Performance Metrics in Clustering Models

The problem of predictive modeling is to create models that have good performance making predictions on new unseen data. Therefore it is critically important to use robust techniques to train and evaluate your models on your available training data. The more reliable the estimate of the performance on your model, the further you can push the performance and be confident it will translate to the operational use of your model.

We will discuss some of the performance metric for clustering models:

1) Silhouette score:

2) contingency_matrix

3) homogeneity_score

Silhouette score:

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b).

To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n_samples - 1.

The **best** value is **1** and the **worst** value is **-1**.Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

Syntax:

from sklearn.metrics import silhouette_score

X= Array of pairwise distances between samples, or a feature array.

labels = Predicted labels for each sample.

score = silhouette_score(X, labels, metric='euclidean', sample_size=None, random_state=None)

Parameters:

- **metric :** string, or callable - The metric to use when calculating distance between instances in a feature array. If metric is a string, it must be one of the options allowed by metrics.pairwise.pairwise_distances. If X is the distance array itself, use metric="precomputed".

- **sample_size :** int or None - The size of the sample to use when computing the Silhouette Coefficient on a random subset of the data. If sample_size is None, no sampling is used.

Arguments:

- **silhouette :** float - Mean Silhouette Coefficient for all samples.

Contingency_matrix:

Contingency table (also known as a cross tabulation or crosstab) is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables. They are heavily used in survey research, business intelligence, engineering and scientific research. They

provide a basic picture of the interrelation between two variables and can help find interactions between them.

Syntax:

from sklearn.metrics.cluster import contingency_matrix

labels_true = Ground truth class labels to be used as a reference

labels_pred = Cluster labels to evaluate

matrix = contingency_matrix(labels_true, labels_pred, eps=None, sparse=False)

Parameters:

- **eps :** None or float, optional. - If a float, that value is added to all values in the contingency matrix. This helps to stop NaN propagation. If None, nothing is adjusted.

- **sparse :** boolean, optional. - If True, return a sparse CSR continency matrix. If eps is not None, and sparse is True, will throw ValueError.

Arguments:

- **contingency :** {array-like, sparse}, shape=[n_classes_true, n_classes_pred] - Matrix such that is the number of samples in true class and in predicted class .If eps is None, the dtype of this array will be integer. If eps is given, the dtype will be float. Will be a scipy.sparse.csr_matrix if sparse=True.

Homogeneity_score:

Homogeneity metric of a cluster labeling given a ground truth. A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class. This metric is independent of the absolute values of the labels: a permutation of the class or cluster label values won't change the score value in any way.

Syntax:

from sklearn.metrics.cluster import homogeneity_score

labels_true = Ground truth class labels to be used as a reference

labels_pred = Cluster labels to evaluate

score = homogeneity_score(labels_true, labels_pred, eps=None, sparse=False)

Attributes:

- **homogeneity** : float - score between 0.0 and 1.0. 1.0 stands for perfectly homogeneous labeling

Example:

from sklearn.metrics.cluster import homogeneity_score

homogeneity_score([0, 0, 1, 1], [1, 1, 0, 0])

1.0