19-Scikit Learn-Clustering Performance Evaluation

There are many functions with the help of which can evaluate the performance of clustering algorithms. Following are some important and mostly used functions given by Scikit Learn for evaluating clustering performance:

19-1-Adjust Rand Index

Rand Index is a function that computes a similarity measures between two clustering. For this computation rand index cosiders all pairs of samples and counting pairs that are assigned in the similar or different clusters in the predicted and true clustering.

$$Adjusted\ RI = (RI - Expected\_RI)\ /\ (max\ (RI) - Expected\_RI)$$

It has two parameters namely **labels_true**, which is ground truth class labels, and **labels_pred**, which are clusters label to evaluate.

```
from sklearn.metrics.cluster import adjusted_rand_score

labels_true=[0,0,1,1,2]
labels_pred=[0,1,1,1,0]
print(adjusted_rand_score(labels_true,labels_pred))
```

output:

0.090

19-2- Mutual Information Based Score

Mutual Information is a function that compute the agreement of two assignments. It ignores the permutations. There are following versions available:

- **Normalized Mutual Information(NMI)**

```
from sklearn.metrics.cluster import normalized_mutual_info_score

labels_true=[0,0,1,1,2]
labels_pred=[0,1,1,1,0]
print(normalized_mutual_info_score(labels_true,labels_pred))
```

output:

0.45

- **Adjusted Mutual Information(AMI)**

```
from sklearn.metrics.cluster import adjusted_mutual_info_score

labels_true=[0,0,1,1,2]
labels_pred=[0,1,1,1,0]
print(adjusted_mutual_info_score(labels_true,labels_pred))
```

output:

0.10

19-3-Fowlkes-Mallows Score

This function measures the similarity of two clustering of a set of points. It may be defined as the geometric mean of the pairwise precision and recall.

$$FMS = \frac{TP}{\sqrt{(TP + FP)\,(TP + FN)}}$$

TP=True Positive:number of pair of points belonging tot he same clusters in true as well as predicted labels both.

FP=False Positive: number of pair of points belonging to the same clusters in true labels but not in the predicted labels.

FN=False Negative: number of pair of points belonging to the same clusters in the predicted labels but not in the true labels.

```
from sklearn.metrics.cluster import fowlkes_mallows_score
labels_true=[0,0,1,1,2]
labels_pred=[0,1,1,1,0]
print(fowlkes_mallows_score(labels_true,labels_pred))
```

output:

0.35

19-4-Silhouette Coefficient

This function will compute the mean Silhouette Coefficient of all samples using the mean intra-cluster distance and the mean nearest-cluster distance for each sample.

S=(b-a)/max(a,b)

a is intra-cluster distance, b is nearest-cluster distance.

```
 from sklearn.metrics import silhouette_score
from sklearn.metrics import pairwise_distances
from sklearn import datasets
import numpy as np
from sklearn.cluster import KMeans
data=datasets.load_iris()
X=data.data
y=data.target
kmeans_model=KMeans(n_clusters=3,random_state=1).fit(X)
labels=kmeans_model.labels_
print(silhouette_score(X,labels,metric='euclidean'))
```

output:

0.55

20- Dimensionality Reduction using PCA

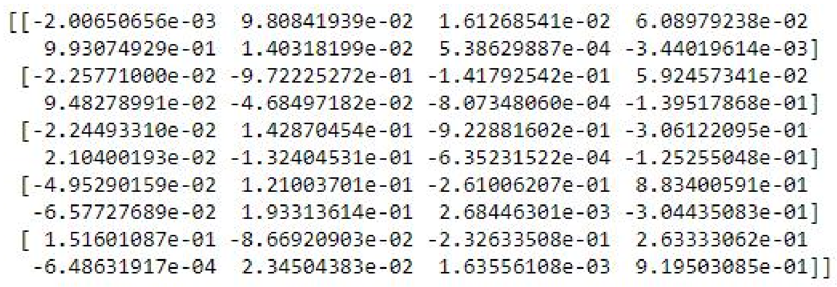
Dimensionality reduction is one of the popular algorithms for dimensionality reduction, an unsupervised machine learning method is used to reduce the number of feature variable for each data sample selecting set of principal features.

PCA is used for linear dimensionality reduction using **Singular Value Decomposition** of the data to project it to a lower dimensional space.

Scikit learn provides **sklearn.decomposition.PCA** module that is implemented as a transformer object which learns n components in its fit() method. in this example PCA used to find best 5 principal components.

```
 import pandas as pd
from pandas import read_csv
from sklearn.decomposition import PCA
data= pd.read_csv('pima-indians-diabetes.csv')
data=pd.DataFrame(data)
data.columns=['preg','plas','pres','skin','test','mass','pedi','age','class']
array=dataframe.values
X=array[:,0:8]
Y=array[:,8]
pca=PCA(n_components=5)
fit=pca.fit(X)
print(fit.components_)
```

output:



```
[[-2.00650656e-03  9.80841939e-02  1.61268541e-02  6.08979238e-02
   9.93074929e-01  1.40318199e-02  5.38629887e-04 -3.44019614e-03]
 [-2.25771000e-02 -9.72225272e-01 -1.41792542e-01  5.92457341e-02
   9.48278991e-02 -4.68497182e-02 -8.07348060e-04 -1.39517868e-01]
 [-2.24493310e-02  1.42870454e-01 -9.22881602e-01 -3.06122095e-01
   2.10400193e-02 -1.32404531e-01 -6.35231522e-04 -1.25255048e-01]
 [-4.95290159e-02  1.21003701e-01 -2.61006207e-01  8.83400591e-01
  -6.57727689e-02  1.93313614e-01  2.68446301e-03 -3.04435083e-01]
 [ 1.51601087e-01 -8.66920903e-02 -2.32633508e-01  2.63333062e-01
  -6.48631917e-04  2.34504383e-02  1.63556108e-03  9.19503085e-01]]
```

20-1- Incremental PCA

Incremental Principal Component Analysis (IPCA) is used to address biggest limitation of PCA and that s PCA only supports batch processing,means all the input data to be processed should fit in the memory. The Scikit-learn provides **sklearn.decomposition.IPCA** module that makes it possible to implement Out-of-Core PCA either by using it partial_fit method on sequentially fetched chunks of data or by enabling use of np.memmap, a memory mapped file, without loading the entire file into memory.

20-2-Kernel PCA

Kernel Prinipal Component Analysis, an extension of PCA, achives on-linear dimensionality reduction using kernels.It supports both ansform and inverse_transform. Scikit-learn provides **sklearn.decomposition.kernelIPCA**module.