

K mediod Clustering when K=4

```
In [1]: 1 import pandas as pd
        2 import numpy as np
        3 import random
        4 import matplotlib.pyplot as plt
        5 import seaborn as sns
        6 from mpl_toolkits.mplot3d import Axes3D
        7 from math import*
        8
        9 from scipy.spatial.distance import pdist,squareform
       10
       11
```

```
In [2]: 1 def manhattan_distance(person1, person2):
        2     distance = 0
        3     distance += abs(person1 - person2)
        4     return distance
```

```
In [3]: 1 Assigned_cluster = pd.DataFrame(np.zeros((196, 1), dtype=int))
        2
```

```
In [4]: 1 dataset = pd.read_csv('Mall_Customers.csv')#-----reading dataset
        2 customer = dataset.to_numpy()
```

calculating Disimilarity MAtrix below

```
In [5]: 1 squareform(pdist(customer[:,2:], metric='euclidean'))#-----calculating dissimilarity matrix
```

```
Out[5]: array([[ 0.          , 42.04759208, 33.03028913, ..., 117.1110584 ,
        124.47489707, 130.15759678],
       [ 42.04759208,  0.          , 75.01333215, ..., 111.7631424 ,
        137.74614332, 122.34786471],
       [ 33.03028913, 75.01333215,  0.          , ..., 129.87686476,
        122.18428704, 143.77065069],
       ...,
       [117.1110584 , 111.7631424 , 129.87686476, ...,  0.          ,
        57.07013229, 14.35270009],
       [124.47489707, 137.74614332, 122.18428704, ..., 57.07013229,
        0.          , 65.03076195],
       [130.15759678, 122.34786471, 143.77065069, ..., 14.35270009,
        65.03076195,  0.          ]])
```

```
In [ ]:
```

```
1
2
```

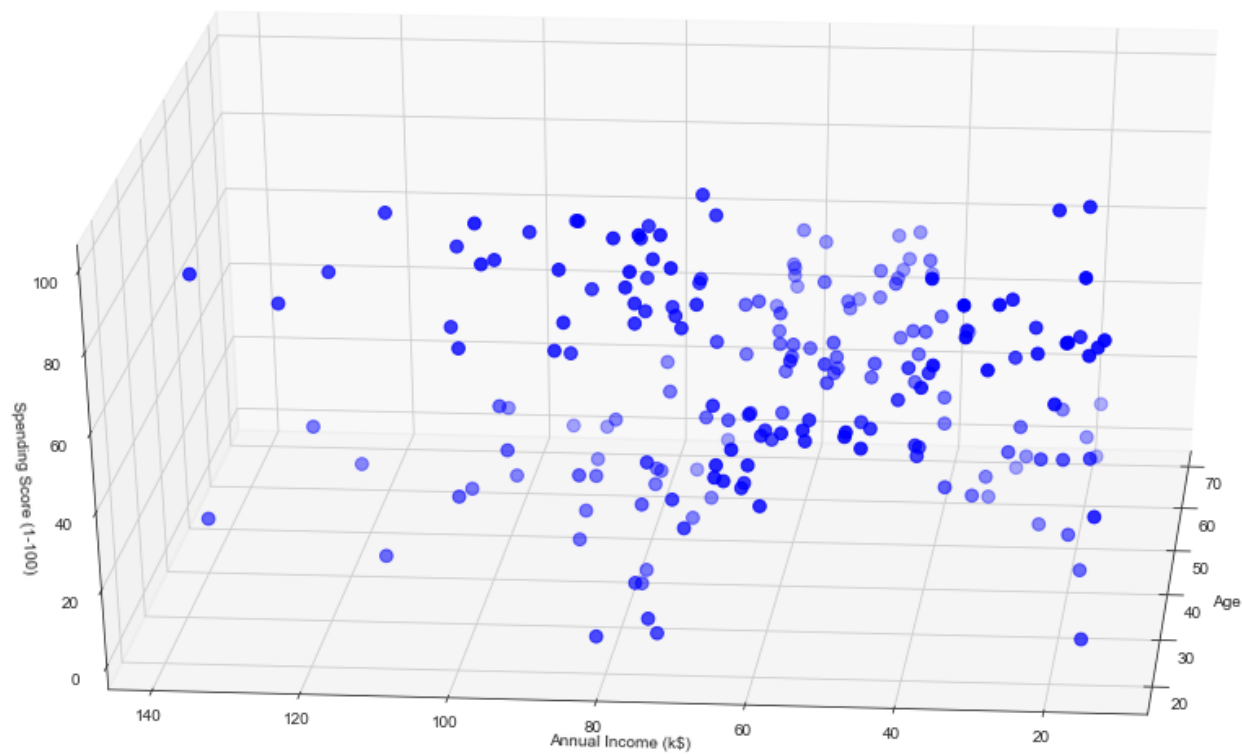
```
In [6]:
```

```
1
2 k = 4 #-----NUMBER OF CLUSTERS
3
```

Plotting graph before clustering

In [7]:

```
1 sns.set_style("white")
2 fig = plt.figure(figsize=(20,10))
3 ax = fig.add_subplot(111, projection='3d')
4 ax.scatter(dataset.Age, dataset["Annual Income (k$)"], dataset["Spending Score (1-100)"], c='blue', s=60)
5 ax.view_init(30, 185)
6 plt.xlabel("Age")
7 plt.ylabel("Annual Income (k$)")
8 ax.set_zlabel('Spending Score (1-100)')
9 plt.show()
```



In []:

1

In [8]:

```
1 mediod = customer[np.random.choice(customer.shape[0], k, replace=False), :]#-----randomly choosing mediods
2
```

In [11]:

```
1 ##### #-----delete the mediods value from dataset
2 customer_new = np.delete(customer, (mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1), axis=0)#---
3 #dataset1 = np.delete(dataset, (mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1), axis=0)#-----
4 dataset_1=dataset.drop(dataset.index[[mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1]])
5
```

```
In [12]: 1 print(mediod[2, 2])
          2 print(customer.shape)
          3 print(customer_new.shape)
          4 distance_age = 0
          5 distance_income = 0
          6 distance_score = 0
          7 total_distance_cost = np.zeros((len(customer_new),k))
          8 print(total_distance_cost.shape)
          9 print(customer_new.shape)
         10 print(dataset_1.shape)
         11 non_mediod = 0
         12
```

```
50
(200, 5)
(196, 5)
(196, 4)
(196, 5)
(196, 5)
```

Calculating distance between mediod and points and assigning clusters to it

In [13]:

```

1  ###*****needs to be modified as the value of k changes
2  #-----comparing when value of k=1
3  old_cost = 0
4  total_cost_cluster = 0
5  difference_cost=0
6  for l in range(len(customer_new)):
7
8      distance_age_k1 = []
9      distance_income_k1 = []
10     distance_score_k1 = []
11     total_distance_k1 = []
12     q=0
13     w=0
14     e=0
15     total=0
16     i=0
17
18     for j in range(len(customer_new)):
19         q = manhattan_distance(mediod[i,2],customer_new[j,2])
20         distance_age_k1.append(q)
21         w = manhattan_distance(mediod[i,3],customer_new[j,3])
22         distance_income_k1.append(w)
23         e = manhattan_distance(mediod[i,4],customer_new[j,4])
24         distance_score_k1.append(e)
25         total = q+w+e
26         total_distance_k1.append(total)
27
28     #-----comparing when value of k=2
29
30     distance_age_k2 = []
31     distance_income_k2 = []
32     distance_score_k2 = []
33     total_distance_k2 = []
34     q=0
35     w=0
36     e=0
37     total=0
38     i=1
39     for j in range(len(customer_new)):
40         q = manhattan_distance(mediod[i,2],customer_new[j,2])
41         distance_age_k2.append(q)

```

```
42     w = manhattan_distance(mediod[i,3],customer_new[j,3])
43     distance_income_k2.append(w)
44     e = manhattan_distance(mediod[i,4],customer_new[j,4])
45     distance_score_k2.append(e)
46     total = q+w+e
47     total_distance_k2.append(total)
48
49     #-----comparing when value of k=3
50     distance_age_k3 = []
51     distance_income_k3 = []
52     distance_score_k3 = []
53     total_distance_k3 = []
54     q=0
55     w=0
56     e=0
57     total=0
58     i=2
59
60
61     for j in range(len(customer_new)):
62         q = manhattan_distance(mediod[i,2],customer_new[j,2])
63         distance_age_k3.append(q)
64         w = manhattan_distance(mediod[i,3],customer_new[j,3])
65         distance_income_k3.append(w)
66         e = manhattan_distance(mediod[i,4],customer_new[j,4])
67         distance_score_k3.append(e)
68         total = q+w+e
69         total_distance_k3.append(total)
70
71
72     #-----comparing when value of k=4
73     distance_age_k4 = []
74     distance_income_k4 = []
75     distance_score_k4 = []
76     total_distance_k4 = []
77     q=0
78     w=0
79     e=0
80     total=0
81     i=3
82     for j in range(len(customer_new)):
83         q = manhattan_distance(mediod[i,2],customer_new[j,2])
```

```

84     distance_age_k4.append(q)
85     w = manhattan_distance(mediod[i,3],customer_new[j,3])
86     distance_income_k4.append(w)
87     e = manhattan_distance(mediod[i,4],customer_new[j,4])
88     distance_score_k4.append(e)
89     total = q+w+e
90     total_distance_k4.append(total)
91
92
93
94
95
96     cost_1 = pd.DataFrame({'Cost_1':total_distance_k1})
97     cost_2 = pd.DataFrame({'Cost_2':total_distance_k2})
98     cost_3 = pd.DataFrame({'Cost_3':total_distance_k3})
99     cost_4 = pd.DataFrame({'Cost_4':total_distance_k4})
100
101
102     dataset_1 = dataset_1.assign(cost_1=cost_1.values,cost_2=cost_2.values,cost_3=cost_3.values,cost_4=cost_4.values)
103
104
105     customer_new = dataset_1.to_numpy()
106     #print(customer_new[0:10, :])#-----combined dataset which displays cost of all k, the last four colour
107
108
109
110
111     #####
112     #----now we will create 4 clusters as the value of k =4 on the basis of total distance or cost
113
114
115     cluster_1 = []
116     cluster_2 = []
117     cluster_3 = []
118     cluster_4 = []
119     q=0
120     w=0
121     e=0
122     r=0
123     i=0
124
125     for i in range(len(customer_new)):

```



```

126
127     if(customer_new[i][5] <= customer_new[i][6] and customer_new[i][5] <= customer_new[i][7] and customer_new[i]
128         #print("five")
129         q = customer_new[i]
130         cluster_1.append(q)
131         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_1"
132
133
134     if(customer_new[i][6] <= customer_new[i][5] and customer_new[i][6] <= customer_new[i][7] and customer_new[i]
135         #print("six")
136         w = customer_new[i]
137         cluster_2.append(w)
138         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_2"
139
140     if(customer_new[i][7] <= customer_new[i][6] and customer_new[i][7] <= customer_new[i][5] and customer_new[i]
141         # print("seven")
142         e = customer_new[i]
143         cluster_3.append(e)
144         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_3"
145
146     if(customer_new[i][8] <= customer_new[i][6] and customer_new[i][8] <= customer_new[i][7] and customer_new[i]
147         # print("eight")
148         r = customer_new[i]
149         cluster_4.append(r)
150         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_4"
151
152     print("lenth is", + len(cluster_4))
153
154     if(len(cluster_4) == 0):
155         cluster_4 = cluster_4_old
156     if(len(cluster_3) == 0):
157         cluster_3 = cluster_3_old
158     if(len(cluster_2) == 0):
159         cluster_2 = cluster_2_old
160     if(len(cluster_1) == 0):
161         cluster_1 = cluster_2_old
162     total_cost_cluster = 0
163     cluster_1 = np.asarray(cluster_1)
164     cluster_2 = np.asarray(cluster_2)
165     cluster_3 = np.asarray(cluster_3)
166     cluster_4 = np.asarray(cluster_4)
167     sum_cluster_1 = cluster_1[:,5].sum(axis=0)#-----adding all the particalura coloumn which is cost of partic

```

```

168 sum_cluster_2 = cluster_2[:,6].sum(axis=0)
169 sum_cluster_3 = cluster_3[:,7].sum(axis=0)#-----adding all the minimum cost of particualr clusters
170 sum_cluster_4 = cluster_4[:,8].sum(axis=0)
171 total_cost_cluster = sum_cluster_1 + sum_cluster_2 + sum_cluster_3 +sum_cluster_4
172 print(total_cost_cluster, l+1)
173 #print(medioid)
174
175 difference_cost= total_cost_cluster - old_cost
176 if (difference_cost<=0):#-----bad cost
177     non_medioid = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
178     non_medioid = np.delete(non_medioid, np.s_[4:9], axis=1)
179     medioid[3]= non_medioid
180     old_cost = total_cost_cluster
181     #print("difference is", + difference_cost)
182 cluster_4_old = cluster_4
183 old_cost = total_cost_cluster
184
185 difference_cost= total_cost_cluster - old_cost
186 if (difference_cost<=0):#-----bad cost
187     non_medioid = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
188     non_medioid = np.delete(non_medioid, np.s_[4:9], axis=1)
189     medioid[2]= non_medioid
190     old_cost = total_cost_cluster
191     #print("difference is", + difference_cost)
192 cluster_3_old = cluster_3
193 old_cost = total_cost_cluster
194
195
196 if (difference_cost<=0):#-----bad cost
197     non_medioid = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
198     non_medioid = np.delete(non_medioid, np.s_[4:9], axis=1)
199     medioid[1]= non_medioid
200     old_cost = total_cost_cluster
201     #print("difference is", + difference_cost)
202 cluster_2_old = cluster_2
203 old_cost = total_cost_cluster
204
205 if (difference_cost<=0):#-----bad cost
206     non_medioid = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
207     non_medioid = np.delete(non_medioid, np.s_[4:9], axis=1)
208     medioid[0]= non_medioid
209     old_cost = total_cost_cluster

```

```
210         #print("difference is", + difference_cost)
211     cluster_1_old = cluster_1
212     old_cost = total_cost_cluster
213
```

```
lenth is 41
7988 1
lenth is 79
13166 2
lenth is 53
13366 3
lenth is 83
13374 4
lenth is 60
12880 5
lenth is 66
14507 6
lenth is 50
14139 7
lenth is 49
14261 8
lenth is 30
15029 9
lenth is 21
14570 10
```

New coloumn added below for assigned cluster

In [14]: 1 dataset_1

Out[14]:

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	cost_1	cost_2	cost_3	cost_4	Assigned_cluster
0	1	Male	19	15	39	132	70	97	110	Cluster_2
1	2	Male	21	15	81	172	114	137	150	Cluster_2
2	3	Female	20	16	6	97	37	62	75	Cluster_2
3	4	Female	23	16	77	165	111	130	143	Cluster_2
4	5	Female	31	17	40	123	81	84	97	Cluster_2
...
195	196	Female	35	120	79	107	169	142	129	Cluster_1
196	197	Female	45	126	28	72	134	107	94	Cluster_1
197	198	Male	32	126	74	105	167	140	127	Cluster_1
198	199	Male	32	137	18	60	122	95	82	Cluster_1
199	200	Male	30	137	83	123	185	162	149	Cluster_1

196 rows × 10 columns

Calculating silhoutte width for each clusters

In [26]:

```

1  distance_age_k2 = []
2  distance_income_k2 = []
3  distance_score_k2 = []
4  total_distance_k2 = []
5  distance_sil = []
6  average_sil = []
7  total=0
8  q=0
9  w=0
10 e=0
11 a=0
12
13
14 total=0
15
16 for i in range(len(cluster_1)):
17     for j in range(len(cluster_1)):#-----calculating distance
18         q = manhattan_distance(cluster_1[i,2],cluster_1[j,2])
19         #distance_age_k2.append(q)
20         w = manhattan_distance(cluster_1[i,3],cluster_1[j,3])
21         #distance_income_k2.append(w)
22         e = manhattan_distance(cluster_1[i,4],cluster_1[j,4])
23         #distance_score.append(e)
24         total = q+w+e
25         distance_sil.append(total/3)
26     a_sil = sum(distance_sil)/len(cluster_1)
27     #print(a_sil)
28     distance_sil.clear()
29
30
31     cluster_min=0
32
33     distance_age_k2 = []
34     distance_income_k2 = []
35     distance_score_k2 = []
36     total_distance_k2 = []
37     distance_sil = []
38     q=0
39     w=0
40     e=0
41     a=0

```

```

42
43 total=0
44 n=0#-----for cluster 1
45 for j in range(len(mediod)):#-----calculating nearest cluster
46     q = manhattan_distance(mediod[n,2],mediod[j,2])
47     #distance_age_k2.append(q)
48     w = manhattan_distance(mediod[n,3],mediod[j,3])
49     #distance_income_k2.append(w)
50     e = manhattan_distance(mediod[n,4],mediod[j,4])
51     #distance_score.append(e)
52     total = q+w+e
53     distance_sil.append(total)
54 #print(distance_sil)
55 index = [i for i in range(0,4)]
56 #print(index)
57 index=np.asarray(index)
58 distance_sil= np.asarray(distance_sil)
59 distance_sil.transpose
60 #print(distance_sil)
61 distance_sil = np.column_stack((index,distance_sil))
62 #print(distance_sil[:,1])
63 minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
64 #print(minval)
65 for b in range(len(distance_sil)):
66     if (minval == distance_sil[b,1]):
67         cluster_min = b
68         #print(cluster_min)
69
70
71
72
73 #-----now find the value of b
74 if (cluster_min == 0):
75     selected_cluster= cluster_1
76 if (cluster_min == 1):
77     selected_cluster= cluster_2
78 if (cluster_min == 2):
79     selected_cluster= cluster_3
80 if (cluster_min == 3):
81     selected_cluster= cluster_4
82 if (cluster_min == 4):
83     selected_cluster= cluster_5

```

```

84
85     distance_age_k2 = []
86     distance_b= []
87     distance_income_k2 = []
88     distance_score_k2 = []
89     total_distance_k2 = []
90     distance_sil = []
91     q=0
92     w=0
93     e=0
94     a=0
95     s=0
96
97     total=0
98
99     for j in range(len(selected_cluster)):#-----calculating distance
100
101
102         q = manhattan_distance(cluster_1[i,2],selected_cluster[j,2])
103         #distance_age_k2.append(q)
104         w = manhattan_distance(cluster_1[i,3],selected_cluster[j,3])
105         #distance_income_k2.append(w)
106         e = manhattan_distance(cluster_1[i,4],selected_cluster[j,4])
107         #distance_score.append(e)
108         total = q+w+e
109         distance_b.append(total)
110     # print(min(distance_b))
111     b = min(distance_b)
112     maxi = max(b, a_sil)
113     #print("a is", + maxi)
114     d = b - a_sil
115     sill = ((b - a_sil) / maxi)#-----silhoutte widht formulae
116     print("Silhoutte width for each point in cluster 1", + sill)
117
118     average_sill.append(sill)
119     s_c1 = sum(average_sill)/len(average_sill)
120     print("Average silhoutee width for cluster 1", + s_c1)
121
122

```

```
Silhoutte width for each point in cluster 1 0.5499999999999999
Silhoutte width for each point in cluster 1 0.48893105629348516
Silhoutte width for each point in cluster 1 0.5996339510409516
Silhoutte width for each point in cluster 1 0.6308243727598568
Silhoutte width for each point in cluster 1 0.6693975081071856
Silhoutte width for each point in cluster 1 0.6170250896057348
Silhoutte width for each point in cluster 1 0.6731182795698925
Silhoutte width for each point in cluster 1 0.5685483870967742
Silhoutte width for each point in cluster 1 0.5504851822711776
Silhoutte width for each point in cluster 1 0.6062048298959987
Silhoutte width for each point in cluster 1 0.7070707070707071
Silhoutte width for each point in cluster 1 0.6251221896383187
Silhoutte width for each point in cluster 1 0.7289359653346174
Silhoutte width for each point in cluster 1 0.6990876507005539
Silhoutte width for each point in cluster 1 0.7337216248506571
Silhoutte width for each point in cluster 1 0.5916601101494886
Silhoutte width for each point in cluster 1 0.7374770521898768
Silhoutte width for each point in cluster 1 0.6575268817204302
Silhoutte width for each point in cluster 1 0.6851851851851852
Silhoutte width for each point in cluster 1 0.720269619643717
Silhoutte width for each point in cluster 1 0.7502986857825568
Silhoutte width for each point in cluster 1 0.7214706902532085
Silhoutte width for each point in cluster 1 0.7089093701996928
Silhoutte width for each point in cluster 1 0.747800586510264
Silhoutte width for each point in cluster 1 0.7643966547192352
Silhoutte width for each point in cluster 1 0.7310577644411103
Silhoutte width for each point in cluster 1 0.7545239968528717
Silhoutte width for each point in cluster 1 0.7237083661159193
Silhoutte width for each point in cluster 1 0.7475678443420378
Silhoutte width for each point in cluster 1 0.7443841158309379
Silhoutte width for each point in cluster 1 0.7557313856766078
Average silhoutee width for cluster 1 0.6770991968983566
```


In [27]:

```

1 distance_age_k2 = []
2 distance_income_k2 = []
3 distance_score_k2 = []
4 total_distance_k2 = []
5 distance_sil = []
6 average_sil = []
7 total=0
8 q=0
9 w=0
10 e=0
11 a=0
12
13 total=0
14
15 for i in range(len(cluster_2)):
16     for j in range(len(cluster_2)):#-----calculating distance
17         q = manhattan_distance(cluster_2[i,2],cluster_2[j,2])
18         #distance_age_k2.append(q)
19         w = manhattan_distance(cluster_2[i,3],cluster_2[j,3])
20         #distance_income_k2.append(w)
21         e = manhattan_distance(cluster_2[i,4],cluster_2[j,4])
22         #distance_score.append(e)
23         total = q+w+e
24         distance_sil.append(total/3)
25     a_sil = sum(distance_sil)/len(cluster_2)
26     #print(a_sil)
27     distance_sil.clear()
28
29
30
31     cluster_min=0
32
33     distance_age_k2 = []
34     distance_income_k2 = []
35     distance_score_k2 = []
36     total_distance_k2 = []
37     distance_sil = []
38     q=0
39     w=0
40     e=0
41     a=0

```

```

42
43 total=0
44 n=1#-----for cluster 2
45 for j in range(len(mediod)):#-----calculating nearest cluster
46     q = manhattan_distance(mediod[n,2],mediod[j,2])
47     #distance_age_k2.append(q)
48     w = manhattan_distance(mediod[n,3],mediod[j,3])
49     #distance_income_k2.append(w)
50     e = manhattan_distance(mediod[n,4],mediod[j,4])
51     #distance_score.append(e)
52     total = q+w+e
53     distance_sil.append(total)
54 #print(distance_sil)
55 index = [i for i in range(0,4)]
56 #print(index)
57 index=np.asarray(index)
58 distance_sil= np.asarray(distance_sil)
59 distance_sil.transpose
60 #print(distance_sil)
61 distance_sil = np.column_stack((index,distance_sil))
62 #print(distance_sil[:,1])
63 minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
64 #print(minval)
65 for b in range(len(distance_sil)):
66     if (minval == distance_sil[b,1]):
67         cluster_min = b
68         #print(cluster_min)
69
70
71
72
73 #-----now find the value of b
74 if (cluster_min == 0):
75     selected_cluster= cluster_1
76 if (cluster_min == 1):
77     selected_cluster= cluster_2
78 if (cluster_min == 2):
79     selected_cluster= cluster_3
80 if (cluster_min == 3):
81     selected_cluster= cluster_4
82 if (cluster_min == 4):
83     selected_cluster= cluster_5

```

```

84
85     distance_age_k2 = []
86     distance_b= []
87     distance_income_k2 = []
88     distance_score_k2 = []
89     total_distance_k2 = []
90     distance_sil = []
91     q=0
92     w=0
93     e=0
94     a=0
95     s=0
96
97     total=0
98
99     for j in range(len(selected_cluster)):#-----calculating distance
100
101
102         q = manhattan_distance(cluster_2[i,2],selected_cluster[j,2])
103         #distance_age_k2.append(q)
104         w = manhattan_distance(cluster_2[i,3],selected_cluster[j,3])
105         #distance_income_k2.append(w)
106         e = manhattan_distance(cluster_2[i,4],selected_cluster[j,4])
107         #distance_score.append(e)
108         total = q+w+e
109         distance_b.append(total)
110     # print(min(distance_b))
111     b = min(distance_b)
112     maxi = max(b, a_sil)
113     #print("a is", + maxi)
114     d = b - a_sil
115     sill = ((b - a_sil) / maxi)#-----silhoutte widht formulae
116     print("Silhoutte width for each point in cluster 2", + sill)
117
118     average_sill.append(sill)
119     s_c2 = sum(average_sill)/len(average_sill)
120     print("Average silhoutee width for cluster 1", + s_c2)
121

```

Silhoutte width for each point in cluster 2 0.7154667837284167

```
Silhoutte width for each point in cluster 2 0.6806546975268585
Silhoutte width for each point in cluster 2 0.5570423743352204
Silhoutte width for each point in cluster 2 0.6713208152784463
Silhoutte width for each point in cluster 2 0.7021908003840447
Silhoutte width for each point in cluster 2 0.6816710100292189
Silhoutte width for each point in cluster 2 0.6340604668962877
Silhoutte width for each point in cluster 2 0.6354793198188163
Silhoutte width for each point in cluster 2 0.5421354764638348
Silhoutte width for each point in cluster 2 0.6746268656716418
Silhoutte width for each point in cluster 2 0.5625049356392638
Silhoutte width for each point in cluster 2 0.5895114590979531
Silhoutte width for each point in cluster 2 0.5617448471926084
Silhoutte width for each point in cluster 2 0.6631753513136075
Silhoutte width for each point in cluster 2 0.6682740145426712
Silhoutte width for each point in cluster 2 0.6722127069570182
Silhoutte width for each point in cluster 2 0.6903445734291507
Silhoutte width for each point in cluster 2 0.714680565426834
Silhoutte width for each point in cluster 2 0.667762880512691
Silhoutte width for each point in cluster 2 0.5710732054015636
Silhoutte width for each point in cluster 2 0.6808116281338408
Silhoutte width for each point in cluster 2 0.6846126510305615
Silhoutte width for each point in cluster 2 0.5294117647058824
Silhoutte width for each point in cluster 2 0.651842826682912
Silhoutte width for each point in cluster 2 0.5539947322212468
Silhoutte width for each point in cluster 2 0.6140724946695094
Silhoutte width for each point in cluster 2 0.6381633499170812
Silhoutte width for each point in cluster 2 0.6905269570514774
Silhoutte width for each point in cluster 2 0.6543811554472535
Silhoutte width for each point in cluster 2 0.5909667349526875
Silhoutte width for each point in cluster 2 0.44928419128845554
Silhoutte width for each point in cluster 2 0.669983416252073
Silhoutte width for each point in cluster 2 0.4650153487879751
Silhoutte width for each point in cluster 2 0.5744007236544549
Silhoutte width for each point in cluster 2 0.5212824765063571
Silhoutte width for each point in cluster 2 0.6169154228855723
Silhoutte width for each point in cluster 2 0.5558872305140964
Silhoutte width for each point in cluster 2 0.5924038344861061
Silhoutte width for each point in cluster 2 0.6170235777633571
Silhoutte width for each point in cluster 2 0.5963659961064246
Silhoutte width for each point in cluster 2 0.5685572139303484
Silhoutte width for each point in cluster 2 0.49366802351876976
Silhoutte width for each point in cluster 2 0.5442524221000262
```

```
Silhoutte width for each point in cluster 2 0.5865268253327955
Silhoutte width for each point in cluster 2 0.5814262023217248
Silhoutte width for each point in cluster 2 0.590049751243781
Silhoutte width for each point in cluster 2 0.5804726368159203
Silhoutte width for each point in cluster 2 0.5743504698728579
Silhoutte width for each point in cluster 2 0.5937418172296413
Silhoutte width for each point in cluster 2 0.574212271973466
Silhoutte width for each point in cluster 2 0.531370923161968
Silhoutte width for each point in cluster 2 0.5526652452025587
Silhoutte width for each point in cluster 2 0.4835820895522388
Silhoutte width for each point in cluster 2 0.5793373557743201
Silhoutte width for each point in cluster 2 0.3547079417726185
Silhoutte width for each point in cluster 2 0.42603648424543944
Silhoutte width for each point in cluster 2 0.4104477611940298
Silhoutte width for each point in cluster 2 0.48428312980551824
Silhoutte width for each point in cluster 2 0.3670110983543819
Silhoutte width for each point in cluster 2 0.2799464217374663
Silhoutte width for each point in cluster 2 0.5571200600769736
Silhoutte width for each point in cluster 2 0.40398009950248764
Silhoutte width for each point in cluster 2 0.38773872572620754
Silhoutte width for each point in cluster 2 0.37744610281923735
Silhoutte width for each point in cluster 2 0.08054963278843849
Silhoutte width for each point in cluster 2 0.007462686567164134
Silhoutte width for each point in cluster 2 0.1630735212824765
Average silhoutee width for cluster 1 0.551302933979199
```

In [28]:

```
1 distance_age_k2 = []
2 distance_income_k2 = []
3 distance_score_k2 = []
4 total_distance_k2 = []
5 distance_sil = []
6 average_sil = []
7 total=0
8 q=0
9 w=0
10 e=0
11 a=0
12
13 total=0
14
15 for i in range(len(cluster_3)):
16     for j in range(len(cluster_3)):#-----calculating distance
17         q = manhattan_distance(cluster_3[i,2],cluster_3[j,2])
18         #distance_age_k2.append(q)
19         w = manhattan_distance(cluster_3[i,3],cluster_3[j,3])
20         #distance_income_k2.append(w)
21         e = manhattan_distance(cluster_3[i,4],cluster_3[j,4])
22         #distance_score.append(e)
23         total = q+w+e
24         distance_sil.append(total/3)
25     a_sil = sum(distance_sil)/len(cluster_3)
26     #print(a_sil)
27     distance_sil.clear()
28
29
30     cluster_min=0
31
32     distance_age_k2 = []
33     distance_income_k2 = []
34     distance_score_k2 = []
35     total_distance_k2 = []
36     distance_sil = []
37     q=0
38     w=0
39     e=0
40     a=0
41
```

```

42 total=0
43 n=2#-----for cluster 2
44 for j in range(len(mediod)):#-----calculating nearest cluster
45     q = manhattan_distance(mediod[n,2],mediod[j,2])
46     #distance_age_k2.append(q)
47     w = manhattan_distance(mediod[n,3],mediod[j,3])
48     #distance_income_k2.append(w)
49     e = manhattan_distance(mediod[n,4],mediod[j,4])
50     #distance_score.append(e)
51     total = q+w+e
52     distance_sil.append(total)
53 #print(distance_sil)
54 index = [i for i in range(0,4)]
55 #print(index)
56 index=np.asarray(index)
57 distance_sil= np.asarray(distance_sil)
58 distance_sil.transpose
59 #print(distance_sil)
60 distance_sil = np.column_stack((index,distance_sil))
61 #print(distance_sil[:,1])
62 minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
63 #print(minval)
64 for b in range(len(distance_sil)):
65     if (minval == distance_sil[b,1]):
66         cluster_min = b
67         #print(cluster_min)
68
69
70
71
72 #-----now find the value of b
73 if (cluster_min == 0):
74     selected_cluster= cluster_1
75 if (cluster_min == 1):
76     selected_cluster= cluster_2
77 if (cluster_min == 2):
78     selected_cluster= cluster_3
79 if (cluster_min == 3):
80     selected_cluster= cluster_4
81 if (cluster_min == 4):
82     selected_cluster= cluster_5
83

```

```

84     distance_age_k2 = []
85     distance_b= []
86     distance_income_k2 = []
87     distance_score_k2 = []
88     total_distance_k2 = []
89     distance_sil = []
90     q=0
91     w=0
92     e=0
93     a=0
94     s=0
95
96     total=0
97
98     for j in range(len(selected_cluster)):#-----calculating distance
99
100
101         q = manhattan_distance(cluster_3[i,2],selected_cluster[j,2])
102         #distance_age_k2.append(q)
103         w = manhattan_distance(cluster_3[i,3],selected_cluster[j,3])
104         #distance_income_k2.append(w)
105         e = manhattan_distance(cluster_3[i,4],selected_cluster[j,4])
106         #distance_score.append(e)
107         total = q+w+e
108         distance_b.append(total)
109     # print(min(distance_b))
110     b = min(distance_b)
111     maxi = max(b, a_sil)
112     #print("a is", + maxi)
113     d = b - a_sil
114     sill = ((b - a_sil) / maxi)#-----silhoutte widht formulae
115     print("Silhoutte width for each point in cluster 3", + sill)
116
117     average_sill.append(sill)
118     s_c3 = sum(average_sill)/len(average_sill)
119     print("Average silhouttee width for cluster 1", + s_c3)
120
121
122

```



```
Silhoutte width for each point in cluster 3 0.7216296296296297
Silhoutte width for each point in cluster 3 0.6790990990990993
Silhoutte width for each point in cluster 3 0.7112820512820514
Silhoutte width for each point in cluster 3 0.7006249999999998
Silhoutte width for each point in cluster 3 0.7
Silhoutte width for each point in cluster 3 0.6585416666666666
Silhoutte width for each point in cluster 3 0.726938775510204
Silhoutte width for each point in cluster 3 0.5637681159420288
Silhoutte width for each point in cluster 3 0.6827083333333334
Silhoutte width for each point in cluster 3 0.6306666666666667
Silhoutte width for each point in cluster 3 0.3079166666666667
Silhoutte width for each point in cluster 3 0.6621333333333334
Silhoutte width for each point in cluster 3 0.6315942028985507
Silhoutte width for each point in cluster 3 0.4395833333333333
Silhoutte width for each point in cluster 3 0.6683950617283949
Silhoutte width for each point in cluster 3 0.5309999999999998
Silhoutte width for each point in cluster 3 0.6152688172043013
Silhoutte width for each point in cluster 3 0.5477192982456139
Silhoutte width for each point in cluster 3 0.5687719298245614
Silhoutte width for each point in cluster 3 0.6358024691358026
Silhoutte width for each point in cluster 3 0.2554166666666664
Silhoutte width for each point in cluster 3 0.42070175438596497
Silhoutte width for each point in cluster 3 0.38410256410256405
Silhoutte width for each point in cluster 3 0.6935135135135135
Silhoutte width for each point in cluster 3 0.5220833333333333
Silhoutte width for each point in cluster 3 -0.07216494845360835
Silhoutte width for each point in cluster 3 0.3248888888888889
Silhoutte width for each point in cluster 3 0.22428571428571434
Silhoutte width for each point in cluster 3 0.48311111111111116
Silhoutte width for each point in cluster 3 0.22615384615384612
Silhoutte width for each point in cluster 3 0.13266666666666668
Silhoutte width for each point in cluster 3 0.41499999999999987
Silhoutte width for each point in cluster 3 0.35333333333333333
Silhoutte width for each point in cluster 3 0.34611111111111111
Silhoutte width for each point in cluster 3 0.6493333333333333
Silhoutte width for each point in cluster 3 -0.3686109440769693
Silhoutte width for each point in cluster 3 0.5114814814814815
Silhoutte width for each point in cluster 3 0.43242424242424227
Silhoutte width for each point in cluster 3 0.5976923076923076
Silhoutte width for each point in cluster 3 0.13266666666666663
Silhoutte width for each point in cluster 3 0.4428571428571429
Silhoutte width for each point in cluster 3 0.5647222222222222
```

```
Silhoutte width for each point in cluster 3 0.6135802469135802
Silhoutte width for each point in cluster 3 0.08809523809523816
Silhoutte width for each point in cluster 3 -0.4942683749157114
Silhoutte width for each point in cluster 3 0.3946031746031749
Silhoutte width for each point in cluster 3 0.3098245614035088
Silhoutte width for each point in cluster 3 0.19958333333333331
Silhoutte width for each point in cluster 3 0.20714285714285716
Silhoutte width for each point in cluster 3 0.26333333333333353
Average silhoutee width for cluster 1 0.4327421765621817
```

In [29]:

```

1 distance_age_k2 = []
2 distance_income_k2 = []
3 distance_score_k2 = []
4 total_distance_k2 = []
5 distance_sil = []
6 average_sil = []
7 total=0
8 q=0
9 w=0
10 e=0
11 a=0
12
13 total=0
14
15 for i in range(len(cluster_4)):
16     for j in range(len(cluster_4)):#-----calculating distance
17         q = manhattan_distance(cluster_4[i,2],cluster_4[j,2])
18         #distance_age_k2.append(q)
19         w = manhattan_distance(cluster_4[i,3],cluster_4[j,3])
20         #distance_income_k2.append(w)
21         e = manhattan_distance(cluster_4[i,4],cluster_4[j,4])
22         #distance_score.append(e)
23         total = q+w+e
24         distance_sil.append(total/3)
25     a_sil = sum(distance_sil)/len(cluster_4)
26     #print(a_sil)
27     distance_sil.clear()
28
29
30
31
32
33     cluster_min=0
34
35     distance_age_k2 = []
36     distance_income_k2 = []
37     distance_score_k2 = []
38     total_distance_k2 = []
39     distance_sil = []
40     q=0
41     w=0

```

```

42     e=0
43     a=0
44
45     total=0
46     n=3#-----for cluster 4
47     for j in range(len(mediod)):#-----calculating nearest cluster
48         q = manhattan_distance(mediod[n,2],mediod[j,2])
49         #distance_age_k2.append(q)
50         w = manhattan_distance(mediod[n,3],mediod[j,3])
51         #distance_income_k2.append(w)
52         e = manhattan_distance(mediod[n,4],mediod[j,4])
53         #distance_score.append(e)
54         total = q+w+e
55         distance_sil.append(total)
56     #print(distance_sil)
57     index = [i for i in range(0,4)]
58     #print(index)
59     index=np.asarray(index)
60     distance_sil= np.asarray(distance_sil)
61     distance_sil.transpose
62     #print(distance_sil)
63     distance_sil = np.column_stack((index,distance_sil))
64     #print(distance_sil[:,1])
65     minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
66     #print(minval)
67     for b in range(len(distance_sil)):
68         if (minval == distance_sil[b,1]):
69             cluster_min = b
70             #print(cluster_min)
71
72
73
74
75     #-----now find the value of b
76     if (cluster_min == 0):
77         selected_cluster= cluster_1
78     if (cluster_min == 1):
79         selected_cluster= cluster_2
80     if (cluster_min == 2):
81         selected_cluster= cluster_3
82     if (cluster_min == 3):
83         selected_cluster= cluster_4

```

```

84     if (cluster_min == 4):
85         selected_cluster= cluster_5
86
87     distance_age_k2 = []
88     distance_b= []
89     distance_income_k2 = []
90     distance_score_k2 = []
91     total_distance_k2 = []
92     distance_sil = []
93     q=0
94     w=0
95     e=0
96     a=0
97     s=0
98
99     total=0
100
101     for j in range(len(selected_cluster)):#-----calculating distance
102
103
104         q = manhattan_distance(cluster_4[i,2],selected_cluster[j,2])
105         #distance_age_k2.append(q)
106         w = manhattan_distance(cluster_4[i,3],selected_cluster[j,3])
107         #distance_income_k2.append(w)
108         e = manhattan_distance(cluster_4[i,4],selected_cluster[j,4])
109         #distance_score.append(e)
110         total = q+w+e
111         distance_b.append(total)
112     # print(min(distance_b))
113     b = min(distance_b)
114     maxi = max(b, a_sil)
115     #print("a is", + maxi)
116     d = b - a_sil
117     sill = ((b - a_sil) / maxi)#-----silhoutte widht formulae
118     print("Silhoutte width for each point in cluster 4", + sill)
119
120     average_sill.append(sill)
121     s_c4 = sum(average_sill)/len(average_sill)
122     print("Average silhouttee width for cluster 1", + s_c4)
123

```

```
Silhoutte width for each point in cluster 4 -0.4558772235786535
Silhoutte width for each point in cluster 4 -0.5785411038209186
Silhoutte width for each point in cluster 4 -0.6812423375561913
Silhoutte width for each point in cluster 4 -0.08657465495608561
Silhoutte width for each point in cluster 4 0.5876532887402452
Silhoutte width for each point in cluster 4 0.12019230769230785
Silhoutte width for each point in cluster 4 0.5002003205128204
Silhoutte width for each point in cluster 4 0.18184885290148434
Silhoutte width for each point in cluster 4 0.6134992458521871
Silhoutte width for each point in cluster 4 0.44673382173382176
Silhoutte width for each point in cluster 4 0.5214342948717948
Silhoutte width for each point in cluster 4 0.5409382284382283
Silhoutte width for each point in cluster 4 0.5182291666666667
Silhoutte width for each point in cluster 4 0.14878542510121456
Silhoutte width for each point in cluster 4 0.4649725274725275
Silhoutte width for each point in cluster 4 0.5467032967032968
Silhoutte width for each point in cluster 4 0.6204309874522641
Silhoutte width for each point in cluster 4 0.6057692307692306
Silhoutte width for each point in cluster 4 0.5057692307692307
Silhoutte width for each point in cluster 4 0.5405518394648829
Silhoutte width for each point in cluster 4 0.4984526967285587
Silhoutte width for each point in cluster 4 0.5620845204178536
Silhoutte width for each point in cluster 4 0.645586785009862
Silhoutte width for each point in cluster 4 0.25993589743589746
Silhoutte width for each point in cluster 4 0.6392525913802508
Silhoutte width for each point in cluster 4 0.5904403567447046
Silhoutte width for each point in cluster 4 0.6556931768796175
Silhoutte width for each point in cluster 4 0.26660839160839184
Silhoutte width for each point in cluster 4 0.5796703296703296
Silhoutte width for each point in cluster 4 0.5902496626180836
Silhoutte width for each point in cluster 4 0.6493589743589743
Silhoutte width for each point in cluster 4 0.6088286713286714
Silhoutte width for each point in cluster 4 0.6677761341222881
Silhoutte width for each point in cluster 4 0.5972222222222221
Silhoutte width for each point in cluster 4 0.6254807692307692
Silhoutte width for each point in cluster 4 0.5843531468531469
Silhoutte width for each point in cluster 4 0.6377060439560438
Silhoutte width for each point in cluster 4 0.6283577533577533
Silhoutte width for each point in cluster 4 0.6567796610169492
Silhoutte width for each point in cluster 4 0.5958073458073458
Silhoutte width for each point in cluster 4 0.3552166224580018
```

```
Silhoutte width for each point in cluster 4 0.6483707264957265
Silhoutte width for each point in cluster 4 0.6044429708222812
Silhoutte width for each point in cluster 4 0.6756844850065189
Silhoutte width for each point in cluster 4 0.5447191697191698
Silhoutte width for each point in cluster 4 0.6309731934731935
Silhoutte width for each point in cluster 4 0.6187232905982907
Silhoutte width for each point in cluster 4 0.6763822115384615
Silhoutte width for each point in cluster 4 0.5754807692307693
Silhoutte width for each point in cluster 4 0.5209276018099546
Silhoutte width for each point in cluster 4 0.6532838506522719
Silhoutte width for each point in cluster 4 0.6814204314204315
Average silhoutte width for cluster 1 0.46897590767698333
```

Average silhoutte width of dataset

```
In [33]: 1 Average_dataset = (s_c1+s_c2+s_c3+s_c4)/4
          2 print("Average silhoutte width of dataset", + Average_dataset)
```

Average silhoutte width of dataset 0.5325300537791802

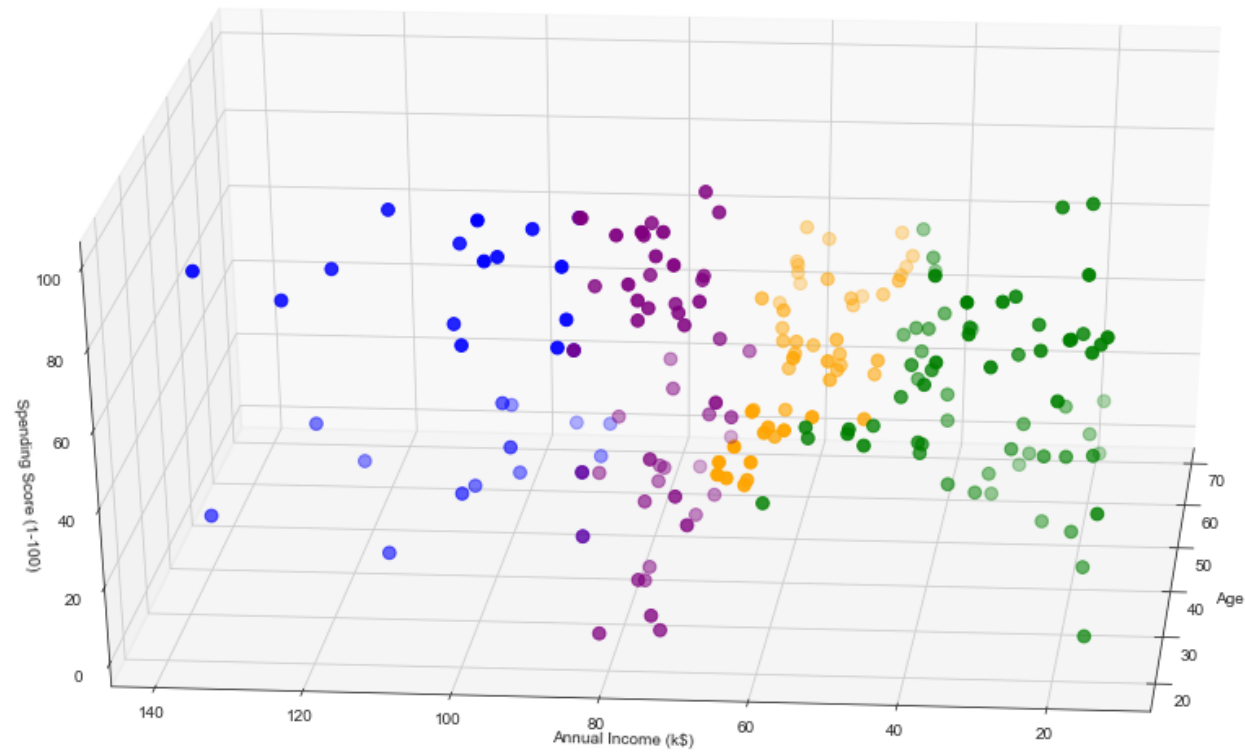
In []:

1

Plotting assigned clusters below

In [31]:

```
1 cluster_1 = np.asarray(cluster_1)
2 cluster_2 = np.asarray(cluster_2)
3 cluster_3 = np.asarray(cluster_3)
4 cluster_4 = np.asarray(cluster_4)
5 sns.set_style("white")
6 fig = plt.figure(figsize=(20,10))
7 ax = fig.add_subplot(111, projection='3d')
8 clusterX1 = cluster_1[:,2].tolist()
9 clusterY1 = cluster_1[:,3].tolist()
10 clusterZ1 = cluster_1[:,4].tolist()
11 clusterX2 = cluster_2[:,2].tolist()
12 clusterY2 = cluster_2[:,3].tolist()
13 clusterZ2 = cluster_2[:,4].tolist()
14 clusterX3 = cluster_3[:,2].tolist()
15 clusterY3 = cluster_3[:,3].tolist()
16 clusterZ3 = cluster_3[:,4].tolist()
17 clusterX4 = cluster_4[:,2].tolist()
18 clusterY4 = cluster_4[:,3].tolist()
19 clusterZ4 = cluster_4[:,4].tolist()
20 ax.scatter(clusterX1, clusterY1, clusterZ1, c='blue', s=60)
21 ax.scatter(clusterX2, clusterY2, clusterZ2, c='green', s=60)
22 ax.scatter(clusterX3, clusterY3, clusterZ3, c='orange', s=60)
23 ax.scatter(clusterX4, clusterY4, clusterZ4, c='purple', s=60)
24 ax.view_init(30, 185)
25 plt.xlabel("Age")
26 plt.ylabel("Annual Income (k$)")
27 ax.set_zlabel('Spending Score (1-100)')
28 plt.show()
29
```

Saving to csv file

```
In [35]: Lakehead Study material\Big data\Assignment1 2\K mediod\Work directory\k_is_4\clusters_k.csv', index = None, header=True)
```

```
In [ ]: 1
```

```
In [ ]: 1
```

