SciKit-Learn API

CONTENTS

BASE CLASSES

(from sklearn.base import x)

BaseEstimator -- Base class for all estimators
ClassifierMixin -- Mixin class for all classifiers
ClusterMixin -- Mixin class for all cluster estimators
RegressorMixin -- Mixin class for all regression estimators/
TransformerMixin -- Mixin class for all transformers
clone (estimator[, safe]) -- Constructs new estimator with the same parameters.

LINEAR MODELS

(from sklearn.linear_model import x)

LinearRegression ([...])
Ordinary least squares Linear Regression.

Ridge [alpha, fit_intercept, max_iter, normalize, solver, tol)
Linear least squares regression. Uses l2 regularization. Built-in multivariate regression support.
RidgeCV ([alphas, ...])
Ridge regression with built-in CV.

Lasso (alpha, copy_X, fit_intercept, max_iter, normalize, positive, precompute, tol)
Estimates sparse coefficients. Uses L1 regularizer. Alpha = degree of sparsity.
LassoCV ([eps, n_alphas, ...])
Lasso with CV

ElasticNet ([alpha, l1_ratio, ...])
Linear regression. Uses combined L1 & L2 regularization. Useful when multiple features are correlated to each other.
ElasticNetCV ([l1_ratio, eps, ...])
Elastic Net, iterative fitting along regularization path

MultiTaskLasso ([alpha, ...])
Estimates sparse coefficients for multiple regression problems (y=2D array)

MultiTaskLassoCV ([eps, ...])
Multi-task L1/L2 Lasso with CV

Lars ([fit_intercept, verbose, ...])
Least Angle Regression - for high-D datasets. Iterative refit of residuals ⇒ sensitive to noise.
LarsCV ([fit_intercept, ...])
Cross-validated Least Angle Regression

LassoLars ([alpha, ...])
Lasso model fit with Least Angle Regression
LassoLarsCV ([fit_intercept, ...])

OrthogonalMatchingPursuit ([...])
OMP algo - constrains # non-zero coefficients (L0 pseudo-norm)
OrthogonalMatchingPursuitCV ([...])

BayesianRidge ([n_iter, tol, ...])
Regularization not hard-coded, but tuned to available data.
ARDRegression ([n_iter, tol, ...])
Similar to Bayes Ridge - doesn't assume spherical Gaussian distribution. (Elliptical instead.)

LogisticRegression ([penalty, ...])
aka logit, max entropy or log-linear classifier. Uses LIBLINEAR library.

SGDClassifier ([loss, penalty, ...])
SGDRegressor ([loss, penalty, ...])
Best use case = when #samples & #features is very large. Multi loss functions, eg loss="log" ⇒ logistic regression; loss="hinge" ⇒ SVM

Perceptron ([penalty, alpha, ...])
Very simple model.

PassiveAggressiveClassifier ([...])
PassiveAggressiveRegressor ([C, ...])
For large-scale learning. No learning rate required; regularization included.

RANSACRegressor ([...])
Iterative algo, estimates params from subset of inliers of dataset. Result depends on probability based on #min_samples.

MultiTaskElasticNet ([alpha, ...])
Multi-task ElasticNet model trained with L1/L2 mixed-norm as regularizer

---

MultiTaskElasticNetCV ([...])
Multi-task L1/L2 ElasticNet with CV

RandomizedLasso ([alpha, ...])
Randomized Lasso.

RandomizedLogisticRegression ([...])
Randomized Logistic Regression

RidgeClassifier ([alpha, ...])
Classifier using Ridge regression.

---

lars_path (X, y[, Xy, Gram, ...])
Least Angle Regression or Lasso using LARS algo

---

lasso_path (X, y[, eps, ...])
Compute Lasso path with coordinate descent

---

lasso_stability_path (X, y[, ...])
Stabiliy path using random Lasso estimates

---

orthogonal_mp (X, y[, ...])
Orthogonal Matching Pursuit (OMP)

---

orthogonal_mp_gram (Gram, Xy[, ...])
Gram Orthogonal Matching Pursuit (OMP)

CLUSTERING

(from sklearn.cluster import x)
** popular unsupervised algorithms*

** classes*
AffinityPropagation ([damping, ...])

AgglomerativeClustering ([...])

DBSCAN ([eps, min_samples, metric, ...])

FeatureAgglomeration ([n_clusters, ...])

KMeans ([n_clusters, init, n_init, ...])

MiniBatchKMeans ([n_clusters, init, ...])

MeanShift ([bandwidth, seeds, ...])

SpectralClustering ([n_clusters, ...])

Ward ([n_clusters, memory, ...])

** functions*
estimate_bandwidth (X[, quantile, ...]) Estimate b/w to use with mean-shift algorithm.

---

k_means (X, n_clusters[, init, ...])
K-means clustering algorithm.

---

ward_tree (X[, connectivity, ...])
Ward clustering based on a feature matrix.

---

affinity_propagation (S[, ...])
Affinity Propagation Clustering of data

---

dbscan (X[, eps, min_samples, ...])
DBSCAN clustering from vector array or distance matrix.

---

mean_shift (X[, bandwidth, seeds, ...])
mean shift clustering using a flat kernel.

---

spectral_clustering (affinity[, ...])
cluster a projection to normalized laplacian.

BiCLUSTERING

(from sklearn.bicluster import x)
** todo*

SpectralBiclustering ([n_clusters, method, ...])

SpectralCoclustering ([n_clusters, ...])

COVARIANCE

(from sklearn.covariance import x)
** The precision matrix defined as inverse covariance also estimated.*

EmpiricalCovariance ([...])
empirical_covariance (X[, ...])
Maximum likelihood covariance estimator

EllipticEnvelope ([...])
Detects outliers in a Gaussian distributed dataset.

GraphLasso ([alpha, mode, tol, ...])
GraphLassoCV ([alphas, ...])
graph_lasso (emp_cov, alpha[, ...])
Sparse inverse covariance with an l1 estimator.

LedoitWolf ([store_precision, ...])
ledoit_wolf (X[, assume_centered, ...])

MinCovDet ([store_precision, ...])
Minimum Covariance Determinant (MCD)

OAS ([store_precision, ...])
oas (X[, assume_centered])

ShrunkCovariance ([...])
shrunk_covariance (emp_cov[, ...])

CROSS VALIDATION

(from sklearn.cross_validation import x)
** todo*

KFold (n[, n_folds, ...])
K-Folds cross validation iterator.

LeaveOneLabelOut (labels[, ...])
Leave-One-Label_Out cross-validation iterator

LeaveOneOut (n[, indices])
Leave-One-Out cross validation iterator.

LeavePLabelOut (labels, p[, ...])
Leave-P-Label_Out cross-validation iterator

LeavePOut (n, p[, indices])
Leave-P-Out cross validation iterator

StratifiedKFold (y[, ...])
Stratified K-Folds cross validation iterator

ShuffleSplit (n[, n_iter, ...])
Random permutation cross-validation iterator.

StratifiedShuffleSplit (y[, ...])
Stratified ShuffleSplit cross validation iterator

train_test_split (*arrays, ...)
Split arrays or matrices into random train and test subsets

cross_val_score (estimator, X)
Evaluate a score by cross-validation

permutation_test_score (...)
Evaluate the significance of a cross-validated score with permutations

check_cv (cv[, X, y, classifier])
Input checker utility for building a CV in a user friendly way.

DATASET LOADERS

(from sklearn.datasets import x)
** todo*

fetch_20newsgroups ([data_home, ...])
fetch_20newsgroups_vectorized ([...]) -- to TFIDF
load_boston () -- regression
load_diabetes () -- regression
load_digits ([n_class]) -- classification
load_iris () -- classification

load_files (container_path[, ...])
Load text files with categories as subfolder names.

load_lfw_pairs ([download_if_missing])
fetch_lfw_pairs ([subset, ...])
Labeled Faces in the Wild (LFW) pairs dataset
fetch_lfw_people ([data_home, ...])
load_lfw_people ([download_if_missing])
Labeled Faces in the Wild (LFW) people dataset

load_linnerud ()
fetch_mldata (dataname[, ...])
fetch_olivetti_faces ([data_home, ...])
fetch_california_housing ([...])
fetch_covtype ([data_home, ...])
load_mlcomp (name_or_id[, set_, ...])

load_sample_image (image_name)

load_sample_images ()

---

load_svmlight_file (f[, n_features, ...])
dump_svmlight_file (X, y, f[, ...])

DATASET SAMPLE GENERATORS

** todo*

make_blobs ([n_samples, n_features, ...])
Gaussian blobs for clustering.

---

make_classification ([n_samples, ...])
Random n-class classification

---

make_circles ([n_samples, shuffle, ...])
2D circle containing a smaller circle

---

make_friedman1 ([n_samples, ...])
Generate “Friedman #1” regression

---

make_friedman2 ([n_samples, noise, ...])
Generate the “Friedman #2” regression

---

make_friedman3 ([n_samples, noise, ...])
Generate the “Friedman #3” regression

---

make_gaussian_quantiles ([mean, ...])
Generate isotropic Gaussian &labels by quantile

---

make_hastie_10_2 ([n_samples, ...])
Generates data for binary classification

---

make_low_rank_matrix ([n_samples, ...])
Generate mostly low rank matrix with bell-shaped singular values

---

make_moons ([n_samples, shuffle, ...])
Make two interleaving half circles

---

make_multilabel_classification ([...])
Random multilabel classification problem.

---

make_regression ([n_samples, ...])
Random regression problem.

---

make_s_curve ([n_samples, noise, ...])
S curve dataset.

---

make_sparse_coded_signal (n_samples, ...)
Sparse combination of dictionary elements.

---

make_sparse_spd_matrix ([dim, ...])
Sparse symmetric definite positive matrix.

---

make_sparse_uncorrelated ([...])
Random regression with sparse uncorrelated design

---

make_spd_matrix (n_dim[, random_state])
Random symmetric, positive-definite matrix.

---

make_swiss_roll ([n_samples, noise, ...])
Generate a swiss roll dataset.

---

make_biclusters (shape, n_clusters)
Array with constant block diagonal structure for biclustering.

---

make_checkerboard (shape, n_clusters)
for biclustering.

MATRIX DECOMPOSITION / DIMENSIONALITY REDUCTION

(from sklearn.decomposition import x)
** todo*

PCA ([n_components, copy, whiten])
Principal component analysis (PCA)

ProjectedGradientNMF ([...])
Non-Negative matrix factorization by Projected Gradient (NMF)

RandomizedPCA ([n_components, ...])
Principal component analysis (PCA) using randomized SVD

KernelPCA ([n_components, ...])
Kernel Principal component analysis (KPCA)

FactorAnalysis ([n_components, ...])
Factor Analysis (FA)

FastICA ([n_components, ...])

TruncatedSVD ([n_components, ...])
Dimensionality reduction using truncated SVD (aka LSA).

NMF ([n_components, init, ...])
Non-Negative matrix factorization by Projected Gradient (NMF)

SparsePCA ([n_components, ...])
Sparse Principal Components Analysis (SparsePCA)

MiniBatchSparsePCA ([...])
Mini-batch Sparse Principal Components Analysis

SparseCoder (dictionary[, ...])
Sparse coding

DictionaryLearning ([...])
Dictionary learning

MiniBatchDictionaryLearning ([...])
Mini-batch dictionary learning

fastica (X[, n_components, ...])
Perform Fast Independent Component Analysis.

dict_learning (X, n_components, ...)
Solves a dictionary learning matrix factorization problem.

dict_learning_online (X[, ...])
Solves a dictionary learning matrix factorization problem online.

sparse_encode (X, dictionary[, ...])
Sparse coding

DUMMY ESTIMATORS

(from sklearn.dummy import x)
** todo*

DummyClassifier ([strategy, ...])
DummyRegressor ([strategy, constant])
Makes predictions using simple rules.

ENSEMBLE METHODS

(from sklearn.ensemble import x)
** todo*

AdaBoostClassifier ([...])
AdaBoostRegressor([base_estimator, ...])

BaggingClassifier ([base_estimator, ...])
BaggingRegressor ([base_estimator, ...])

ExtraTreesClassifier ([...])
ExtraTreesRegressor ([n_estimators, ...])

GradientBoostingClassifier ([loss, ...])
GradientBoostingRegressor ([loss, ...])

RandomForestClassifier ([...])
RandomTreesEmbedding ([...])
RandomForestRegressor ([...])

FEATURE EXTRACTION

(from sklearn.feature_extraction import x)
** todo*

DictVectorizer ([dtype, ...])
Transforms lists of feature-value maps to vectors.

FeatureHasher ([...])
Implements feature hashing, aka the hashing trick.

FEATURE EXTRACTION (IMAGES)

(from sklearn.feature_extraction.image import x)
** todo*

img_to_graph (img[, ...])
Graph of the pixel-to-pixel gradient connections

grid_to_graph (n_x, n_y)
Graph of the pixel-to-pixel connections

extract_patches_2d (...)
Reshape a 2D image into a collection of patches

reconstruct_from_patches_2d (...)
Reconstruct image from all of its patches.

PatchExtractor ([...])
Extracts patches from a collection of images

FEATURE EXTRACTION (TEXT)

(from sklearn.feature_extraction.text import x)
** todo*

CountVectorizer ([...])
Convert collection of text docs to matrix of token counts

HashingVectorizer ([...])
Convert collection of text docs to matrix of token occurrences

TfidfTransformer ([...])
Transform count matrix to normalized tf or tf-idf

TfidfVectorizer ([...])
Convert collection of raw docs to matrix of TF-IDF features.

FEATURE SELECTION

(from sklearn.feature_selection import x)
** todo*

GenericUnivariateSelect ([...])
Univariate - configurable strategy.

SelectPercentile ([...])
Select features using pctage of highest scores.

SelectKBest ([score_func, k])
Select features using k highest scores.

SelectFpr ([score_func, alpha])
Filter: Select pvalues below alpha using FPR test.

SelectFdr ([score_func, alpha])
Filter: Select pvalues for est false discovery rate

SelectFwe ([score_func, alpha])
Filter: Select p-values using Family-wise error rate

RFE (estimator[, ...])
Feature ranking with recursive feature elimination.
RFECV (estimator[, step, ...])
RFE with CV'd selection of best number of features.

VarianceThreshold ([threshold])
Removes all low-variance features.

chi2 (X, y)
Find chi-squared stat for each class/feature combo.

f_classif (X, y)
Compute Anova F-value for the provided sample

f_regression (X, y[, center])
Univariate linear regression tests

GAUSSIAN PROCESSES

(from sklearn.gaussian_process.correlation_models or regression_models import x)

GaussianProcess ([regr, ...]) -- base class

** correlation models*
absolute_exponential (...)
squared_exponential (...) -- RBF
generalized_exponential (...)
pure_nugget (...) -- spatial independence
cubic (...)
linear (...)

** Zero/1st/2nd-order regression models*
constant (x) -- Zero order polynomial RM.
linear (x) -- 1st order polynomial RM.
quadratic (x) -- 2nd order polynomial RM.

GRID SEARCH

(from sklearn.grid_search import x)

** utilities for fine-tuning estimator parameters.*

GridSearchCV (estimator, param_grid)
Exhaustive search for estimator.

ParameterGrid (param_grid)
Grid of parameters with a discrete #values for each.

ParameterSampler (...[, random_state])
Generates params sampled from given distributions.

RandomizedSearchCV (estimator, ...)
Randomized search on hyper parameters.

ISOTONIC REGRESSION

(from sklearn.isotonic import x)
** todo*

IsotonicRegression ([y_min, y_max, ...])
Isotonic regression model.

isotonic_regression (y[, ...])
Solve the isotonic regression model:

check_increasing (x, y)
Determine whether y is monotonically correlated with x.

KERNEL APPROXIMATION

(from sklearn.kernel_approximation import x)
** todo*

AdditiveChi2Sampler ([...])
Approximate feature map for additive chi2 kernel

Nystroem ([kernel, ...])
Approximate kernel map using subset of training data.

RBFSampler ([gamma, ...])
Approximates feature map of an RBF kernel by Monte Carlo approximation of its Fourier transform.

SkewedChi2Sampler ([...])
Approximates feature map of the “skewed chi-squared” kernel by Monte Carlo approximation of its Fourier transform.

DISCRIMINANT ANALYSIS

(from sklearn.lda or .qda import x)
** todo*

lda.LDA ([n_components, priors])
Linear Discriminant Analysis (LDA)

qda.QDA ([priors, reg_param])
Quadratic Discriminant Analysis (QDA)

MANIFOLD LEARNING

(from sklearn.manifold import x)
** todo*

LocallyLinearEmbedding ([...])
Locally Linear Embedding

Isomap ([n_neighbors, n_components, ...])
Isomap Embedding

MDS ([n_components, metric, n_init, ...])
Multidimensional scaling

SpectralEmbedding ([n_components, ...])
For non-linear dimensionality reduction

TSNE ([n_components, perplexity, ...])
t-distributed Stochastic Neighbor Embedding.

locally_linear_embedding (X, ...)
LLE analysis on the data

spectral_embedding (adjacency[, ...])
Project sample on 1st eigenvectors of graph Laplacian.

METRICS/SCORING (CLASSIFICATION)

(from sklearn.metrics import x)
** todo*

accuracy_score (y_true, y_pred[, ...])

auc (x, y[, reorder])
Area Under Curve - trapezoidal rule

average_precision_score (y_true, y_score)

classification_report (y_true, y_pred)
text summary report

confusion_matrix (y_true, y_pred[, ...])

precision_recall_curve (y_true, ...)
precision_recall_fscore_support (...)
precision_score (y_true, y_pred[, ...])
recall_score (y_true, y_pred[, ...])

---

f1_score (y_true, y_pred[, labels, ...])
aka balanced F-score or F-measure

---

fbeta_score (y_true, y_pred, beta[, ...])

---

hamming_loss (y_true, y_pred[, classes])

---

hinge_loss (y_true, pred_decision[, ...])

---

jaccard_similarity_score (y_true, y_pred)

---

log_loss (y_true, y_pred[, eps, ...])
aka logistic loss, aka cross-entropy loss

---

matthews_corrcoef (y_true, y_pred)
for binary classes

---

roc_auc_score (y_true, y_score[, ...])
Compute Area Under the Curve (AUC) from prediction scores

---

roc_curve (y_true, y_score[, ...])
Receiver operating characteristic (ROC)

---

zero_one_loss (y_true, y_pred[, ...])
Zero-one classification loss.

METRICS/SCORING (REGRESSION)

(from sklearn.metrics import x)
** todo*

explained_variance_score (y_true, y_pred)

---

mean_absolute_error (y_true, y_pred)

---

mean_squared_error (y_true, y_pred[, ...])
MSE

---

r2_score (y_true, y_pred[, sample_weight])
R2 = coefficient of determination

METRICS/SCORING (CLUSTERING)

(from sklearn.metrics import x)
** supervised: "ground truth" class values are used.*
** unsupervised: measures quality of model itself.*

adjusted_mutual_info_score (...)
Adjusted Mutual info between two clusterings

---

adjusted_rand_score (labels_true, ...)
Rand index adjusted for chance

---

completeness_score (labels_true, ...)
Completeness of cluster labels given ground truth

---

homogeneity_completeness_v_measure (...)
homogeneity, completeness, and V-Measure scores

---

homogeneity_score (labels_true, ...)
Homogeneity of cluster labels given a ground truth

---

mutual_info_score (labels_true, ...)
Mutual info between two clusters
normalized_mutual_info_score (...)
Mutual info between two clusters, normalized

---

silhouette_score (X, labels[, ...])
Mean Silhouette Coefficient of all samples.
silhouette_samples (X, labels[, metric])
Silhouette Coefficient for each sample.

---

v_measure_score (labels_true, labels_pred)
V-measure cluster labeling given a ground truth.

METRICS/SCORING (PAIRWISE)

(from sklearn.metrics.pairwise import x)
** distance metrics: if d(a,b) < d(a,c), then a&b are "more similar". Exact likeness: d=0.*
** Kernels = measures of similarity.*

additive_chi2_kernel (X[, Y])
Additive chi-squared kernel between observations in X and Y

---

chi2_kernel (X[, Y, gamma])
Exponential chi-squared kernel X and Y.

---

distance_metrics ()
Valid metrics for pairwise_distances.

euclidean_distances (X[, Y, ...])
Considering the rows of X (and Y=X) as vectors, find distance matrix between each pair of vectors.

kernel_metrics ()
Valid metrics for pairwise_kernels

manhattan_distances (X[, Y, ...])
L1 distances between vectors in X and Y.

pairwise_distances (X[, Y, ...])
Distance matrix from vector array X and optional Y.

pairwise_kernels (X[, Y, ...])
Kernel between arrays X and optional array Y.

polynomial_kernel (X[, Y, ...])
Polynomial kernel between X and Y

rbf_kernel (X[, Y, gamma])
Gaussian kernel between X & Y

GAUSSIAN MIXTURES

(from sklearn.mixture import x)
** todo*

GMM ([n_components, covariance_type, ...])
Gaussian Mixture Model

DPGMM ([n_components, ...])
Variational Inference for an Infinite GMM

VBGMM ([n_components, ...])
Variational inference for a GMM

MULTICLASS/LABEL CLASSIFICATION

(from sklearn.multiclass import x)
** one-vs-rest, one-vs-all, one-vs-one, ECC outputs. meta-estimators (base estimator required during construction.)*

OneVsRestClassifier (estimator[, ...])
fit_ovr (estimator, X, y[, n_jobs])
predict_ovr (estimators, ...)

OneVsOneClassifier (estimator[, ...])
fit_ovo (estimator, X, y[, n_jobs])
predict_ovo (estimators, classes, X)

OutputCodeClassifier (estimator[, ...])
fit_ecoc (estimator, X, y[, ...])
predict_ecoc (estimators, classes, ...)

NAIVE BAYES

(from sklearn.naive_bayes import x)
** supervised learning methods with strong (naive) feature independence assumptions*

GaussianNB
Gaussian Naive Bayes (GaussianNB)

MultinomialNB([alpha, ...])
NB classifier for multinomial models

BernoulliNB ([alpha, binarize, ...])
NB classifier for multivariate Bernoulli models.

NEAREST NEIGHBORS

(from sklearn.neighbors import x)
** todo*

NearestNeighbors ([n_neighbors, ...])
Unsupervised learner for neighbor searches.

KNeighborsClassifier ([...])
KNeighborsRegressor ([n_neighbors, ...])
Classification, Regression based on k-nearest vote.

RadiusNeighborsClassifier ([...])
RadiusNeighborsRegressor ([radius, ...])
Classification, Regression based on neighbors within a fixed radius.

NearestCentroid ([metric, ...])
Nearest centroid classifier.

BallTree
BallTree for fast generalized N-point problems
KDTree
KDTree for fast generalized N-point problems

DistanceMetric
DistanceMetric class

KernelDensity ([bandwidth, ...])
Kernel Density Estimation

kneighbors_graph (X, n_neighbors[, ...])
Computes the (weighted) graph of k-Neighbors for points in X

radius_neighbors_graph (X, radius)
Computes the (weighted) graph of Neighbors for points in X

LEARNING CURVE EVALUATION

(from sklearn.learning_curve import x)
** utilities to evaluate models w.r.t. a variable.*

learning_curve (estimator, X, y)

validation_curve (estimator, ...)

CROSS DECOMPOSITION

(from sklearn.cross_decomposition import x)
** todo*

PLSRegression ([...])
PLS regression

PLSCanonical([...])
PLS-C2A algorithm

CCA ([n_components, ...])
CCA Canonical Correlation Analysis.

PLSSVD ([n_components, ...])
Partial Least Square SVD

PIPELINES

(from sklearn.pipeline import x)
** tools to build composite (chained) estimators*

Pipeline (steps)
Pipeline of transforms with a final estimator.

FeatureUnion (transformer_list[, ...])
Concatenate results of multiple transformers.

make_pipeline (*steps)
Construct a Pipeline from given estimators.

make_union (*transformers)
Construct a FeatureUnion from given transformers.

DATA PREP

(from sklearn.preprocessing import x)
** scaling, centering, normalization, binarization, imputation methods*

Binarizer ([threshold, copy])
Binarize data (set values to 0 or 1) via threshold
binarize (X[, threshold, copy])
Boolean thresholding of array or scipy.sparse matrix

Imputer ([missing_values, ...])
Fills in missing values.

KernelCenterer
Center a kernel matrix

LabelBinarizer ([neg_label, ...])
Binarize labels in a one-vs-all fashion
label_binarize (y, classes[, ...])
Binarize labels in a one-vs-all fashion

LabelEncoder
Encode labels; value between 0 and n_classes-1.

MultiLabelBinarizer ([classes, ...])
Transform between iterable of iterables and a multilabel format

MinMaxScaler ([feature_range, copy])
Scaling each feature to a given range.

Normalizer ([norm, copy])
Scales samples individually to unit norm
normalize (X[, norm, axis, copy])
Normalize a dataset along any axis

OneHotEncoder ([n_values, ...])
Encode integer category features using one-of-K scheme.

StandardScaler ([copy, ...])
Standardize features by removing the mean and scaling to unit variance
scale (X[, axis, with_mean, ...])
Standardize a dataset along any axis

PolynomialFeatures ([degree, ...])
Generate polynomial and interaction features.

add_dummy_feature (X[, value])
Augment dataset with an additional dummy feature.

RANDOM PROJECTION

(from sklearn.random_projection import x)
** Reduce data dimensionality by trading accuracy for dataset size. Pairwise distances are preserved.*

GaussianRandomProjection ([...])
Gaussian random projection

SparseRandomProjection ([...])
Sparse random projection

johnson_lindenstrauss_min_dim (...)
Find 'safe' #components for randomly projection

SEMI-SUPERVISED LEARNING

(from sklearn.semi_supervised import x)
** Use case: small amounts of labeled data, large amounts fo unlabeled data.*

LabelPropagation ([kernel, ...])
Label Propagation classifier

LabelSpreading ([kernel, ...])
LabelSpreading model

SVM ESTIMATORS

SVC([C, kernel, degree, gamma, coef0, ...])
C-Support Vector Classification.

LinearSVC([penalty, loss, dual, tol, C, ...])
Linear Support Vector Classification.

uSVC([nu, kernel, degree, gamma, ...])
Nu-Support Vector Classification.

SVR([kernel, degree, gamma, coef0, tol, ...])
Epsilon-Support Vector Regression.

LinearSVR([epsilon, tol, C, loss, ...])
Linear Support Vector Regression.

NuSVR([nu, C, kernel, degree, gamma, ...])
Nu Support Vector Regression.

OneClassSVM([kernel, degree, gamma, ...])
Unsupervised Outlier Detection.

l1_min_c(X, y[, loss, fit_intercept, ...])
Return lowest bound for C such that C in (l1_min_C, infinity) the model is guaranteed not to be empty.

svm.libsvm.fit
Train the model using libsvm (low-level method)

svm.libsvm.decision_function
Predict margin (libsvm name: predict_values)

svm.libsvm.predict
Predict target values of X given a model

svm.libsvm.predict_proba
Predict probabilities

svm.libsvm.cross_validation
Binding of the cross-validation routine

DECISION TREES

tree.DecisionTreeClassifier([criterion, ...])
tree.DecisionTreeRegressor([criterion, ...])
Basic DT classifier & regressor

tree.ExtraTreeClassifier([criterion, ...])
tree.ExtraTreeRegressor([criterion, ...])
Randomized DT classifier & regressor

tree.export_graphviz
Export a decision tree in DOT format.

UTILITIES

check_random_state(seed)
Turn seed into a np.random.RandomState instance

---

estimator_checks.check_estimator(Estimator)
Check if estimator adheres to scikit-learn conventions.

---

resample(*arrays, **options)
Resample arrays or sparse matrices in a consistent way

---

shuffle(*arrays, **options)
Shuffle arrays or sparse matrices in a consistent way