kenchi Documentation

Release 0.10.0a1

Author

Sep 30, 2018

Contents:

CHAPTER 1

kenchi package

1.1 Subpackages

1.1.1 kenchi.datasets package

1.1.1.1 Submodules

kenchi.datasets.base.**load_pendigits**(*random_state=None*, *return_X_y=False*, *subset='kriegel11'*)

Load and return the pendigits dataset.

Kriegel's structure (subset='kriegel11') :

anomalous class	class 4
n_samples	9868
n_outliers	20
n_features	16
contamination	0.002

Goldstein's global structure (subset='goldstein12-global') :

anomalous class	classes 0, 1, 2, 3, 4, 5, 6, 7, 9
n_samples	809
n_outliers	90
n_features	16
contamination	0.111

Goldstein's local structure (subset='goldstein12-local') :

anomalous class	class 4
n_samples	6724
n_outliers	10
n_features	16
contamination	0.001

Parameters

- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **return_X_y** (*bool, default False*) – If True, return `(data, target)` instead of a Bunch object.
- **subset** (*str, default 'kriegel11'*) – Specify the structure. Valid options are ['goldstein12-global'|'goldstein12-local'|'kriegel11'].

Returns **data** – Dictionary-like object.

Return type Bunch

References

Examples

```
>>> from kenchi.datasets import load_pendigits
>>> pendigits = load_pendigits(subset='kriegel11')
>>> pendigits.data.shape
(9868, 16)
>>> pendigits = load_pendigits(subset='goldstein12-global')
>>> pendigits.data.shape
(809, 16)
>>> pendigits = load_pendigits(subset='goldstein12-local')
>>> pendigits.data.shape
(6724, 16)
```

kenchi.datasets.base.**load_pima**(*return_X_y=False*)

Load and return the Pima Indians diabetes dataset.

anomalous class	class 1
n_samples	768
n_outliers	268
n_features	8
contamination	0.349

Parameters **return_X_y** (*bool, default False*) – If True, return `(data, target)` instead of a Bunch object.

Returns **data** – Dictionary-like object.

Return type Bunch

References

Examples

```
>>> from kenchi.datasets import load_pima
>>> pima = load_pima()
>>> pima.data.shape
(768, 8)
```

kenchi.datasets.base.**load_wdbc**(*random_state=None*, *return_X_y=False*, *subset='kriegel11'*)

Load and return the breast cancer Wisconsin dataset.

Goldstein's structure (subset='goldstein12') :

anomalous class	malignant
n_samples	367
n_outliers	10
n_features	30
contamination	0.027

Kriegel's structure (subset='kriegel11') :

anomalous class	malignant
n_samples	367
n_outliers	10
n_features	30
contamination	0.027

Sugiyama's structure (subset='sugiyama13') :

anomalous class	malignant
n_samples	569
n_outliers	212
n_features	30
contamination	0.373

Parameters

- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **return_X_y** (*bool, default False*) – If True, return `(data, target)` instead of a Bunch object.
- **subset** (*str, default 'kriegel11'*) – Specify the structure. Valid options are ['goldstein12'|'kriegel11'|'sugiyama13'].

Returns **data** – Dictionary-like object.

Return type Bunch

References

Examples

```
>>> from kenchi.datasets import load_wdbc
>>> wdbc = load_wdbc(subset='goldstein12')
>>> wdbc.data.shape
(367, 30)
>>> wdbc = load_wdbc(subset='kriegel11')
>>> wdbc.data.shape
(367, 30)
>>> wdbc = load_wdbc(subset='sugiyama13')
>>> wdbc.data.shape
(569, 30)
```

kenchi.datasets.base.**load_wilt**(*return_X_y=False*)

Load and return the wilt dataset.

anomalous class	class 'w'
n_samples	4839
n_outliers	261
n_features	5
contamination	0.053

Parameters **return_X_y** (*bool, default False*) – If True, return `(data, target)` instead of a Bunch object.

Returns **data** – Dictionary-like object.

Return type Bunch

References

Examples

```
>>> from kenchi.datasets import load_wilt
>>> wilt = load_wilt()
>>> wilt.data.shape
(4839, 5)
```

kenchi.datasets.samples_generator.**make_blobs**(*centers=5, center_box=(-10.0, 10.0), cluster_std=1.0, contamination=0.02, n_features=25, n_samples=500, random_state=None, shuffle=True*)

Generate isotropic Gaussian blobs with outliers.

Parameters

- **centers** (*int or array-like of shape (n_centers, n_features), default 5*) – Number of centers to generate, or the fixed center locations.
- **center_box** (*pair of floats (min, max), default (-10.0, 10.0)*) – Bounding box for each cluster center when centers are generated at random.

- **cluster_std** (*float or array-like of shape (n_centers,), default 1.0*) – Standard deviation of the clusters.
- **contamination** (*float, default 0.02*) – Proportion of outliers in the data set.
- **n_features** (*int, default 25*) – Number of features for each sample.
- **n_samples** (*int, default 500*) – Number of samples.
- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **shuffle** (*bool, default True*) – If True, shuffle samples.

Returns

- **X** (*array-like of shape (n_samples, n_features)*) – Generated data.
- **y** (*array-like of shape (n_samples,)*) – Return -1 for outliers and +1 for inliers.

References

Examples

```
>>> from kenchi.datasets import make_blobs
>>> X, y = make_blobs(n_samples=10, n_features=2, contamination=0.1)
>>> X.shape
(10, 2)
>>> y.shape
(10,)
```

1.1.1.2 Module contents

1.1.2 kenchi.outlier_detection package

1.1.2.1 Submodules

class kenchi.outlier_detection.angle_based.**FastABOD**(*algorithm='auto', contamination=0.1, leaf_size=30, metric='minkowski', novelty=False, n_jobs=1, n_neighbors=20, p=2, metric_params=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Fast Angle-Based Outlier Detector (FastABOD).

Parameters

- **algorithm** (*str, default 'auto'*) – Tree algorithm to use. Valid algorithms are ['kd_tree'|'ball_tree'|'auto'].
- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **leaf_size** (*int, default 30*) – Leaf size of the underlying tree.
- **metric** (*str or callable, default 'minkowski'*) – Distance metric to use.

- **novelty** (*bool, default False*) – If True, you can use predict, decision_function and anomaly_score on new unseen data and not on the training data.
- **n_jobs** (*int, default 1*) – Number of jobs to run in parallel. If -1, then the number of jobs is set to the number of CPU cores.
- **n_neighbors** (*int, default 20*) – Number of neighbors.
- **p** (*int, default 2*) – Power parameter for the Minkowski metric.
- **metric_params** (*dict, default None*) – Additioal parameters passed to the requested metric.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

n_neighbors_
int – Actual number of neighbors used for `kneighbors` queries.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import FastABOD
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = FastABOD(n_neighbors=3)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

X_
array-like of shape (n_samples, n_features) – Training data.

class kenchi.outlier_detection.base.**BaseOutlierDetector**
Bases: sklearn.base.BaseEstimator, abc.ABC

Base class for all outlier detectors in kenchi.

References

anomaly_score(*X=None*, *normalize=False*)
Compute the anomaly score for each sample.

Parameters

- **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, compute the anomaly score for each training sample.
- **normalize** (*bool, default False*) – If True, return the normalized anomaly score.

Returns **anomaly_score** – Anomaly score for each sample.

Return type array-like of shape (n_samples,)

decision_function(*X=None*, *threshold=None*)
Compute the decision function of the given samples.

Parameters

- **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, compute the decision function of the given training samples.
- **threshold** (*float, default None*) – User-provided threshold.

Returns **shiftted_score_samples** – Shifted opposite of the anomaly score for each sample. Negative scores represent outliers and positive scores represent inliers.

Return type array-like of shape (n_samples,)

fit(*X*, *y=None*)
Fit the model according to the given training data.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Training data.
- **y** (*ignored*) –

Returns **self** – Return self.

Return type object

fit_predict(*X*, *y=None*)
Fit the model according to the given training data and predict if a particular training sample is an outlier or not.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Training Data.
- **y** (*ignored*) –

Returns **y_pred** – Return -1 for outliers and +1 for inliers.

Return type array-like of shape (n_samples,)

plot_anomaly_score(*X=None*, *normalize=False*, ***kwargs*)
Plot the anomaly score for each sample.

Parameters

- **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, plot the anomaly score for each training samples.
- **normalize** (*bool, default False*) – If True, plot the normalized anomaly score.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **bins** (*int, str or array-like, default 'auto'*) – Number of hist bins.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **hist** (*bool, default True*) – If True, plot a histogram of anomaly scores.
- **kde** (*bool, default True*) – If True, plot a gaussian kernel density estimate.
- **title** (*string, default None*) – Axes title. To disable, pass None.

- **xlabel** (*string, default 'Samples'*) – X axis title label. To disable, pass None.
- **xlim** (*tuple, default None*) – Tuple passed to `ax.xlim`.
- **ylabel** (*string, default 'Anomaly score'*) – Y axis title label. To disable, pass None.
- **ylim** (*tuple, default None*) – Tuple passed to `ax.ylim`.
- ****kwargs** (*dict*) – Other keywords passed to `ax.plot`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

plot_roc_curve(*X*, *y*, ***kwargs*)

Plot the Receiver Operating Characteristic (ROC) curve.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Data.
- **y** (*array-like of shape (n_samples,)*) – Labels.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **title** (*string, default 'ROC curve'*) – Axes title. To disable, pass None.
- **xlabel** (*string, default 'FPR'*) – X axis title label. To disable, pass None.
- **ylabel** (*string, default 'TPR'*) – Y axis title label. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `ax.plot`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

predict(*X=None*, *threshold=None*)

Predict if a particular sample is an outlier or not.

Parameters

- **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, predict if a particular training sample is an outlier or not.
- **threshold** (*float, default None*) – User-provided threshold.

Returns **y_pred** – Return -1 for outliers and +1 for inliers.

Return type array-like of shape (n_samples,)

predict_proba(*X=None*)

Predict class probabilities for each sample.

Parameters **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, predict if a particular training sample is an outlier or not.

Returns **y_score** – Class probabilities.

Return type array-like of shape (n_samples, n_classes)

score_samples(*X=None*)

Compute the opposite of the anomaly score for each sample.

Parameters **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, compute the opposite of the anomaly score for each training sample.

Returns **score_samples** – Opposite of the anomaly score for each sample.

Return type array-like of shape (n_samples,)

to_pickle(*filename*, ***kwargs*)
Persist an outlier detector object.

Parameters

- **filename** (*str or pathlib.Path*) – Path of the file in which it is to be stored.
- **kwargs** (*dict*) – Other keywords passed to `sklearn.externals.joblib.dump`.

Returns **filenames** – List of file names in which the data is stored.

Return type list

class kenchi.outlier_detection.classification_based.**OCSVM**(*cache_size=200*, *gamma='scale'*, *max_iter=-1*, *nu=0.5*, *shrinking=True*, *tol=0.001*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

One Class Support Vector Machines (only RBF kernel).

Parameters

- **cache_size** (*float, default 200*) – Specify the size of the kernel cache (in MB).
- **gamma** (*float, default 'scale'*) – Kernel coefficient. If gamma is 'scale', 1 / (n_features * np.std(X)) will be used instead.
- **max_iter** (*int, optional default -1*) – Maximum number of iterations.
- **nu** (*float, default 0.5*) – An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Should be in the interval (0, 1].
- **shrinking** (*bool, default True*) – If True, use the shrinking heuristic.
- **tol** (*float, default 0.001*) – Tolerance to declare convergence.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import OCSVM
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = OCSVM(gamma=1e-03, nu=0.25)
```

(continues on next page)

(continued from previous page)

```
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

dual_coef_
array-like of shape (1, n_SV) – Coefficients of the support vectors in the decision function.

intercept_
array-like of shape (1,) – Constant in the decision function.

support_
array-like of shape (n_SV) – Indices of support vectors.

support_vectors_
array-like of shape (n_SV, n_features) – Support vectors.

class kenchi.outlier_detection.clustering_based.**MiniBatchKMeans**(*batch_size=100*, *contamination=0.1*, *init='k-means++'*, *init_size=None*, *max_iter=100*, *max_no_improvement=10*, *n_clusters=8*, *n_init=3*, *random_state=None*, *reassignment_ratio=0.01*, *tol=0.0*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Outlier detector using K-means clustering.

Parameters

- **batch_size** (*int, optional, default 100*) – Size of the mini batches.
- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **init** (*str or array-like, default 'k-means++'*) – Method for initialization. Valid options are ['k-means++'|'random'].
- **init_size** (*int, default: 3 * batch_size*) – Number of samples to randomly sample for speeding up the initialization.
- **max_iter** (*int, default 100*) – Maximum number of iterations.
- **max_no_improvement** (*int, default 10*) – Control early stopping based on the consecutive number of mini batches that does not yield an improvement on the smoothed inertia. To disable convergence detection based on inertia, set max_no_improvement to None.
- **n_clusters** (*int, default 8*) – Number of clusters.
- **n_init** (*int, default 3*) – Number of initializations to perform.
- **random_state** (*int or RandomState instance, default None*) – Seed of the pseudo random number generator.

- **reassignment_ratio** (*float, default 0.01*) – Control the fraction of the maximum number of counts for a center to be reassigned.
- **tol** (*float, default 0.0*) – Tolerance to declare convergence.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import MiniBatchKMeans
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = MiniBatchKMeans(n_clusters=1, random_state=0)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

cluster_centers_
array-like of shape (n_clusters, n_features) – Coordinates of cluster centers.

inertia_
float – Value of the inertia criterion associated with the chosen partition.

labels_
array-like of shape (n_samples,) – Label of each point.

class kenchi.outlier_detection.density_based.**LOF**(*algorithm='auto', contamination='auto', leaf_size=30, metric='minkowski', novelty=False, n_jobs=1, n_neighbors=20, p=2, metric_params=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Local Outlier Factor.

Parameters

- **algorithm** (*str, default 'auto'*) – Tree algorithm to use. Valid algorithms are ['kd_tree'|'ball_tree'|'auto'].
- **contamination** (*float, default 'auto'*) – Proportion of outliers in the data set. Used to define the threshold.
- **leaf_size** (*int, default 30*) – Leaf size of the underlying tree.
- **metric** (*str or callable, default 'minkowski'*) – Distance metric to use.
- **novelty** (*bool, default False*) – If True, you can use predict, decision_function and anomaly_score on new unseen data and not on the training data.
- **n_jobs** (*int, default 1*) – Number of jobs to run in parallel. If -1, then the number of jobs is set to the number of CPU cores.

- **n_neighbors** (*int, default 20*) – Number of neighbors.
- **p** (*int, default 2*) – Power parameter for the Minkowski metric.
- **metric_params** (*dict, default None*) – Additioal parameters passed to the requested metric.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import LOF
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = LOF(n_neighbors=3)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

X_
array-like of shape (n_samples, n_features) – Training data.

n_neighbors_
int – Actual number of neighbors used for `kneighbors` queries.

negative_outlier_factor_
array-like of shape (n_samples,) – Opposite LOF of the training samples.

class kenchi.outlier_detection.distance_based.**KNN**(*aggregate=False, algorithm='auto', contamination=0.1, leaf_size=30, metric='minkowski', novelty=False, n_jobs=1, n_neighbors=20, p=2, metric_params=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Outlier detector using k-nearest neighbors algorithm.

Parameters

- **aggregate** (*bool, default False*) – If True, return the sum of the distances from k nearest neighbors as the anomaly score.
- **algorithm** (*str, default 'auto'*) – Tree algorithm to use. Valid algorithms are ['kd_tree'|'ball_tree'|'auto'].
- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.

- **leaf_size** (*int, default 30*) – Leaf size of the underlying tree.
- **metric** (*str or callable, default 'minkowski'*) – Distance metric to use.
- **novelty** (*bool, default False*) – If True, you can use predict, decision_function and anomaly_score on new unseen data and not on the training data.
- **n_jobs** (*int, default 1*) – Number of jobs to run in parallel. If -1, then the number of jobs is set to the number of CPU cores.
- **n_neighbors** (*int, default 20*) – Number of neighbors.
- **p** (*int, default 2*) – Power parameter for the Minkowski metric.
- **metric_params** (*dict, default None*) – Additioal parameters passed to the requested metric.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

n_neighbors_
int – Actual number of neighbors used for `kneighbors` queries.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import KNN
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = KNN(n_neighbors=3)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

X_
array-like of shape (n_samples, n_features) – Training data.

class kenchi.outlier_detection.distance_based.**OneTimeSampling**(*contamination=0.1*, *metric='euclidean'*, *novelty=False*, *n_subsamples=20*, *random_state=None*, *metric_params=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

One-time sampling.

Parameters

- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **metric** (*str, default 'euclidean'*) – Distance metric to use.
- **novelty** (*bool, default False*) – If True, you can use predict, decision_function and anomaly_score on new unseen data and not on the training data.
- **n_subsamples** (*int, default 20*) – Number of random samples to be used.
- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **metric_params** (*dict, default None*) – Additional parameters passed to the requested metric.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

subsamples_
array-like of shape (n_subsamples,) – Indices of subsamples.

S_
array-like of shape (n_subsamples, n_features) – Subset of the given training data.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import OneTimeSampling
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = OneTimeSampling(n_subsamples=3, random_state=0)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

class kenchi.outlier_detection.ensemble.**IForest**(*bootstrap=False, contamination='auto', max_features=1.0, max_samples='auto', n_estimators=100, n_jobs=1, random_state=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Isolation forest (iForest).

Parameters

- **bootstrap** (*bool, False*) – If True, individual trees are fit on random subsets of the training data sampled with replacement. If False, sampling without replacement is performed.

- **contamination** (*float, default 'auto'*) – Proportion of outliers in the data set. Used to define the threshold.
- **max_features** (*int or float, default 1.0*) – Number of features to draw from X to train each base estimator.
- **max_samples** (*int ,float or str, default 'auto'*) – Number of samples to draw from X to train each base estimator.
- **n_estimators** (*int, default 100*) – Number of base estimators in the ensemble.
- **n_jobs** (*int*) – Number of jobs to run in parallel. If -1, then the number of jobs is set to the number of CPU cores.
- **random_state** (*int or RandomState instance, default None*) – Seed of the pseudo random number generator.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import IForest
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = IForest(random_state=0)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

estimators_
list – Collection of fitted sub-estimators.

estimators_samples_
int – Subset of drawn samples for each base estimator.

max_samples_
int – Actual number of samples.

class kenchi.outlier_detection.reconstruction_based.**PCA**(*contamination=0.1, iterated_power='auto', n_components=None, random_state=None, svd_solver='auto', tol=0.0, whiten=False*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Outlier detector using Principal Component Analysis (PCA).

Parameters

- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **iterated_power** (*int, default 'auto'*) – Number of iterations for the power method computed by svd_solver == 'randomized'.
- **n_components** (*int, float, or string, default None*) – Number of components to keep.
- **random_state** (*int or RandomState instance, default None*) – Seed of the pseudo random number generator.
- **svd_solver** (*string, default 'auto'*) – SVD solver to use. Valid solvers are ['auto'|'full'|'arpack'|'randomized'].
- **tol** (*float, default 0.0*) – Tolerance to declare convergence for singular values computed by svd_solver == 'arpack'.
- **whiten** (*bool, default False*) – If True, the `components_` vectors are multiplied by the square root of n_samples and then divided by the singular values to ensure uncorrelated outputs with unit component-wise variances.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import PCA
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = PCA()
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

components_
array-like of shape (n_components, n_features) – Principal axes in feature space, representing the directions of maximum variance in the data.

explained_variance_
array-like of shape (n_components,) – Amount of variance explained by each of the selected components.

explained_variance_ratio_
array-like of shape (n_components,) – Percentage of variance explained by each of the selected components.

mean_
array-like of shape (n_features,) – Per-feature empirical mean, estimated from the training set.

n_components_
int – Estimated number of components.

noise_variance_
float – Estimated noise covariance following the Probabilistic PCA model from Tipping and Bishop 1999.

singular_values_
array-like of shape (n_components,) – Singular values corresponding to each of the selected components.

class kenchi.outlier_detection.statistical.**GMM**(*contamination=0.1*, *covariance_type='full'*, *init_params='kmeans'*, *max_iter=100*, *means_init=None*, *n_components=1*, *n_init=1*, *precisions_init=None*, *random_state=None*, *reg_covar=1e-06*, *tol=0.001*, *warm_start=False*, *weights_init=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Outlier detector using Gaussian Mixture Models (GMMs).

Parameters

- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **covariance_type** (*str, default 'full'*) – String describing the type of covariance parameters to use. Valid options are ['full'|'tied'|'diag'|'spherical'].
- **init_params** (*str, default 'kmeans'*) – Method used to initialize the weights, the means and the precisions. Valid options are ['kmeans'|'random'].
- **max_iter** (*int, default 100*) – Maximum number of iterations.
- **means_init** (*array-like of shape (n_components, n_features), default None*) – User-provided initial means.
- **n_init** (*int, default 1*) – Number of initializations to perform.
- **n_components** (*int, default 1*) – Number of mixture components.
- **precisions_init** (*array-like, default None*) – User-provided initial precisions.
- **random_state** (*int or RandomState instance, default None*) – Seed of the pseudo random number generator.
- **reg_covar** (*float, default 1e-06*) – Non-negative regularization added to the diagonal of covariance.
- **tol** (*float, default 1e-03*) – Tolerance to declare convergence.
- **warm_start** (*bool, default False*) – If True, the solution of the last fitting is used as initialization for the next call of `fit`.
- **weights_init** (*array-like of shape (n_components,), default None*) – User-provided initial weights.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import GMM
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = GMM(random_state=0)
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

converged_
bool – True when convergence was reached in `fit`, False otherwise.

covariances_
array-like – Covariance of each mixture component.

lower_bound_
float – Log-likelihood of the best fit of EM.

means_
array-like of shape (n_components, n_features) – Mean of each mixture component.

n_iter_
int – Number of step used by the best fit of EM to reach the convergence.

precisions_
array-like – Precision matrix for each component in the mixture.

precisions_cholesky_
array-like – Cholesky decomposition of the precision matrices of each mixture component.

weights_
array-like of shape (n_components,) – Weight of each mixture components.

class kenchi.outlier_detection.statistical.**HBOS**(*bins='auto'*, *contamination=0.1*, *novelty=False*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Histogram-based outlier detector.

Parameters

- **bins** (*int or str, default 'auto'*) – Number of hist bins.
- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **novelty** (*bool, default False*) – If True, you can use predict, decision_function and anomaly_score on new unseen data and not on the training data.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

bin_edges_
array-like – Bin edges.

data_max_
array-like of shape (n_features,) – Per feature maximum seen in the data.

data_min_
array-like of shape (n_features,) – Per feature minimum seen in the data.

hist_
array-like – Values of the histogram.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import HBOS
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = HBOS()
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

class kenchi.outlier_detection.statistical.**KDE**(*algorithm='auto', atol=0.0, bandwidth=1.0, breadth_first=True, contamination=0.1, kernel='gaussian', leaf_size=40, metric='euclidean', rtol=0.0, metric_params=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Outlier detector using Kernel Density Estimation (KDE).

Parameters

- **algorithm** (*str, default 'auto'*) – Tree algorithm to use. Valid algorithms are ['kd_tree'|'ball_tree'|'auto'].
- **atol** (*float, default 0.0*) – Desired absolute tolerance of the result.
- **bandwidth** (*float, default 1.0*) – Bandwidth of the kernel.
- **breadth_first** (*bool, default True*) – If true, use a breadth-first approach to the problem. Otherwise use a depth-first approach.
- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **kernel** (*str, default 'gaussian'*) – Kernel to use. Valid kernels are ['gaussian'|'tophat'|'epanechnikov'|'exponential'|'linear'|'cosine'].
- **leaf_size** (*int, default 40*) – Leaf size of the underlying tree.
- **metric** (*str, default 'euclidean'*) – Distance metric to use.
- **rtol** (*float, default 0.0*) – Desired relative tolerance of the result.
- **metric_params** (*dict, default None*) – Additional parameters to be passed to the requested metric.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import KDE
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = KDE()
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

X_
array-like of shape (n_samples, n_features) – Training data.

class kenchi.outlier_detection.statistical.**SparseStructureLearning**(*alpha=0.01*, *assume_centered=False*, *contamination=0.1*, *enet_tol=0.0001*, *max_iter=100*, *mode='cd'*, *tol=0.0001*, *apcluster_params=None*)

Bases: *kenchi.outlier_detection.base.BaseOutlierDetector*

Outlier detector using sparse structure learning.

Parameters

- **alpha** (*float, default 0.01*) – Regularization parameter.
- **assume_centered** (*bool, default False*) – If True, data are not centered before computation.
- **contamination** (*float, default 0.1*) – Proportion of outliers in the data set. Used to define the threshold.
- **enet_tol** (*float, default 1e-04*) – Tolerance for the elastic net solver used to calculate the descent direction. This parameter controls the accuracy of the search direction for a given column update, not of the overall parameter estimate. Only used for mode='cd'.
- **max_iter** (*integer, default 100*) – Maximum number of iterations.
- **mode** (*str, default 'cd'*) – Lasso solver to use: coordinate descent or LARS.
- **tol** (*float, default 1e-04*) – Tolerance to declare convergence.

- **apcluster_params** (*dict, default None*) – Additional parameters passed to `sklearn.cluster.affinity_propagation`.

anomaly_score_
array-like of shape (n_samples,) – Anomaly score for each training data.

contamination_
float – Actual proportion of outliers in the data set.

threshold_
float – Threshold.

labels_
array-like of shape (n_features,) – Label of each feature.

References

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import SparseStructureLearning
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = SparseStructureLearning()
>>> det.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

covariance_
array-like of shape (n_features, n_features) – Estimated covariance matrix.

featurewise_anomaly_score(*X*)
Compute the feature-wise anomaly scores for each sample.

Parameters **X** (*array-like of shape (n_samples, n_features)*) – Data.

Returns **anomaly_score** – Feature-wise anomaly scores for each sample.

Return type array-like of shape (n_samples, n_features)

graphical_model_
networkx Graph – GGM.

isolates_
array-like of shape (n_isolates,) – Indices of isolates.

location_
array-like of shape (n_features,) – Estimated location.

n_iter_
int – Number of iterations run.

partial_corrcoef_
array-like of shape (n_features, n_features) – Partial correlation coefficient matrix.

plot_graphical_model(***kwargs*)
Plot the Gaussian Graphical Model (GGM).

Parameters

- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **title** (*string, default 'GGM (n_clusters, n_features, n_isolates)'*) – Axes title. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `nx.draw_networkx`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

plot_partial_corrcoef(***kwargs*)
Plot the partial correlation coefficient matrix.

Parameters

- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **cbar** (*bool, default True.*) – If Ture, to draw a colorbar.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **title** (*string, default 'Partial correlation'*) – Axes title. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `ax.pcolormesh`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

precision_
array-like of shape (n_features, n_features) – Estimated pseudo inverse matrix.

1.1.2.2 Module contents

1.2 Submodules

class kenchi.metrics.**LeeLiuScorer**
Bases: `object`

Lee-Liu scorer.

References

class kenchi.metrics.**NegativeMVAUCScorer**(*data_max*, *data_min*, *interval=(0.9, 0.999)*, *n_offsets=1000*, *n_uniform_samples=1000*, *random_state=None*)
Bases: `object`

Negative MV AUC scorer.

Parameters

- **data_max** (*array-like of shape (n_features,)*) – Per feature maximum seen in the data.
- **data_min** (*array-like of shape (n_features,)*) – Per feature minimum seen in the data.
- **interval** (*tuple, default (0.9, 0.999)*) – Interval of probabilities.
- **n_offsets** (*int, default 1000*) – Number of offsets.
- **n_uniform_samples** (*int, default 1000*) – Number of samples which are drawn from the uniform distribution over the hypercube enclosing the data.
- **random_state** (*int or RandomState instance, default None*) – Seed of the pseudo random number generator.

References

kenchi.pipeline.**make_pipeline**(**steps*)

Construct a Pipeline from the given estimators. This is a shorthand for the Pipeline constructor; it does not require, and does not permit, naming the estimators. Instead, their names will be set to the lowercase of their types automatically.

Parameters ***steps** (*list*) – List of estimators.

Returns **p**

Return type *Pipeline*

Examples

```
>>> from kenchi.outlier_detection import MiniBatchKMeans
>>> from kenchi.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler()
>>> det = MiniBatchKMeans()
>>> pipeline = make_pipeline(scaler, det)
```

class kenchi.pipeline.**Pipeline**(*steps*, *memory=None*)

Bases: sklearn.pipeline.Pipeline

Pipeline of transforms with a final estimator.

Parameters

- **steps** (*list*) – List of (name, transform) tuples (implementing fit/transform) that are chained, in the order in which they are chained, with the last object an estimator.
- **memory** (*instance of joblib.Memory or string, default None*) – Used to cache the fitted transformers of the pipeline. By default, no caching is performed. If a string is given, it is the path to the caching directory. Enabling caching triggers a clone of the transformers before fitting. Therefore, the transformer instance given to the pipeline cannot be inspected directly. Use the attribute named_steps or steps to inspect estimators within the pipeline. Caching the transformers is advantageous when fitting is time consuming.

named_steps

dict – Read-only attribute to access any step parameter by user given name. Keys are step names and values are steps parameters.

Examples

```
>>> import numpy as np
>>> from kenchi.outlier_detection import MiniBatchKMeans
>>> from kenchi.pipeline import Pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> X = np.array([
...     [0., 0.], [1., 1.], [2., 0.], [3., -1.], [4., 0.],
...     [5., 1.], [6., 0.], [7., -1.], [8., 0.], [1000., 1.]
... ])
>>> det = MiniBatchKMeans(n_clusters=1, random_state=0)
>>> scaler = StandardScaler()
>>> pipeline = Pipeline([('scaler', scaler), ('det', det)])
>>> pipeline.fit_predict(X)
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1, -1])
```

anomaly_score (*X=None*, ***kwargs*)
Apply transforms, and compute the anomaly score for each sample with the final estimator.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Data. If None, compute the anomaly score for each training samples.
- **normalize** (*bool, default False*) – If True, return the normalized anomaly score.

Returns **anomaly_score** – Anomaly score for each sample.

Return type array-like of shape (n_samples,)

featurewise_anomaly_score (*X*)
Apply transforms, and compute the feature-wise anomaly scores for each sample with the final estimator.

Parameters **X** (*array-like of shape (n_samples, n_features)*) – Data.

Returns **anomaly_score** – Feature-wise anomaly scores for each sample.

Return type array-like of shape (n_samples, n_features)

plot_anomaly_score (*X=None*, ***kwargs*)
Apply transoforms, and plot the anomaly score for each sample with the final estimator.

Parameters

- **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, plot the anomaly score for each training samples.
- **normalize** (*bool, default False*) – If True, plot the normalized anomaly score.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **bins** (*int, str or array-like, default 'auto'*) – Number of hist bins.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **hist** (*bool, default True*) – If True, plot a histogram of anomaly scores.
- **kde** (*bool, default True*) – If True, plot a gaussian kernel density estimate.
- **title** (*string, default None*) – Axes title. To disable, pass None.

- **xlabel** (*string, default 'Samples'*) – X axis title label. To disable, pass None.
- **xlim** (*tuple, default None*) – Tuple passed to `ax.xlim`.
- **ylabel** (*string, default 'Anomaly score'*) – Y axis title label. To disable, pass None.
- **ylim** (*tuple, default None*) – Tuple passed to `ax.ylim`.
- ****kwargs** (*dict*) – Other keywords passed to `ax.plot`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

plot_graphical_model
Apply transforms, and plot the Gaussian Graphical Model (GGM) with the final estimator.

Parameters

- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **title** (*string, default 'GGM (n_clusters, n_features, n_isolates)'*) – Axes title. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `nx.draw_networkx`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

plot_partial_corrcoef
Apply transforms, and plot the partial correlation coefficient matrix with the final estimator.

Parameters

- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **cbar** (*bool, default True.*) – If True, draw a colorbar.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **title** (*string, default 'Partial correlation'*) – Axes title. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `ax.pcolormesh`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

plot_roc_curve (*X*, *y*, ***kwargs*)
Apply transoforms, and plot the Receiver Operating Characteristic (ROC) curve with the final estimator.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Data.
- **y** (*array-like of shape (n_samples,)*) – Labels.

- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **title** (*string, default 'ROC curve'*) – Axes title. To disable, pass None.
- **xlabel** (*string, default 'FPR'*) – X axis title label. To disable, pass None.
- **ylabel** (*string, default 'TPR'*) – Y axis title label. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `ax.plot`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

score_samples(*X=None*)

Apply transforms, and compute the opposite of the anomaly score for each sample with the final estimator.

Parameters **X** (*array-like of shape (n_samples, n_features), default None*) – Data. If None, compute the opposite of the anomaly score for each training sample.

Returns **score_samples** – Opposite of the anomaly score for each sample.

Return type array-like of shape (n_samples,)

to_pickle(*filename*, ***kwargs*)

Persist a pipeline object.

Parameters

- **filename** (*str or pathlib.Path*) – Path of the file in which it is to be stored.
- **kwargs** (*dict*) – Other keywords passed to `sklearn.externals.joblib.dump`.

Returns **filenames** – List of file names in which the data is stored.

Return type list

kenchi.plotting.**plot_anomaly_score**(*anomaly_score*, *ax=None*, *bins='auto'*, *figsize=None*, *filename=None*, *hist=True*, *kde=True*, *threshold=None*, *title=None*, *xlabel='Samples'*, *xlim=None*, *ylabel='Anomaly score'*, *ylim=None*, ***kwargs*)

Plot the anomaly score for each sample.

Parameters

- **anomaly_score** (*array-like of shape (n_samples,)*) – Anomaly score for each sample.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **bins** (*int, str or array-like, default 'auto'*) – Number of hist bins.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **hist** (*bool, default True*) – If True, plot a histogram of anomaly scores.
- **kde** (*bool, default True*) – If True, plot a gaussian kernel density estimate.
- **threshold** (*float, default None*) – Threshold.
- **title** (*string, default None*) – Axes title. To disable, pass None.
- **xlabel** (*string, default 'Samples'*) – X axis title label. To disable, pass None.

- **xlim** (*tuple, default None*) – Tuple passed to `ax.xlim`.
- **ylabel** (*string, default 'Anomaly score'*) – Y axis title label. To disable, pass None.
- **ylim** (*tuple, default None*) – Tuple passed to `ax.ylim`.
- ****kwargs** (*dict*) – Other keywords passed to `ax.plot`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

Examples

```
>>> import matplotlib.pyplot as plt
>>> from kenchi.datasets import load_wdbc
>>> from kenchi.outlier_detection import MiniBatchKMeans
>>> from kenchi.plotting import plot_anomaly_score
>>> X, _ = load_wdbc(random_state=0, return_X_y=True)
>>> det = MiniBatchKMeans(random_state=0).fit(X)
>>> anomaly_score = det.anomaly_score(X, normalize=True)
>>> plot_anomaly_score(
...     anomaly_score, threshold=det.threshold_, linestyle='', marker='.'
... )
<matplotlib.axes._subplots.AxesSubplot object at 0x...>
>>> plt.show()
```

kenchi.plotting.**plot_graphical_model**(*G*, *ax=None*, *figsize=None*, *filename=None*, *random_state=None*, *title='GGM'*, ***kwargs*)

Plot the Gaussian Graphical Model (GGM).

Parameters

- **G** (*networkx Graph*) – GGM.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **random_state** (*int, RandomState instance, default None*) – Seed of the pseudo random number generator.
- **title** (*string, default 'GGM'*) – Axes title. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `nx.draw_networkx`.

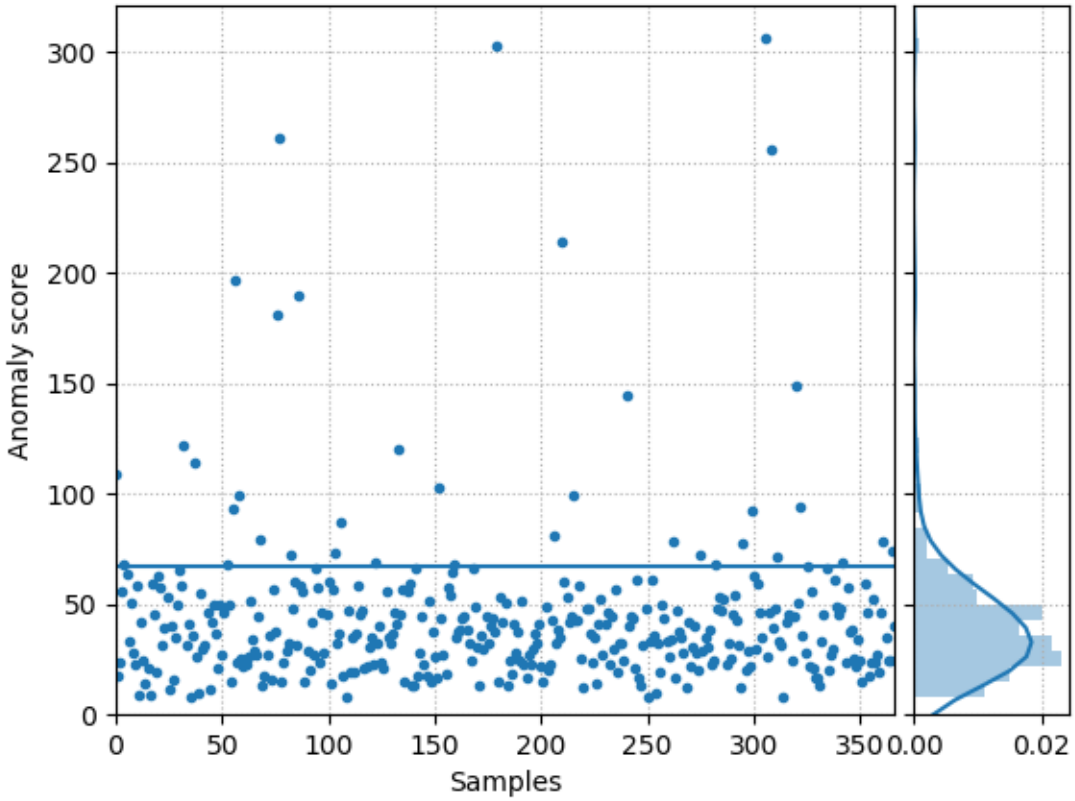
Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

Examples

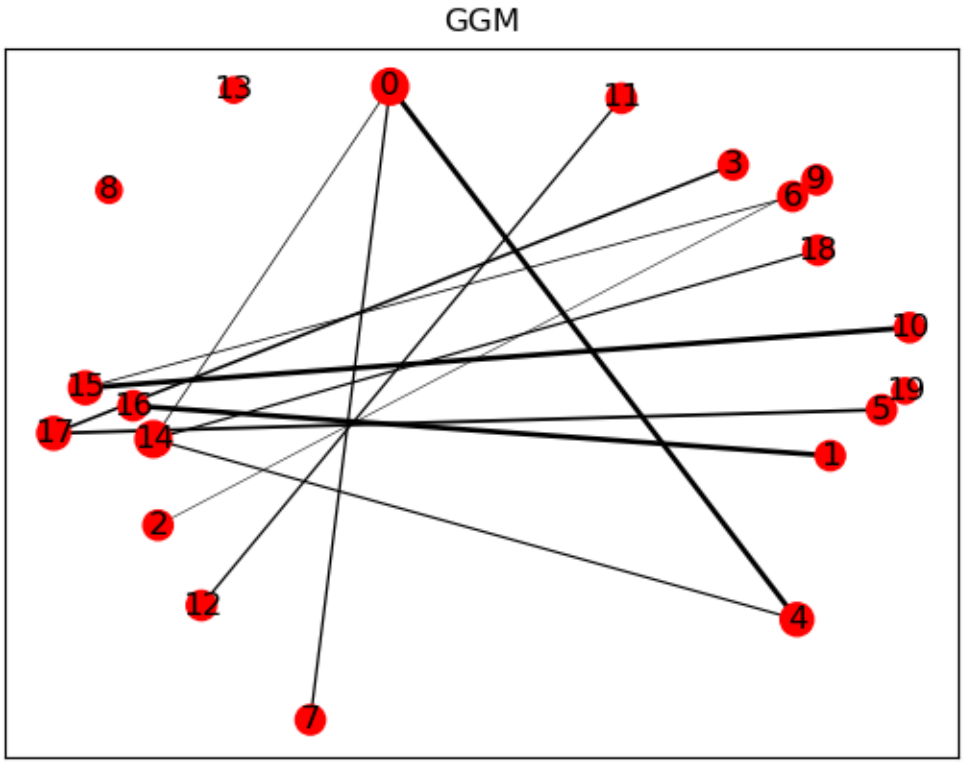
```
>>> import matplotlib.pyplot as plt
>>> import networkx as nx
>>> from kenchi.plotting import plot_graphical_model
>>> from sklearn.datasets import make_sparse_spd_matrix
>>> A = make_sparse_spd_matrix(dim=20, norm_diag=True, random_state=0)
```

(continues on next page)


Anomaly score
Samples
0
50
100
150
200
250
300
350
0.00
0.02

(continued from previous page)

```
>>> G = nx.from_numpy_matrix(A)
>>> plot_graphical_model(G, random_state=0)
<matplotlib.axes._subplots.AxesSubplot object at 0x...>
>>> plt.show()
```



kenchi.plotting.**plot_partial_corrcoef**(*partial_corrcoef*, *ax=None*, *cbar=True*, *figsize=None*, *filename=None*, *title='Partial correlation'*, ***kwargs*)

Plot the partial correlation coefficient matrix.

Parameters

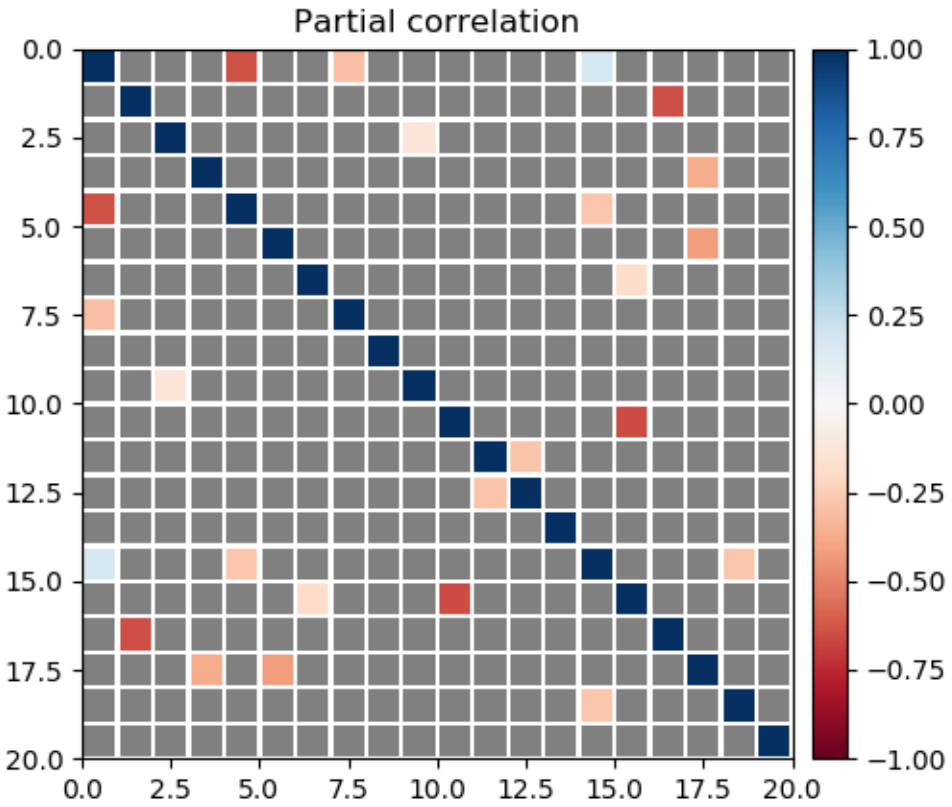
- **partial_corrcoef** (*array-like of shape (n_features, n_features)*) – Partial correlation coefficient matrix.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.
- **cbar** (*bool, default True.*) – If True, draw a colorbar.
- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **title** (*string, default 'Partial correlation'*) – Axes title. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `ax.pcolormesh`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

Examples

```
>>> import matplotlib.pyplot as plt
>>> from kenchi.plotting import plot_partial_corrcoef
>>> from sklearn.datasets import make_sparse_spd_matrix
>>> A = make_sparse_spd_matrix(dim=20, norm_diag=True, random_state=0)
>>> plot_partial_corrcoef(A)
<matplotlib.axes._subplots.AxesSubplot object at 0x...>
>>> plt.show()
```



kenchi.plotting.**plot_roc_curve**(*y_true*, *y_score*, *ax=None*, *figsize=None*, *filename=None*, *title='ROC curve'*, *xlabel='FPR'*, *ylabel='TPR'*, ***kwargs*)

Plot the Receiver Operating Characteristic (ROC) curve.

Parameters

- **y_true** (*array-like of shape (n_samples,)*) – True Labels.
- **y_score** (*array-like of shape (n_samples,)*) – Target scores.
- **ax** (*matplotlib Axes, default None*) – Target axes instance.

- **figsize** (*tuple, default None*) – Tuple denoting figure size of the plot.
- **filename** (*str, default None*) – If provided, save the current figure.
- **title** (*string, default 'ROC curve'*) – Axes title. To disable, pass None.
- **xlabel** (*string, default 'FPR'*) – X axis title label. To disable, pass None.
- **ylabel** (*string, default 'TPR'*) – Y axis title label. To disable, pass None.
- ****kwargs** (*dict*) – Other keywords passed to `ax.plot`.

Returns **ax** – Axes on which the plot was drawn.

Return type matplotlib Axes

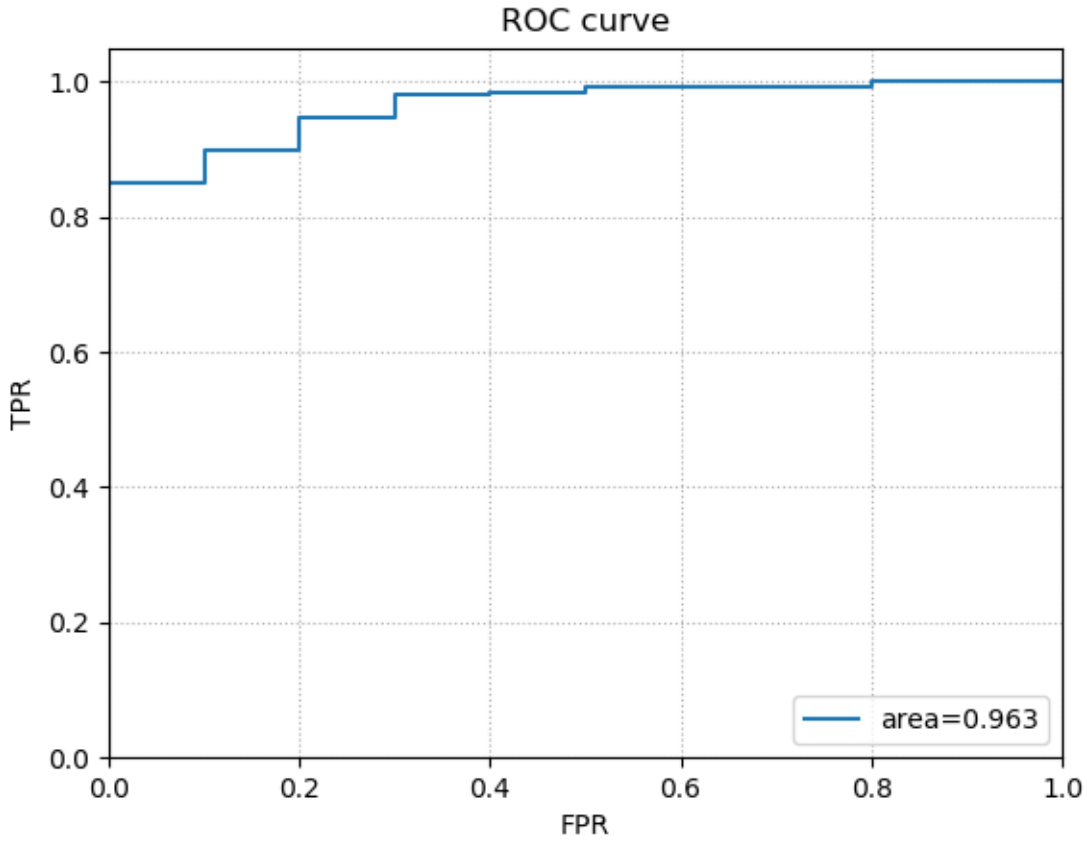
Examples

```
>>> import matplotlib.pyplot as plt
>>> from kenchi.datasets import load_wdbc
>>> from kenchi.outlier_detection import MiniBatchKMeans
>>> from kenchi.plotting import plot_roc_curve
>>> X, y = load_wdbc(random_state=0, return_X_y=True)
>>> det = MiniBatchKMeans(random_state=0).fit(X)
>>> score_samples = det.score_samples(X)
>>> plot_roc_curve(y, score_samples)
<matplotlib.axes._subplots.AxesSubplot object at 0x...>
>>> plt.show()
```

kenchi.utils.**check_contamination**(*contamination*, *low=0.0*, *high=0.5*)

Raise ValueError if the contamination is not valid.

1.3 Module contents


ROC curve
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0
area=0.963

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

k

Index

O

P

S

T

W

X