

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```


```
1 import warnings
2 warnings.simplefilter("ignore")
```

```
1 colnames=['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
2 print(len(colnames))
```

```
4
```

```
1 iris_data=pd.read_csv('/content/drive/MyDrive/iris.txt', sep=" ", names=colnames,header=None)
```

```
1 iris_data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	
0	5.7	4.4	1.5	0.4	
1	5.5	4.2	1.4	0.2	
2	5.2	4.1	1.5	0.1	
3	5.8	4.0	1.2	0.2	
4	5.4	3.9	1.7	0.4	

```
1 iris_data.shape
```

```
(150, 4)
```

```
1 def euclidean_distance(point1, point2):
2     dist = np.linalg.norm(point1 - point2)
```

```
3     return dist
```

```
1 def manhattan_distance(point1, point2):
```

```
2     return np.sum([abs(value1 - value2) for value1, value2 in zip(point1, point2)])
```

```
1 def cosine_similarity(point1,point2):
```

```
2     return np.dot(point1, point2) / (np.linalg.norm(point1) * np.linalg.norm(point2))
```

```
1 def sse_k(data,cluster,centroids,K,metric):
```

```
2     sse = []
```

```
3     count = [0]*K
```

```
4     dist1 = 0
```

```
5     dist2 = 0
```

```
6     dist3 = 0
```

```
7     for i in range(data.shape[0]):
```

```
8         if cluster[i] == 1:
```

```
9             dist1+= (euclidean_distance(data[i],centroids[0])**2)
```

```
10            count[0]+=1
```

```
11        elif cluster[i] == 2:
```

```
12            dist2+= (euclidean_distance(data[i],centroids[1])**2)
```

```
13            count[1]+=1
```

```
14        elif cluster[i] == 3:
```

```
15            dist3+= (euclidean_distance(data[i],centroids[2])**2)
```

```
16            count[2]+=1
```

```
17        # print(count[0])
```

```
18        # print(count[1])
```

```
19        # print(count[2])
```

```
20        # print(dist1)
```

```
21        # print(dist2)
```

```
22        # print(dist3)
```

```
23        #sse_1 = (dist1/count1)+(dist2/count2)+(dist3/count3)
```

```
24        #sse.append(sse_1)
```

```
25        sse_1 = dist1+dist2+dist3
```

```
26
```

```
27        #sse_total = np.array(sse)
```

```
28        return sse_1,count
```

```

1 def ssb(data,count,centroids,K):
2     cent_1 = [0]*len(centroids[0])
3     for i in range(K):
4         cent_1 += centroids[i]
5
6     cent_1 /= K
7
8
9     print(cent_1)
10    value = 0
11    for i in range(K):
12        print((euclidean_distance(centroids[i],cent_1)**2))
13        value+= count[i]*(euclidean_distance(centroids[i],cent_1)**2)
14        print(value,i)
15    return value

```

```

1 import random
2 from collections import defaultdict
3 def K_Means_predict_man(data,K,max_iter,rand_seed):
4     centroids = defaultdict(int)
5     cluster = [0]*iris_data1.shape[0]
6     random.seed(rand_seed)
7     mylist = np.arange(data.shape[0])
8     list1 = mylist.tolist()
9     x = random.sample(list1,K)
10    #print(x)
11    for i in range(K):
12        #initializing 1st cluster center
13        num1 = x[i]
14        #print(data[num1])
15        centroids[i] = data[num1]
16
17    iter=0
18    #print(2)
19
20    for iteration in range(max_iter):
21        iter+=1
22        labels=defaultdict(list)
23        #print(data.shape)

```

```

24     #print(centroids)
25
26     for keys in range(K):
27         labels[keys]=[]
28
29     for datapoint1 in range(len(data)):
30         distance=[]
31         for datapoint2 in range(K):
32             dist=manhattan_distance(data[datapoint1],centroids[datapoint2])
33             #print("Dp",data[i])
34             #print("Cent",centroids[j])
35             #print(dist)
36             distance.append(dist)
37         min_distance=min(distance)
38         index=distance.index(min_distance)
39         labels[index].append(data[datapoint1])
40         cluster[datapoint1] = index+1
41         centroid_old=dict(centroids)
42
43     for i in range(K):
44         label=labels[i]
45
46         centroid_new=np.mean(label,axis=0)
47         centroids[i]=centroid_new
48         flag=1
49
50     for i in range(K):
51         a=centroids[i]
52         b=centroid_old[i]
53         temp = 0
54         for i in range(len(a)):
55             d = abs(a[i] - b[i])
56             temp+=d
57         if temp !=0:
58             flag = 0
59
60
61     if flag==1:
62         break

```

```
63 #print(iter)
64 return labels,centroids,cluster,iter
65
```

```
1 import random
2 from collections import defaultdict
3 def K_Means_predict(data,K,max_iter,rand_seed):
4     centroids = defaultdict(int)
5     cluster = [0]*iris_data1.shape[0]
6     random.seed(rand_seed)
7     mylist = np.arange(data.shape[0])
8     list1 = mylist.tolist()
9     x = random.sample(list1,K)
10    print("Initial Cluster Center Indices \n",x)
11    print("Initial Cluster Centers\n")
12    for i in range(K):
13        #initializing 1st cluster center
14        num1 = x[i]
15        print(data[num1])
16        centroids[i] = data[num1]
17
18    iter=0
19    #print(2)
20
21    for iteration in range(max_iter):
22        iter+=1
23        labels=defaultdict(list)
24        #print(data.shape)
25        #print(centroids)
26
27        for keys in range(K):
28            labels[keys]=[]
29
30        for datapoint1 in range(len(data)):
31            distance=[]
32            for datapoint2 in range(K):
33                dist=euclidean_distance(data[datapoint1],centroids[datapoint2])
34                #print("Dp",data[i])
```

```

35     #print("Cent",centroids[j])
36     #print(dist)
37     distance.append(dist)
38     min_distance=min(distance)
39     index=distance.index(min_distance)
40     labels[index].append(data[datapoint1])
41     cluster[datapoint1] = index+1
42     centroid_old=dict(centroids)
43
44     for i in range(K):
45         label=labels[i]
46
47         centroid_new=np.mean(label,axis=0)
48         centroids[i]=centroid_new
49         flag=1
50
51     for i in range(K):
52         a=centroids[i]
53         b=centroid_old[i]
54         temp = 0
55         for i in range(len(a)):
56             d = abs(a[i] - b[i])
57             temp+=d
58         if temp !=0:
59             flag = 0
60
61
62     if flag==1:
63         break
64     #print(iter)
65     return labels,centroids,cluster,iter
66

```

```

1 import random
2 from collections import defaultdict
3 def K_Means_predict_man1(data,K,max_iter,rand_seed):
4     centroids = defaultdict(int)
5     cluster = [0]*iris_data1.shape[0]
6     random.seed(rand_seed)

```

```

7  mylist = np.arange(data.shape[0])
8  list1 = mylist.tolist()
9  x = random.sample(list1,K)
10 #print(x)
11 for i in range(K):
12 #initializing 1st cluster center
13     num1 = x[i]
14     #print(data[num1])
15     centroids[i] = data[num1]
16
17 iter=0
18 #print(2)
19
20 for iteration in range(max_iter):
21     iter+=1
22     labels=defaultdict(list)
23     #print(data.shape)
24     #print(centroids)
25
26     for keys in range(K):
27         labels[keys]=[]
28
29     for datapoint1 in range(len(data)):
30         distance=[]
31         for datapoint2 in range(K):
32             dist=manhattan_distance(data[datapoint1],centroids[datapoint2])
33             #print("Dp",data[i])
34             #print("Cent",centroids[j])
35             #print(dist)
36             distance.append(dist)
37         min_distance=min(distance)
38         index=distance.index(min_distance)
39         labels[index].append(data[datapoint1])
40         cluster[datapoint1] = index+1
41         centroid_old=dict(centroids)
42
43     for i in range(K):
44         label=labels[i]
45

```

```

46     centroid_new=np.median(label,axis=0)
47     centroids[i]=centroid_new
48     flag=1
49
50     for i in range(K):
51         a=centroids[i]
52         b=centroid_old[i]
53         temp = 0
54         for i in range(len(a)):
55             d = abs(a[i] - b[i])
56             temp+=d
57         if temp !=0:
58             flag = 0
59
60
61     if flag==1:
62         break
63     #print(iter)
64     return labels,centroids,cluster,iter
65

```

```

1 import random
2 from collections import defaultdict
3 def K_Means_predict1(data,K,max_iter,rand_seed):
4     centroids = defaultdict(int)
5     cluster = [0]*iris_data1.shape[0]
6     random.seed(rand_seed)
7     mylist = np.arange(data.shape[0])
8     list1 = mylist.tolist()
9     x = random.sample(list1,K)
10    #print(x)
11    for i in range(K):
12        #initializing 1st cluster center
13        num1 = x[i]
14        centroids[i] = data[num1]
15
16    iter=0
17    #print(2)
18

```



```
19 for iteration in range(max_iter):
20     iter+=1
21     labels=defaultdict(list)
22     #print(data.shape)
23     #print(centroids)
24
25     for keys in range(K):
26         labels[keys]=[]
27
28     for datapoint1 in range(len(data)):
29         distance=[]
30         for datapoint2 in range(K):
31             dist=euclidean_distance(data[datapoint1],centroids[datapoint2])
32             #print("Dp",data[i])
33             #print("Cent",centroids[j])
34             #print(dist)
35             distance.append(dist)
36         min_distance=min(distance)
37         index=distance.index(min_distance)
38         labels[index].append(data[datapoint1])
39         cluster[datapoint1] = index+1
40         centroid_old=dict(centroids)
41
42     for i in range(K):
43         label=labels[i]
44
45         centroid_new=np.median(label,axis=0)
46         centroids[i]=centroid_new
47         flag=1
48
49     for i in range(K):
50         a=centroids[i]
51         b=centroid_old[i]
52         temp = 0
53         for i in range(len(a)):
54             d = abs(a[i] - b[i])
55             temp+=d
56         if temp !=0:
57             flag = 0
```

```
58
59
60     if flag==1:
61         break
62     #print(iter)
63     return labels,centroids,cluster,iter
64
```

```
1 import random
2 from collections import defaultdict
3 def K_Means_predict_cos(data,K,max_iter,rand_seed):
4     centroids = defaultdict(int)
5     cluster = [0]*iris_data1.shape[0]
6     random.seed(rand_seed)
7     mylist = np.arange(data.shape[0])
8     list1 = mylist.tolist()
9     x = random.sample(list1,K)
10    #print(x)
11    for i in range(K):
12        #initializing 1st cluster center
13        num1 = x[i]
14        centroids[i] = data[num1]
15
16    iter=0
17    #print(2)
18
19    for iteration in range(max_iter):
20        iter+=1
21        labels=defaultdict(list)
22        #print(data.shape)
23        #print(centroids)
24
25        for keys in range(K):
26            labels[keys]=[]
27
28        for datapoint1 in range(len(data)):
29            distance=[]
30            for datapoint2 in range(K):
31                dist=cosine_similarity(data[datapoint1],centroids[datapoint2])
```

```

32     #print("Dp",data[i])
33     #print("Cent",centroids[j])
34     #print(dist)
35     distance.append(dist)
36     min_distance=max(distance)
37     index=distance.index(min_distance)
38     labels[index].append(data[datapoint1])
39     cluster[datapoint1] = index+1
40     centroid_old=dict(centroids)
41
42     for i in range(K):
43         label=labels[i]
44
45         centroid_new=np.mean(label,axis=0)
46         #print(centroid_new)
47         centroids[i]=centroid_new
48         flag=1
49
50     for i in range(K):
51         a=centroids[i]
52         b=centroid_old[i]
53         temp = 0
54         #print(a)
55         #print(b)
56         for i in range(len(a)):
57             d = abs(a[i] - b[i])
58             temp+=d
59         if temp !=0:
60             flag = 0
61
62
63     if flag==1:
64         break
65     #print(iter)
66     return labels,centroids,cluster,iter
67

```

```

1 import random
2 from collections import defaultdict

```

```

3 def K_Means_predict_cos1(data,K,max_iter,rand_seed):
4     centroids = defaultdict(int)
5     cluster = [0]*iris_data1.shape[0]
6     random.seed(rand_seed)
7     mylist = np.arange(data.shape[0])
8     list1 = mylist.tolist()
9     x = random.sample(list1,K)
10    #print(x)
11    for i in range(K):
12        #initializing 1st cluster center
13        num1 = x[i]
14        centroids[i] = data[num1]
15
16    iter=0
17    #print(2)
18
19    for iteration in range(max_iter):
20        iter+=1
21        labels=defaultdict(list)
22        #print(data.shape)
23        #print(centroids)
24
25        for keys in range(K):
26            labels[keys]=[]
27
28        for datapoint1 in range(len(data)):
29            distance=[]
30            for datapoint2 in range(K):
31                dist=cosine_similarity(data[datapoint1],centroids[datapoint2])
32                #print("Dp",data[i])
33                #print("Cent",centroids[j])
34                #print(dist)
35                distance.append(dist)
36            min_distance=max(distance)
37            index=distance.index(min_distance)
38            labels[index].append(data[datapoint1])
39            cluster[datapoint1] = index+1
40            centroid_old=dict(centroids)
41

```

```

42     for i in range(K):
43         label=labels[i]
44
45         centroid_new=np.median(label,axis=0)
46         centroids[i]=centroid_new
47         flag=1
48
49     for i in range(K):
50         a=centroids[i]
51         b=centroid_old[i]
52         temp = 0
53         for i in range(len(a)):
54             d = abs(a[i] - b[i])
55             temp+=d
56         if temp !=0:
57             flag = 0
58
59
60     if flag==1:
61         break
62     #print(iter)
63     return labels,centroids,cluster,iter
64

```

```

1 sse_1 = []
2 sse_2 = []
3 sse_3 = []
4 count11 = []
5 ssb_k = []
6 tss_k = []
7 from collections import defaultdict
8 iris_data1 = iris_data.to_numpy()
9 for rand_seed in range(20):
10     classes,centroids,cluster,iter=K_Means_predict(iris_data1,3,10000,rand_seed)
11     classes1,centroids1,cluster1,iter1=K_Means_predict1(iris_data1,3,10000,rand_seed)
12     classes2,centroids2,cluster2,iter2=K_Means_predict_man(iris_data1,3,10000,rand_seed)
13     classes3,centroids3,cluster3,iter3=K_Means_predict_man1(iris_data1,3,10000,rand_seed)
14     classes4,centroids4,cluster4,iter4=K_Means_predict_cos(iris_data1,3,10000,rand_seed)
15     classes5,centroids5,cluster5,iter5=K_Means_predict_cos1(iris_data1,3,10000,rand_seed)

```

```

16 for i in range(0,3):
17     classes[i]=np.array(classes[i]).tolist()
18 for i in range(0,3):
19     classes1[i]=np.array(classes1[i]).tolist()
20 print("Iteration=%d \n"%rand_seed)
21 print("Euclidean")
22 print("Mean \n")
23 print("Max Iteratins Run=%d \n"%iter)
24 print("Final Centroids:",centroids)
25 SSE,count = sse_k(iris_data1,cluster,centroids,3,1)
26 sse_1.append(SSE)
27 SSB = ssb(iris_data1,count,centroids,3)
28 ssb_k.append(SSB)
29 count11.append(count)
30 ssb_k.append(SSB)
31 count11.append(count)
32 print("Total SSE =%f"%SSE)
33 print("Total SSB =%2f \n"%SSB)
34 print("\n")
35 for i in range(0,3):
36     print("Cluster %d"%i,len(classes[i]))
37 print("Median \n")
38 print("Max Iteratins Run=%d \n"%iter1)
39 print("Final Centroids:",centroids1)
40 SSE,count = sse_k(iris_data1,cluster1,centroids1,3,1)
41 print("Total SSE =%2f"%SSE)
42 #print("Total SSB =%2f \n"%SSB)
43 print("\n")
44 for i in range(0,3):
45     print("Cluster %d"%i,len(classes1[i]))
46 print("\n")
47 print("Manhattan")
48 print("Mean \n")
49 print("Max Iteratins Run=%d \n"%iter2)
50 print("Final Centroids:",centroids2)
51 SSE,count = sse_k(iris_data1,cluster2,centroids2,3,2)
52 print("Total SSE =%2f"%SSE)
53 print("\n")
54 for i in range(0,3):

```

```

55     print("Cluster %d"%i,len(classes2[i]))
56     print("Median \n")
57     print("Max Iteratins Run=%d \n"%iter3)
58     print("Final Centroids:",centroids3)
59     SSE,count = sse_k(iris_data1,cluster3,centroids3,3,2)
60     print("Total SSE =%2f"%SSE)
61     print("\n")
62     for i in range(0,3):
63         print("Cluster %d"%i,len(classes3[i]))
64         print("\n")
65         print("Cosine")
66         print("Mean \n")
67         print("Max Iteratins Run=%d \n"%iter4)
68         print("Final Centroids:",centroids4)
69         SSE,count = sse_k(iris_data1,cluster4,centroids4,3,3)
70         print("Total SSE =%2f"%SSE)
71         print("\n")
72         for i in range(0,3):
73             print("Cluster %d"%i,len(classes4[i]))
74             print("Median \n")
75             print("Max Iteratins Run=%d \n"%iter5)
76             print("Final Centroids:",centroids5)
77             SSE,count = sse_k(iris_data1,cluster5,centroids5,3,3)
78             sse_3.append(SSE)
79             print("Total SSE =%2f"%SSE)
80             print("\n")
81             for i in range(0,3):
82                 print("Cluster %d"%i,len(classes5[i]))
83                 print("\n")
84                 print("*****XXXXX*****")
85                 #print(centroids)

```

*****XXXXX*****

Initial Cluster Center Indices

[14, 23, 21]

Initial Cluster Centers

[5.3 3.7 1.5 0.2]

[5.5 3.5 1.3 0.2]

```
[5.1 3.5 1.4 0.2]
```

```
Iteration=2
```

```
Euclidean
```

```
Mean
```

```
Max Iteratins Run=5
```

```
Final Centroids: defaultdict(<class 'int'>, {0: array([6.30103093, 2.88659794, 4.95876289, 1.69587629]), 1: array([5.28333333, 3.70833333, 6.30103093, 2.88659794])})
```

```
97
```

```
24
```

```
29
```

```
123.79587628865977
```

```
3.8495833333333342
```

```
15.808275862068967
```

```
[5.45340878 3.22244835 2.69037307 0.75719823]
```

```
6.857967680295207
```

```
665.2228649886351 0
```

```
1.9304210810745257
```

```
711.5529709344238 1
```

```
1.8379977178688256
```

```
764.8549047526197 2
```

```
Total SSE =143.453735
```

```
Total SSB =764.854905
```

```
Cluster 0 97
```

```
Cluster 1 24
```

```
Cluster 2 29
```

```
Median
```

```
Max Iteratins Run=3
```

```
Final Centroids: defaultdict(<class 'int'>, {0: array([6.3, 2.9, 4.9, 1.6]), 1: array([5.4, 3.8, 1.5, 0.2]), 2: array([4.9, 3.2, 1.4, 0.2])})
```

```
98
```

```
17
```

```
35
```

```
130.01000000000002
```

```
2.7500000000000013
```

```
14.420000000000003
```

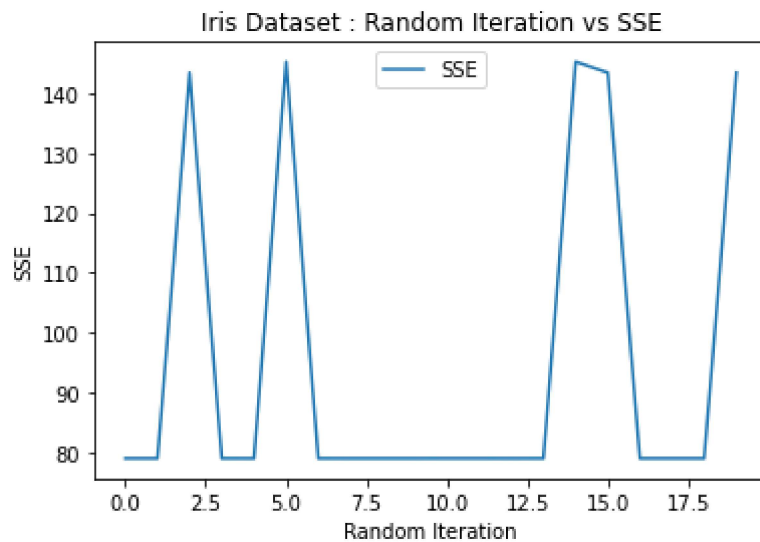
```
Total SSE =147.180000
```



```
Cluster 0 98  
Cluster 1 17  
Cluster 2 35
```

```
1 center = []  
2 for i in range(len(centroids)):  
3     center.append(centroids[i])  
4 cent = list(center)  
5 cent_1 = np.array(cent)
```

```
1 iterations = np.arange(20)  
2 #print(sse_1)  
3 x = plt.subplot( )  
4 x.plot(iterations, sse_1, label='SSE')  
5 #x.plot(k_1, cv_auc, label='AUC CV')  
6 plt.title('Iris Dataset : Random Iteration vs SSE')  
7 plt.xlabel('Random Iteration')  
8 plt.ylabel('SSE')  
9 x.legend()  
10 plt.show()
```

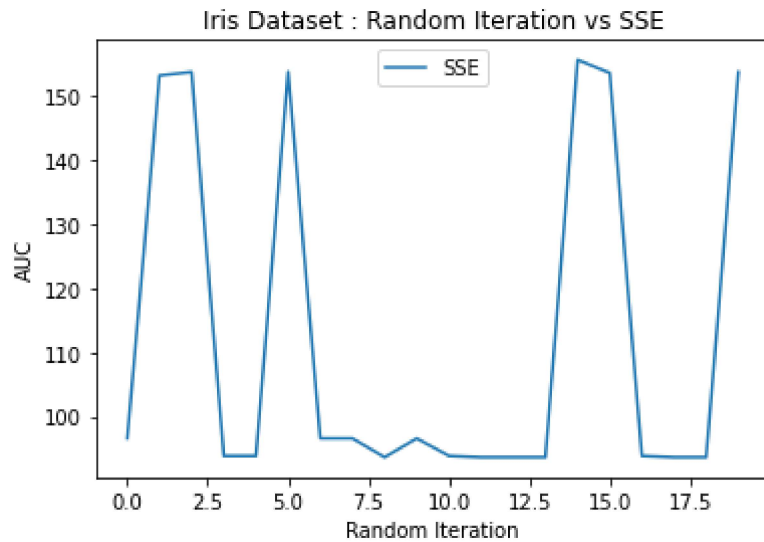


```

1 iterations = np.arange(20)
2 print(sse_3)
3 x = plt.subplot( )
4 x.plot(iterations, sse_3, label='SSE')
5 #x.plot(k_1, cv_auc, label='AUC CV')
6 plt.title('Iris Dataset : Random Iteration vs SSE')
7 plt.xlabel('Random Iteration')
8 plt.ylabel('AUC')
9 x.legend()
10 plt.show()

```

[96.56999999999998, 153.15000000000003, 153.70000000000002, 93.83999999999999, 93.83999999999999, 153.74, 96.57, 96.56999999999998, 93.61



```

1 import seaborn as sns
2 df_iris = pd.DataFrame(iris_data1, columns = colnames)
3 df_iris['cluster'] = cluster
4 sns.FacetGrid(df_iris, hue="cluster", size=5, hue_kws={"marker":["o", "o", "o", "x"]}).map(plt.scatter, "sepal_length", "sepal_width").add_

1 plt.scatter(df_iris['sepal_length'], df_iris['sepal_width'],c=df_iris['cluster'], s=50, cmap='viridis')
2 centers = cent_1
3 plt.scatter(centers[:,0], centers[:,1], c='black', s=200, alpha=0.8);

```

```
1 plt.scatter(df_iris['petal_length'], df_iris['petal_width'],c=df_iris['cluster'], s=50, cmap='viridis')
2 centers = cent_1
3 plt.scatter(centers[:,2], centers[:,3], c='black', s=200, alpha=0.8);
```

```
1 print(classes)
```

```
1 print(cluster)
```

```
1 with open('cluster8.txt', 'w') as f:
2     for i in cluster:
3         f.write(str(i) + "\n")
```

```
1 sse = []
2 count1 = []
3 for rand_seed in range(1):
4     classes,centroids,cluster,iter4=K_Means_predict(iris_data1,3,10000,rand_seed)
5     SSE,count = sse_k(iris_data1,cluster,centroids,3)
6     sse.append(SSE)
7     count1.append(count)
8     print("Total SSE =%2f"%SSE)
9     #classes1,centroids1,cluster1,iter2=K_Means_predict_cos1(iris_data1,3,10000,rand_seed)
10    for i in range(0,3):
11        classes[i]=np.array(classes[i]).tolist()
12    for i in range(0,3):
13        classes1[i]=np.array(classes1[i]).tolist()
14
15    print("Iteration=%d \n"%rand_seed)
16    print("Mean \n")
17    # for i in range(0,3):
18    #     print(len(classes[i]))
19    # print("Median \n")
20    # for i in range(0,3):
21    #     print(len(classes1[i]))
22    # print("\n")
23    #print(centroids)
```

```

1 # from collections import defaultdict
2 # iris_data1 = iris_data.to_numpy()
3 # for rand_seed in range(5):
4 #     classes,centroids,cluster=K_Means_predict(iris_data2,3,10000,rand_seed)
5 #     for i in range(0,3):
6 #         classes[i]=np.array(classes[i]).tolist()
7
8 #     for i in range(0,3):
9 #         print(len(classes[i]))
10 #     #print(centroids)
11
12 rand_seed = 0
13 classes,centroids,cluster,iter=K_Means_predict_cos1(iris_data1,3,10000,rand_seed)
14 for i in range(0,3):
15     classes[i]=np.array(classes[i]).tolist()
16
17 for i in range(0,3):
18     print(len(classes[i]))

```

```

1 print(cluster)

```

```

1 with open('cluster101.txt', 'w') as f:
2     for i in cluster:
3         f.write(str(i) + "\n")

```

```


1 SSE = sse_k(iris_data1,cluster,centroids,3)

```

```

1 print(SSE)

```



[Colab paid products](#) - [Cancel contracts here](#)