**Homework 10**

201635825 오성원

**HW10-1**

|  |
| --- |
| Source Code |
| import pandas as pd  import numpy as np  from sklearn import preprocessing  from sklearn.cluster import KMeans  import seaborn as sns  import matplotlib.pyplot as plt  #Read the file  df = pd.read\_excel('C:/python\_file/HW10\_dataset1.xlsx')  #print all column name  print("Attribute name")  for i in range(len(df.columns)):  print("{0} . {1}".format(i+1, df.columns[i]))  #Select the target column  target\_column=int(input("\nSelect the target: "))  print("---------------------------------------------------------------------------------------------------")  target=df.columns[target\_column-1]  #It is divided into input data and target data.  X=np.array(df.drop([target], 1).astype(float))  Y=np.array(df[target])  #Normalize using MinMaxScaler  scaler=preprocessing.MinMaxScaler()  X\_scaled=scaler.fit\_transform(X)  #A function that outputs data belonging to each cluster.  def print\_cluster(data, kmeans, k):  cluster\_row=[[],[],[]]  for i in range(len(kmeans.labels\_)):  if kmeans.labels\_[i]==0:  cluster\_row[0].append(i)  elif kmeans.labels\_[i]==1:  cluster\_row[1].append(i)  elif kmeans.labels\_[i]==2:  cluster\_row[2].append(i)  for i in range(k):  print("Cluster",i+1)  cluster=df.loc[cluster\_row[i], : ]  print(cluster,"\n")    #A function that calculate prediction  def calculate\_prediction(kmeans,X,Y):  correct=0  for i in range(len(X)):  predict\_me=np.array(X[i].astype(float))  predict\_me=predict\_me.reshape(-1,len(predict\_me))  prediction=kmeans.predict(predict\_me)  if prediction[0]==Y[i]:  correct+=1  print("Prediction: ",correct/len(X))  kmeans=KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=600,  n\_clusters=2,n\_init=10, n\_jobs=1, precompute\_distances='auto',  random\_state=None, tol=0.0001,verbose=0)  kmeans.fit(X\_scaled)  print("Result in K=2")  print\_cluster(df, kmeans, 2)  if target\_column==4:  calculate\_prediction(kmeans, X,Y)  if target\_column==5:  calculate\_prediction(kmeans, X,Y)  print("---------------------------------------------------------------------------------------------------")  kmeans=KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=600,  n\_clusters=3,n\_init=10, n\_jobs=1, precompute\_distances='auto',  random\_state=None, tol=0.0001,verbose=0)  kmeans.fit(X\_scaled)  print("Result in K=3")  print\_cluster(df, kmeans, 3) |

|  |
| --- |
| Screen shot |
|  |

|  |
| --- |
| Screen shot |
|  |

|  |
| --- |
| Screen shot |
|  |

**HW10-2 (Change Parameters)**

|  |
| --- |
| Source Code |
| import pandas as pd  import numpy as np  from sklearn import preprocessing  from sklearn.cluster import KMeans  import seaborn as sns  import matplotlib.pyplot as plt  #Read the file  df = pd.read\_excel('C:/python\_file/HW10\_dataset1.xlsx')  #print all column name  print("Attribute name")  for i in range(len(df.columns)):  print("{0} . {1}".format(i+1, df.columns[i]))  #Select the target column  target\_column=int(input("\nSelect the target: "))  value\_max\_iter=int(input("Enter the Max\_iter (Initial:600): "))  print("---------------------------------------------------------------------------------------------------")  target=df.columns[target\_column-1]  #It is divided into input data and target data.  X=np.array(df.drop([target], 1).astype(float))  Y=np.array(df[target])  #Normalize using MinMaxScaler  scaler=preprocessing.MinMaxScaler()  X\_scaled=scaler.fit\_transform(X)  #A function that outputs data belonging to each cluster.  def print\_cluster(data, kmeans, k):  cluster\_row=[[],[],[]]  for i in range(len(kmeans.labels\_)):  if kmeans.labels\_[i]==0:  cluster\_row[0].append(i)  elif kmeans.labels\_[i]==1:  cluster\_row[1].append(i)  elif kmeans.labels\_[i]==2:  cluster\_row[2].append(i)  for i in range(k):  print("Cluster",i+1)  cluster=df.loc[cluster\_row[i], : ]  print(cluster,"\n")    #A function that calculate prediction  def calculate\_prediction(kmeans,X,Y):  correct=0  for i in range(len(X)):  predict\_me=np.array(X[i].astype(float))  predict\_me=predict\_me.reshape(-1,len(predict\_me))  prediction=kmeans.predict(predict\_me)  if prediction[0]==Y[i]:  correct+=1  print("Prediction: ",correct/len(X))  kmeans=KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=300,  n\_clusters=2,n\_init=10, n\_jobs=1, precompute\_distances='auto',  random\_state=None, tol=0.0001,verbose=0)  kmeans.fit(X\_scaled)  print("Result in max\_iter=600")  print\_cluster(df, kmeans, 2)  if target\_column==4:  calculate\_prediction(kmeans, X,Y)  if target\_column==5:  calculate\_prediction(kmeans, X,Y)  print("---------------------------------------------------------------------------------------------------")  kmeans=KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=value\_max\_iter,  n\_clusters=2,n\_init=10, n\_jobs=1, precompute\_distances='auto',  random\_state=None, tol=0.0001,verbose=0)  kmeans.fit(X\_scaled)  print("Result in max\_iter=",value\_max\_iter)  print\_cluster(df, kmeans, 2) |

|  |
| --- |
| Screen shot |
|  |

**HW10-3**

|  |
| --- |
| Source Code |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from scipy.cluster.hierarchy import dendrogram, linkage  #Read the file  df = pd.read\_excel('C:/python\_file/HW10\_dataset2.xlsx')  df\_list = df.values.tolist()  #-----------------------------------------------------  #Single Linkage Hierarchical Clustering  linked = linkage(df, 'single')  dendrogram(linked, orientation='top',  labels=[1,2,3,4,5],  distance\_sort='descending',  show\_leaf\_counts=True)  plt.title("Single Linkage Clustering")  plt.show()  #-----------------------------------------------------  #Complete Linkage Hierarchical Clustering  linked = linkage(df, 'complete')  dendrogram(linked, orientation='top',  labels=[1,2,3,4,5],  distance\_sort='descending',  show\_leaf\_counts=True)  plt.title("Complete Linkage Clustering")  plt.show()  #-----------------------------------------------------  #Average Linkage Hierarchical Clustering  linked = linkage(df, 'average')  dendrogram(linked, orientation='top',  labels=[1,2,3,4,5],  distance\_sort='descending',  show\_leaf\_counts=True)  plt.title("Average Linkage Clustering")  plt.show()  #-----------------------------------------------------  #Centroid Linkage Hierarchical Clustering  linked = linkage(df, 'centroid')  dendrogram(linked, orientation='top',  labels=[1,2,3,4,5],  distance\_sort='descending',  show\_leaf\_counts=True)  plt.title("Centroid Linkage Clustering")  plt.show() |

|  |
| --- |
| Screen shot |
|  |

|  |
| --- |
| Screen shot |
|  |

|  |
| --- |
| Screen shot |
|  |

|  |
| --- |
| Screen shot |
|  |