HW13 Report

인대영 (2022311154)

2022-06-09

1. (a) After saving Fisher's Iris data, we normalize the data using 'MinMaxScaler' function. We choose our bandwidth for mean-shift cluster analysis based on the estimation

$$\lambda = \left(\frac{4\sigma^5}{3n}\right)^{0.2}$$

Taking $n=150$, $\hat{\sigma}=1$, we have $\lambda \simeq 0.39$

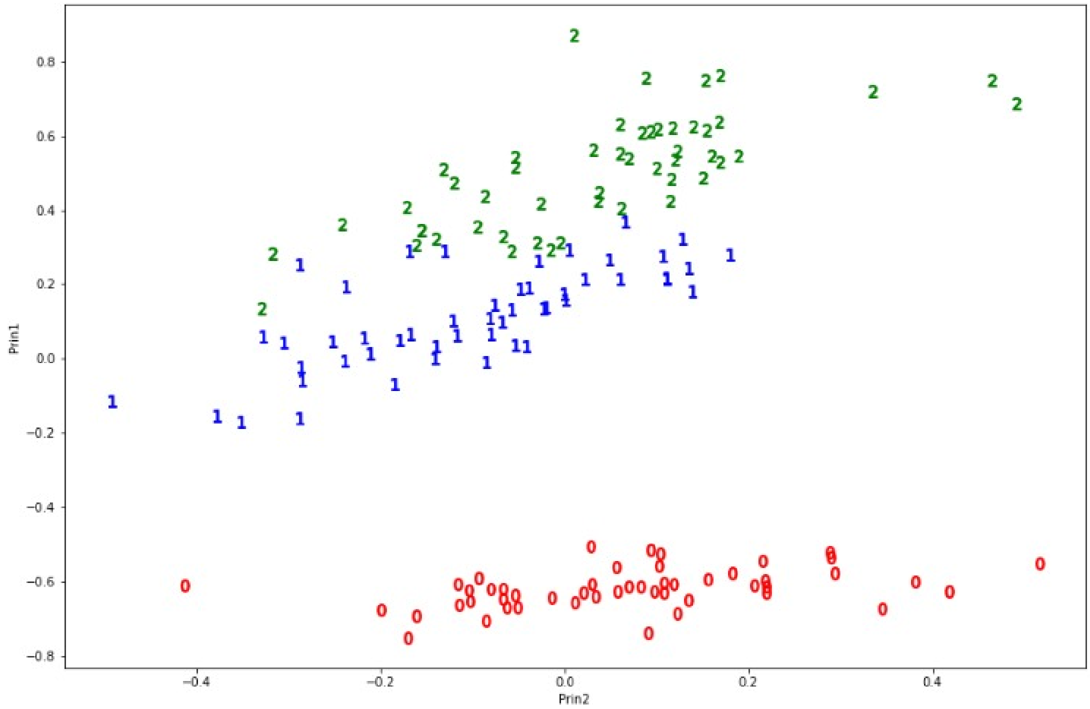
Now, we use 'MeanShift' function from 'sklearn.cluster' module to perform mean-shift cluster analysis. From the fit, we obtain 3 cluster labels. We save the list of labels as 'y_kde'.

```
In [4]: #Bandwidth estimation
        bw = (4/(3*150))**0.2
        bw

Out[4]: 0.3888387116587077

In [5]: from sklearn.cluster import MeanShift
        kde = MeanShift(bandwidth = bw).fit(iris2)

In [6]: #3 clusters
        y_kde = pd.Series(kde.labels_, name = "cluster_kde")
        y_kde

Out[6]: 0      0
        1      0
        2      0
        3      0
        4      0
              ..
        145    2
        146    2
        147    2
        148    2
        149    2
        Name: cluster_kde, Length: 150, dtype: int64
```

(b) We obtain the first two principal components of the data using 'PCA' function from the 'sklearn.decomposition' module. We then plot the first two principle scores with the cluster label obtained in the code in #1(a).

In [7]:
```
#(b) Plot the first two principal component scores with the cluster label
#Perform PCA to plot in 2D
from sklearn.decomposition import PCA
x_pca = iris2.iloc[:,:4]
pca = PCA(n_components = 2)
pc = pca.fit_transform(x_pca)
```

In [8]:
```
#Plot first two principal component scores with cluster label based on mean-shift method
plt.figure(figsize = (15,10))
plt.xlabel('Prin2') ; plt.ylabel('Prin1')
color = {0:'red', 1:'blue', 2:'green'}
for i in range(len(iris2)):
    plt.scatter(pc[i,1], pc[i,0],
                marker = "$ {} $".format(kde.labels_[i]),
                s = 100,
                c = color[kde.labels_[i]])
```



(c) We rewrite the code in HW #12 to repeat hierarchical and K-means analysis and we save the resulting cluster labels as 'y_hier' and 'y_kmean' respectively. We compare the analyses pairwise using confusion matrices to inspect the similarity between each set of cluster labels.

In [11]:
```
#Compare Hiearchical and K-means
from sklearn.metrics import confusion_matrix
C1 = pd.DataFrame(confusion_matrix(y_hier, y_kmean),
                  index = np.sort(y_hier.unique()),
                  columns = np.sort(y_kmean.unique()))
C1['Total'] = C1.sum(axis = 1) #row sum
C1.loc['Total',:] = C1.sum(axis = 0) # column sum
C1.index.names = ['Hierarchical'] ; C1.columns.names = ['K-means']
C1.astype(int)
```

Out[11]:

K-means / Hierarchical	0	1	2	Total
0	50	0	0	50
1	0	61	9	70
2	0	0	30	30
Total	50	61	39	150

In [13]:
```
#Compare Hierarchical and Mean-shift
C2 = pd.DataFrame(confusion_matrix(y_hier, y_kde),
                  index = np.sort(y_hier.unique()),
                  columns = np.sort(y_kde.unique()))
C2['Total'] = C2.sum(axis = 1) #row sum
C2.loc['Total',:] = C2.sum(axis = 0) # column sum
C2.index.names = ['Hierarchical'] ; C2.columns.names = ['Mean-shift']
C2.astype(int)
```

Out[13]:

Mean-shift / Hierarchical	0	1	2	Total
0	50	0	0	50
1	0	50	20	70
2	0	0	30	30
Total	50	50	50	150

In [15]:
```
#Compare K-means and Mean-shift
C3 = pd.DataFrame(confusion_matrix(y_kmean, y_kde),
                  index = np.sort(y_kmean.unique()),
                  columns = np.sort(y_kde.unique()))
C3['Total'] = C3.sum(axis = 1) #row sum
C3.loc['Total',:] = C3.sum(axis = 0) # column sum
C3.index.names = ['K-means'] ; C2.columns.names = ['Mean-shift']
C3.astype(int)
```

Out[15]:

K-means	0	1	2	Total
0	50	0	0	50
1	0	47	14	61
2	0	3	36	39
Total	50	50	50	150

The accuracy levels of each confusion matrix are 94%, 86.67%, and 88.67%. Although all of them are very high, we may say that the hierarchical and K-means clusters are the most similar. Also, the cluster obtained from mean-shift method is slightly different from the other two clusters. We also obtain a table to compare all three labels simultaneously.

In [17]:
```
#Compare cluster labels
pd.DataFrame({'cluster_hier': y_hier, 'cluster_kmean': y_kmean, 'cluster_kde': y_kde})
```

Out[17]:

	cluster_hier	cluster_kmean	cluster_kde
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
...	...	...	...
145	2	2	2
146	1	1	2
147	1	2	2
148	2	2	2
149	1	1	2

150 rows × 3 columns