INTELLIGENT DATA ANALYSIS CS5152/6052

HOMEWORK #2 – PART II

2. Now use the same data set for the following sequence of steps.

a. For this problem consider only the (x,y) coordinates of the data points and ignore their class labels. Perform k-means clustering of these data points for values of k to be 3, 5, 7, 9, and 11. For each value of k run the clustering algorithm (Scikit or Matlab only) 6 times (with initial cluster centers being selected randomly) and report the following: (i) total SSE for each clustering run, (ii) Average SSE and its standard deviation for each k value, and (iii) the minimum and maximum SSE values for each value of k.

Solution:

```
#!/usr/bin/env python
# coding: utf-8

import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd


df = pd.read_excel("HW2Data.xlsx",usecols = "A,B,D", header = None, names = ['X1','X2', 'Y'])
np_data = df.to_numpy()


X = np_data[:,[0,1]]
y = np_data[:,2]
x1 = X[:,0]
x2 = X[:,1]


cluster_dict = {}
sse_dict = {}
cluster_list = [3,5,7,9,11]
for i in cluster_list:
    cluster_dict[i] = []
    sse_dict[i] = []
    for j in range(6):
        k_means = KMeans(n_clusters=i,init="random")
        model = k_means.fit(X)
        sse_dict[i].append(model.inertia_)
        cluster_dict[i].append(model)


sse_df = pd.DataFrame(index = cluster_list,columns = ["Epoch1","Epoch2","Epoch3","Epoch 4","Epoch5","Epoch6"])
for k,v in sse_dict.items():
    sse_df.loc[k] = [float(i) for i in v]
sse_df = sse_df.astype(float)


```

```
sse_df


from statistics import mean,stdev,pstdev
metrics_df = pd.DataFrame(index=cluster_list,columns=["Total SSE","Average SSE","Std. D
ev","min SSE","max SSE"])
for k,v in sse_dict.items():
    metrics_df.loc[k] = np.sum(v),mean(v),stdev(v),min(v),max(v)

metrics_df

```

In [28]:
```
sse_df
```

Out[28]:

	Epoch1	Epoch2	Epoch3	Epoch4	Epoch5	Epoch6
3	144037.227959	144037.227959	144039.237581	144037.227959	144037.227959	144037.227959
5	71332.515400	71332.515400	71341.048904	71341.048904	71332.515400	71332.515400
7	44857.752172	44857.855200	44861.630711	44902.731952	44857.752172	44861.630711
9	31158.520220	33614.745213	31144.081727	31143.336004	31140.517822	31155.266034
11	24530.078043	25007.227164	25125.145740	24958.814580	24544.091654	24578.137024

In [30]:
```
metrics_df
```

Out[30]:

	Total SSE	Average SSE	Std. Dev	min SSE	max SSE
3	864225	144038	0.820425	144037	144039
5	428012	71335.4	4.40668	71332.5	71341
7	269199	44866.6	17.821	44857.8	44902.7
9	189356	31559.4	1006.93	31140.5	33614.7
11	148743	24790.6	268.672	24530.1	25125.1

b. Consider the best clustering obtained for k=5. Each data point is assigned a cluster number by your clustering algorithm. Plot all the given data points on the grid after assigning a different color/symbol to members of each cluster. Write the cluster-specific SSE for each individual cluster on the plot.

Solution:

The best clustering obtained would be the model with the least SSE. That would be the model at Epoch-1.

```
least_sse = sse_dict.get(5).index(min(sse_dict.get(5)))
model_index = cluster_dict.get(5)
model_least_sse = model_index[least_sse]
model_least_sse

```

```
centroids = model_least_sse.cluster_centers_
labels = model_least_sse.labels_

#Calculating local sse
def calculate_localsse(data,cx,cy,i,clusters):
    sse = [((x-cx)**2 + (y-cy)**2) for x,y in data[labels==i]]
    total = sum(sse)
    return total

sse = []
for i, (cx, cy) in enumerate(centroids):
    local_sse = calculate_localsse(X,cx,cy,i,model_least_sse)
    sse.append(local_sse)


total_sse = pd.DataFrame(index=[0,1,2,3,4],columns = ["SSE"])
for index,item in enumerate(sse):
    total_sse.loc[index] = item

total_sse


fig, ax = plt.subplots(figsize=(13,10))
plotter_dict = {0:{'c':'green','marker':'o'},
             1:{'c':'yellow','marker':'^'},
             2:{'c':'blue','marker':'H'},
             3:{'c':'purple','marker':'X'},
             4:{'c':'red','marker':'s'}}

for index,label in enumerate([0,1,2,3,4]):

    ax.scatter(
        x1[np.where(labels == label)], x2[np.where(labels == label)],
        s=40,
        c=plotter_dict.get(label)['c'],
        marker=plotter_dict.get(label)['marker'],
        edgecolor='black',
        label='Cluster - '+str(index)
    )

ax.legend()

for index,centroid in enumerate(centroids):
    ax.scatter(centroid[0],centroid[1],marker="*",
                s = 150,
                c = "black"
              )
    ax.text(centroid[0],centroid[1]+3,"sse = " + str(int(total_sse.at[index,'SSE'])),bb
ox=dict(facecolor='lightblue',alpha=0.8))

plt.savefig("clustering_5.png")
```

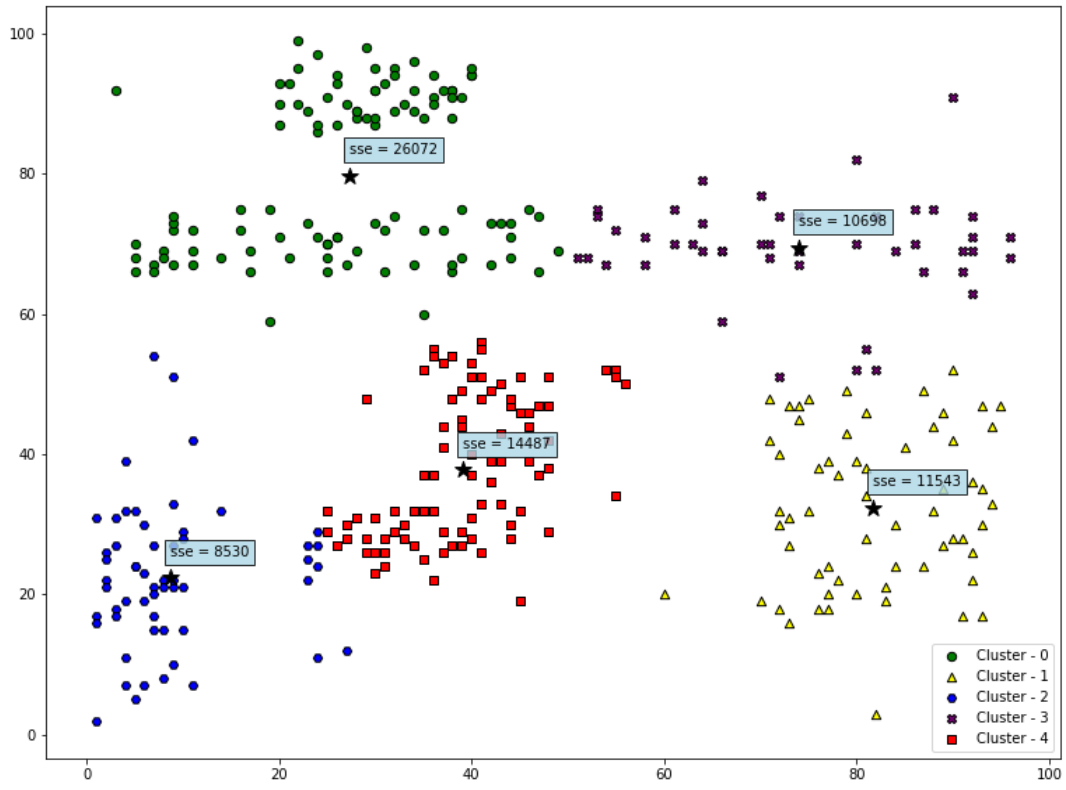
```
In [11]: total_sse = pd.DataFrame(index=[0,1,2,3,4],columns = ["SSE"])
         for index,item in enumerate(sse):
             total_sse.loc[index] = item

         total_sse
```

Out[11]:

	SSE
0	26072.3
1	11543
2	8530.86
3	10698.5
4	14487.9

Considering Epoch-1's model, the plot is as follows :

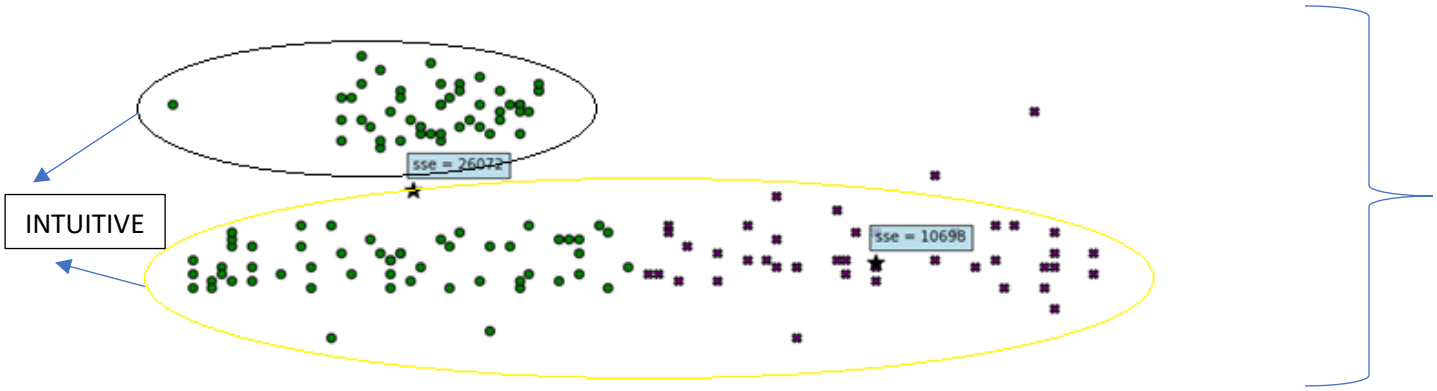


c. Comment on the cluster boundaries obtained by your clustering algorithm, focusing on how they are different from your intuitive idea of five clusters from this dataset.

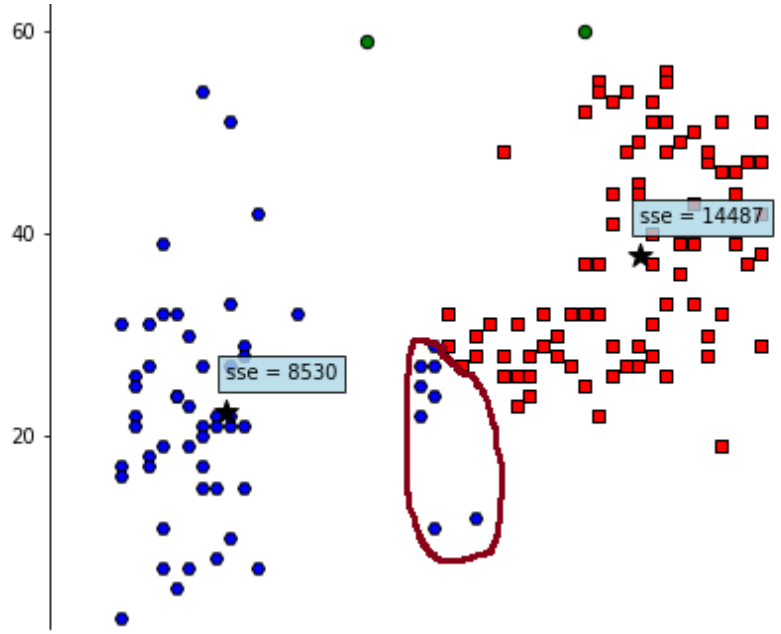
Solution:

The cluster boundaries obtained from the best epoch with k = 5, results in clusters that are slightly unusual than the intuitive sense of cluster formation.

For example, Cluster – 0 would be just the first half portion of the actual cluster, intuitively, in the sense that, we would focus on the density of the datapoints to form a cluster.

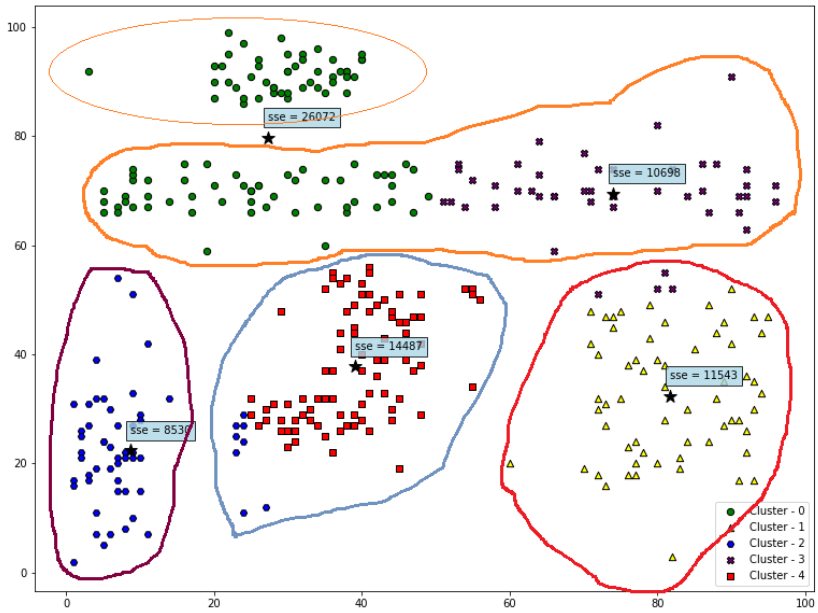


Additionally, another bunch of 8 datapoints seem to be closer to Cluster – 4 (red points) than to Cluster – 2 (blue points), but, the model categorizes them as to belong to Cluster – 2 as opposed to cluster -4.



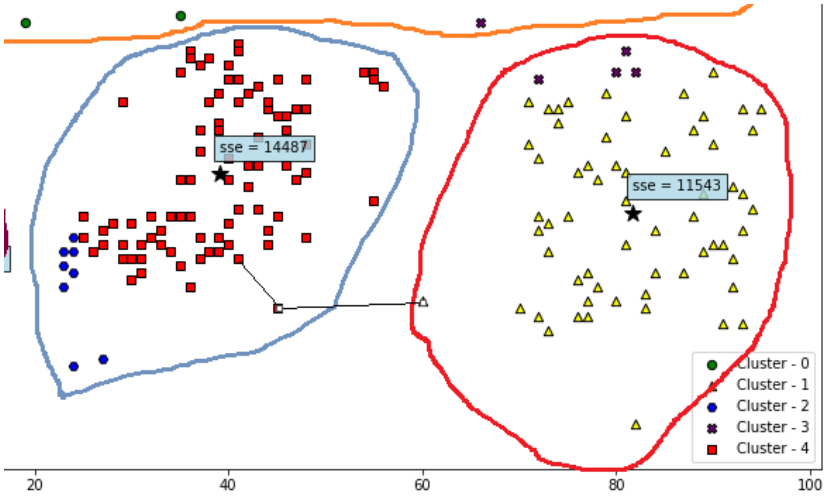
d. On the data plot, mark the boundaries of the five clusters that you think are intuitively distinct clusters. Justify the boundaries you have drawn, giving reasons for preferring the boundaries you have drawn.

Solution:



The boundaries I think that would be formed are as above since K-means can generate clusters of random shapes and non-linear boundaries.

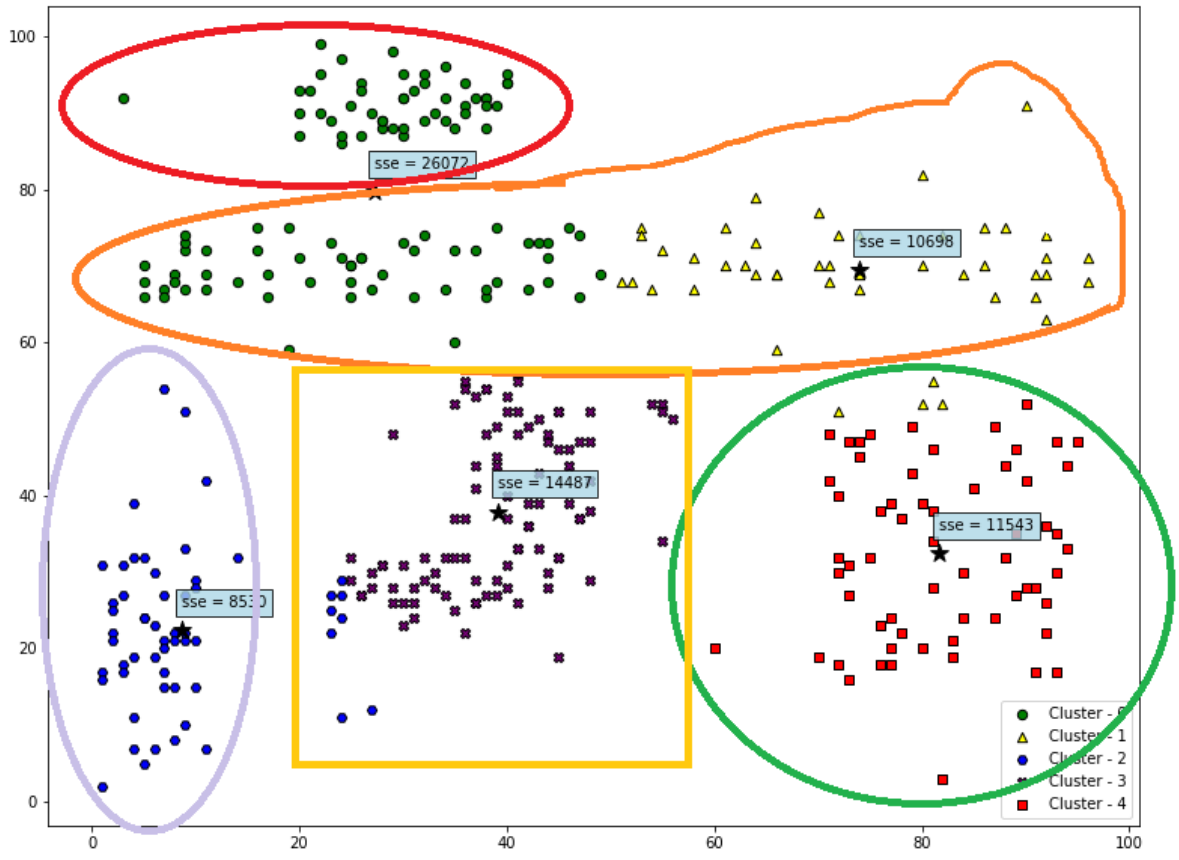
- I have chosen clusters based on each datapoint's similarity influenced by its distance ; hence, those group of datapoints that form a dense proportion form a cluster. Any anomalies/outliers are also categorized based on the distance.
- If a datapoint appears equidistant to 2 different data points belonging to different dense regions, I estimate it to belong to that cluster whose distance from the dense region is lesser, giving lesser priority to the distance from an outlier.



e. Compute the Rand Index between the clustering used in parts (b) and (d) above. Show your work and steps performed to arrive at the Rand Index value. How can we interpret the meaning of the Rand index obtained by you?

Solution:

Referring an obsolete version of (d) as calculations of clustering for rand index was done based on this color coding.



2 e) Rand-Index Calculation:

Solution: clustering between (b) and (d).

Total no of comparisons = $\frac{N(N-1)}{2}$

$= \frac{360 \times 359}{2} = 180 \times 359$

$= 64,620$ comparisons.

	Same cluster	Different cluster
Same Cluster	a	b
Different cluster	c	d.

Total no of positives = True positives + False positives

Total positives = TP + FP

Cluster 0 : $44C_2$

cluster 1 : $101C_2$

cluster 2 : $48C_2$

cluster 3 : $102C_2$

cluster 4 : $65C_2$

Total positives = $44C_2 + 101C_2 + 48C_2 + 102C_2 + 65C_2$

$= 946 + 5050 + 1128 + 5151 + 2080$

$= 14,355$

True positives = Cluster - 0 $44C_2$
Cluster - 1 $59C_2 + 42C_2$
Cluster - 2 $48C_2$
Cluster - 3 $94C_2 + 8C_2$
Cluster 4 $61C_2 + 4C_2$

$\sum_{i \in C_i}$

True positives (TP) = $44C_2 + 59C_2 + 42C_2 + 48C_2 + 94C_2 + 8C_2 + 61C_2 + 4C_2$

$= \underbrace{946}_{0} + \underbrace{1711 + 861}_{1} + \underbrace{1128}_{2} + \underbrace{4371 + 28}_{3} + \underbrace{1830 + 6}_{4}$

TP = 10,881

False positive (FP) = Total positives − True positives

~~= 64,620 − 10~~

= 14,355 − 10,881 = 3,474

FP = 3,474

Total negatives = True negatives + False negatives
Total negatives = TN + FN

Total negatives = Total pairs − Total positives
= 64,620 − 14,355
= 50,265

False negatives (FN)

$= [44 \times 59] + [42 \times 4]$

(44: green in green; 59: green in another cluster; 42: yellow in yellow; 4: yellow in another cluster)

$+ [48 \times 8] + 0 + 0$

(48: blue in blue; 8: blue in another cluster; 0: purple; 0: red)

$= 2596 + 168 + 384$

$= 3148$

$\boxed{FN = 3148}$

TN = Total negatives − False negatives (FN)

$= 50,265 - 3,148$

$= 47,117$

$\boxed{TN = 47,117}$

$$\text{Rand Index} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{10881 + 47117}{14355 + 50265}$$

$$= \frac{57,998}{64,620} = 0.8975.$$

0.89 is the value obtained for the rand-index. This means that the proportion of the datapoints in (d) which were placed in the wrong cluster when compared to the clustering obtained in (b) is relatively lower.

This is the same as accuracy we calculate in classification. To base the quality of clustering on this value depends on the use-case.

If the emphasis is on getting as many true positives and true negatives as possible, then this metric is a good measure. If the cost of getting a false negative is high, then this may not be the best way to evaluate the categorization.