Scikit-learn

MACHINE LEARNING TOOLS

Python

Java

.net

We need scalable Machine Learning/Data Mining Algorithms

Java, Scala

MLlib

Python, Scala

Statistical Analysis

C, FORTRAN Languages

Machine Learning in Python

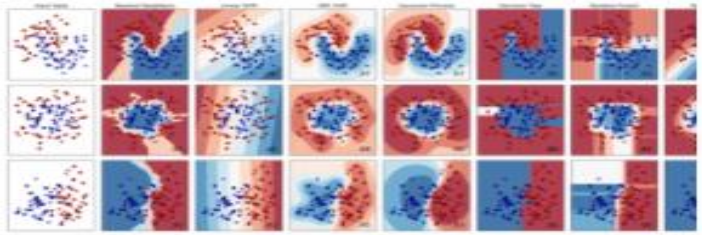
Getting Started | Release Highlights for 0.23 | GitHub

- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

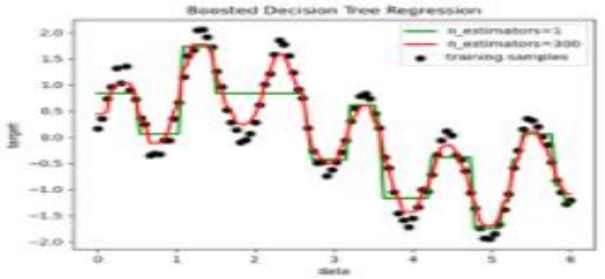


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

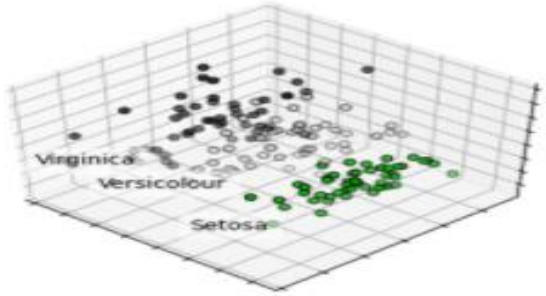
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

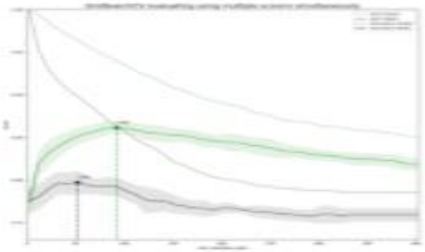
Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

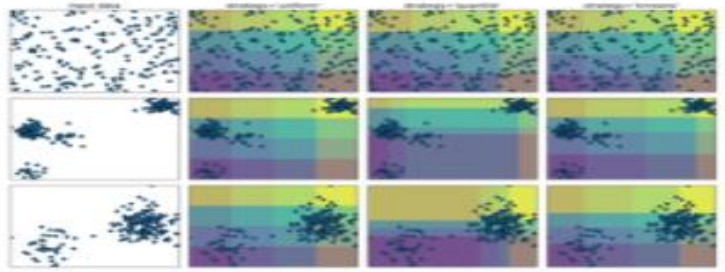
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Installing the latest release

Operating System Windows macOS Linux

Packager pip conda

Use conda environment

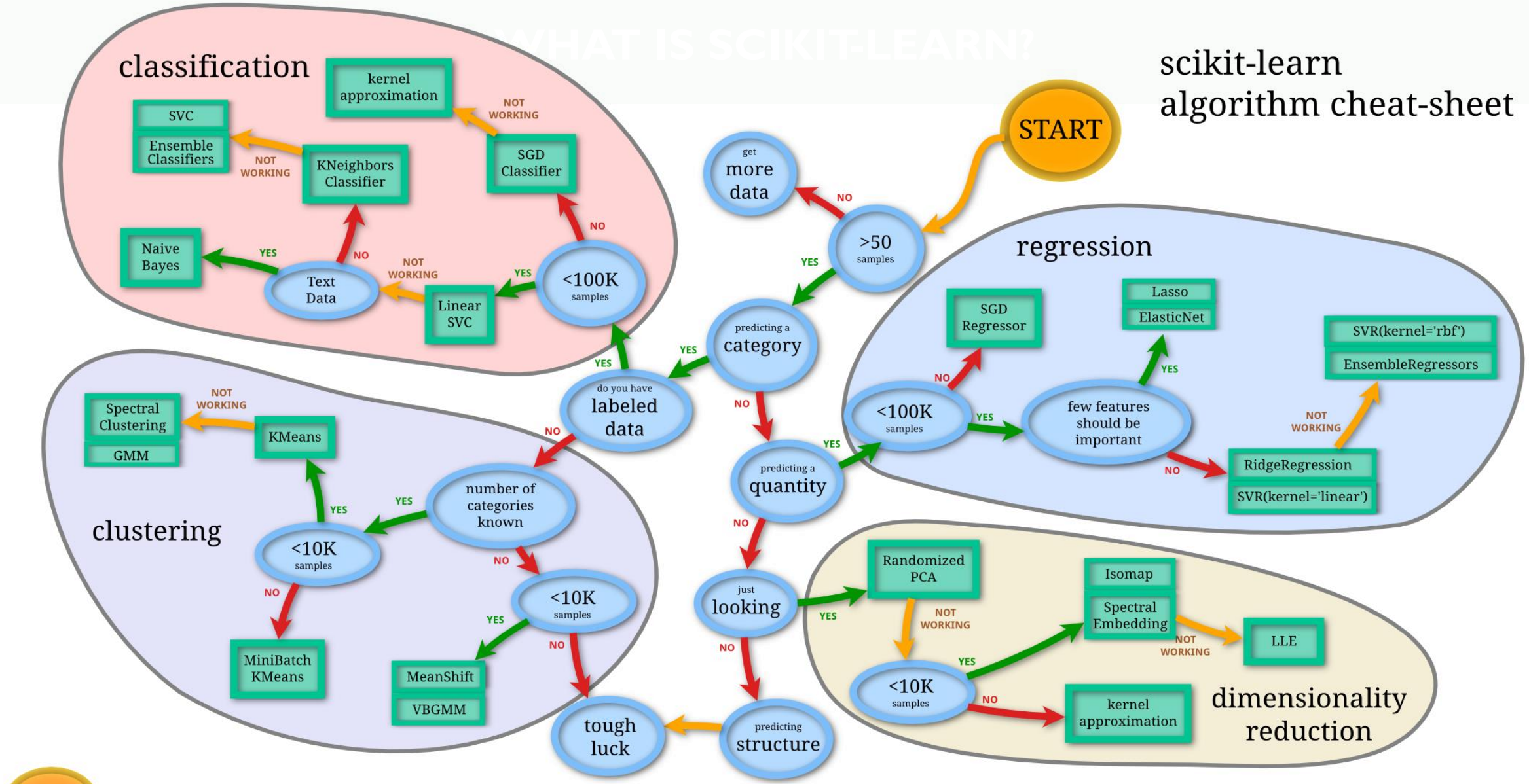
Install conda (no administrator permission required).
Then run:

```
$ conda install scikit-learn
```

In order to check your installation you can use

```
$ conda list scikit-learn # to see which scikit-learn version is installed
$ conda list # to see all packages installed in the active conda environment
$ python -c "import sklearn; sklearn.show_versions()"
```

Note that in order to avoid potential conflicts with other packages it is strongly recommended to use a virtual environment, e.g. python3 `virtualenv` (see python3 virtualenv documentation) or conda environments.



Back

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Prev Up Next

scikit-learn 0.23.2
Other versions

Please cite us if you use the software.

Welcome to scikit-learn
scikit-learn Tutorials
Getting Started
User Guide
1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection
5. Visualizations
6. Dataset transformations
7. Dataset loading utilities
8. Computing with scikit-learn
Glossary of Common Terms and API Elements
Examples
API Reference
Developer's Guide

User Guide

1. Supervised learning

Toggle Menu

Sklearn APIs

(Note: Refer latest API documentation)

sklearn.base: Base classes and utility functions

Base classes for all estimators.

Used for VotingClassifier

Base classes

base.BaseEstimator	Base class for all estimators in scikit-learn
base.BiclusterMixin	Mixin class for all bicluster estimators in scikit-learn
base.ClassifierMixin	Mixin class for all classifiers in scikit-learn.
base.ClusterMixin	Mixin class for all cluster estimators in scikit-learn.
base.DensityMixin	Mixin class for all density estimators in scikit-learn.
base.RegressorMixin	Mixin class for all regression estimators in scikit-learn.
base.TransformerMixin	Mixin class for all transformers in scikit-learn.
feature_selection.SelectorMixin	Transformer mixin that performs feature selection given a support mask

Functions

base.clone(estimator, *[, safe])	Constructs a new estimator with the same parameters.
base.is_classifier(estimator)	Return True if the given estimator is (probably) a classifier.
base.is_regressor(estimator)	Return True if the given estimator is (probably) a regressor.
config_context(**new_config)	Context manager for global scikit-learn configuration
get_config()	Retrieve current values for configuration set by set_config
set_config([assume_finite, working_memory, ...])	Set global scikit-learn configuration
show_versions()	Print useful debugging information"

sklearn.calibration: Probability Calibration

Calibration of predicted probabilities.

User guide: See the Probability calibration section for further details.

calibration.CalibratedClassifierCV([...])	Probability calibration with isotonic regression or logistic regression.
calibration.calibration_curve(y_true, y_prob, *)	Compute true and predicted probabilities for a calibration curve.

sklearn.datasets : Datasets

The sklearn.datasets module includes utilities to load datasets, including methods to load and fetch popular reference datasets. It also features some artificial data generators.

User guide: See the Dataset loading utilities section for further details.

Loaders

datasets.clear_data_home ([data_home])	Delete all the content of the data home cache.
datasets.dump_svmlight_file (X, y, f[, ...])	Dump the dataset in svmlight / libsvm file format.
datasets.fetch_20newsgroups ([data_home, ...])	Load the filenames and data from the 20 newsgroups dataset (classification).
datasets.fetch_20newsgroups_vectorized ([...])	Load the 20 newsgroups dataset and vectorize it into token counts (classification).
datasets.fetch_california_housing ([...])	Load the California housing dataset (regression).
datasets.fetch_covtype ([data_home, ...])	Load the covertype dataset (classification).
datasets.fetch_kddcup99 ([subset, data_home, ...])	Load the kddcup99 dataset (classification).
datasets.fetch_lfw_pairs ([subset, ...])	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
datasets.fetch_lfw_people ([data_home, ...])	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
datasets.fetch_olivetti_faces ([data_home, ...])	Load the Olivetti faces data-set from AT&T (classification).
datasets.fetch_openml ([name, version, ...])	Fetch dataset from openml by name or dataset id.
datasets.fetch_rcv1 ([data_home, subset, ...])	Load the RCV1 multilabel dataset (classification).

sklearn.impute : Impute

Transformers for missing value imputation

User guide: See the Imputation of missing values section for further details.

impute.SimpleImputer ([missing_values, ...])	Imputation transformer for completing missing values.
impute.IterativeImputer ([estimator, ...])	Multivariate imputer that estimates each feature from all the others.
impute.MissingIndicator ([missing_values, ...])	Binary indicators for missing values.

`sklearn.preprocessing` : Preprocessing and Normalization

The `sklearn.preprocessing` module includes scaling, centering, normalization, binarization and imputation methods.

User guide: See the Preprocessing data section for further details.

`preprocessing.Binarizer` ([threshold, copy])	Binarize data (set feature values to 0 or 1) according to a threshold
`preprocessing.FunctionTransformer` ([func, ...])	Constructs a transformer from an arbitrary callable.
`preprocessing.KBinsDiscretizer` ([n_bins, ...])	Bin continuous data into intervals.
`preprocessing.KernelCenterer` ()	Center a kernel matrix
`preprocessing.LabelBinarizer` ([neg_label, ...])	Binarize labels in a one-vs-all fashion
`preprocessing.LabelEncoder`	Encode labels with value between 0 and n_classes-1.
`preprocessing.MultiLabelBinarizer` ([classes, ...])	Transform between iterable of iterables and a multilabel format
`preprocessing.MaxAbsScaler` ([copy])	Scale each feature by its maximum absolute value.
`preprocessing.MinMaxScaler` ([feature_range, copy])	Transforms features by scaling each feature to a given range.
`preprocessing.Normalizer` ([norm, copy])	Normalize samples individually to unit norm.

sklearn.discriminant_analysis : Discriminant Analysis

Linear Discriminant Analysis and Quadratic Discriminant Analysis

User guide: See the Linear and Quadratic Discriminant Analysis section for further details.

discriminant_analysis.LinearDiscriminantAnalysis ([...])	Linear Discriminant Analysis
discriminant_analysis.QuadraticDiscriminantAnalysis ([...])	Quadratic Discriminant Analysis

sklearn.cluster : Clustering

The sklearn.cluster module gathers popular unsupervised clustering algorithms.

User guide: See the Clustering section for further details.

Classes

cluster.AffinityPropagation([damping, ...])	Perform Affinity Propagation Clustering of data.
cluster.AgglomerativeClustering([...])	Agglomerative Clustering
cluster.Birch([threshold, branching_factor, ...])	Implements the Birch clustering algorithm.
cluster.DBSCAN([eps, min_samples, metric, ...])	Perform DBSCAN clustering from vector array or distance matrix.
cluster.OPTICS([min_samples, max_eps, ...])	Estimate clustering structure from vector array
cluster.FeatureAgglomeration([n_clusters, ...])	Agglomerate features.
cluster.KMeans([n_clusters, init, n_init, ...])	K-Means clustering
cluster.MiniBatchKMeans([n_clusters, init, ...])	Mini-Batch K-Means clustering
cluster.MeanShift([bandwidth, seeds, ...])	Mean shift clustering using a flat kernel.
cluster.SpectralClustering([n_clusters, ...])	Apply clustering to a projection of the normalized Laplacian.

Functions

cluster.affinity_propagation(S[, ...])	Perform Affinity Propagation Clustering of data
cluster.cluster_optics_dbscan(reachability, ...)	Performs DBSCAN extraction for an arbitrary epsilon.
cluster.cluster_optics_xi(reachability, ...)	Automatically extract clusters according to the Xi-steep method.
cluster.compute_optics_graph(X, min_samples, ...)	Computes the OPTICS reachability graph.
cluster.dbscan(X[, eps, min_samples, ...])	Perform DBSCAN clustering from vector array or distance matrix.
cluster.estimate_bandwidth(X[, quantile, ...])	Estimate the bandwidth to use with the mean-shift algorithm.
cluster.k_means(X, n_clusters[, ...])	K-means clustering algorithm.
cluster.mean_shift(X[, bandwidth, seeds, ...])	Perform mean shift clustering of data using a flat kernel.
cluster.spectral_clustering(affinity[, ...])	Apply clustering to a projection of the normalized Laplacian.
cluster.ward_tree(X[, connectivity, ...])	Ward clustering based on a Feature matrix.

`sklearn.naive_bayes` : Naive Bayes

The `sklearn.naive_bayes` module implements Naive Bayes algorithms. These are supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.

User guide: See the Naive Bayes section for further details.

`naive_bayes.BernoulliNB` ([alpha, binarize, ...])	Naive Bayes classifier for multivariate Bernoulli models.
`naive_bayes.GaussianNB` ([priors, var_smoothing])	Gaussian Naive Bayes (GaussianNB)
`naive_bayes.MultinomialNB` ([alpha, ...])	Naive Bayes classifier for multinomial models
`naive_bayes.ComplementNB` ([alpha, fit_prior, ...])	The Complement Naive Bayes classifier described in Rennie et al.

sklearn.neighbors : Nearest Neighbors

The sklearn.neighbors module implements the k-nearest neighbors algorithm.

User guide: See the Nearest Neighbors section for further details.

neighbors.BallTree	BallTree for fast generalized N-point problems
neighbors.DistanceMetric	DistanceMetric class
neighbors.KDTree	KDTree for fast generalized N-point problems
neighbors.KernelDensity ([bandwidth, ...])	Kernel Density Estimation
neighbors.KNeighborsClassifier ([...])	Classifier implementing the k-nearest neighbors vote.
neighbors.KNeighborsRegressor ([n_neighbors, ...])	Regression based on k-nearest neighbors.
neighbors.LocalOutlierFactor ([n_neighbors, ...])	Unsupervised Outlier Detection using Local Outlier Factor (LOF)
neighbors.RadiusNeighborsClassifier ([...])	Classifier implementing a vote among neighbors within a given radius
neighbors.RadiusNeighborsRegressor ([radius, ...])	Regression based on neighbors within a fixed radius.
neighbors.NearestCentroid ([metric, ...])	Nearest centroid classifier.
neighbors.NearestNeighbors ([n_neighbors, ...])	Unsupervised learner for implementing

`sklearn.svm`: Support Vector Machines

The `sklearn.svm` module includes Support Vector Machine algorithms.

User guide: See the Support Vector Machines section for further details.

Estimators

`svm.LinearSVC` ([penalty, loss, dual, tol, C, ...])	Linear Support Vector Classification.
`svm.LinearSVR` ([epsilon, tol, C, loss, ...])	Linear Support Vector Regression.
`svm.NuSVC` ([nu, kernel, degree, gamma, ...])	Nu-Support Vector Classification.
`svm.NuSVR` ([nu, C, kernel, degree, gamma, ...])	Nu Support Vector Regression.
`svm.OneClassSVM` ([kernel, degree, gamma, ...])	Unsupervised Outlier Detection.
`svm.SVC` ([C, kernel, degree, gamma, coef0, ...])	C-Support Vector Classification.
`svm.SVR` ([kernel, degree, gamma, coef0, tol, ...])	Epsilon-Support Vector Regression.
`svm.l1_min_c` (X, y[, loss, fit_intercept, ...])	Return the lowest bound for C such that for C in (l1_min_C, infinity) the model is guaranteed not to be empty.

`sklearn.tree`: Decision Trees

The `sklearn.tree` module includes decision tree-based models for classification and regression.

User guide: See the Decision Trees section for further details.

`tree.DecisionTreeClassifier` ([criterion, ...])	A decision tree classifier.
`tree.DecisionTreeRegressor` ([criterion, ...])	A decision tree regressor.
`tree.ExtraTreeClassifier` ([criterion, ...])	An extremely randomized tree classifier.
`tree.ExtraTreeRegressor` ([criterion, ...])	An extremely randomized tree regressor.
`tree.export_graphviz` (decision_tree[, ...])	Export a decision tree in DOT format.
`tree.plot_tree` (decision_tree[, max_depth, ...])	Plot a decision tree.
`tree.export_text` (decision_tree[, ...])	Build a text report showing the rules of a decision tree.

sklearn.ensemble : Ensemble Methods

The sklearn.ensemble module includes ensemble-based methods for classification, regression and anomaly detection.

User guide: See the Ensemble methods section for further details.

ensemble.AdaBoostClassifier ([...])	An AdaBoost classifier.
ensemble.AdaBoostRegressor ([base_estimator, ...])	An AdaBoost regressor.
ensemble.BaggingClassifier ([base_estimator, ...])	A Bagging classifier.
ensemble.BaggingRegressor ([base_estimator, ...])	A Bagging regressor.
ensemble.ExtraTreesClassifier ([...])	An extra-trees classifier.
ensemble.ExtraTreesRegressor ([n_estimators, ...])	An extra-trees regressor.
ensemble.GradientBoostingClassifier ([loss, ...])	Gradient Boosting for classification.
ensemble.GradientBoostingRegressor ([loss, ...])	Gradient Boosting for regression.
ensemble.IsolationForest ([n_estimators, ...])	Isolation Forest Algorithm
ensemble.RandomForestClassifier ([...])	A random forest classifier.
ensemble.RandomForestRegressor ([...])	A random forest regressor.
ensemble.RandomTreesEmbedding ([...])	An ensemble of totally random trees.
ensemble.VotingClassifier (estimators[, ...])	Soft Voting/Majority Rule classifier for unfitted estimators.
ensemble.VotingRegressor (estimators[, ...])	Prediction voting regressor for unfitted estimators.
ensemble.HistGradientBoostingRegressor ([...])	Histogram-based Gradient Boosting Regression Tree.
ensemble.HistGradientBoostingClassifier ([...])	Histogram-based Gradient Boosting Classification Tree.

sklearn.metrics : Metrics

See the Model evaluation: quantifying the quality of predictions section and the Pairwise metrics, Affinities and Kernels section of the user guide for further details.

The `sklearn.metrics` module includes score functions, performance metrics and pairwise metrics and distance computations.

Classification metrics

See the Classification metrics section of the user guide for further details.

`metrics.accuracy_score` (y_true, y_pred[, …])	Accuracy classification score.
`metrics.auc` (x, y[, reorder])	Compute Area Under the Curve (AUC) using the trapezoidal rule
`metrics.average_precision_score` (y_true, y_score)	Compute average precision (AP) from prediction scores
`metrics.balanced_accuracy_score` (y_true, y_pred)	Compute the balanced accuracy
`metrics.brier_score_loss` (y_true, y_prob[, …])	Compute the Brier score.
`metrics.classification_report` (y_true, y_pred)	Build a text report showing the main classification metrics
`metrics.cohen_kappa_score` (y1, y2[, labels, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.

sklearn.metrics : Metrics

Classification metrics

metrics.confusion_matrix (y_true, y_pred[, ...])	Compute confusion matrix to evaluate the accuracy of a classification
metrics.f1_score (y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score (y_true, y_pred, beta[, ...])	Compute the F-beta score
metrics.hamming_loss (y_true, y_pred[, ...])	Compute the average Hamming loss.
metrics.hinge_loss (y_true, pred_decision[, ...])	Average hinge loss (non-regularized)
metrics.jaccard_score (y_true, y_pred[, ...])	Jaccard similarity coefficient score
metrics.log_loss (y_true, y_pred[, eps, ...])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef (y_true, y_pred[, ...])	Compute the Matthews correlation coefficient (MCC)
metrics.multilabel_confusion_matrix (y_true, ...)	Compute a confusion matrix for each class or sample
metrics.precision_recall_curve (y_true, ...)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support (...)	Compute precision, recall, F-measure and support for each class
metrics.precision_score (y_true, y_pred[, ...])	Compute the precision
metrics.recall_score (y_true, y_pred[, ...])	Compute the recall
metrics.roc_auc_score (y_true, y_score[, ...])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

sklearn.metrics : Metrics

Regression metrics

See the Regression metrics section of the user guide for further details.

metrics.explained_variance_score (y_true, y_pred)	Explained variance regression score function
metrics.max_error (y_true, y_pred)	max_error metric calculates the maximum residual error.
metrics.mean_absolute_error (y_true, y_pred)	Mean absolute error regression loss
metrics.mean_squared_error (y_true, y_pred[, …])	Mean squared error regression loss
metrics.mean_squared_log_error (y_true, y_pred)	Mean squared logarithmic error regression loss
metrics.median_absolute_error (y_true, y_pred)	Median absolute error regression loss
metrics.r2_score (y_true, y_pred[, …])	R^2 (coefficient of determination) regression score function.

sklearn.model_selection: Model Selection

Splitter Classes

model_selection.GroupKFold ([n_splits])	K-fold iterator variant with non-overlapping groups.
model_selection.GroupShuffleSplit ([…])	Shuffle-Group(s)-Out cross-validation iterator
model_selection.KFold ([n_splits, shuffle, …])	K-Folds cross-validator
model_selection.LeaveOneGroupOut	Leave One Group Out cross-validator
model_selection.LeavePGroupsOut (n_groups)	Leave P Group(s) Out cross-validator
model_selection.LeaveOneOut	Leave-One-Out cross-validator
model_selection.LeavePOut (p)	Leave-P-Out cross-validator
model_selection.PredefinedSplit (test_fold)	Predefined split cross-validator
model_selection.RepeatedKFold ([n_splits, …])	Repeated K-Fold cross validator.
model_selection.RepeatedStratifiedKFold ([…])	Repeated Stratified K-Fold cross validator.
model_selection.ShuffleSplit ([n_splits, …])	Random permutation cross-validator
model_selection.StratifiedKFold ([n_splits, …])	Stratified K-Folds cross-validator
model_selection.StratifiedShuffleSplit ([…])	Stratified ShuffleSplit cross-validator
model_selection.TimeSeriesSplit ([n_splits, …])	Time Series cross-validator

sklearn.model_selection: Model Selection

Splitter Functions

model_selection.check_cv([cv, y, classifier])	Input checker utility for building a cross-validator
model_selection.train_test_split(*arrays, …)	Split arrays or matrices into random train and test subsets

Hyper-parameter optimizers

model_selection.GridSearchCV(estimator, …)	Exhaustive search over specified parameter values for an estimator.
model_selection.ParameterGrid(param_grid)	Grid of parameters with a discrete number of values for each.
model_selection.ParameterSampler(…[, …])	Generator on parameters sampled from given distributions.
model_selection.RandomizedSearchCV(…[, …])	Randomized search on hyper parameters.
model_selection.fit_grid_point(X, y, …[, …])	Run fit on one set of parameters.

sklearn.model_selection: Model Selection

Model validation

model_selection.cross_validate (estimator, X)	Evaluate metric(s) by cross-validation and also record fit/score times.
model_selection.cross_val_predict (estimator, X)	Generate cross-validated estimates for each input data point
model_selection.cross_val_score (estimator, X)	Evaluate a score by cross-validation
model_selection.learning_curve (estimator, X, y)	Learning curve.
model_selection.permutation_test_score (…)	Evaluate the significance of a cross-validated score with permutations
model_selection.validation_curve (estimator, …)	Validation curve.

sklearn.pipeline : Pipeline

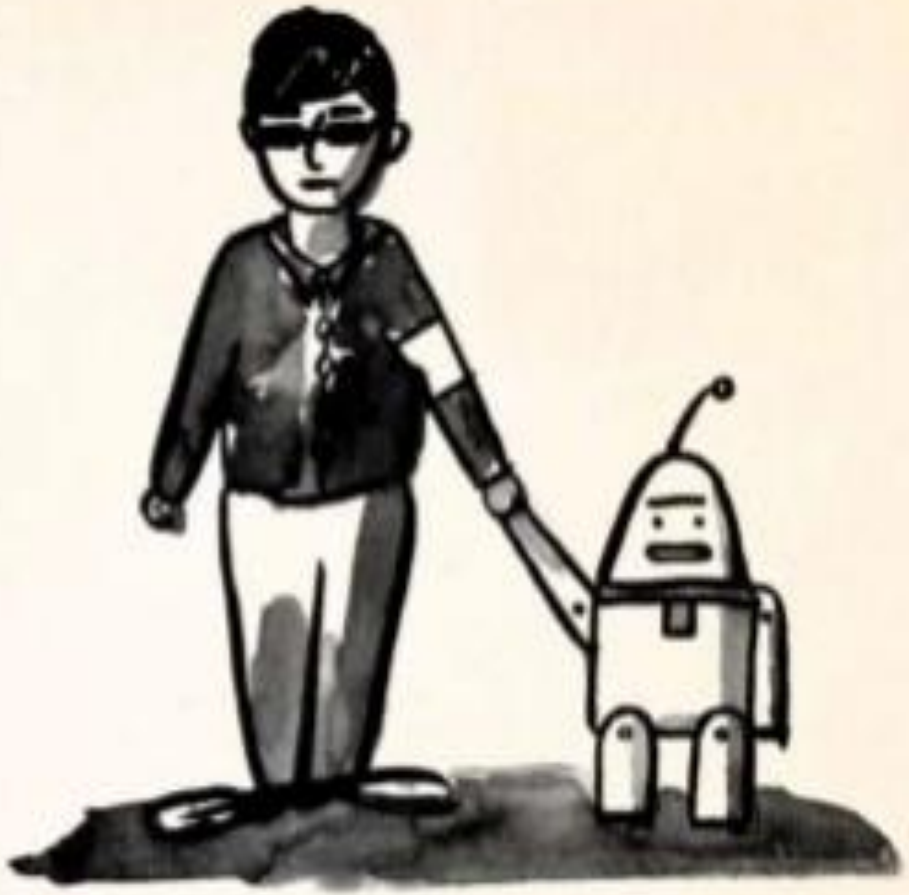
The sklearn.pipeline module implements utilities to build a composite estimator, as a chain of transforms and estimators.

pipeline.FeatureUnion (transformer_list[, ...])	Concatenates results of multiple transformer objects.
pipeline.Pipeline (steps[, memory, verbose])	Pipeline of transforms with a final estimator.
pipeline.make_pipeline (*steps, **kwargs)	Construct a Pipeline from the given estimators.
pipeline.make_union (*transformers, **kwargs)	Construct a FeatureUnion from the given transformers.

An introduction to machine learning with scikit-learn

- Machine learning: the problem setting
- Loading an example dataset
- Learning and predicting
- Model persistence
- Conventions

Refer: https://scikit-learn.org/stable/tutorial/index.html



Thank You!

QUESTIONS?