Intelligent Data Analysis CS5152/6052
Homework #2
Due Date(s): Problem #1 is due by March 30th, 9PM; Problem #2 is due by April 4th, 9PM.

Consider the data in a 2-dimensional space given in the attached file. It includes points belonging to two different classes. A plot of the data points showing their distribution is attached with the asignment. Perform the following tasks with this dataset.

Include your response to each part of a homework question in a single pdf file and upload it on Blackboard. There are two assignments created on Blackboard, one for responses to Question#1 and the other for Question#2. Two separate submissions are required by their respective due dates.

1. Your task is to design a good classifier for this data set. You can use only one of the following two toolboxes: Scikit library (Python), or Matlab. For this question we are going to find a non-linear SVM classifier that fits this data. Perform the following steps and report your work and results as stated below.
    a. Use non-linear SVM with Radial-basis kernel, to design classifier for the two classes of this dataset. Randomly select 75%% of the data points as training data and use the remaining 25% as the test data. Create and test an SVM model for at least 6 different values of the regularization parameter (this parameter is named differently in each toolbox, read the documentation for your toolbox). For each of the six runs, show the chosen parameter value, confusion matrix, and accuracy, and precision values. Plot the accuracy, precision, and recall values for the six parameter values.

Gamma=1

```
the Gamma is  1.0
Accuracy of Classifier:0.633333
Precision of Classifier:0.547945
Recall of Classifier:1.000000
Confusion matrix of Classifier: [[40  0]
 [33 17]]
```

Gamma = 0.5

```
the Gamma is  0.5
Accuracy of Classifier:0.833333
Precision of Classifier:0.766667
Recall of Classifier:0.978723
Confusion matrix of Classifier: [[46  1]
 [14 29]]
```

Gamma = 0.75

```
the Gamma is  0.75
Accuracy of Classifier:0.688889
Precision of Classifier:0.583333
Recall of Classifier:0.921053
Confusion matrix of Classifier: [[35  3]
 [25 27]]
```

Gamma = 0.6

```
the Gamma is  0.6
Accuracy of Classifier:0.733333
Precision of Classifier:0.942857
Recall of Classifier:0.600000
Confusion matrix of Classifier: [[33 22]
 [ 2 33]]
```

Gamma = 0.55

```
the Gamma is  0.55
Accuracy of Classifier:0.822222
Precision of Classifier:0.741379
Recall of Classifier:0.977273
Confusion matrix of Classifier: [[43  1]
 [15 31]]
```

Gamma = 0.525

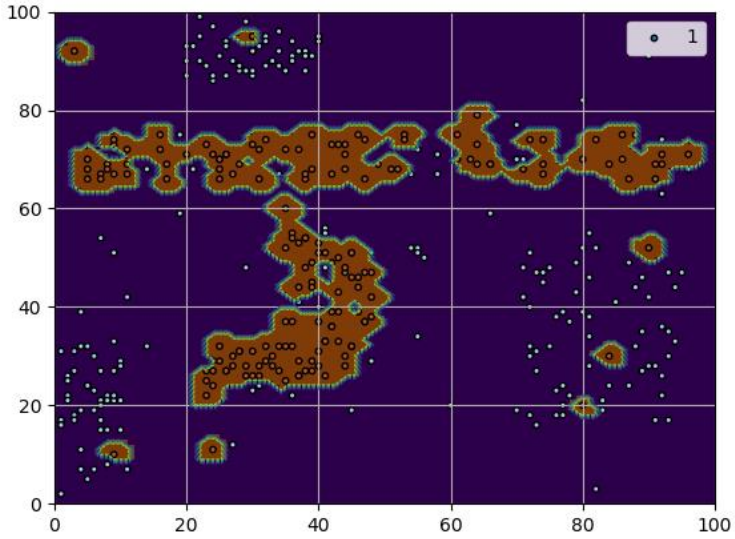
```
the Gamma is  0.525
Accuracy of Classifier:0.866667
Precision of Classifier:0.824561
Recall of Classifier:0.959184
Confusion matrix of Classifier: [[47  2]
 [10 31]]
```

Base on the output shown, when gamma = 0.525, the accuracy will be the highest.

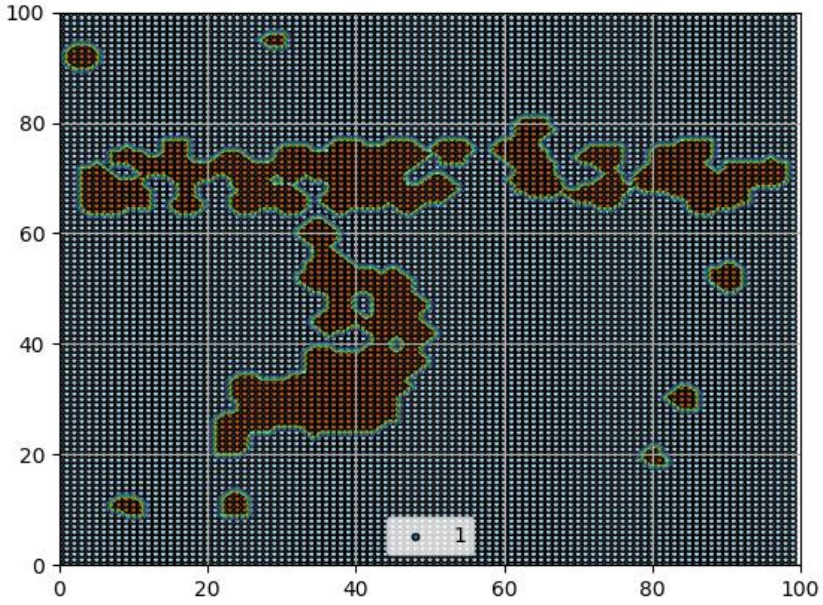
b. Select the model having the highest accuracy value from the six models generate above. For this model consider all the points of the 100X100 grid as the test points. Find the class for each of these test points. Use two different colors for denoting the predicted class label for a point and create the plot with corresponding colors for each point of the full grid. Your output will be a grid in which each point will be plotted using one of the two colors.

We choose gamma = 0.525 which gives the highest accuracy is 0.86667.

Blue points are class 1
Red points are class 2



For all data in the grid:
Blue points are class 1
Red points are class 2

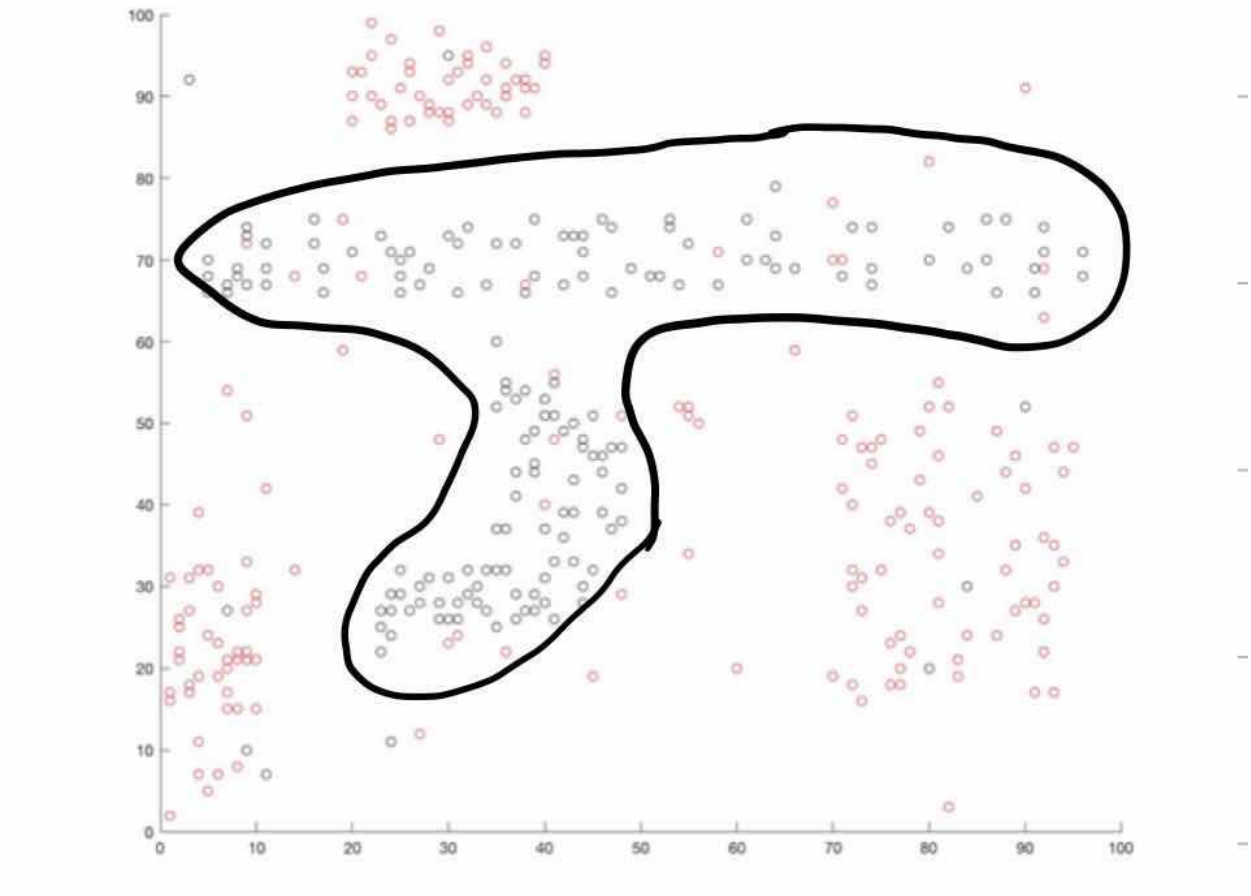


c. The grid generated in (b) above shows the boundaries learned by the classifier to predict the two classes. Comment on the following:

   i. How efficient and effective are these boundaries, given data distribution of the two classes.

   The boundaries can **effectively** divide two classes. Base on the output plot the boundaries are **not efficient**, the boundary divides the data into small area that some of the area only contain one data point.

   ii. On a plot of original data points draw the ideal boundaries that you would like to see from your intuitive point of view. Give reason for your choosing the boundaries that you drawn.



   I will divide the plot as shown in the figure, the boundary can efficient and effective dived the two classes of data.

   iii. Which classifier, in your view, can learn the boundaries closest to the intuitive ones you have suggested? Give reasons for your answer.

Non-linear SVM and KNN are both ok with learning the boundaries but in this model, SVM fits the model too much, so I choose KNN with low k value which will not classify the data too much.

d. What is the role of the regularization parameter and how does it work? (Maximum 200 words)

The regularization parameter can control the results. In this model, the regularization is gamma, it can control the radius of the area that influenced by support vectors. In conclusion, it will control the shape of output boundaries.

Code for 1:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
import pandas as pd
from sklearn.preprocessing import StandardScaler
gammavalue=float(input('gamma=  '))
data =
pd.read_excel('D:/LEARN/GRADUATE2020/IDA/assignment2/HW2Data.xlsx',header=Non
e)
clf = svm.NuSVC(gamma=gammavalue)
x = data[[0,1]]
y = data[3]
#change the gamma as the parameter
x_train = x.sample(int(0.75*len(x)))
x_test = x.loc[x.index.difference(x_train.index)]
y_train = y.loc[x_train.index]
y_test = y.loc[x.index.difference(y_train.index)]

#fit model
clf.fit(x_train,y_train)
#test model
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
y_test_p = clf.predict(x_test)
print('the Gamma is ',gammavalue)
print ('Accuracy of Classifier:%f'%clf.score(x_test,y_test))
print('Precision of Classifier:%f'%precision_score(y_test,y_test_p))
print('Recall of Classifier:%f'%recall_score(y_test,y_test_p))
print('Confusion matrix of Classifier:', confusion_matrix(y_test,y_test_p))


##question b
scaler =StandardScaler()
x_b = x
y_b = y
y_b_p = clf.predict(x_b)
```

```
# plot the decision function for each datapoint on the grid
xx, yy = np.meshgrid(np.linspace(0, 100, 100),
                     np.linspace(0, 100, 100))

# evaluate decision function in a grid
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
gamma = 0.8125

plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()), aspect='auto',
           origin='lower', cmap=plt.cm.PuOr_r)
contours = plt.contour(xx, yy, Z,  linewidths=0.5,linestyles='dashed')
plt.scatter(x_b.values[:,0], x_b.values[:,1],s=10,c=y_b_p,
cmap=plt.cm.Paired,edgecolors='k')

plt.grid(True)
plt.legend(y_b_p)
plt.axis([0, 100, 0, 100])
plt.show()


# print ('Accuracy of Classifier:%f'%clf.score(x,y_b_p))
# print('Precision of Classifier:%f'%precision_score(y,y_b_p))
# print('Recall of Classifier:%f'%recall_score(y,y_b_p))
# print('Confusion matrix of Classifier:'%confusion_matrix(y,y_b_p))


# for all data in grid

plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()), aspect='auto',
           origin='lower', cmap=plt.cm.PuOr_r)
k = list(range(0,100))
#create data for all
every1=[]
every2 = k*100
for i in k:
    every1 = [i] * 100+every1
xxinput = np.column_stack((every1,every2))
y_all = clf.predict(xxinput)
fig = plt.figure()
contours = plt.contour(xx, yy, Z,  linewidths=0.5,linestyles='dashed')
plt.scatter(xxinput[:,0],xxinput[:,1],s=10,c=y_all,
cmap=plt.cm.Paired,edgecolors='k')
plt.axis([0, 100, 0, 100])
plt.show()

plt.grid(True)
plt.legend(y_all)
```

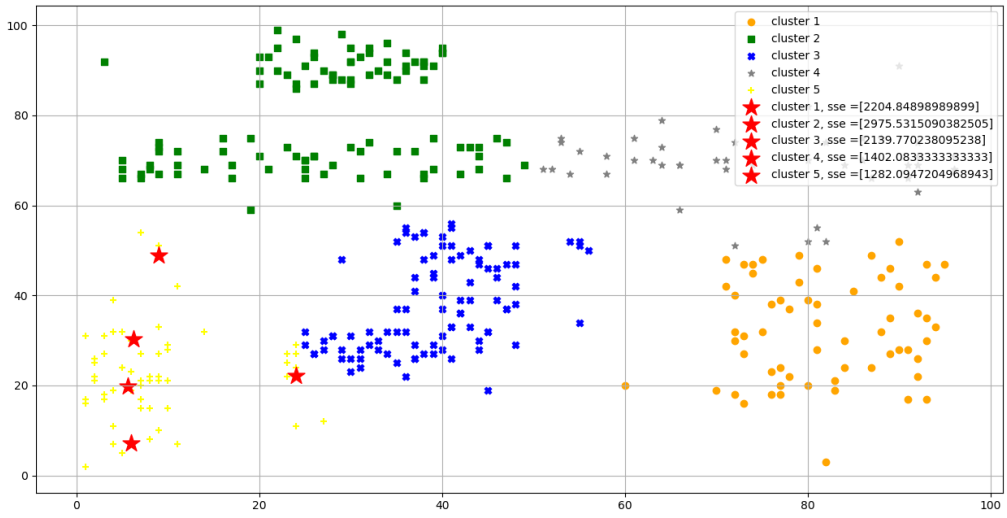
2. Now use the same data set for the following sequence of steps.
    a. For this problem consider only the (x,y) coordinates of the data points and ignore their class labels. Perform k-means clustering of these data points for values of k to be 3, 5, 7, 9, and 11. For each value of k run the clustering 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.

number is 3 , the SSE is : 144037.22795882093
number is 5 , the SSE is : 71332.51539973082
number is 7 , the SSE is : 44857.75217158108
number is 9 , the SSE is : 31114.275369003524
number is 11 , the SSE is : 24572.99778730638
number is 3 , the SSE is : 144037.22795882093
number is 5 , the SSE is : 71332.51539973082
number is 7 , the SSE is : 44857.855199968304
number is 9 , the SSE is : 31117.839274184367
number is 11 , the SSE is : 24528.916082293774
number is 3 , the SSE is : 144037.22795882093
number is 5 , the SSE is : 71332.51539973082
number is 7 , the SSE is : 44857.75217158108
number is 9 , the SSE is : 31144.08172728054
number is 11 , the SSE is : 24581.700929037637
number is 3 , the SSE is : 144037.22795882093
number is 5 , the SSE is : 71332.51539973082
number is 7 , the SSE is : 44857.75217158108
number is 9 , the SSE is : 31114.275369003524
number is 11 , the SSE is : 24486.861840188987
number is 3 , the SSE is : 144037.22795882093
number is 5 , the SSE is : 71332.51539973082
number is 7 , the SSE is : 44857.855199968304
number is 9 , the SSE is : 31117.093550821704
number is 11 , the SSE is : 24503.878637029524
number is 3 , the SSE is : 144037.22795882093
number is 5 , the SSE is : 71332.51539973082
number is 7 , the SSE is : 44857.75217158108
number is 9 , the SSE is : 31114.275369003524
number is 11 , the SSE is : 24572.99778730638
average SSE and std for each:
number is : 3
average is:
144037.22795882096

std is :
2.9103830456733704e-11
max is : 144037.22795882093
min is : 144037.22795882093
number is : 5
average is:
71332.51539973082
std is :
0.0
max is : 71332.51539973082
min is : 71332.51539973082
number is : 7
average is:
44857.78651437682
std is :
0.048568047507289915
max is : 44857.855199968304
min is : 44857.75217158108
number is : 9
average is:
31120.306776549533
std is :
10.729983696166801
max is : 31144.08172728054
min is : 31114.275369003524
number is : 11
average is:
24541.225510527114
std is :
36.87576955849614
max is : 24581.700929037637
min is : 24486.861840188987

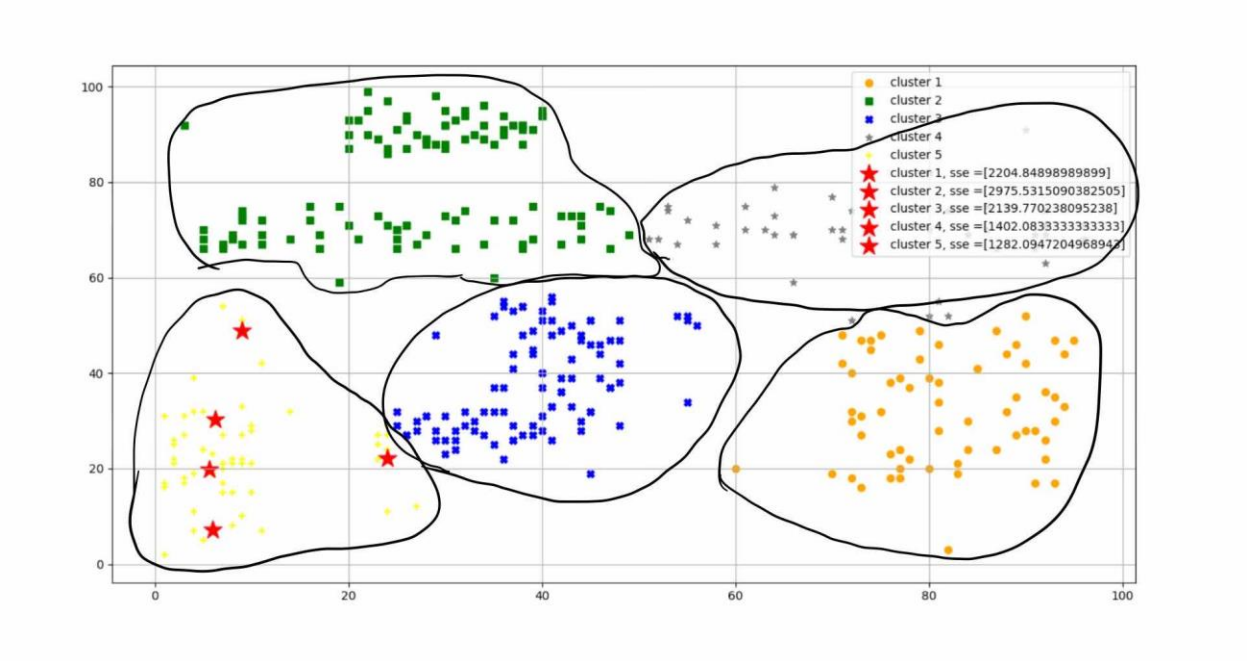
b. Consider the best clustering obtained for k=5. Each data point is assigned a cluster number by your clustering algorithm. Plot all the given data points on the grid after assigning a different color/symbol to members of each cluster. Write the cluster-specific SSE for each individual cluster on the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

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.
e



These five cluster boundaries can directly show the data classes. All these classed data points are gathered around to the predict class center point. For the clusters SSE, they all located in the center of a cluster of data.

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.



I have draw five boundaries on the plot, each of them can effective and efficiently divided the classes. All the boundary can include the clusters points and the boundary will not too specific so the model will not be over fit.

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?

By verify the TP, TN, FP, FN of the clustered datapoints, we can easily find the recall, accuracy, precision and F-sorce. RI = (TP+TN)/(TP+FP+FN+TN)
Code for e:

```
#Rand Index
from sklearn.cluster import KMeans
import numpy as np
from sklearn import svm
import pandas as pd

data =
pd.read_excel('D:/LEARN/GRADUATE2020/IDA/assignment2/HW2Data.xlsx',header=Non
e)
x = data[[0,1]]
y = data[[3]]
pd.to_numeric(x[0])
pd.to_numeric(x[1])
kmodel = KMeans(n_clusters=5)
yp=kmodel.fit_predict(x)
#RI = （TP+TN）/（TP+FP+FN+TN）
bimg=KMeans(n_clusters=5)
bimg.fit_predict(x.iloc[yp==0,:])
```

```
SSE1 = []
SSE1.append(bimg.inertia_)
bimg.fit_predict(x.iloc[yp==1,:])
SSE2 = []
SSE2.append(bimg.inertia_)
bimg.fit_predict(x.iloc[yp==2,:])
SSE3 = []
SSE3.append(bimg.inertia_)
bimg.fit_predict(x.iloc[yp==3,:])
SSE4 = []
SSE4.append(bimg.inertia_)
bimg.fit_predict(x.iloc[yp==4,:])
SSE5 = []
SSE5.append(bimg.inertia_)

def f_score(cluster, labels):
    TP, TN, FP, FN = 0, 0, 0, 0
    n = len(labels)
    # a lookup table
    for i in range(n):
        if i not in cluster:
            continue
        for j in range(i + 1, n):
            if j not in cluster:
                continue
            same_label = (labels[i] == labels[j])
            same_cluster = (cluster[i] == cluster[j])
            if same_cluster:
                if same_label:
                    TP += 1
                else:
                    FP += 1
            elif same_label:
                FN += 1
            else:
                TN += 1
    precision = TP / (TP + FP)
    recall = TP / (TP + FN)
    fscore = 2 * precision * recall / (precision + recall)
    return fscore, precision, recall, TP + FP + FN + TN

labels=[0,1,2,3,4]
fscore, precision, recall, tptnfpfn=f_score(x, yp)
```

```
import numpy as np
from sklearn import svm
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples,silhouette_score
import matplotlib.cm as cm
import random

data =
pd.read_excel('D:/LEARN/GRADUATE2020/IDA/assignment2/HW2Data.xlsx',header=Non
e)
x = data[[0,1]]
y = data[[3]]
kxy=data[[0,1,3]]
pd.to_numeric(x[0])
pd.to_numeric(x[1])

#change the gamma as the parameter

#fit model
def distEclud(vecA, vecB):
    'find the distance'
    return np.sum(np.power(vecA - vecB, 2))


def test_Kmeans_nclusters(data_train):
    'find the SSE value for different k'
    data_train = data_train.values
    nums = [3,5,7,9,11]
    SSE = []
    for num in nums:
        sse = 0
        kmodel = KMeans(n_clusters=num, n_jobs=4)
        kmodel.fit(data_train)
        # centers
        cluster_ceter_list = kmodel.cluster_centers_
        cluster_list = kmodel.labels_.tolist()
        for index in range(len(data)):
            cluster_num = cluster_list[index]
            sse += distEclud(data_train[index, :],
cluster_ceter_list[cluster_num])
        print("number is ", num, ", the SSE is : ", sse)
        SSE.append(sse)
    return nums, SSE,kmodel
totalSSE=[]
for i in range(6):
    nums, SSE ,kmodel= test_Kmeans_nclusters(x)
    totalSSE.append(SSE)
def dispavg(sse,nums):
```

```
    avgsse=[]
    for i in range(len(sse)-1):
        k=0
        g=[]
        for j in range(len(sse)):
            k=k+sse[j][i]
            g.append(sse[j][i])
        stdg=np.std(g)
        maxv=max(g)
        minv=min(g)
        avgsse.append(k/(len(sse)))
        print('number is :',nums[i])
        print('average is:')
        print(k/(len(sse)))
        print('std is :')
        print(stdg)
        print('max is :',maxv)
        print('min is :',minv)
    return avgsse,stdg,g
print('average SSE and std for each:')
avgsse,stdg,g=dispavg(totalSSE,nums)



#b_____________________________________________
#_____________________________________________
kmodel = KMeans(n_clusters=5)
y=kmodel.fit_predict(x)
# centers
cluster_ceter_list = kmodel.cluster_centers_
cluster_list = kmodel.labels_.tolist()
#print
r1 = pd.Series(kmodel.labels_).value_counts()

r2 = pd.DataFrame(kmodel.cluster_centers_)

r = pd.concat([r2, r1], axis = 1) #
r.columns = list(x.columns) + [3] #
#plot figure
from sklearn.manifold import TSNE
tsne = TSNE()
tsne.fit_transform(x) #
tsne = pd.DataFrame(tsne.embedding_, index = x.index) #
import matplotlib.pyplot as plt
bimg=KMeans(n_clusters=1)
bimg.fit_predict(x.iloc[y==0,:])
SSE1 = []
SSE1.append(bimg.inertia_)
bimg.fit_predict(x.iloc[y==1,:])
SSE2 = []
```

```
SSE2.append(bimg.inertia_)
bimg.fit_predict(x.iloc[y==2,:])
SSE3 = []
SSE3.append(bimg.inertia_)
bimg.fit_predict(x.iloc[y==3,:])
SSE4 = []
SSE4.append(bimg.inertia_)
bimg.fit_predict(x.iloc[y==4,:])
SSE5 = []
SSE5.append(bimg.inertia_)
plt.scatter(x.iloc[y==0,0],x.iloc[y==0,1],c="orange",marker="o",label="cluste
r 1")
plt.scatter(x.iloc[y==1,0],x.iloc[y==1,1],c="green",marker="s",label="cluster
2")
plt.scatter(x.iloc[y==2,0],x.iloc[y==2,1],c="blue",marker="X",label="cluster
3")
plt.scatter(x.iloc[y==3,0],x.iloc[y==3,1],c="gray",marker="*",label="cluster
4")
plt.scatter(x.iloc[y==4,0],x.iloc[y==4,1],c="yellow",marker="+",label="cluste
r 5")
plt.scatter(bimg.cluster_centers_[0,0],bimg.cluster_centers_[0,1],s=250,marke
r="*",c="red" ,label="cluster 1, sse ="+str(SSE1))
plt.scatter(bimg.cluster_centers_[1,0],bimg.cluster_centers_[1,1],s=250,marke
r="*",c="red" ,label="cluster 2, sse ="+str(SSE2))
plt.scatter(bimg.cluster_centers_[2,0],bimg.cluster_centers_[2,1],s=250,marke
r="*",c="red" ,label="cluster 3, sse ="+str(SSE3))
plt.scatter(bimg.cluster_centers_[3,0],bimg.cluster_centers_[3,1],s=250,marke
r="*",c="red" ,label="cluster 4, sse ="+str(SSE4))
plt.scatter(bimg.cluster_centers_[4,0],bimg.cluster_centers_[4,1],s=250,marke
r="*",c="red" ,label="cluster 5, sse ="+str(SSE5))
plt.legend()
plt.grid()
plt.show()
```