**Experiment 3**

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 installation instructions for installing scikit-learn.

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 and evaluation, and many other utilities.

**Fitting and predicting: estimator basics**

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

Here is a simple example where we fit a RandomForestClassifier to some very basic data:



The fit method generally accepts 2 inputs:

The samples matrix (or design matrix) 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.

The target values 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.

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

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:



**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

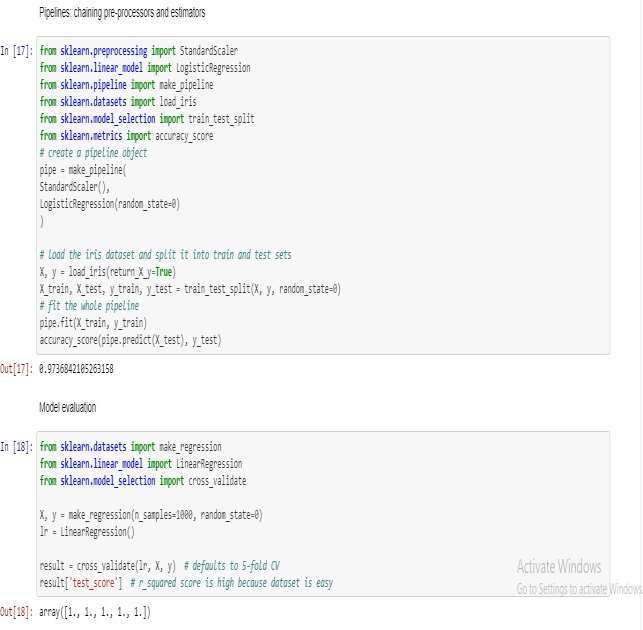
estimator objects (they actually all inherit from the same BaseEstimator class). The transformer objects don’t have a predict method but rather a transform method that outputs a newly transformed sample matrix X:



**Pipelines: chaining pre-processors and estimators**

Transformers and estimators (predictors) can be combined together into a single unifying object: a 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.

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



**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 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 cross-validation.

We here briefly show how to perform a 5-fold cross-validation procedure, using the 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 User Guide for more details:

