Overview

The scikit-learn project is an increasingly popular machine learning library written in Python. It is designed to be simple and efficient, useful to both experts and non-experts, and reusable in a variety of contexts. The primary aim of the project is to provide a compendium of efficient implementations of classic, well-established machine learning algorithms.

Scikit-Learn: Machine Learning in the Python ecosystem

Arnaud Joly, Gilles Louppe, and all scikit-learn contributors

Unsupervised learning: clustering (KMeans, Ward, ...), matrix decomposition (PCA, ICA, ...), outlier detection.

Model selection and evaluation: cross-validation, grid-search, lots of metrics. ... and many more! (See our Reference).

Collaborative development

Around 10 core developers and more than 100 occasional contributors from all around the world. All are working together on GitHub with strict coding guidelines, including style consistency, unit-test coverage, documentation, examples and code review.

Who is using scikit-learn?

In the Bioinformatics and Modelling research unit, scikit-learn is used by several researchers to perform gene interaction discovery. Bioinformatics image analysis such as in the Cytomine project or more fundamental research in machine learning. The research unit is also actively contributing to the library with two researchers as core contributors.

In the applied science faculty of the University of Liège (ULg), scikit-learn is also actively contributing to the library with two researchers as core contributors.

scikit-learn is widely used in the industry (e.g. Spotify, Evernote, Phimeca) and in the academic world (e.g. ULg, Inria, Télécom ParisTech).

A simple and unified API

All objects in scikit-learn share a uniform and limited API consisting of three complementary interfaces:

- an estimator interface for building and fitting models;
- a predictor interface for making predictions;
- a transformer interface for converting data.

All of them takes as input data which is structured as Numpy arrays or Scipy sparse matrices.

Data analysis in Python

Let us illustrate the various components of the scientific Python ecosystem for the analysis of scientific data. We consider genetic data from the HapMap project which catalogs common genetic variants in human beings from 11 human populations from different parts of the world.

1) Data preprocessing with scikit-learn and pandas

Loading, converting and merging raw data files is easily done using scikit-learn, pandas, NumPy or other Python battery included modules.

Once data are loaded, scikit-learn and pandas provide data preprocessing tools to extract and to normalize data. For instance, there is missing values in the HapMap dataset that can be inferred using the most frequent values associated to each SNP.

2) Data visualization with matplotlib

In exploratory analysis, data visualization plays an important role in identifying interesting patterns. However data are often high dimensional. scikit-learn implements several dimensionality reduction methods. For illustration, let’s study chromosome 15. First, we reduce the dimensionality of the data from 33800 to 2.

Conclusions

Python data-oriented packages tend to complement and integrate smoothly with each other. Together, they provide a powerful, flexible and coherent working environment for real-world scientific data analysis. Python is good and viable replacement to Matlab and R.

Scikit-Learn complements this ecosystem with machine learning algorithms and data analysis utilities.

www.scikit-learn.org