Project_4_4

July 28, 2020

1 Project 4.4. Spectf Heart dataset clustering

[1]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.metrics.cluster import normalized_mutual_info_score
from sklearn.metrics.cluster import adjusted_rand_score
from sklearn.metrics.cluster import contingency_matrix
from sklearn.metrics.cluster import adjusted_rand_score
from sklearn.cluster import KMeans
```

1.1 Read datase

[2]:
```
dataset_path = './data/SPECT.train'
dataset_df = pd.read_csv(dataset_path)

y = dataset_df['class']
X = dataset_df.drop(['class'], axis=1)

X
```

[2]:

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	…	f13	f14	f15	f16	f17	\
0	1	0	0	0	1	0	0	0	1	1	…	0	1	1	0	0	
1	1	0	0	1	1	0	0	0	1	1	…	0	1	1	0	0	
2	1	1	0	1	0	1	0	0	1	0	…	0	1	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	…	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	1	0	…	0	1	0	1	1	
..	..	..	..	..	..	..	..	..	..	..	…	…	…	…	…	…	
75	0	1	0	0	0	1	0	0	0	0	…	0	0	0	0	0	
76	0	1	0	0	0	1	1	0	0	1	…	0	0	1	0	0	
77	0	1	0	0	0	1	0	0	0	0	…	0	0	0	0	0	
78	0	0	0	1	1	0	0	1	0	0	…	1	1	1	0	0	
79	0	1	0	0	0	1	0	0	0	0	…	0	0	0	0	0	

	f18	f19	f20	f21	f22
0	0	0	0	0	0

```
1     0    0    0    0    0
2     0    0    0    0    0
3     0    0    0    1    1
4     0    0    0    0    0
..   …    …    …    …    …
75    0    0    0    0    0
76    0    0    1    1    0
77    0    0    1    0    0
78    0    0    0    0    1
79    0    0    0    0    0

[80 rows x 22 columns]
```

1.2 Fit the model

[3]:
```
cluster_number = 2
k_means_model = KMeans(n_clusters=cluster_number)
k_means_model.fit(X)
k_means_labels = k_means_model.labels_

print('\nOriginal classes: ',y.values)
print('\nk-means clustering: ', k_means_labels)
```

```
Original classes:  [0 1 0 1 0 1 1 1 1 0 1 0 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 0 1 1
0 0 1 1 0 1 1
 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 1 0]

k-means clustering:  [1 1 1 0 0 1 1 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 0
1 0 0 1 1 1 1 1
 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 1 0]
```

1.3 Validate the trained model

[4]:
```
# This cells code have been brout from stack overflow website by the address:
# https://stackoverflow.com/questions/49586742/
 ↪rand-index-function-clustering-performance-evaluation

from scipy.special import comb
def rand_index_score(clusters, classes):
    tp_plus_fp = comb(np.bincount(clusters), 2).sum()
    tp_plus_fn = comb(np.bincount(classes), 2).sum()
    A = np.c_[(clusters, classes)]
```

```
    tp = sum(comb(np.bincount(A[A[:, 0] == i, 1]), 2).sum()
             for i in set(clusters))
    fp = tp_plus_fp - tp
    fn = tp_plus_fn - tp
    tn = comb(len(A), 2) - tp - fp - fn
    return (tp + tn) / (tp + fp + fn + tn)
```

[5]:
```
k_means_RI = adjusted_rand_score(k_means_labels, y)
print('K-means RI result:', k_means_RI)

# NMI metrics
k_means_NMI = normalized_mutual_info_score(k_means_labels, y)
print('\nK-means NMI result:', k_means_NMI)

# Purity metrics
k_means_CM = contingency_matrix(k_means_labels, y)
k_means_purity = np.sum(np.amax(k_means_CM, axis=0)) / np.sum(k_means_CM)
print('\nK-means purity result:', k_means_purity)

# AIR metrics
k_means_ARI = rand_index_score(k_means_labels, y)
print('\nK-means ARI result:', k_means_ARI)
```

```
K-means RI result: 0.18225830526513898

K-means NMI result: 0.10777535947061594

K-means purity result: 0.725

K-means ARI result: 0.5962025316455696
```