sklearn.model_selection.train_test_split

sklearn.model_selection.train_test_split(**arrays*, *test_size=None*, *train_size=None*, *random_state=None*, *shuffle=True*, *stratify=None*) [source]

Split arrays or matrices into random train and test subsets.

Quick utility that wraps input validation and `next(ShuffleSplit().split(X, y))` and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

Read more in the User Guide.

Parameters:

arrays : *sequence of indexables with same length / shape[0]
Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*
If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*
Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*
Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Returns:

splitting : *list, length=2 * len(arrays)*
List containing train-test split of inputs.

New in version 0.16: If the input is sparse, the output will be a `scipy.sparse.csr_matrix`. Else, output type is the same as the input type.

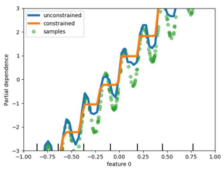
Examples

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]
```

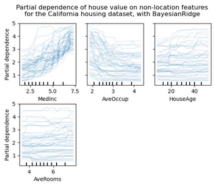
```
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]
```

```
>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```

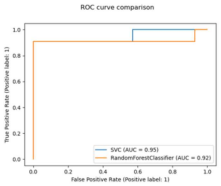
Examples using `sklearn.model_selection.train_test_split`



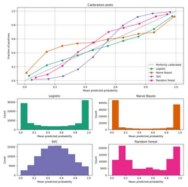
Release Highlights for scikit-learn 0.23



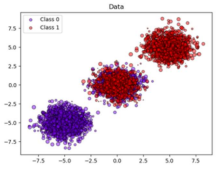
Release Highlights for scikit-learn 0.24



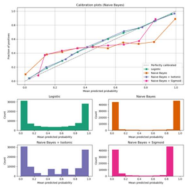
Release Highlights for scikit-learn 0.22



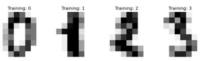
Comparison of Calibration of Classifiers



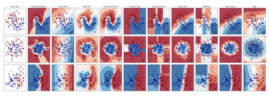
Probability calibration of classifiers



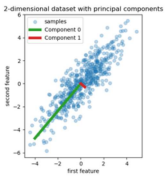
Probability Calibration curves



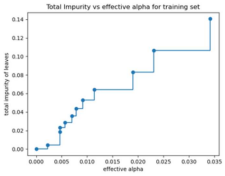
Recognizing hand-written digits



Classifier comparison

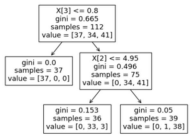


Principal Component Regression vs Partial Least Squares Regression

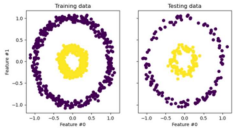


Post pruning decision trees with cost complexity pruning

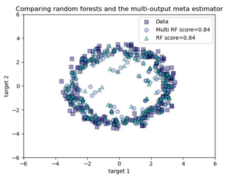
Toggle Menu



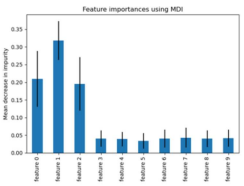
[Understanding the decision tree structure]



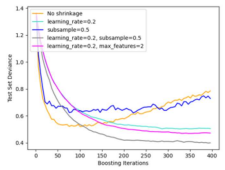
[Kernel PCA]



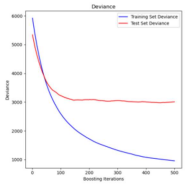
[Comparing random forests and the multi-output meta estimator]



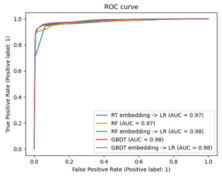
[Feature importances with a forest of trees]



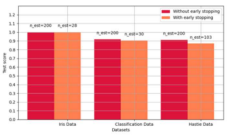
[Gradient Boosting regularization]



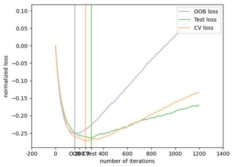
[Gradient Boosting regression]



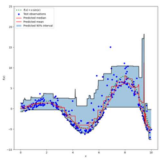
[Feature transformations with ensembles of trees]



[Early stopping of Gradient Boosting]

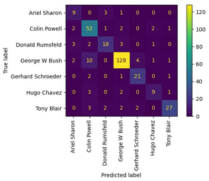


[Gradient Boosting Out-of-Bag estimates]

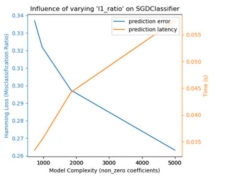


[Prediction Intervals for Gradient Boosting Regression]

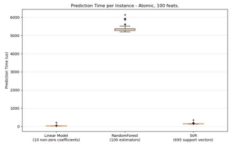
[Image denoising using kernel PCA]



[Faces recognition example using eigenfaces and SVMs]

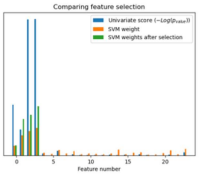


[Model Complexity Influence]

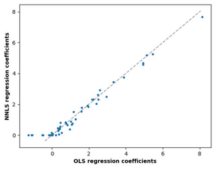


[Prediction Latency]

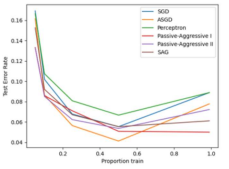
[Pipeline ANOVA SVM]



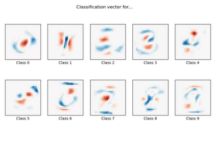
[Univariate Feature Selection]



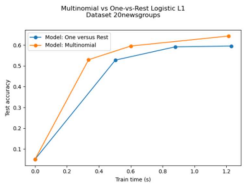
[Non-negative least squares]



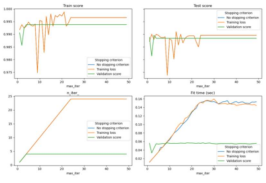
[Comparing various online solvers]



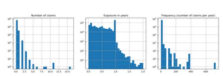
[MNIST classification using multinomial logistic + L1]



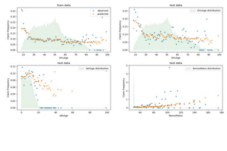
[Multiclass sparse logistic regression on 20newgroups]



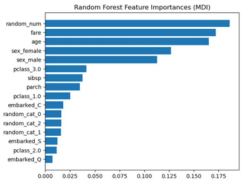
[Early stopping of Stochastic Gradient]



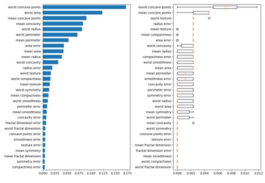
[Poisson regression and non-normal loss]



[Tweedie regression on insurance claims]



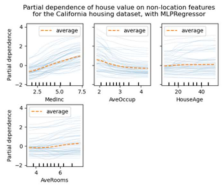
[Permutation Importance vs Random Forest]



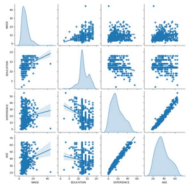
[Permutation Importance with Multicollinear or Correlated Features]

Toggle Menu

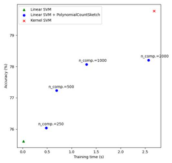
Feature Importance



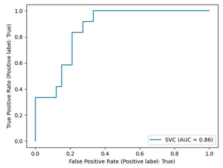
Partial Dependence and Individual Conditional Expectation Plots



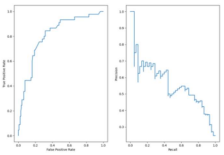
Common pitfalls in the interpretation of coefficients of linear models



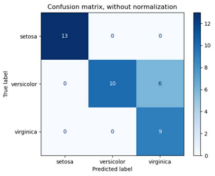
Scalable learning with polynomial kernel approximation



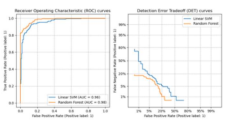
ROC Curve with Visualization API



Visualizations with Display Objects

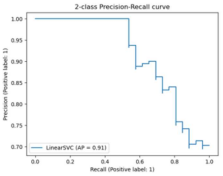


Confusion matrix

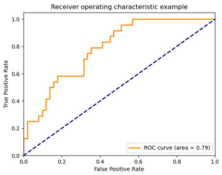


Detection error trade-off (DET) curve

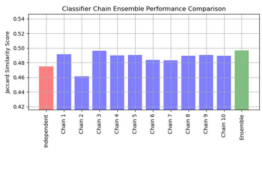
Parameter estimation using grid search with cross-validation



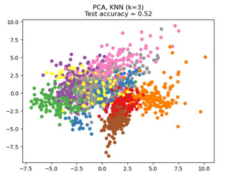
Precision-Recall



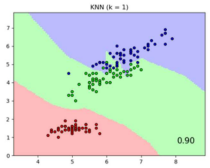
Receiver Operating Characteristic (ROC)



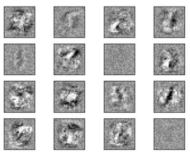
Classifier Chain



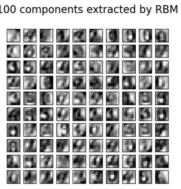
Dimensionality Reduction with Neighborhood Components Analysis



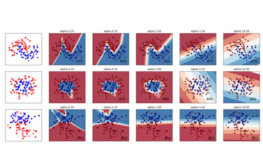
Comparing Nearest Neighbors with and without Neighborhood Components Analysis



Visualization of MLP weights on MNIST

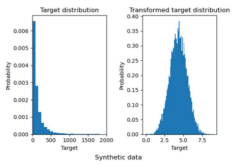


Restricted Boltzmann Machine features for digit classification

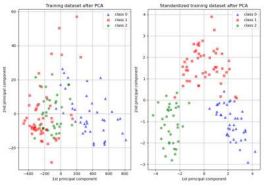


Varying regularization in Multi-layer Perceptron

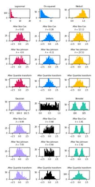
Column Transformer with Mixed Types



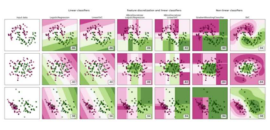
Effect of transforming the targets in regres-sion model



Importance of Feature Scaling



Map data to a normal distribution



Feature discretization

Semi-supervised Classification on a Text Dataset