

## K mediod clustering when K = 5

```
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 [27]: 1
2 Assigned_cluster = pd.DataFrame(np.zeros((195, 1), dtype=int))
3
```

```
In [4]: 1 def silhoutte(a,b):
2     average_a = sum(a)/len(a)
3
4
```

```
In [5]: 1 dataset = pd.read_csv('Mall_Customers.csv')
2 customer = dataset.to_numpy()
```

## Calcluating dissimilarity matrix

```
In [6]: 1 squareform(pdist(customer[:,2:], metric='euclidean'))#-----calculating dissimilarity matrix
```

```
Out[6]: 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
3
```

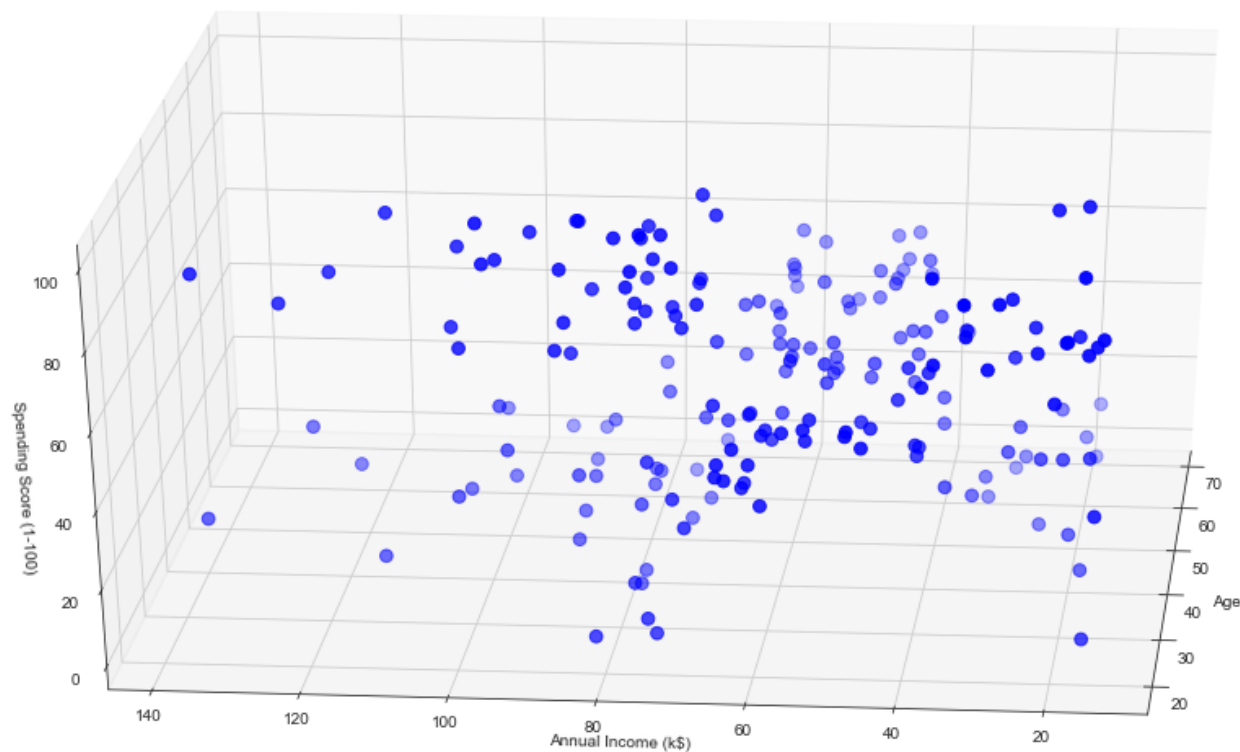
```
In [8]: 1 print(customer[1][:])
       2 k = 5 #-----NUMBER OF CLUSTERS
       3 print(dataset[1:10])
```

```
[2 'Male' 21 15 81]
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
1           2    Male   21                15                81
2           3  Female   20                16                 6
3           4  Female   23                16                77
4           5  Female   31                17                40
5           6  Female   22                17                76
6           7  Female   35                18                 6
7           8  Female   23                18                94
8           9    Male   64                19                 3
9          10  Female   30                19                72
```

## Plotting graph before clustering

In [9]:

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



```
In [10]: 1 mediod = customer[np.random.choice(customer.shape[0], k, replace=False), :]#-----randomly choosing mediods
2
```

```
In [11]: 1 print(mediod[:])
2 #print(customer[128])
3
4 #print(dataset)
5 print(dataset.iloc[49][2])
```

```
[[108 'Male' 54 63 46]
 [147 'Male' 48 77 36]
 [113 'Female' 38 64 42]
 [30 'Female' 23 29 87]
 [120 'Female' 50 67 57]]
31
```

```
In [12]: 1 mediod[0,0] - 1
```

Out[12]: 107

```
In [13]: 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, mediod[4,0] -
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, mediod[4,0]
5
```

```
In [14]: 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
```

```
38
(200, 5)
(195, 5)
(195, 5)
(195, 5)
(195, 5)
```

## Calculating distance between mediod and points and assigning clusters to it

In [15]:

```

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(100):
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)):#-----calculating distance
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)):#-----calculating distance
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)):#-----calculating distance
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     #-----comparing when value of k=5
97     distance_age_k5 = []
98     distance_income_k5 = []
99     distance_score_k5 = []
100    total_distance_k5 = []
101    q=0
102    w=0
103    e=0
104    total=0
105    i=4
106    for j in range(len(customer_new)):
107        q = manhattan_distance(mediod[i,2],customer_new[j,2])
108        distance_age_k5.append(q)
109        w = manhattan_distance(mediod[i,3],customer_new[j,3])
110        distance_income_k5.append(w)
111        e = manhattan_distance(mediod[i,4],customer_new[j,4])
112        distance_score_k5.append(e)
113        total = q+w+e
114        total_distance_k5.append(total)
115
116
117
118    #print(len(total_distance_k1))
119    #print(len(total_distance_k2))
120
121    #print(len(total_distance_k3))
122
123    #print(len(total_distance_k4))
124
125    cost_1 = pd.DataFrame({'Cost_1':total_distance_k1})
```



```

126 cost_2 = pd.DataFrame({'Cost_2':total_distance_k2})
127 cost_3 = pd.DataFrame({'Cost_3':total_distance_k3})
128 cost_4 = pd.DataFrame({'Cost_4':total_distance_k4})
129 cost_5 = pd.DataFrame({'Cost_5':total_distance_k5})
130
131 #dfn = pd.concat([dataset_1,cost_1,cost_2,cost_3,cost_4], axis=1)
132 #dfn= pd.merge(dataset_1, cost_1,cost_2,cost_3,cost_4,)
133 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)
134 #print(df_col)
135
136 customer_new = dataset_1.to_numpy()
137 #print(customer_new[0:10, :])#-----combined dataset which displays cost of all k, the last four columns
138
139 #print(dataset_1.shape)
140 #print("hello")
141 #print(cost_1.shape)
142 #print(cost_2.shape)
143 #print(cost_3.shape)
144 #print(cost_4.shape)
145 #print(customer_new.shape)
146 #print(dataset_1.shape)
147
148
149
150 #*****
151 #----now we will create 4 clusters as the value of k =4 on the basis of total distance or cost
152
153 cluster_1 = []
154 cluster_2 = []
155 cluster_3 = []
156 cluster_4 = []
157 cluster_5 = []
158
159 q=0
160 w=0
161 e=0
162 r=0
163 t=0
164 i=0
165
166 for i in range(len(customer_new)):#-----select the points with minimum distance
167

```

```
168     if(customer_new[i][5] <= customer_new[i][6] and customer_new[i][5] <= customer_new[i][7] and customer_new[i]
169         #print("five")
170         q = customer_new[i]
171         cluster_1.append(q)
172         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_1"
173
174
175
176     if(customer_new[i][6] <= customer_new[i][5] and customer_new[i][6] <= customer_new[i][7] and customer_new[i]
177         #print("six")
178         w = customer_new[i]
179         cluster_2.append(w)
180         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_2"
181
182
183     if(customer_new[i][7] <= customer_new[i][6] and customer_new[i][7] <= customer_new[i][5] and customer_new[i]
184         # print("seven")
185         e = customer_new[i]
186         cluster_3.append(e)
187         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_3"
188
189
190     if(customer_new[i][8] <= customer_new[i][6] and customer_new[i][8] <= customer_new[i][7] and customer_new[i]
191         # print("eight")
192         r = customer_new[i]
193         cluster_4.append(r)
194         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_4"
195
196
197     if(customer_new[i][9] <= customer_new[i][6] and customer_new[i][9] <= customer_new[i][7] and customer_new[i]
198         # print("eight")
199         t = customer_new[i]
200         cluster_5.append(t)
201         dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_5"
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

```

210     cluster_1 = cluster_2_old
211     total_cost_cluster = 0
212     difference_cost=0
213     cluster_1 = np.asarray(cluster_1)
214     cluster_2 = np.asarray(cluster_2)
215     cluster_3 = np.asarray(cluster_3)
216     cluster_4 = np.asarray(cluster_4)
217     cluster_5 = np.asarray(cluster_5)
218     sum_cluster_1 = cluster_1[:,5].sum(axis=0)#-----adding all the particalura coloumn which is cost of partic
219     sum_cluster_2 = cluster_2[:,6].sum(axis=0)
220     sum_cluster_3 = cluster_3[:,7].sum(axis=0)#-----adding all the minimum cost of particualr clusters
221     sum_cluster_4 = cluster_4[:,8].sum(axis=0)
222     sum_cluster_5 = cluster_5[:,9].sum(axis=0)
223     total_cost_cluster = sum_cluster_1 + sum_cluster_2 + sum_cluster_3 +sum_cluster_4+sum_cluster_5
224     print(total_cost_cluster, l+1)
225     #print(mediod)
226
227     difference_cost = total_cost_cluster - old_cost
228     if (difference_cost<=0):#-----bad cost
229         non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
230         non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
231         mediod[4]= non_mediod
232         old_cost = total_cost_cluster
233         print("difference is", + difference_cost)
234     cluster_5_old = cluster_5
235     old_cost = total_cost_cluster
236
237     difference_cost= total_cost_cluster - old_cost
238     if (difference_cost<=0):#-----bad cost
239         non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
240         non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
241         mediod[3]= non_mediod
242         old_cost = total_cost_cluster
243         #print("difference is", + difference_cost)
244     cluster_4_old = cluster_4
245     old_cost = total_cost_cluster
246
247     difference_cost= total_cost_cluster - old_cost
248     if (difference_cost<=0):#-----bad cost
249         non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
250         non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
251         mediod[2]= non_mediod

```

```

252     old_cost = total_cost_cluster
253     #print("difference is", + difference_cost)
254     cluster_3_old = cluster_3
255     old_cost = total_cost_cluster
256
257
258     if (difference_cost<=0):#-----bad cost
259         non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
260         non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
261         mediod[1]= non_mediod
262         old_cost = total_cost_cluster
263         #print("difference is", + difference_cost)
264     cluster_2_old = cluster_2
265     old_cost = total_cost_cluster
266
267     if (difference_cost<=0):#-----bad cost
268         non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), : ]#-----for cluster
269         non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
270         mediod[0]= non_mediod
271         old_cost = total_cost_cluster
272         print("difference is", + difference_cost)
273     cluster_1_old = cluster_1
274     old_cost = total_cost_cluster

```

```

lenth is 46
8041 1
difference is 0
lenth is 132
9722 2
difference is 0
lenth is 138
9732 3
difference is 0
lenth is 141
9376 4
difference is -356
difference is 0
lenth is 73
14350 5
difference is 0

```

```
length is 32
14015 6
difference is -335
```

## New coloumn added below for assigned cluster

In [16]: 1 dataset\_1

Out[16]:

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	cost_1	cost_2	cost_3	cost_4	cost_5	Assigned_cluster
0	1	Male	19	15	39	92	121	145	132	61	Cluster_5
1	2	Male	21	15	81	132	161	185	172	101	Cluster_5
2	3	Female	20	16	6	57	86	110	97	26	Cluster_5
3	4	Female	23	16	77	125	154	178	165	94	Cluster_5
4	5	Female	31	17	40	79	108	132	123	48	Cluster_5
...	...	...	...	...	...	...	...	...	...	...	...
195	196	Female	35	120	79	173	188	128	107	178	Cluster_4
196	197	Female	45	126	28	118	133	73	72	143	Cluster_4
197	198	Male	32	126	74	177	192	132	105	182	Cluster_4
198	199	Male	32	137	18	132	147	87	60	137	Cluster_4
199	200	Male	30	137	83	199	214	154	123	204	Cluster_4

195 rows × 11 columns

In [ ]: 1

## Calculating silhoutte width of each cluster

In [18]:

```

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_1)):
16     for j in range(len(cluster_1)):#-----calculating distance
17         q = manhattan_distance(cluster_1[i,2],cluster_1[j,2])
18         #distance_age_k2.append(q)
19         w = manhattan_distance(cluster_1[i,3],cluster_1[j,3])
20         #distance_income_k2.append(w)
21         e = manhattan_distance(cluster_1[i,4],cluster_1[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_1)
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=0#-----for cluster 1
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(0,5)]
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_1[i,2],selected_cluster[j,2])
102         #distance_age_k2.append(q)
103         w = manhattan_distance(cluster_1[i,3],selected_cluster[j,3])
104         #distance_income_k2.append(w)
105         e = manhattan_distance(cluster_1[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 1", + sill)
116
117     average_sill.append(sill)
118     s_c1 = sum(average_sill)/len(average_sill)
119     print("Average silhouttee width for cluster 4", + s_c1)

```

Silhoutte width for each point in cluster 1 -0.020990764063811503  
 Silhoutte width for each point in cluster 1 0.5105774728416237  
 Silhoutte width for each point in cluster 1 0.0411051212938001



```
Silhoutte width for each point in cluster 1 0.37454485269778215
Silhoutte width for each point in cluster 1 0.38384433962264153
Silhoutte width for each point in cluster 1 0.574374725756911
Silhoutte width for each point in cluster 1 0.6068376068376068
Silhoutte width for each point in cluster 1 0.6663522012578618
Silhoutte width for each point in cluster 1 0.6634346610761706
Silhoutte width for each point in cluster 1 0.3485600794438926
Silhoutte width for each point in cluster 1 0.6051076805793787
Silhoutte width for each point in cluster 1 0.3991090146750522
Silhoutte width for each point in cluster 1 0.6482275586049171
Silhoutte width for each point in cluster 1 0.3325934147243804
Silhoutte width for each point in cluster 1 0.7407407407407406
Silhoutte width for each point in cluster 1 0.7386443046820405
Silhoutte width for each point in cluster 1 0.7338013748720199
Silhoutte width for each point in cluster 1 0.44389275074478635
Silhoutte width for each point in cluster 1 0.550896808758444
Silhoutte width for each point in cluster 1 0.6749907510173881
Silhoutte width for each point in cluster 1 0.4935609463911351
Silhoutte width for each point in cluster 1 0.6176999101527404
Silhoutte width for each point in cluster 1 0.32861635220125784
Silhoutte width for each point in cluster 1 0.702912942734194
Silhoutte width for each point in cluster 1 0.20215633423180598
Silhoutte width for each point in cluster 1 0.7045392398140554
Silhoutte width for each point in cluster 1 -0.21326076199901048
Silhoutte width for each point in cluster 1 0.687140372005888
Silhoutte width for each point in cluster 1 0.5050314465408805
Silhoutte width for each point in cluster 1 0.6870385561936013
Silhoutte width for each point in cluster 1 0.6745283018867924
Silhoutte width for each point in cluster 1 0.5031446540880504
Silhoutte width for each point in cluster 1 0.6424343322234555
Silhoutte width for each point in cluster 1 0.24752920035938925
Silhoutte width for each point in cluster 1 0.4852201257861637
Silhoutte width for each point in cluster 1 0.6398921832884097
Silhoutte width for each point in cluster 1 0.26708595387840656
Silhoutte width for each point in cluster 1 0.5263877495214658
Silhoutte width for each point in cluster 1 0.3423742138364779
Silhoutte width for each point in cluster 1 0.6710092842168314
Silhoutte width for each point in cluster 1 0.12735849056603796
Silhoutte width for each point in cluster 1 -0.12757201646090546
Silhoutte width for each point in cluster 1 0.6731057202755315
Silhoutte width for each point in cluster 1 0.6399371069182389
Silhoutte width for each point in cluster 1 0.17385444743935327
```

```
Silhoutte width for each point in cluster 1 0.7037977745524915
Silhoutte width for each point in cluster 1 0.31335553089160195
Silhoutte width for each point in cluster 1 0.6013390139987829
Silhoutte width for each point in cluster 1 0.6625871154173042
Silhoutte width for each point in cluster 1 0.35674353598881897
Silhoutte width for each point in cluster 1 0.38050314465408813
Silhoutte width for each point in cluster 1 0.3116701607267647
Silhoutte width for each point in cluster 1 0.37705899970050905
Average silhoutee width for cluster 4 0.47029103883366474
```

In [19]:

```

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(0,5)]
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 2", + s_c2)

```

Silhoutte width for each point in cluster 2 0.7674603174603175  
 Silhoutte width for each point in cluster 2 0.8441358024691358

```
Silhoutte width for each point in cluster 2 0.8287385129490393
Silhoutte width for each point in cluster 2 0.8812083973374295
Silhoutte width for each point in cluster 2 0.8748373666406454
Silhoutte width for each point in cluster 2 0.8877344877344877
Silhoutte width for each point in cluster 2 0.8733398121153223
Silhoutte width for each point in cluster 2 0.8883116883116883
Silhoutte width for each point in cluster 2 0.8766360345307713
Silhoutte width for each point in cluster 2 0.84992784992785
Silhoutte width for each point in cluster 2 0.8443093549476528
Silhoutte width for each point in cluster 2 0.8277197057684863
Silhoutte width for each point in cluster 2 0.7970827970827971
Silhoutte width for each point in cluster 2 0.8617216117216118
Silhoutte width for each point in cluster 2 0.8423280423280424
Silhoutte width for each point in cluster 2 0.8134038800705469
Silhoutte width for each point in cluster 2 0.8329554043839759
Silhoutte width for each point in cluster 2 0.8330026455026455
Silhoutte width for each point in cluster 2 0.839105339105339
Silhoutte width for each point in cluster 2 0.8335179032853451
Silhoutte width for each point in cluster 2 0.8059964726631392
Average silhoutee width for cluster 2 0.8430225441112508
```

In [20]:

```

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(0,5)]
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 3", + s_c3)
120
121

```

Silhoutte width for each point in cluster 3 0.6430341147322279

```
Silhoutte width for each point in cluster 3 0.7204245848313645
Silhoutte width for each point in cluster 3 0.6773861059575347
Silhoutte width for each point in cluster 3 0.6472892187177901
Silhoutte width for each point in cluster 3 0.5723905723905724
Silhoutte width for each point in cluster 3 0.7157287157287158
Silhoutte width for each point in cluster 3 0.553030303030303
Silhoutte width for each point in cluster 3 0.6917388167388168
Silhoutte width for each point in cluster 3 0.574468085106383
Silhoutte width for each point in cluster 3 0.7007328183798773
Silhoutte width for each point in cluster 3 0.5869107744107743
Silhoutte width for each point in cluster 3 0.7035742035742036
Silhoutte width for each point in cluster 3 0.7064083457526081
Silhoutte width for each point in cluster 3 0.8022650749923477
Silhoutte width for each point in cluster 3 0.6594735231098867
Silhoutte width for each point in cluster 3 0.7982954545454546
Silhoutte width for each point in cluster 3 0.6433425160697888
Silhoutte width for each point in cluster 3 0.7953379953379953
Silhoutte width for each point in cluster 3 0.7349250076522804
Silhoutte width for each point in cluster 3 0.7695133149678604
Silhoutte width for each point in cluster 3 0.791798582843359
Silhoutte width for each point in cluster 3 0.655196028077384
Silhoutte width for each point in cluster 3 0.8069264069264069
Silhoutte width for each point in cluster 3 0.7799204162840526
Silhoutte width for each point in cluster 3 0.7942279942279943
Silhoutte width for each point in cluster 3 0.77706643903827
Silhoutte width for each point in cluster 3 0.6484563949352682
Silhoutte width for each point in cluster 3 0.7967836257309941
Silhoutte width for each point in cluster 3 0.7568330362448009
Silhoutte width for each point in cluster 3 0.7567676767676768
Silhoutte width for each point in cluster 3 0.7963977120603627
Silhoutte width for each point in cluster 3 0.780273321449792
Silhoutte width for each point in cluster 3 0.7786273954498253
Average silhouttee width for cluster 3 0.715622562910999
```

In [21]:

```

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(0,5)]
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 4", + s_c4)

```

```
Silhoutte width for each point in cluster 4 0.5827900912646675
Silhoutte width for each point in cluster 4 0.5296610169491525
Silhoutte width for each point in cluster 4 0.501755993281417
Silhoutte width for each point in cluster 4 0.5740604274134118
Silhoutte width for each point in cluster 4 0.5655994978028877
Silhoutte width for each point in cluster 4 0.5875706214689266
Silhoutte width for each point in cluster 4 0.5682878899533284
Silhoutte width for each point in cluster 4 0.6042843691148776
Silhoutte width for each point in cluster 4 0.5963983050847459
Silhoutte width for each point in cluster 4 0.6152542372881356
Silhoutte width for each point in cluster 4 0.6174334140435834
Silhoutte width for each point in cluster 4 0.6433308769344143
Silhoutte width for each point in cluster 4 0.6615914966963516
Silhoutte width for each point in cluster 4 0.7413887370147622
Silhoutte width for each point in cluster 4 0.6617493199414102
Silhoutte width for each point in cluster 4 0.737775180206507
Silhoutte width for each point in cluster 4 0.7049450898241605
Silhoutte width for each point in cluster 4 0.7787918296392872
Silhoutte width for each point in cluster 4 0.7465160075329567
Silhoutte width for each point in cluster 4 0.710222047037183
Silhoutte width for each point in cluster 4 0.7259391416394092
Silhoutte width for each point in cluster 4 0.7756648752928207
Silhoutte width for each point in cluster 4 0.6663207655943735
Silhoutte width for each point in cluster 4 0.7817154596815613
Silhoutte width for each point in cluster 4 0.718351119481063
Silhoutte width for each point in cluster 4 0.7883812331122574
Silhoutte width for each point in cluster 4 0.7704476314645807
Silhoutte width for each point in cluster 4 0.6800286969778496
Silhoutte width for each point in cluster 4 0.7781073446327683
Silhoutte width for each point in cluster 4 0.7992379450794901
Silhoutte width for each point in cluster 4 0.7884494664155681
Silhoutte width for each point in cluster 4 0.6901398977670165
Silhoutte width for each point in cluster 4 0.7801669618011636
Silhoutte width for each point in cluster 4 0.797539869841951
Silhoutte width for each point in cluster 4 0.7319843191513893
Silhoutte width for each point in cluster 4 0.794388370469117
Silhoutte width for each point in cluster 4 0.7926433465560764
Silhoutte width for each point in cluster 4 0.7611228813559322
Silhoutte width for each point in cluster 4 0.805225988700565
Silhoutte width for each point in cluster 4 0.8096672944130572
Silhoutte width for each point in cluster 4 0.7963128159381504
Silhoutte width for each point in cluster 4 0.7943262411347517
```

```
Silhoutte width for each point in cluster 4 0.6998280520756571
Silhoutte width for each point in cluster 4 0.8099481623856951
Silhoutte width for each point in cluster 4 0.8160824968460315
Silhoutte width for each point in cluster 4 0.6501883239171374
Silhoutte width for each point in cluster 4 0.8071025020177564
Silhoutte width for each point in cluster 4 0.800961101370219
Silhoutte width for each point in cluster 4 0.6928787281566154
Silhoutte width for each point in cluster 4 0.7932664591260233
Silhoutte width for each point in cluster 4 0.7356971231898176
Silhoutte width for each point in cluster 4 0.7937196163447642
Silhoutte width for each point in cluster 4 0.7410129493525324
Silhoutte width for each point in cluster 4 0.7795601291364004
Silhoutte width for each point in cluster 4 0.7856189008731382
Silhoutte width for each point in cluster 4 0.6664863565332372
Silhoutte width for each point in cluster 4 0.7879869164436514
Silhoutte width for each point in cluster 4 0.738479872881356
Silhoutte width for each point in cluster 4 0.78954802259887
Average silhoutee width for cluster 4 0.7193887089532538
```

In [22]:

```

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_5)):
16     for j in range(len(cluster_5)):#-----calculating distance
17         q = manhattan_distance(cluster_5[i,2],cluster_5[j,2])
18         #distance_age_k2.append(q)
19         w = manhattan_distance(cluster_5[i,3],cluster_5[j,3])
20         #distance_income_k2.append(w)
21         e = manhattan_distance(cluster_5[i,4],cluster_5[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_5)
26     #print(a_sil)
27     distance_sil.clear()
28
29     cluster_min=0
30
31     distance_age_k2 = []
32     distance_income_k2 = []
33     distance_score_k2 = []
34     total_distance_k2 = []
35     distance_sil = []
36     q=0
37     w=0
38     e=0
39     a=0
40
41     total=0

```



```

42 n=4#-----for cluster 5
43 for j in range(len(mediod)):#-----calculating nearest cluster
44     q = manhattan_distance(mediod[n,2],mediod[j,2])
45     #distance_age_k2.append(q)
46     w = manhattan_distance(mediod[n,3],mediod[j,3])
47     #distance_income_k2.append(w)
48     e = manhattan_distance(mediod[n,4],mediod[j,4])
49     #distance_score.append(e)
50     total = q+w+e
51     distance_sil.append(total)
52 #print(distance_sil)
53 index = [i for i in range(0,5)]
54 #print(index)
55 index=np.asarray(index)
56 distance_sil= np.asarray(distance_sil)
57 distance_sil.transpose
58 #print(distance_sil)
59 distance_sil = np.column_stack((index,distance_sil))
60 #print(distance_sil[:,1])
61 minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
62 #print(minval)
63 for b in range(len(distance_sil)):
64     if (minval == distance_sil[b,1]):
65         cluster_min = b
66         #print(cluster_min)
67
68
69
70
71 #-----now find the value of b
72 if (cluster_min == 0):
73     selected_cluster= cluster_1
74 if (cluster_min == 1):
75     selected_cluster= cluster_2
76 if (cluster_min == 2):
77     selected_cluster= cluster_3
78 if (cluster_min == 3):
79     selected_cluster= cluster_4
80 if (cluster_min == 4):
81     selected_cluster= cluster_5
82
83 distance_age_k2 = []

```

```

84     distance_b= []
85     distance_income_k2 = []
86     distance_score_k2 = []
87     total_distance_k2 = []
88     distance_sil = []
89     q=0
90     w=0
91     e=0
92     a=0
93     s=0
94
95     total=0
96
97     for j in range(len(selected_cluster)):#-----calculating distance
98
99
100         q = manhattan_distance(cluster_5[i,2],selected_cluster[j,2])
101         #distance_age_k2.append(q)
102         w = manhattan_distance(cluster_5[i,3],selected_cluster[j,3])
103         #distance_income_k2.append(w)
104         e = manhattan_distance(cluster_5[i,4],selected_cluster[j,4])
105         #distance_score.append(e)
106         total = q+w+e
107         distance_b.append(total)
108     # print(min(distance_b))
109     b = min(distance_b)
110     maxi = max(b, a_sil)
111     #print("a is", + maxi)
112     d = b - a_sil
113     sill = ((b - a_sil) / maxi)#-----silhoutte widht formulae
114     print("Silhoutte width for each point in cluster 5", + sill)
115
116     average_sill.append(sill)
117     s_c5 = sum(average_sill)/len(average_sill)
118     print("Average silhoutee width for cluster 1", + s_c5)

```

```

Silhoutte width for each point in cluster 5 0.7643518518518518
Silhoutte width for each point in cluster 5 0.8021097046413502
Silhoutte width for each point in cluster 5 0.7223577235772358
Silhoutte width for each point in cluster 5 0.8004629629629629

```

```

Silhoutte width for each point in cluster 5 0.7339181286549707
Silhoutte width for each point in cluster 5 0.8007824726134585
Silhoutte width for each point in cluster 5 0.6661538461538461
Silhoutte width for each point in cluster 5 0.7458730158730158
Silhoutte width for each point in cluster 5 0.5355555555555557
Silhoutte width for each point in cluster 5 0.7931216931216931
Silhoutte width for each point in cluster 5 0.6879781420765027
Silhoutte width for each point in cluster 5 0.550595238095238
Silhoutte width for each point in cluster 5 0.7971807628524047
Silhoutte width for each point in cluster 5 0.69008547008547
Silhoutte width for each point in cluster 5 0.8010954616588419
Silhoutte width for each point in cluster 5 0.712551440329218
Silhoutte width for each point in cluster 5 0.81593567251462
Silhoutte width for each point in cluster 5 0.6487758945386064
Silhoutte width for each point in cluster 5 0.6656746031746031
Silhoutte width for each point in cluster 5 0.6921568627450981
Silhoutte width for each point in cluster 5 0.7851254480286738
Silhoutte width for each point in cluster 5 0.5178649237472767
Silhoutte width for each point in cluster 5 0.7585858585858587
Silhoutte width for each point in cluster 5 0.7444444444444444
Silhoutte width for each point in cluster 5 0.7054421768707482
Silhoutte width for each point in cluster 5 0.626984126984127
Silhoutte width for each point in cluster 5 0.7512962962962964
Silhoutte width for each point in cluster 5 0.6503831417624523
Silhoutte width for each point in cluster 5 0.725136612021858
Silhoutte width for each point in cluster 5 0.673049645390071
Average silhoutte width for cluster 1 0.712167639240278

```

## Average Silhoutte width of dataset

```

In [23]: 1 Average_dataset = (s_c1+s_c2+s_c3+s_c4+s_c5)/5
          2 print("Average silhoutte width of dataset", + Average_dataset)

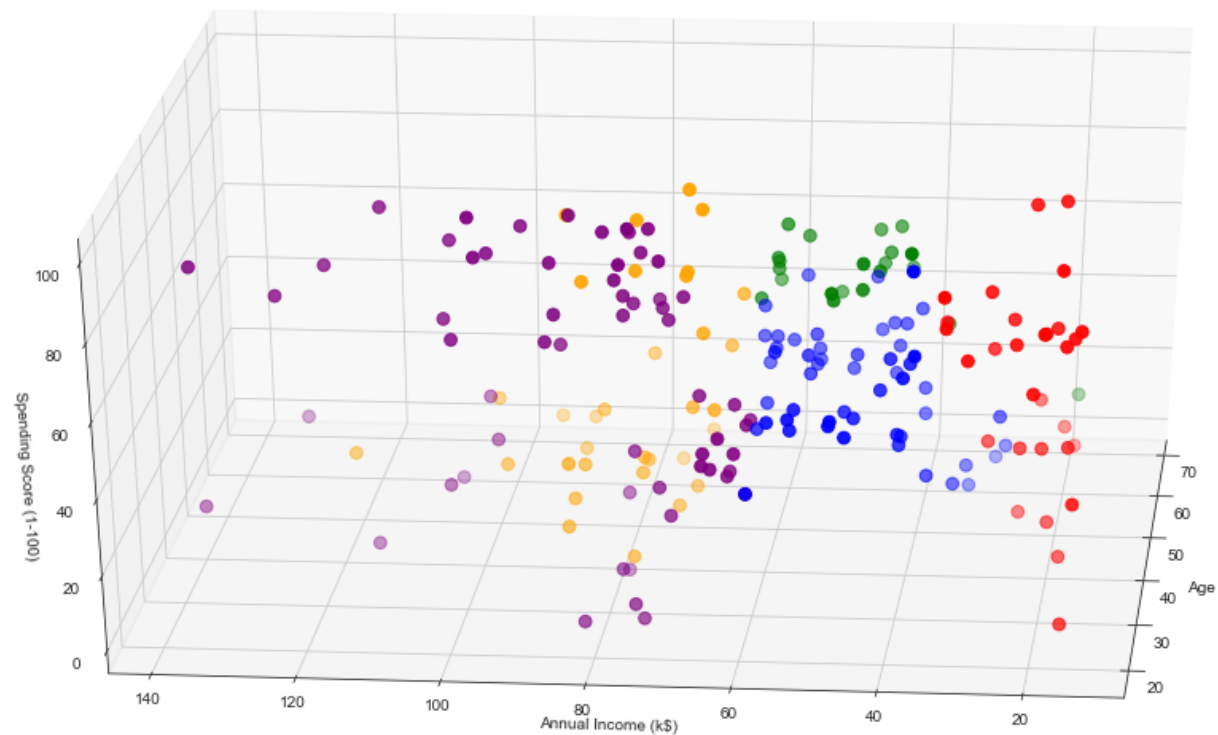
```

Average silhoutte width of dataset 0.6920984988098893

## Plotting assigned clusters below

In [25]:

```
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 cluster_5 = np.asarray(cluster_5)
6 sns.set_style("white")
7 fig = plt.figure(figsize=(20,10))
8 ax = fig.add_subplot(111, projection='3d')
9 clusterX1 = cluster_1[:,2].tolist()
10 clusterY1 = cluster_1[:,3].tolist()
11 clusterZ1 = cluster_1[:,4].tolist()
12 clusterX2 = cluster_2[:,2].tolist()
13 clusterY2 = cluster_2[:,3].tolist()
14 clusterZ2 = cluster_2[:,4].tolist()
15 clusterX3 = cluster_3[:,2].tolist()
16 clusterY3 = cluster_3[:,3].tolist()
17 clusterZ3 = cluster_3[:,4].tolist()
18 clusterX4 = cluster_4[:,2].tolist()
19 clusterY4 = cluster_4[:,3].tolist()
20 clusterZ4 = cluster_4[:,4].tolist()
21 clusterX5 = cluster_5[:,2].tolist()
22 clusterY5 = cluster_5[:,3].tolist()
23 clusterZ5 = cluster_5[:,4].tolist()
24 ax.scatter(clusterX1, clusterY1, clusterZ1, c='blue', s=60)
25 ax.scatter(clusterX2, clusterY2, clusterZ2, c='green', s=60)
26 ax.scatter(clusterX3, clusterY3, clusterZ3, c='orange', s=60)
27 ax.scatter(clusterX4, clusterY4, clusterZ4, c='purple', s=60)
28 ax.scatter(clusterX5, clusterY5, clusterZ5, c='red', s=60)
29 ax.view_init(30, 185)
30 plt.xlabel("Age")
31 plt.ylabel("Annual Income (k$)")
32 ax.set_zlabel('Spending Score (1-100)')
33 plt.show()
```



## Saving to csv file

```
In [26]: Lakehead Study material\Big data\Assignment 2\K mediod\Work directory\k_is_5\clusters_k.csv', index = None, header=True)
```

```
In [ ]: 1
```

```
In [ ]: 1  
        2
```

