**# Other feature extraction algorithms:**

# https://scikit-learn.org/stable/modules/manifold.html#multi-dimensional-scaling-mds

# https://pydiffmap.readthedocs.io/en/master/reference/diffusion\_map.html

**1. scikit‑learn Manifold‑Learning Methods**

These methods live under sklearn.manifold or sklearn.decomposition and require only pip install scikit-learn.

1. **Kernel PCA** (sklearn.decomposition.KernelPCA)
   * Nonlinear extension of PCA via kernels (e.g., RBF, polynomial).
   * **Implementation**:

from sklearn.decomposition import KernelPCA

kpca = KernelPCA(n\_components=2, kernel='rbf', gamma=0.1)

X\_kpca = kpca.fit\_transform(X)

1. [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html?utm_source=chatgpt.com)
2. **t‑SNE** (sklearn.manifold.TSNE)
   * Converts pairwise similarities into low‑dimensional embeddings by minimizing a KL divergence.
   * **Implementation**:

from sklearn.manifold import TSNE

tsne = TSNE(n\_components=2, perplexity=30)

X\_tsne = tsne.fit\_transform(X)

1. [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html?utm_source=chatgpt.com)
2. **Isomap** (sklearn.manifold.Isomap)
   * Preserves manifold’s geodesic distances via k‑NN graphs.
   * **Implementation**:

from sklearn.manifold import Isomap

iso = Isomap(n\_components=2, n\_neighbors=5)

X\_iso = iso.fit\_transform(X)

1. [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.Isomap.html?utm_source=chatgpt.com)
2. **Locally Linear Embedding (LLE)** (sklearn.manifold.LocallyLinearEmbedding)
   * Preserves local linear structures. Supports variants via method parameter:
     + method='standard' (classic LLE)
     + method='modified', method='hessian' (Hessian LLE), method='ltsa' (Local Tangent Space Alignment)
   * **Implementation**:

from sklearn.manifold import LocallyLinearEmbedding

lle = LocallyLinearEmbedding(n\_components=2, method='standard')

X\_lle = lle.fit\_transform(X)

1. [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.LocallyLinearEmbedding.html?utm_source=chatgpt.com)
2. **Multidimensional Scaling (MDS)** (sklearn.manifold.MDS)
   * Finds embeddings preserving pairwise distances (metric or non‑metric).
   * **Implementation**:

from sklearn.manifold import MDS

mds = MDS(n\_components=2, metric=False)

X\_mds = mds.fit\_transform(dist\_matrix)

1. [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html?utm_source=chatgpt.com)
2. **Spectral Embedding** (Laplacian Eigenmaps, sklearn.manifold.SpectralEmbedding)
   * Constructs graph‑Laplacian and uses its eigenvectors for embedding.
   * **Implementation**:

from sklearn.manifold import SpectralEmbedding

se = SpectralEmbedding(n\_components=2, affinity='nearest\_neighbors')

X\_se = se.fit\_transform(X)

1. [Scikit-learn](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.SpectralEmbedding.html?utm_source=chatgpt.com)

**2. Specialized Packages**

These techniques live in separate libraries—installable via pip install <package>—and excel at capturing complex nonlinear structures.

1. **UMAP** (umap-learn)
   * Balances local and global data structure preservation with graph‑based fuzzy simplicial sets.
   * **Implementation**:

import umap

reducer = umap.UMAP(n\_components=2, n\_neighbors=15, min\_dist=0.1)

X\_umap = reducer.fit\_transform(X)

1. [umap-learn.readthedocs.io](https://umap-learn.readthedocs.io/?utm_source=chatgpt.com)
2. **Diffusion Maps** (pydiffmap)
   * Builds a diffusion operator over the data to reveal intrinsic geometry.
   * **Implementation**:

from pydiffmap import diffusion\_map as dm

mydmap = dm.DiffusionMap.from\_sklearn(n\_evecs=2, alpha=0.5)

X\_dmap = mydmap.fit\_transform(X)

1. [pydiffmap.readthedocs.io](https://pydiffmap.readthedocs.io/en/master/reference/diffusion_map.html?utm_source=chatgpt.com)
2. **Self‑Organizing Maps (SOMs)** (MiniSom)
   * Neural‑network grid that maps high‑dimensional inputs to a 2D/3D lattice.
   * **Implementation**:

from minisom import MiniSom

som = MiniSom(x=10, y=10, input\_len=X.shape[1], sigma=1.0, learning\_rate=0.5)

som.train\_random(X, num\_iteration=1000)

weights = som.get\_weights() # 10×10×dim map

1. [GitHub](https://github.com/JustGlowing/minisom?utm_source=chatgpt.com)
2. **Contractive Autoencoder** (PyTorch)
   * Enforces robustness by penalizing the Jacobian of the encoder mapping.
   * **Implementation**:

# See https://github.com/avijit9/Contractive\_Autoencoder\_in\_Pytorch

# train as usual, with contractive loss term

1. [GitHub](https://github.com/avijit9/Contractive_Autoencoder_in_Pytorch?utm_source=chatgpt.com)
2. **Variational Autoencoder (VAE)** (Keras)
   * Probabilistic autoencoder learning latent distributions under KL‑regularization.
   * **Implementation**:

# Keras example: https://keras.io/examples/generative/vae/

1. [Keras](https://keras.io/examples/generative/vae/?utm_source=chatgpt.com)
2. **PHATE** (phate)
   * Heat‑diffusion based embedding preserving local and global structure, popular in bioinformatics.
   * **Implementation**:

import phate

phate\_op = phate.PHATE(n\_components=2)

X\_phate = phate\_op.fit\_transform(X)

1. [PyPI](https://pypi.org/project/phate/?utm_source=chatgpt.com)
2. **TriMap** (trimap)
   * Uses triplet constraints (“i closer to j than k”) to optimize embeddings.
   * **Implementation**:

import trimap

tri = trimap.TRIMAP(n\_dims=2)

X\_trimap = tri.fit\_transform(X)

1. [PyPI](https://pypi.org/project/trimap/?utm_source=chatgpt.com)
2. **Kepler Mapper** (kmapper)
   * Topological‑data‑analysis Mapper algorithm producing simplicial complexes.
   * **Implementation**:

import kmapper as km

mapper = km.KeplerMapper(verbose=1)

proj = mapper.fit\_transform(X, projection=[0,1])

graph = mapper.map(proj, X, cover=km.Cover(n\_cubes=10), clusterer=km.cluster.DBSCAN())

1. [kepler-mapper.scikit-tda.org](https://kepler-mapper.scikit-tda.org/?utm_source=chatgpt.com)

**3. Gaussian‑Process‑Based Latent Variable Models**

1. **GPLVM** (GPy)
   * Learns latent embeddings via a Gaussian‑process generative model.
   * **Implementation**:

import GPy

model = GPy.models.GPLVM(Y, input\_dim=2, kernel=GPy.kern.RBF(2))

model.optimize()

X\_gplvm = model.X.values

1. [gpy.readthedocs.io](https://gpy.readthedocs.io/en/devel/_modules/GPy/models/gplvm.html?utm_source=chatgpt.com)
2. **Bayesian GPLVM** (GPflow or GPyTorch)
   * Bayesian treatment of latent inputs via variational inference.
   * **Implementation**:

# GPflow example:

# https://gpflow.github.io/GPflow/2.4.0/notebooks/basics/GPLVM.html

1. [gpflow.github.io](https://gpflow.github.io/GPflow/2.4.0/notebooks/basics/GPLVM.html?utm_source=chatgpt.com)

This list spans 17 proven methods—each unsupervised and nonlinear—complete with ready‑to‑use implementations. If you need code snippets, advice on hyperparameter tuning, or help selecting the right approach for your dataset, just let me know!