K mediod clustering when K = 5

In [1]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from math import*

from scipy.spatial.distance import pdist,squareform


```

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

In [27]:
```

Assigned_cluster = pd.DataFrame(np.zeros((195, 1), dtype=int))

```

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


```

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

Calcluating dissimlarity matrix

In [6]:

```
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 [ ]:

```



```

In [8]:

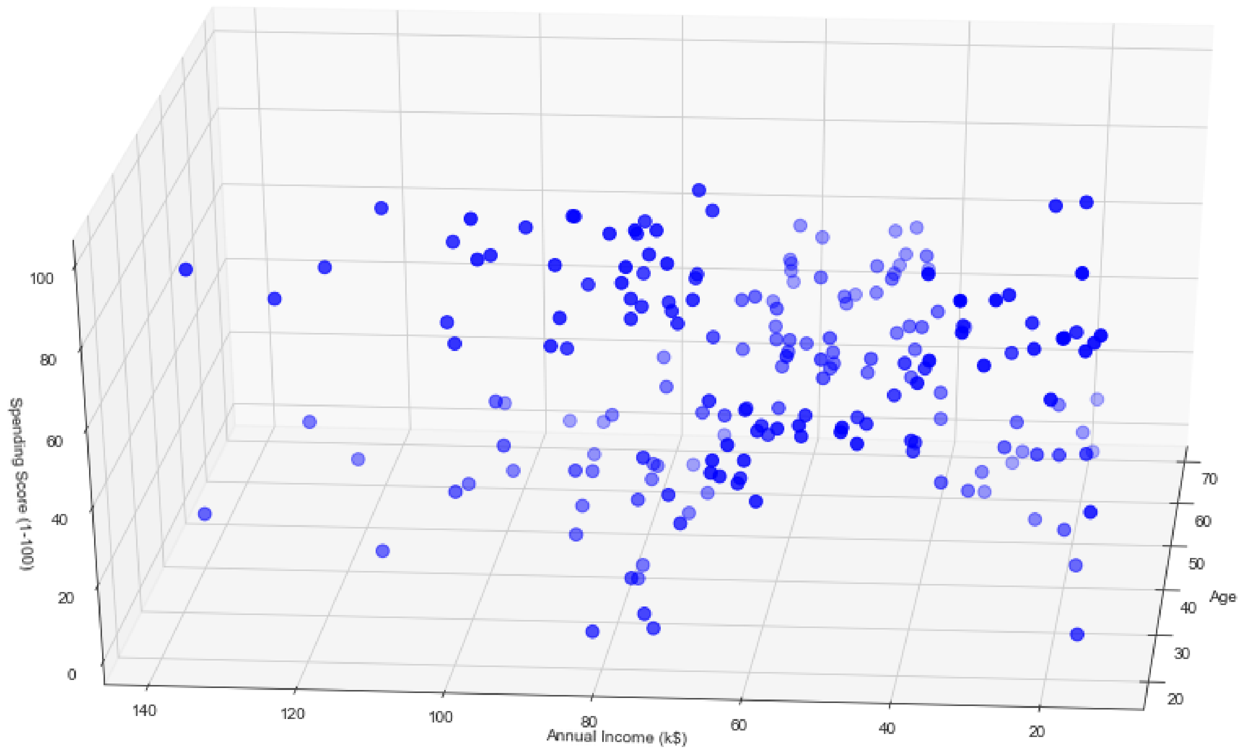
```
print(customer[1][:])
k = 5 #---------------NUMBER OF CLUSTERS
print(dataset[1:10])
```

```
[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]:

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



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

```

In [11]:
```
print(mediod[:])
#print(customer[128])

#print(dataset)
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]:
```
mediod[0,0] - 1
```

Out[12]:
```
107
```

In [13]:
```
###**********************                          #----------delete the mediods value from dataset
customer_new = np.delete(customer, (mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1,mediod[4,0] -
#dataset1 = np.delete(dataset, (mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1), axis=0)#-------
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]

```

In [14]:

```
print(mediod[2, 2])
print(customer.shape)
print(customer_new.shape)
distance_age = 0
distance_income = 0
distance_score = 0
total_distance_cost = np.zeros((len(customer_new),k))
print(total_distance_cost.shape)
print(customer_new.shape)
print(dataset_1.shape)
non_mediod = 0

```

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

Calculating distance between mediod and points and assigning clusters to it

In [15]:

```
###************************-----------------needs to be modified as the value of k changes
#--------------------------comparing when value of k=1
old_cost = 0
total_cost_cluster = 0
difference_cost=0
for l in range(100):

    distance_age_k1 = []
    distance_income_k1 = []
    distance_score_k1 = []
    total_distance_k1 = []
    q=0
    w=0
    e=0
    total=0
    i=0

    for j in range(len(customer_new)):#------------------------------calculating distance
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k1.append(q)
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k1.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k1.append(e)
        total = q+w+e
        total_distance_k1.append(total)

    #--------------------------comparing when value of k=2

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    q=0
    w=0
    e=0
    total=0
    i=1
    for j in range(len(customer_new)):#------------calcualting distance
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k2.append(q)
```

```
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k2.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k2.append(e)
        total = q+w+e
        total_distance_k2.append(total)

    #--------------------------comparing when value of k=3
    distance_age_k3 = []
    distance_income_k3 = []
    distance_score_k3 = []
    total_distance_k3 = []
    q=0
    w=0
    e=0
    total=0
    i=2


    for j in range(len(customer_new)):#-----------------------------------calculating distance
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k3.append(q)
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k3.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k3.append(e)
        total = q+w+e
        total_distance_k3.append(total)


    #--------------------------comparing when value of k=4
    distance_age_k4 = []
    distance_income_k4 = []
    distance_score_k4 = []
    total_distance_k4 = []
    q=0
    w=0
    e=0
    total=0
    i=3
    for j in range(len(customer_new)):
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
```

```
        distance_age_k4.append(q)
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k4.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k4.append(e)
        total = q+w+e
        total_distance_k4.append(total)





    #--------------------------comparing when value of k=5
    distance_age_k5 = []
    distance_income_k5 = []
    distance_score_k5 = []
    total_distance_k5 = []
    q=0
    w=0
    e=0
    total=0
    i=4
    for j in range(len(customer_new)):
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k5.append(q)
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k5.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k5.append(e)
        total = q+w+e
        total_distance_k5.append(total)



    #print(len(total_distance_k1))
    #print(len(total_distance_k2))

    #print(len(total_distance_k3))

    #print(len(total_distance_k4))

    cost_1 = pd.DataFrame({'Cost_1':total_distance_k1})
```

```
    cost_2 = pd.DataFrame({'Cost_2':total_distance_k2})
    cost_3 = pd.DataFrame({'Cost_3':total_distance_k3})
    cost_4 = pd.DataFrame({'Cost_4':total_distance_k4})
    cost_5 = pd.DataFrame({'Cost_5':total_distance_k5})

    #dfn = pd.concat([dataset_1,cost_1,cost_2,cost_3,cost_4], axis=1)
    #dfn= pd.merge(dataset_1, cost_1,cost_2,cost_3,cost_4,)
    dataset_1 = dataset_1.assign(cost_1=cost_1.values,cost_2=cost_2.values,cost_3=cost_3.values,cost_4=cost_4.value
    #print(df_col)

    customer_new = dataset_1.to_numpy()
    #print(customer_new[0:10, :])#---------------combined dataset which displays cost of all k, the last four colour

    #print(dataset_1.shape)
    #print("hello")
    #print(cost_1.shape)
    #print(cost_2.shape)
    #print(cost_3.shape)
    #print(cost_4.shape)
    #print(customer_new.shape)
    #print(dataset_1.shape)



    #**************************************
    #----now we will create 4 clusters as the value of k =4 on the basis of total distance or cost

    cluster_1 = []
    cluster_2 = []
    cluster_3 = []
    cluster_4 = []
    cluster_5 = []

    q=0
    w=0
    e=0
    r=0
    t=0
    i=0

    for i in range(len(customer_new)):#-------------select the points with minimum distance

```

```
        if(customer_new[i][5] <= customer_new[i][6] and customer_new[i][5] <= customer_new[i][7] and customer_new[i
            #print("five")
            q = customer_new[i]
            cluster_1.append(q)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_1"



        if(customer_new[i][6] <= customer_new[i][5] and customer_new[i][6] <= customer_new[i][7] and customer_new[i
            #print("six")
            w = customer_new[i]
            cluster_2.append(w)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_2"

        if(customer_new[i][7] <= customer_new[i][6] and customer_new[i][7] <= customer_new[i][5] and customer_new[i
           # print("seven")
            e = customer_new[i]
            cluster_3.append(e)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_3"

        if(customer_new[i][8] <= customer_new[i][6] and customer_new[i][8] <= customer_new[i][7] and customer_new[i
           # print("eight")
            r = customer_new[i]
            cluster_4.append(r)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_4"

        if(customer_new[i][9] <= customer_new[i][6] and customer_new[i][9] <= customer_new[i][7] and customer_new[i
           # print("eight")
            t = customer_new[i]
            cluster_5.append(t)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_5"

    print("lenth is", + len(cluster_5))
    if(len(cluster_5) == 0):
        cluster_5 = cluster_5_old
    if(len(cluster_4) == 0):
        cluster_4 = cluster_4_old
    if(len(cluster_3) == 0):
        cluster_3 = cluster_3_old
    if(len(cluster_2) == 0):
        cluster_2 = cluster_2_old
    if(len(cluster_1) == 0):
```

```
        cluster_1 = cluster_2_old
    total_cost_cluster = 0
    difference_cost=0
    cluster_1 = np.asarray(cluster_1)
    cluster_2 = np.asarray(cluster_2)
    cluster_3 = np.asarray(cluster_3)
    cluster_4 = np.asarray(cluster_4)
    cluster_5 = np.asarray(cluster_5)
    sum_cluster_1 = cluster_1[:,5].sum(axis=0)#----------adding all the particalura coloumn which is cost of partic
    sum_cluster_2 = cluster_2[:,6].sum(axis=0)
    sum_cluster_3 = cluster_3[:,7].sum(axis=0)#-------adding all the minimum cost of particualr clusters
    sum_cluster_4 = cluster_4[:,8].sum(axis=0)
    sum_cluster_5 = cluster_5[:,9].sum(axis=0)
    total_cost_cluster = sum_cluster_1 + sum_cluster_2 + sum_cluster_3 +sum_cluster_4+sum_cluster_5
    print(total_cost_cluster, l+1)
    #print(mediod)

    difference_cost = total_cost_cluster - old_cost
    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
        mediod[4]= non_mediod
        old_cost = total_cost_cluster
        print("difference is", + difference_cost)
    cluster_5_old = cluster_5
    old_cost = total_cost_cluster

    difference_cost= total_cost_cluster - old_cost
    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
        mediod[3]= non_mediod
        old_cost = total_cost_cluster
        #print("difference is", + difference_cost)
    cluster_4_old = cluster_4
    old_cost = total_cost_cluster

    difference_cost= total_cost_cluster - old_cost
    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
        mediod[2]= non_mediod
```

```
        old_cost = total_cost_cluster
        #print("difference is", + difference_cost)
    cluster_3_old = cluster_3
    old_cost = total_cost_cluster


    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
        mediod[1]= non_mediod
        old_cost = total_cost_cluster
        #print("difference is", + difference_cost)
    cluster_2_old = cluster_2
    old_cost = total_cost_cluster

    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:10], axis=1)
        mediod[0]= non_mediod
        old_cost = total_cost_cluster
        print("difference is", + difference_cost)
    cluster_1_old = cluster_1
    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
```

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

New coloumn added below for assigned cluster

In [16]:
```
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 [ ]:
```

```

Calculating silhoutte width of each cluster

In [18]:
```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_1)):
    for j in range(len(cluster_1)):#------------calcualting distance
        q = manhattan_distance(cluster_1[i,2],cluster_1[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_1[i,3],cluster_1[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_1[i,4],cluster_1[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_1)
    #print(a_sil)
    distance_sil.clear()


    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0

```

```
    total=0
    n=0#--------------for cluster 1
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,5)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5

```

```
    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_1[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_1[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_1[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 1", +  sill)

    average_sill.append(sill)
s_c1 = sum(average_sill)/len(average_sill)
print("Average silhoutee 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]:

```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_2)):
    for j in range(len(cluster_2)):#------------calcualting distance
        q = manhattan_distance(cluster_2[i,2],cluster_2[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_2[i,3],cluster_2[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_2[i,4],cluster_2[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_2)
    #print(a_sil)
    distance_sil.clear()



    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
```

```python

    total=0
    n=1#--------------for cluster 2
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,5)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5
```

```
    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_2[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_2[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_2[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#-----------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 2", +  sill)

    average_sill.append(sill)
s_c2 = sum(average_sill)/len(average_sill)
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]:
```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_3)):
    for j in range(len(cluster_3)):#------------calcualting distance
        q = manhattan_distance(cluster_3[i,2],cluster_3[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_3[i,3],cluster_3[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_3[i,4],cluster_3[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_3)
    #print(a_sil)
    distance_sil.clear()


    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0

```

```
    total=0
    n=2#--------------for cluster 2
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,5)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5

```

```
    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_3[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_3[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_3[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 3", +  sill)

    average_sill.append(sill)
s_c3 = sum(average_sill)/len(average_sill)
print("Average silhoutee width for cluster 3", + s_c3)


```

```
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 silhoutee width for cluster 3 0.715622562910999
```

In [21]:

```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_4)):
    for j in range(len(cluster_4)):#------------calcualting distance
        q = manhattan_distance(cluster_4[i,2],cluster_4[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_4[i,3],cluster_4[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_4[i,4],cluster_4[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_4)
    #print(a_sil)
    distance_sil.clear()





    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
```

```
    e=0
    a=0

    total=0
    n=3#--------------for cluster 4
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,5)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
```

```
    if (cluster_min == 4):
            selected_cluster= cluster_5

    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_4[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_4[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_4[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 4", +  sill)

    average_sill.append(sill)
s_c4 = sum(average_sill)/len(average_sill)
print("Average silhoutee 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]:
```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_5)):
    for j in range(len(cluster_5)):#------------calcualting distance
        q = manhattan_distance(cluster_5[i,2],cluster_5[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_5[i,3],cluster_5[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_5[i,4],cluster_5[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_5)
    #print(a_sil)
    distance_sil.clear()

    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0

    total=0
```

```
    n=4#--------------for cluster 5
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,5)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5

    distance_age_k2 = []
```

```
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_5[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_5[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_5[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 5", +  sill)

    average_sill.append(sill)
s_c5 = sum(average_sill)/len(average_sill)
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 silhoutee width for cluster 1 0.712167639240278
```

Average Silhoutte width of dataset

In [23]:

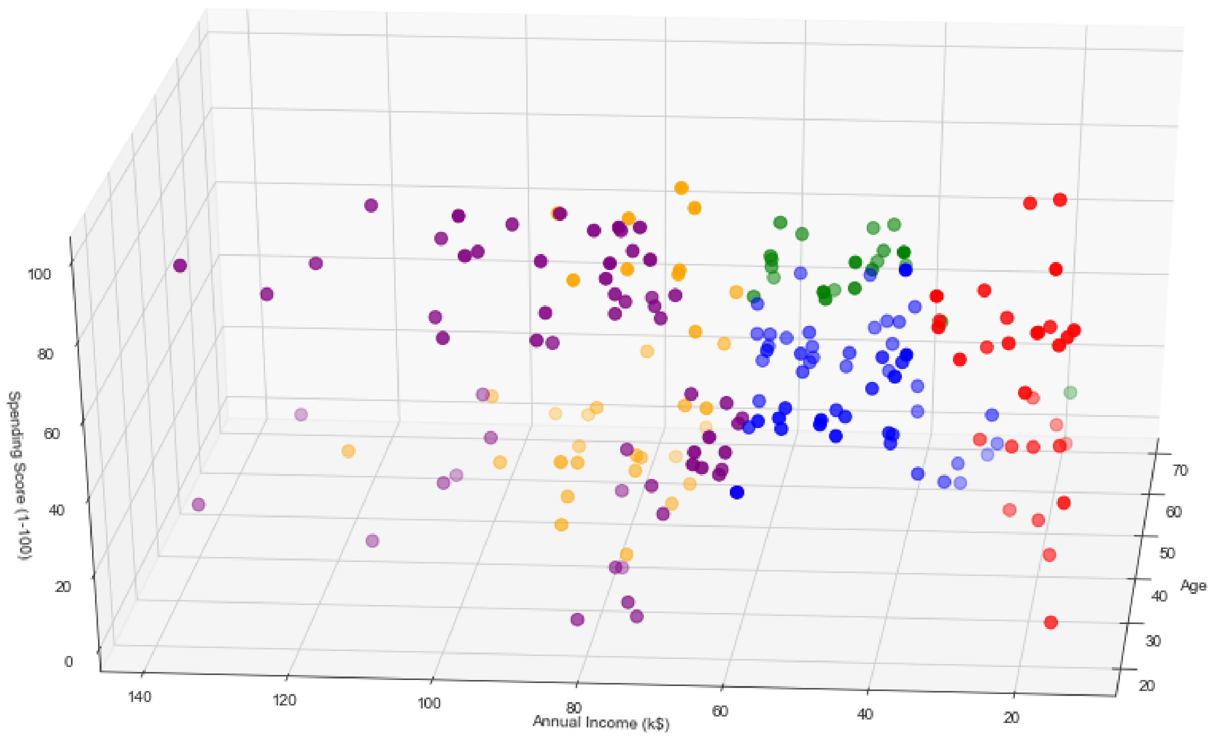
```
Average_dataset = (s_c1+s_c2+s_c3+s_c4+s_c5)/5
print("Average silhoutte width of dataset", + Average_dataset)
```

```
Average silhoutte width of dataset 0.6920984988098893
```

Plotting assigned clusters below

In [25]:

```
cluster_1 = np.asarray(cluster_1)
cluster_2 = np.asarray(cluster_2)
cluster_3 = np.asarray(cluster_3)
cluster_4 = np.asarray(cluster_4)
cluster_5 = np.asarray(cluster_5)
sns.set_style("white")
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
clusterX1 = cluster_1[:,2].tolist()
clusterY1 = cluster_1[:,3].tolist()
clusterZ1 = cluster_1[:,4].tolist()
clusterX2 = cluster_2[:,2].tolist()
clusterY2 = cluster_2[:,3].tolist()
clusterZ2 = cluster_2[:,4].tolist()
clusterX3 = cluster_3[:,2].tolist()
clusterY3 = cluster_3[:,3].tolist()
clusterZ3 = cluster_3[:,4].tolist()
clusterX4 = cluster_4[:,2].tolist()
clusterY4 = cluster_4[:,3].tolist()
clusterZ4 = cluster_4[:,4].tolist()
clusterX5 = cluster_5[:,2].tolist()
clusterY5 = cluster_5[:,3].tolist()
clusterZ5 = cluster_5[:,4].tolist()
ax.scatter(clusterX1, clusterY1, clusterZ1, c='blue', s=60)
ax.scatter(clusterX2, clusterY2, clusterZ2, c='green', s=60)
ax.scatter(clusterX3, clusterY3, clusterZ3, c='orange', s=60)
ax.scatter(clusterX4, clusterY4, clusterZ4, c='purple', s=60)
ax.scatter(clusterX5, clusterY5, clusterZ5, c='red', s=60)
ax.view_init(30, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
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 [ ]:
```

```

In [ ]:
```


```