusion, the result of k-modes of Equal-width is terrible.
② Equal-depth without reducing dimension:
samples of 1, 6, 10, 11, 12, 13 are cluster 1
samples of 2, 3, 4, 8 are cluster 2
samples of 5, 7, 9, 14, 15 are cluster 3
The result of cluster seems not bad, every cluster has about average number of samples.

reducing dimension:





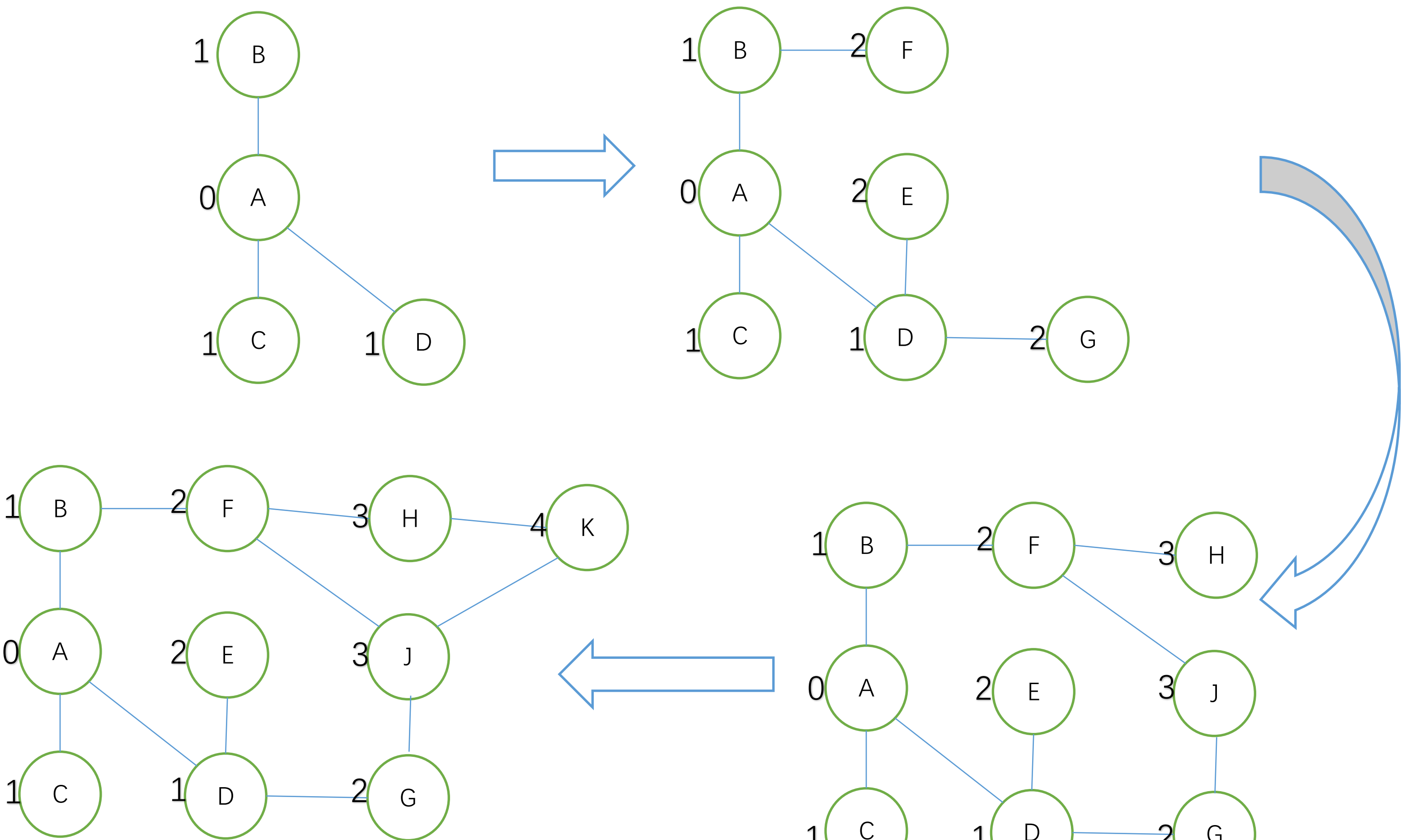
The clusters are different between 2-dimensions and 3-dimensions:
2-dimension: 0: [2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15], 1: [1], 2: [10]
3-dimension: 0: [1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15], 1: [9], 2: [10]
In conclusion, the result of k-modes of Equal-depth is also not good.

Draw a conclusion about part c), the k-means algorithm has better performance before and after reducing dimensions than k-modes algorithm. But the data must be done normalized.
The performance of equal-depth is better than equal-width before reducing dimension. However, these two ways will loss information after reducing dimension seriously.

Answer 2:

a)

Breadth First Search:



path: ABCDFEGHJK

A- K:

① A -> B -> F -> H -> K

② A -> B -> F -> J -> K

③ A -> D -> G -> J -> K

All shortest paths between every two nodes:

	A	B	C	D	E	F	G	H	J	K
A	-	AB	AC	AD	ADE	ABF	ADG	ABFH	ABFJ ADGJ	ABFHK ABFJK ADGJK
B	-	-	BAC	BAD	BFE	BF	BADG BFJG BFEG	BFH	BFJ	BFHK BFJK

C	-	-	-	CD	CDE	CABF CDEF	CDG	CABFH CDEFH CDGJH	CDGJ	CDGJK
D	-	-	-	-	DE	DEF	DG	DEFH DGJH	DGJ	DGJK
E	-	-	-	-	-	EF	EG	EFH	EFJ EGJ	EFHK EGJK EFJK
F	-	-	-	-	-	-	FJG FEG	FH	FJ	FHK FJK
G	-	-	-	-	-	-	-	GJH	GJ	GJK
H	-	-	-	-	-	-	-	-	HJ	HK
J	-	-	-	-	-	-	-	-	-	JK

b)

The total number of shortest paths between every node:

σAB=1	σAC=1	σAD=1	σAE=1	σAF=1	σAG=1	σAH=1	σAJ=2	σAK=3
σBC=1	σBD=1	σBE=1	σBF=1	σBG=3	σBH=1	σBJ=1	σBK=2	
σCD=1	σCE=1	σCF=2	σCG=1	σCH=3	σCJ=1	σCK=1		
σDE=1	σDF=1	σDG=1	σDH=2	σDJ=1	σDK=1			
σEF=1	σEG=1	σEH=1	σEJ=2	σEK=3				
σFG=2	σFH=1	σFJ=1	σFK=2					
σGH=1	σGJ=1	σGK=1						
σHJ=1	σHK=1							
σJK=1								

The number of shortest paths of above table that pass along with every pair of neighbor node:

	AB	AC	AD	BF	CD	DE	DG	EG	EF	FH	FJ	GJ	HJ	HK	JK
AB	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
AD	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
AE	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
AF	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
AG	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
AH	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
AJ	1	0	1	1	0	0	1	0	0	0	1	1	0	0	0
AK	2	0	1	2	0	0	1	0	0	1	1	1	0	1	2
BC	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
BD	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

BE	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
BF	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
BG	1	0	1	2	0	0	1	1	1	0	1	1	0	0	0
BH	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
BJ	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
BK	0	0	0	2	0	0	0	0	0	1	1	0	0	1	1
CD	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
CE	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
CF	1	1	0	1	1	1	0	0	1	0	0	0	0	0	0
CG	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
CH	1	1	0	1	2	1	1	0	1	2	0	1	1	0	0
CJ	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0
CK	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1
DE	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
DF	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
DG	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
DH	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0
DJ	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
DK	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
EF	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
EG	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
EH	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
EJ	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0
EK	0	0	0	0	0	0	0	1	2	1	1	1	0	1	2
FG	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0
FH	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
FJ	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
FK	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
GH	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
GJ	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
GK	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
HJ	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
HK	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
JK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

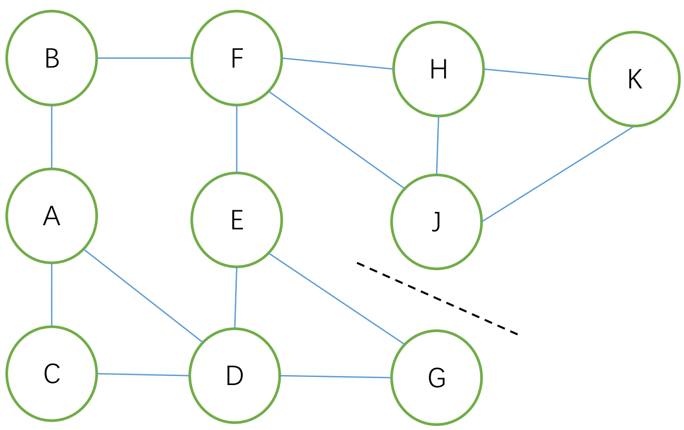
Edge-Betweenness values of the edges:

Vertex	A	B	C	D	E	F	G	H	J	K
A	0	7.333	2.833	5.167						
B	7.333	0				9.667				
C	2.833		0	6.167						
D	5.167		6.167	0	5.333		9.0			

E				5.333	0	7.333	2.667			
F		9.667			7.333	0		6.833	5.5	
G				9.0	2.667		0		**10.333**	
H						6.833		0	2.833	2.667
J						5.5	**10.333**	2.833	0	6.333
K								2.667	6.333	0

c)

According to the result of part b), Edge-Betweenness value of JG is largest, so remove edge JG:



Update all shortest paths between every two nodes:

	A	B	C	D	E	F	G	H	J	K
A	-	AB	AC	AD	ADE	ABF	ADG	ABFH	ABFJ	ABFHK ABFJK
B	-	-	BAC	BAD	BFE	BF	BADG BFEG	BFH	BFJ	BFHK BFJK
C	-	-	-	CD	CDE	CABF CDEF	CDG	CABFH CDEFH	CABFJ	CABFHK CABFJK
D	-	-	-	-	DE	DEF	DG	DEFH	DEFJ	DEFHK DEFJK
E	-	-	-	-	-	EF	EG	EFH	EFJ	EFHK EFJK
F	-	-	-	-	-	-	FEG	FH	FJ	FHK FJK
G	-	-	-	-	-	-	-	GEFH	GEFJ	GEFJK GEFHK
H	-	-	-	-	-	-	-	-	HJ	HK

J	-	-	-	-	-	-	-	-	-	JK

Update the total number of shortest paths between every node:

σAB=1	σAC=1	σAD=1	σAE=1	σAF=1	σAG=1	σAH=1	σAJ=1	σAK=2
σBC=1	σBD=1	σBE=1	σBF=1	σBG=2	σBH=1	σBJ=1	σBK=2	
σCD=1	σCE=1	σCF=2	σCG=1	σCH=2	σCJ=1	σCK=2		
σDE=1	σDF=1	σDG=1	σDH=1	σDJ=1	σDK=2			
σEF=1	σEG=1	σEH=1	σEJ=1	σEK=2				
σFG=1	σFH=1	σFJ=1	σFK=2					
σGH=1	σGJ=1	σGK=2						
σHJ=1	σHK=1							
σJK=1								

Update the number of shortest paths of above table that pass along with every pair of neighbor node:

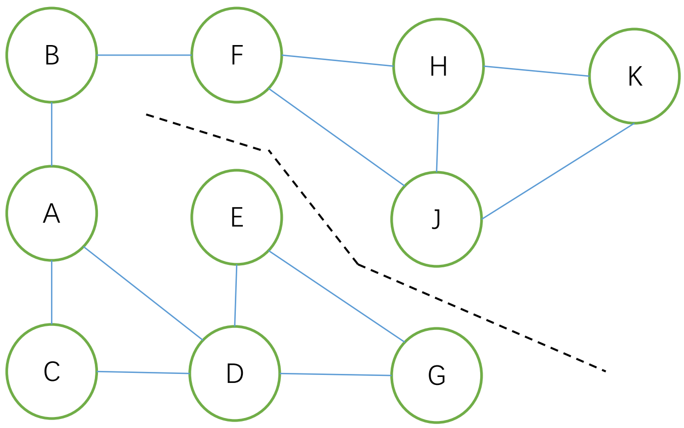
	AB	AC	AD	BF	CD	DE	DG	EG	EF	FH	FJ	HJ	HK	JK
AB	1	0	0	0	0	0	0	0	0	0	0	0	0	0
AC	0	1	0	0	0	0	0	0	0	0	0	0	0	0
AD	0	0	1	0	0	0	0	0	0	0	0	0	0	0
AE	0	0	1	0	0	1	0	0	0	0	0	0	0	0
AF	1	0	0	1	0	0	0	0	0	0	0	0	0	0
AG	0	0	1	0	0	0	1	0	0	0	0	0	0	0
AH	1	0	0	1	0	0	0	0	0	1	0	0	0	0
AJ	1	0	0	1	0	0	0	0	0	0	1	0	0	0
AK	2	0	0	2	0	0	0	0	0	1	1	0	1	1
BC	1	1	0	0	0	0	0	0	0	0	0	0	0	0
BD	1	0	1	0	0	0	0	0	0	0	0	0	0	0
BE	0	0	0	1	0	0	0	0	1	0	0	0	0	0
BF	0	0	0	1	0	0	0	0	0	0	0	0	0	0
BG	1	0	1	1	0	0	1	1	1	0	0	0	0	0
BH	0	0	0	1	0	0	0	0	0	1	0	0	0	0
BJ	0	0	0	1	0	0	0	0	0	0	1	0	0	0
BK	0	0	0	2	0	0	0	0	0	1	1	0	1	1
CD	0	0	0	0	1	0	0	0	0	0	0	0	0	0
CE	0	0	0	0	1	1	0	0	0	0	0	0	0	0
CF	1	1	0	1	1	1	0	0	1	0	0	0	0	0
CG	0	0	0	0	1	0	1	0	0	0	0	0	0	0
CH	1	1	0	1	1	1	0	0	1	2	0	0	0	0
CJ	1	1	0	1	0	0	0	0	0	0	1	0	0	0
CK	2	2	0	2	0	0	0	0	0	1	1	0	1	1
DE	0	0	0	0	0	1	0	0	0	0	0	0	0	0

DF	0	0	0	0	0	1	0	0	1	0	0	0	0	0
DG	0	0	0	0	0	0	1	0	0	0	0	0	0	0
DH	0	0	0	0	0	1	0	0	1	1	0	0	0	0
DJ	0	0	0	0	0	1	0	0	1	0	1	0	0	0
DK	0	0	0	0	0	2	0	0	2	1	1	0	1	1
EF	0	0	0	0	0	0	0	0	1	0	0	0	0	0
EG	0	0	0	0	0	0	0	1	0	0	0	0	0	0
EH	0	0	0	0	0	0	0	0	1	1	0	0	0	0
EJ	0	0	0	0	0	0	0	0	1	0	1	0	0	0
EK	0	0	0	0	0	0	0	0	2	1	1	0	1	1
FG	0	0	0	0	0	0	0	1	1	0	0	0	0	0
FH	0	0	0	0	0	0	0	0	0	1	0	0	0	0
FJ	0	0	0	0	0	0	0	0	0	0	1	0	0	0
FK	0	0	0	0	0	0	0	0	0	1	1	0	1	1
GH	0	0	0	0	0	0	0	1	1	1	0	0	0	0
GJ	0	0	0	0	0	0	0	1	1	0	1	0	0	0
GK	0	0	0	0	0	0	0	2	2	1	1	0	1	1
HJ	0	0	0	0	0	0	0	0	0	0	0	1	0	0
HK	0	0	0	0	0	0	0	0	0	0	0	0	1	0
JK	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Edge-Betweenness values of the edges except JG:

Vertex	A	B	C	D	E	F	G	H	J	K
A	0	10.5	5.0	4.5						
B	10.5	0				12.5				
C	5.0		0	4.0						
D	4.5		4.0	0	8.0		3.5			
E				8.0	0	**14.5**	5.5			
F		12.5			**14.5**	0		10.5	10.5	
G				3.5	5.5		0			
H						10.5		0	1	4.5
J						10.5		1	0	4.5
K								4.5	4.5	0

According to the result of above table, Edge-Betweenness value of FE is largest, so remove edge FE:



Update all shortest paths between every two nodes again:

	A	B	C	D	E	F	G	H	J	K
A	-	AB	AC	AD	ADE	ABF	ADG	ABFH	ABFJ	ABFHK ABFJK
B	-	-	BAC	BAD	BADE	BF	BADG	BFH	BFJ	BFHK BFJK
C	-	-	-	CD	CDE	CABF	CDG	CABFH	CABFJ	CABFHK CABFJK
D	-	-	-	-	DE	DABF	DG	DABFH	DABFJ	DABFHK DABFJK
E	-	-	-	-	-	EDABF	EG	EDABFH	EDABFJ	EDABFHK EDABFJK
F	-	-	-	-	-	-	FBADG	FH	FJ	FHK FJK
G	-	-	-	-	-	-	-	GDABFH	GDABFJ	GDABFHK GDABFJK
H	-	-	-	-	-	-	-	-	HJ	HK
J	-	-	-	-	-	-	-	-	-	JK

Update the total number of shortest paths between every node again:

σAB=1	σAC=1	σAD=1	σAE=1	σAF=1	σAG=1	σAH=1	σAJ=1	σAK=2
σBC=1	σBD=1	σBE=1	σBF=1	σBG=1	σBH=1	σBJ=1	σBK=2	
σCD=1	σCE=1	σCF=1	σCG=1	σCH=1	σCJ=1	σCK=2		
σDE=1	σDF=1	σDG=1	σDH=1	σDJ=1	σDK=2			
σEF=1	σEG=1	σEH=1	σEJ=1	σEK=2				
σFG=1	σFH=1	σFJ=1	σFK=2					
σGH=1	σGJ=1	σGK=2						
σHJ=1	σHK=1							

σJK=1								

Update the number of shortest paths of above table that pass along with every pair of neighbor node again:

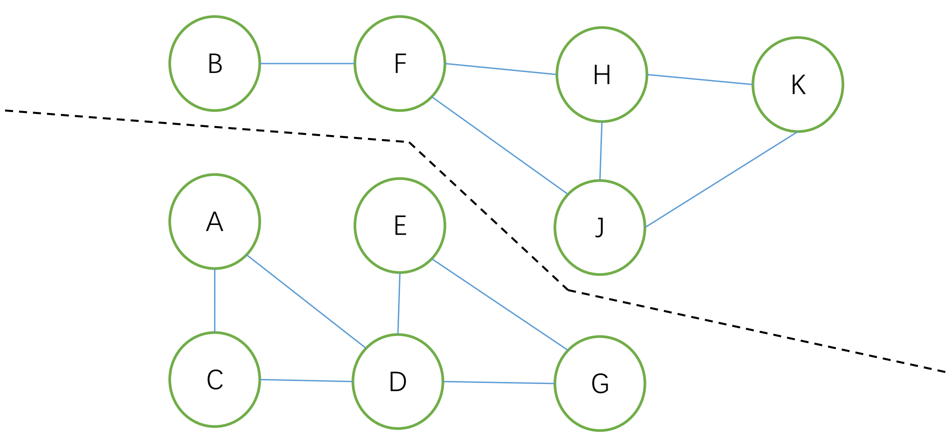
	AB	AC	AD	BF	CD	DE	DG	EG	FH	FJ	HJ	HK	JK
AB	1	0	0	0	0	0	0	0	0	0	0	0	0
AC	0	1	0	0	0	0	0	0	0	0	0	0	0
AD	0	0	1	0	0	0	0	0	0	0	0	0	0
AE	0	0	1	0	0	1	0	0	0	0	0	0	0
AF	1	0	0	1	0	0	0	0	0	0	0	0	0
AG	0	0	1	0	0	0	1	0	0	0	0	0	0
AH	1	0	0	1	0	0	0	0	1	0	0	0	0
AJ	1	0	0	1	0	0	0	0	0	1	0	0	0
AK	2	0	0	2	0	0	0	0	1	1	0	1	1
BC	1	1	0	0	0	0	0	0	0	0	0	0	0
BD	1	0	1	0	0	0	0	0	0	0	0	0	0
BE	1	0	1	0	0	1	0	0	0	0	0	0	0
BF	0	0	0	1	0	0	0	0	0	0	0	0	0
BG	1	0	1	0	0	0	1	0	0	0	0	0	0
BH	0	0	0	1	0	0	0	0	1	0	0	0	0
BJ	0	0	0	1	0	0	0	0	0	1	0	0	0
BK	0	0	0	2	0	0	0	0	1	1	0	1	1
CD	0	0	0	0	1	0	0	0	0	0	0	0	0
CE	0	0	0	0	1	1	0	0	0	0	0	0	0
CF	1	1	0	1	0	0	0	0	0	0	0	0	0
CG	0	0	0	0	1	0	1	0	0	0	0	0	0
CH	1	1	0	1	0	0	0	0	1	0	0	0	0
CJ	1	1	0	1	0	0	0	0	0	1	0	0	0
CK	2	2	0	2	0	0	0	0	1	1	0	1	1
DE	0	0	0	0	0	1	0	0	0	0	0	0	0
DF	1	0	1	1	0	0	0	0	0	0	0	0	0
DG	0	0	0	0	0	0	1	0	0	0	0	0	0
DH	1	0	1	1	0	0	0	0	1	0	0	0	0
DJ	1	0	1	1	0	0	0	0	0	1	0	0	0
DK	2	0	2	2	0	0	0	0	1	1	0	1	1
EF	1	0	1	1	0	1	0	0	0	0	0	0	0
EG	0	0	0	0	0	0	0	1	0	0	0	0	0
EH	1	0	1	1	0	1	0	0	1	0	0	0	0
EJ	1	0	1	1	0	1	0	0	0	1	0	0	0
EK	2	0	2	2	0	2	0	0	1	1	0	1	1

FG	1	0	1	1	0	0	1	1	0	0	0	0	0
FH	0	0	0	0	0	0	0	0	1	0	0	0	0
FJ	0	0	0	0	0	0	0	0	0	1	0	0	0
FK	0	0	0	0	0	0	0	0	1	1	0	1	1
GH	1	0	1	1	0	0	1	1	1	0	0	0	0
GJ	1	0	1	1	0	0	1	1	0	1	0	0	0
GK	2	0	2	2	0	0	2	2	1	1	0	1	1
HJ	0	0	0	0	0	0	0	0	0	0	1	0	0
HK	0	0	0	0	0	0	0	0	0	0	0	1	0
JK	0	0	0	0	0	0	0	0	0	0	0	0	1

Edge-Betweenness values of the edges except JG and FE:

Vertex	A	B	C	D	E	F	G	H	J	K
A	0	**25**	6	18						
B	**25**	0				24				
C	6		0	3						
D	18		3	0	8		8			
E				8	0		5			
F		24				0		10.5	10.5	
G				8	5		0			
H						10.5		0	1	4.5
J						10.5		1	0	4.5
K								4.5	4.5	0

According to the result of above table, Edge-Betweenness values of AB is largest, so remove edge AB:



Finally, we discover two communities in the graph, one community includes A, C, D, E and G, another community includes B, F, H, K and J.

ref1:

```python
# -*- encoding:utf-8 -*-

# Name: SUN RUI    ID:18083229g

import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D

def load_data(file_path):
    df_data = pd.read_csv(file_path)
    df_data.columns = ["Age", "Sex", "MonthlyIncome", "MaritalStatus",
"ServicePlan", "ExtraUsage"]
    scaled_data = preprocessing.scale(df_data)
    return scaled_data

def do_k_means(data):
    centers = np.vstack((data[0], data[7], data[14]))
    k_means = KMeans(n_clusters=3, init=centers, n_init=1,
max_iter=1000)
    return k_means.fit_predict(data), k_means.fit(data).cluster_centers_

def plot_distribution_2D(cluster_index, data, cluster_center2):
    cluster0 = []
    cluster1 = []
    cluster2 = []
    for i in range(len(cluster_index)):
        if cluster_index[i] == 0:
            cluster0.append(data[i])
        if cluster_index[i] == 1:
            cluster1.append(data[i])
        if cluster_index[i] == 2:
            cluster2.append(data[i])
    color_ls = ["red", "green", "blue"]
    all_clusters = [cluster0, cluster1, cluster2]
    for each_cluster, color in zip(all_clusters, color_ls):
        for each_item in each_cluster:
            X = each_item[0]
            Y = each_item[1]
            plt.scatter(X, Y, s=200, c=color, alpha=.5)
    for each_center,color in zip(cluster_center2, color_ls):
        plt.scatter(each_center[0], each_center[1], s=200, c=color,
alpha=.5, marker="*")
    plt.xlim(-3, 3)
    plt.ylim(-3, 3)
    plt.show()
```

ref1:

```
def plot_distribution_3D(cluster_index, data, cluster_center3):
    cluster0 = []
    cluster1 = []
    cluster2 = []
    for i in range(len(cluster_index)):
        if cluster_index[i] == 0:
            cluster0.append(data[i])
        if cluster_index[i] == 1:
            cluster1.append(data[i])
        if cluster_index[i] == 2:
            cluster2.append(data[i])
    color_ls = ["red", "green", "blue"]
    all_clusters = [cluster0, cluster1, cluster2]
    fig = plt.figure()
    ax = Axes3D(fig)
    for each_cluster, color in zip(all_clusters, color_ls):
        for each_item in each_cluster:
            X = each_item[0]
            Y = each_item[1]
            Z = each_item[2]
            ax.scatter(X, Y, Z, s=200, c=color, alpha=.5)
    for each_center,color in zip(cluster_center3, color_ls):
        plt.scatter(each_center[0], each_center[1], s=200, c=color,
alpha=.5, marker="*")
    ax.set_zlabel('Z')
    ax.set_ylabel('Y')
    ax.set_xlabel('X')
    plt.show()

def print_clusters(cluster_index):
    cluster_dict = {0: [], 1: [], 2: []}
    for i in range(len(cluster_index)):
        cluster_dict[cluster_index[i]].append(i+1)
    print("cluster result: {}".format(cluster_dict))


if __name__ == '__main__':
    data = load_data("data.csv")

    pca2 = PCA(2)
    reduce_data2 = pca2.fit_transform(data)
    cluster_index2, cluster_center2 = do_k_means(reduce_data2)
    print_clusters(cluster_index2)
    plot_distribution_2D(cluster_index2, reduce_data2, cluster_center2)

    pca3 = PCA(3)
    reduce_data3 = pca3.fit_transform(data)
    cluster_index3, cluster_center3 = do_k_means(reduce_data3)
    print_clusters(cluster_index3)
    plot_distribution_3D(cluster_index3, reduce_data3, cluster_center3)
```

ref2:

```
# -*- encoding:utf-8 -*-

# Name: SUN RUI    ID:18083229g

import pandas as pd
import numpy as np
from kmodes.kmodes import KModes
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D

def load_data(file_path):
    df_data = pd.read_csv(file_path)
    df_data.columns = ["Age", "Sex", "MonthlyIncome", "MaritalStatus",
"ServicePlan", "ExtraUsage"]
    one_hot_data = np.array(pd.get_dummies(df_data))
    return one_hot_data

def do_k_modes(data):
    centroids = np.vstack((data[0], data[7], data[14]))
    k_modes = KModes(n_clusters=3, init=centroids, n_init=1,
max_iter=10000)
    return k_modes.fit_predict(data), k_modes.cluster_centroids_

def plot_distribution_2D(cluster_index, data, cluster_center2):
    cluster0 = []
    cluster1 = []
    cluster2 = []
    for i in range(len(cluster_index)):
        if cluster_index[i] == 0:
            cluster0.append(data[i])
        if cluster_index[i] == 1:
            cluster1.append(data[i])
        if cluster_index[i] == 2:
            cluster2.append(data[i])
    color_ls = ["red", "green", "blue"]
    all_clusters = [cluster0, cluster1, cluster2]
    for each_cluster, color in zip(all_clusters, color_ls):
        for each_item in each_cluster:
            X = each_item[0]
            Y = each_item[1]
            plt.scatter(X, Y, s=200, c=color, alpha=.5)
    for each_center,color in zip(cluster_center2, color_ls):
        plt.scatter(each_center[0], each_center[1], s=200, c=color,
alpha=.5, marker="*")
    plt.xlim(-3, 3)
    plt.ylim(-3, 3)
    plt.show()
```

ref2:

```python
def plot_distribution_3D(cluster_index, data, cluster_center3):
    cluster0 = []
    cluster1 = []
    cluster2 = []
    for i in range(len(cluster_index)):
        if cluster_index[i] == 0:
            cluster0.append(data[i])
        if cluster_index[i] == 1:
            cluster1.append(data[i])
        if cluster_index[i] == 2:
            cluster2.append(data[i])
    color_ls = ["red", "green", "blue"]
    all_clusters = [cluster0, cluster1, cluster2]
    fig = plt.figure()
    ax = Axes3D(fig)
    for each_cluster, color in zip(all_clusters, color_ls):
        for each_item in each_cluster:
            X = each_item[0]
            Y = each_item[1]
            Z = each_item[2]
            ax.scatter(X, Y, Z, s=200, c=color, alpha=.5)
    for each_center,color in zip(cluster_center3, color_ls):
        plt.scatter(each_center[0], each_center[1], s=200, c=color,
alpha=.5, marker="*")
    ax.set_zlabel('Z')
    ax.set_ylabel('Y')
    ax.set_xlabel('X')
    plt.show()

def print_clusters(cluster_index):
    cluster_dict = {0: [], 1: [], 2: []}
    for i in range(len(cluster_index)):
        cluster_dict[cluster_index[i]].append(i+1)
    print("cluster result: {}".format(cluster_dict))

if __name__ == '__main__':
    data = load_data("data_depth.csv")    # data_width.csv
    cluster_index1, cluster_center1 = do_k_modes(data)
    print_clusters(cluster_index1)
    pca2 = PCA(2)
    reduce_data2 = pca2.fit_transform(data)
    cluster_index2, cluster_center2 = do_k_modes(reduce_data2)
    print_clusters(cluster_index2)
    plot_distribution_2D(cluster_index2, reduce_data2, cluster_center2)
    pca3 = PCA(3)
    reduce_data3 = pca3.fit_transform(data)
    cluster_index3, cluster_center3 = do_k_modes(reduce_data3)
    print_clusters(cluster_index3)
    plot_distribution_3D(cluster_index3, reduce_data3, cluster_center3)
```