***class*sklearn.cluster.KMeans(*n\_clusters****=8***,***\****, *init****='k-means++'***, *n\_init****='auto'***, *max\_iter****=300***, *tol****=0.0001***, *verbose****=0***, *random\_state****=None***, *copy\_x****=True***, *algorithm****='lloyd'***)**

n\_clusters : int, default=8

        The number of clusters to form as well as the number of

        centroids to generate.

    init : {'k-means++', 'random'}, callable or array-like of shape \

            (n\_clusters, n\_features), default='k-means++'

        Method for initialization:

'k-means++' : selects initial cluster centroids using sampling based on an empirical probability distribution of the points' contribution to the overall inertia. This technique speeds up convergence, and is theoretically proven to be :math:`\\mathcal{O}(\\log k)`-optimal.

        See the description of `n\_init` for more details.

        'random': choose `n\_clusters` observations (rows) at random from data for the initial centroids.

If an array is passed, it should be of shape (n\_clusters, n\_features) and gives the initial centers.

If a callable is passed, it should take arguments X, n\_clusters and a random state and return an initialization.

    n\_init : int, default=10

*Number of time the k-means algorithm will* be run with different

        centroid seeds. The final results will be the best output of

        n\_init consecutive runs in terms of inertia.

    max\_iter : int, default=300

        Maximum number of iterations of the k-means algorithm for a

        single run.

**tol** : float, default=1e-4

        Relative tolerance with regards to Frobenius norm of the difference in the cluster centers of two consecutive iterations to declare convergence.

    random\_state : int, RandomState instance or None, default=None

.

    copy\_x : bool, default=True

        When pre-computing distances it is more numerically accurate to centerthe data first.

If copy\_x is True (default), then the original data is not modified. If False, the original data is modified, and put back

        before the function returns, but small numerical differences may be introduced by subtracting and then adding the data mean. Note that if the original data is not C-contiguous, a copy will be made even if copy\_x is False. If the original data is sparse, but not in CSR format,a copy will be made even if copy\_x is False.

    algorithm : {"lloyd", "elkan", "auto", "full"}, default="lloyd"

        K-means algorithm to use. The classical EM-style algorithm is `"lloyd"`.

        The `"elkan"` variation can be more efficient on some datasets with well-defined clusters, by using the triangle inequality. However it's more memory intensive due to the allocation of an extra array of shape

        `(n\_samples, n\_clusters)`.

**Attributes:**

cluster\_centers\_ : ndarray of shape (n\_clusters, n\_features)

labels\_ : ndarray of shape (n\_samples,)

inertia\_ : float (Sum of squared distances of samples to their closest cluster center, weighted by the sample weights if provided.)

n\_iter\_ : int

n\_features\_in\_: int

feature\_names\_in\_: ndarray of shape (`n\_features\_in\_`,)

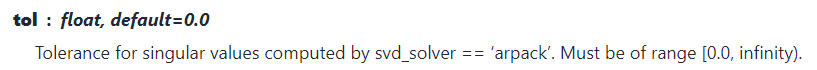
***class*sklearn.decomposition.PCA(*n\_components****=None***,***\****, *copy****=True***, *whiten****=False***, *svd\_solver****='auto'***, *tol****=0.0***, *iterated\_power****='auto'***, *n\_oversamples****=10***, *power\_iteration\_normalizer****='auto'***, *random\_state****=None***)**

图形用户界面, 文本, 应用程序, 电子邮件

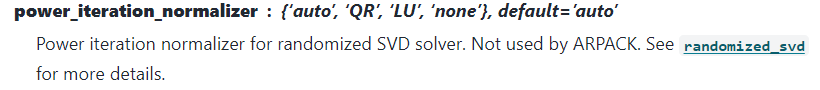
描述已自动生成图形用户界面, 文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面

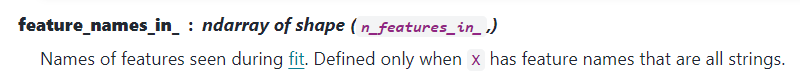
中度可信度描述已自动生成图形用户界面, 文本

描述已自动生成

手机屏幕截图

描述已自动生成图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**t-Distributed Stochastic Neighbor Embedding (t-SNE)** is a non-linear dimensionality reduction technique commonly used for visualizing high-dimensional datasets. t-SNE minimizes the divergence between two distributions: one that measures pairwise similarities of the data points in the high-dimensional space and another that measures similarities in the low-dimensional embedding. It is particularly effective for visualizing clusters or groups in complex datasets.

t-SNE [1] is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

It is highly recommended to use another dimensionality reduction method (e.g. PCA for dense data or TruncatedSVD for sparse data) to reduce the number of dimensions to a reasonable amount (e.g. 50) if the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples. For more tips see Laurens van der Maaten’s FAQ [2].

文本, 信件

描述已自动生成

图形用户界面, 文本, 应用程序, Word

描述已自动生成徽标

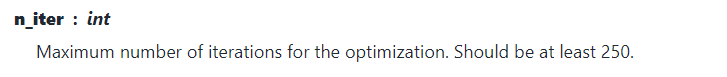
低可信度描述已自动生成图形用户界面, 文本

描述已自动生成图片包含 表格

描述已自动生成文本

描述已自动生成图形用户界面, 文本

描述已自动生成文本

描述已自动生成图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

LDA (Latent Dirichlet Allocation) is a statistical model commonly used for topic modeling in natural language processing. It is a generative probabilistic model that aims to explain observations by representing them as a mixture of multiple topics, where each topic is a distribution over words. Essentially, LDA is used to identify the topics that are present within a collection of documents.

LDA assumes that each document is generated from a random mix of latent topics, and each topic is represented as a distribution over words. The model uses a Dirichlet prior to ensure that each document can be represented by a few prominent topics, promoting sparsity and interpretability.

To provide an example using scikit-learn (sklearn) version 1.5.2, I'll demonstrate how to apply LDA to a simple dataset and plot the loss (log-likelihood) curve to illustrate the convergence process. Here is an example using scikit-learn's LatentDirichletAllocation:

***class*sklearn.decomposition.LatentDirichletAllocation(*n\_components****=10***,***\****, *doc\_topic\_prior****=None***, *topic\_word\_prior****=None***, *learning\_method****='batch'***, *learning\_decay****=0.7***, *learning\_offset****=10.0***, *max\_iter****=10***, *batch\_size****=128***, *evaluate\_every****=1***, *total\_samples****=1000000.0***, *perp\_tol****=0.1***, *mean\_change\_tol****=0.001***, *max\_doc\_update\_iter****=100***, *n\_jobs****=None***, *verbose****=0***, *random\_state****=None***)**

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成图形用户界面, 文本

描述已自动生成图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

文本, 信件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 文本

描述已自动生成图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成