Getting Started

The purpose of this guide is to illustrate some of the main features that `scikit-learn` provides. It assumes a very basic working knowledge of machine learning practices (model fitting, predicting, cross-validation, etc.). Please refer to our :ref:`installation instructions <installation-instructions>` for installing `scikit-learn`.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 4);** ***backlink***
>
> Unknown interpreted text role "ref".

`Scikit-learn` is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities.

Fitting and predicting: estimator basics

`Scikit-learn` provides dozens of built-in machine learning algorithms and models, called :term:`estimators`. Each estimator can be fitted to some data using its :term:`fit` method.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 18);** ***backlink***
>
> Unknown interpreted text role "term".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 18);** ***backlink***
>
> Unknown interpreted text role "term".

Here is a simple example where we fit a :class:`~sklearn.ensemble.RandomForestClassifier` to some very basic data:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 22);** ***backlink***
>
> Unknown interpreted text role "class".

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf = RandomForestClassifier(random_state=0)
>>> X = [[ 1,  2,  3],  # 2 samples, 3 features
...      [11, 12, 13]]
>>> y = [0, 1]  # classes of each sample
>>> clf.fit(X, y)
RandomForestClassifier(random_state=0)
```

The :term:`fit` method generally accepts 2 inputs:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 33);** ***backlink***
>
> Unknown interpreted text role "term".

- The samples matrix (or design matrix) :term:`X`. The size of `X` is typically `(n_samples, n_features)`, which means that samples are represented as rows and features are represented as columns.

  > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 35);** ***backlink***
  >
  > Unknown interpreted text role "term".

- The target values :term:`y` which are real numbers for regression tasks, or integers for classification (or any other discrete set of values). For unsupervized learning tasks, `y` does not need to be specified. `y` is usually 1d array where the `i` th entry corresponds to the target of the `i` th sample (row) of `X`.

  > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-

resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst, line 38); *backlink*

Unknown interpreted text role "term".

Both X and y are usually expected to be numpy arrays or equivalent :term:`array-like` data types, though some estimators work with other formats such as sparse matrices.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst, line 44); *backlink*

Unknown interpreted text role "term".

Once the estimator is fitted, it can be used for predicting target values of new data. You don't need to re-train the estimator:

```
>>> clf.predict(X)  # predict classes of the training data
array([0, 1])
>>> clf.predict([[4, 5, 6], [14, 15, 16]])  # predict classes of new data
array([0, 1])
```

Transformers and pre-processors

Machine learning workflows are often composed of different parts. A typical pipeline consists of a pre-processing step that transforms or imputes the data, and a final predictor that predicts target values.

In scikit-learn, pre-processors and transformers follow the same API as the estimator objects (they actually all inherit from the same BaseEstimator class). The transformer objects don't have a :term:`predict` method but rather a :term:`transform` method that outputs a newly transformed sample matrix X:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst, line 63); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst, line 63); *backlink*

Unknown interpreted text role "term".

```
>>> from sklearn.preprocessing import StandardScaler
>>> X = [[0, 15],
...      [1, -10]]
>>> # scale data according to computed scaling values
>>> StandardScaler().fit(X).transform(X)
array([[-1.,  1.],
       [ 1., -1.]])
```

Sometimes, you want to apply different transformations to different features: the :ref:`ColumnTransformer<column_transformer>` is designed for these use-cases.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst, line 77); *backlink*

Unknown interpreted text role "ref".

Pipelines: chaining pre-processors and estimators

Transformers and estimators (predictors) can be combined together into a single unifying object: a :class:`~sklearn.pipeline.Pipeline`. The pipeline offers the same API as a regular estimator: it can be fitted and used for prediction with fit and predict. As we will see later, using a pipeline will also prevent you from data leakage, i.e. disclosing some testing data in your training data.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst, line 84); *backlink*

Unknown interpreted text role "class".

In the following example, we :ref:`load the Iris dataset <datasets>`, split it into train and test sets, and compute the accuracy score of a pipeline on the test data:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 91); *backlink*

Unknown interpreted text role "ref".

```
>>> from sklearn.preprocessing import StandardScaler
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
...
>>> # create a pipeline object
>>> pipe = make_pipeline(
...     StandardScaler(),
...     LogisticRegression()
... )
...
>>> # load the iris dataset and split it into train and test sets
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
...
>>> # fit the whole pipeline
>>> pipe.fit(X_train, y_train)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
>>> # we can now use it like any other estimator
>>> accuracy_score(pipe.predict(X_test), y_test)
0.97...
```

Model evaluation

Fitting a model to some data does not entail that it will predict well on unseen data. This needs to be directly evaluated. We have just seen the :func:`~sklearn.model_selection.train_test_split` helper that splits a dataset into train and test sets, but `scikit-learn` provides many other tools for model evaluation, in particular for :ref:`cross-validation <cross_validation>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 123); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 123); *backlink*

Unknown interpreted text role "ref".

We here briefly show how to perform a 5-fold cross-validation procedure, using the :func:`~sklearn.model_selection.cross_validate` helper. Note that it is also possible to manually iterate over the folds, use different data splitting strategies, and use custom scoring functions. Please refer to our :ref:`User Guide <cross_validation>` for more details:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 130); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 130); *backlink*

Unknown interpreted text role "ref".

```
>>> from sklearn.datasets import make_regression
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.model_selection import cross_validate
...
>>> X, y = make_regression(n_samples=1000, random_state=0)
>>> lr = LinearRegression()
...
>>> result = cross_validate(lr, X, y)  # defaults to 5-fold CV
>>> result['test_score']  # r_squared score is high because dataset is easy
array([1., 1., 1., 1., 1.])
```

Automatic parameter searches

All estimators have parameters (often called hyper-parameters in the literature) that can be tuned. The generalization power of an estimator often critically depends on a few parameters. For example a :class:`~sklearn.ensemble.RandomForestRegressor` has a `n_estimators` parameter that determines the number of trees in the forest, and a `max_depth` parameter that determines the maximum depth of each tree. Quite often, it is not clear what the exact values of these parameters should be since they depend on the data at hand.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 150); *backlink*

Unknown interpreted text role "class".

`Scikit-learn` provides tools to automatically find the best parameter combinations (via cross-validation). In the following example, we randomly search over the parameter space of a random forest with a :class:`~sklearn.model_selection.RandomizedSearchCV` object. When the search is over, the :class:`~sklearn.model_selection.RandomizedSearchCV` behaves as a :class:`~sklearn.ensemble.RandomForestRegressor` that has been fitted with the best set of parameters. Read more in the :ref:`User Guide <grid_search>`:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 159); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 159); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 159); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 159); *backlink*

Unknown interpreted text role "ref".

```
>>> from sklearn.datasets import fetch_california_housing
>>> from sklearn.ensemble import RandomForestRegressor
>>> from sklearn.model_selection import RandomizedSearchCV
>>> from sklearn.model_selection import train_test_split
>>> from scipy.stats import randint
...
>>> X, y = fetch_california_housing(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
...
>>> # define the parameter space that will be searched over
>>> param_distributions = {'n_estimators': randint(1, 5),
...                        'max_depth': randint(5, 10)}
...
>>> # now create a searchCV object and fit it to the data
>>> search = RandomizedSearchCV(estimator=RandomForestRegressor(random_state=0),
...                             n_iter=5,
...                             param_distributions=param_distributions,
...                             random_state=0)
>>> search.fit(X_train, y_train)
RandomizedSearchCV(estimator=RandomForestRegressor(random_state=0), n_iter=5,
                   param_distributions={'max_depth': ...,
                                        'n_estimators': ...},
                   random_state=0)
>>> search.best_params_
{'max_depth': 9, 'n_estimators': 4}

>>> # the search object now acts like a normal random forest estimator
>>> # with max_depth=9 and n_estimators=4
>>> search.score(X_test, y_test)
0.73...
```

Note

In practice, you almost always want to :ref:`search over a pipeline <composite_grid_search>`, instead of a single estimator. One of the main reasons is that if you apply a pre-processing step to the whole dataset without using a pipeline, and then perform any kind of cross-validation, you would be breaking the fundamental assumption of independence between training and testing data. Indeed, since you pre-processed the data using the whole dataset, some information about the test sets are available to the train sets. This will lead to over-estimating the generalization power of the estimator (you can read more in this Kaggle post).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 201); *backlink*

Unknown interpreted text role "ref".

Using a pipeline for cross-validation and searching will largely keep you from this common pitfall.

Next steps

We have briefly covered estimator fitting and predicting, pre-processing steps, pipelines, cross-validation tools and automatic hyper-parameter searches. This guide should give you an overview of some of the main features of the library, but there is much more to `scikit-learn`!

Please refer to our :ref:`user_guide` for details on all the tools that we provide. You can also find an exhaustive list of the public API in the :ref:`api_ref`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 224); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 224); *backlink*

Unknown interpreted text role "ref".

You can also look at our numerous :ref:`examples <general_examples>` that illustrate the use of `scikit-learn` in many different contexts.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 228); *backlink*

Unknown interpreted text role "ref".

The :ref:`tutorials <tutorial_menu>` also contain additional learning resources.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\(scikit-learn-main)(doc)getting_started.rst`, line 231); *backlink*

Unknown interpreted text role "ref".