SSLearn: A Semi-Supervised Learning library for Python

José Luis Garrido-Labrador[1][0000−0002−3441−4223], Jesús Manuel Maudes-Raedo[1][0000−0001−8808−412X], Juan José Rodríguez[1][0000−0002−3291−2739], and César Ignacio García-Osorio[1][0000−0002−1206−1084]

Departamento de Ingeniería Informática, Universidad de Burgos, Escuela Politécnica Superior, Avd. Cantabria s/n, 09006 Burgos, Spain
{jlgarrido,jmaudes,jjrodriguez,cgosorio}@ubu.es

Abstract. This work is about the Python library for semi-supervised learning (*SSLearn*) presented in SoftwareX [1]. It offer a variety of wrappers and restricted set classification algorithms, and utilities to make easier the use of semi-supervised learning in Python and implement new algorithms. Is compatible with Scikit-learn and open source under the BSD 3-Clause License.

Keywords: Semi-supervised learning · Python library · Self-training · Co-training · Restricted set classification

1 Introduction

Semi-supervised learning (SSL) has been an important machine learning paradigm for almost three decades. Although Python users increasingly rely on *scikit-learn* as their main machine learning library, its native support for SSL is limited to a small set of methods, namely Label Propagation, Label Spreading, and Self-training. To address this gap, we present *SSLearn*, a Python library for semi-supervised learning built on the *scikit-learn* API and inspired by earlier work developed for the KEEL software [3]. *SSLearn* provides a collection of wrapper-based SSL algorithms, together with methods for restricted set classification, a novel constrained classification paradigm.

In addition, *SSLearn* includes tools for semi-supervised regression and for conducting experiments and developing new algorithms. The library is open source under the BSD 3-Clause License and is available on PyPI[1] and GitHub[2].

2 Related work

Several software solutions for data mining include support for semi-supervised learning (SSL). Among them, three stand out: two Python libraries and one

[1] https://pypi.org/project/sslearn/
[2] https://github.com/jlgarridol/sslearn

Table 1. Comparison of related work.

	SSLearn	*LAMDA-SSL*	*scikit-learn*	*KEEL*
Year (last update)	2025	2022	2025	2018
Type	Library	Library	Library	GUI App
License	BSD 3-Clause	MIT	BSD 3-Clause	GPLv3
Language	Python	Python	Python	Java
Wrapper algorithms	12	4	1	12
RSC for SSL	Yes (2)	No	No	No
Deep learning SSL	No	Yes (18)	No	No
Other SSL	No	Yes (8)	Yes (2)	No

Java desktop application. *LAMDA-SSL* [2] is arguably the most comprehensive Python library for SSL, with a strong emphasis on image processing. It includes 30 algorithms, most of which are intrinsically semi-supervised or related to semi-supervised preprocessing. However, its coverage of wrapper-based methods is limited to four algorithms, whereas *SSLearn* currently focuses exclusively on wrapper-based SSL and provides twelve algorithms in this family. A relevant advantage of wrapper methods is that they can be combined with any supervised classifier, including decision trees, nearest neighbors, and deep learning models.

The second Python library is *scikit-learn*, which includes three SSL algorithms, as noted in the introduction. Although *scikit-learn* is the de facto standard machine learning library in Python, its SSL support remains limited.

Finally, *KEEL* is a Java-based desktop application that provides a broad collection of data mining algorithms, including 12 wrapper-based SSL methods. In contrast with *SSLearn*, however, *KEEL* is not a Python library, which makes it less convenient for integration into modern Python-based machine learning workflows and research environments.

Table 1 summarizes the comparison between these tools and *SSLearn*.

3 Software description

The *SSLearn* library is organized into four main components: data manipulation, wrapper-based algorithms, restricted set classification, and utilities.

3.1 Data manipulation

The `datasets` module provides functions for loading and preprocessing datasets. It supports CSV and `.dat` files from KEEL.

The `model_selection` module includes tools for creating artificially labeled and unlabeled datasets from fully labeled data.

3.2 Wrapper-based algorithms

The `wrapper` module is the core of the library and contains implementations of wrapper-based algorithms for semi-supervised learning, including the following methods:

- `SelfTraining`: a wrapper that iteratively labels the most confident unlabeled samples and adds them to the training set.
- `Setred`: similar to `SelfTraining`, but it uses a rejection mechanism to avoid adding noisy samples to the training set.
- `CoTraining`: combines two classifiers trained on different views of the data.
- `CoReg`: a co-training strategy for regression problems.
- `CoTrainingByCommittee`: creates a committee of classifiers and labels the samples on which the committee members reach sufficient agreement.
- `DemocraticCoLearning`: a co-training method that combines predictions through majority voting and confidence measures to label samples.
- `Rasco`: a wrapper that uses a random subspace method to generate different views of the data and label samples through a consensus mechanism.
- `RelRasco`: a variant of `Rasco` that uses a relevance measure to select the features for each view through tournament selection.
- `CoForest`: a semi-supervised adaptation of Random Forest.
- `TriTraining`: a wrapper that trains three classifiers and labels samples for one classifier when the other two agree on the prediction.
- `DeTriTraining`: a variant of `TriTraining` that uses a depuration mechanism based on clustering to remove noisy samples from the training set.
- `TriTrainingRegressor`: the regression version of `TriTraining`.

3.3 Restricted set classification

The `restricted` module contains implementations of algorithms for restricted set classification.

- `WhoIsWhoClassifier`: a wrapper that uses "cannot-link" constraints for assigning labels.
- `probability_fusion`: a method that uses both "cannot-link" and "must-link" constraints, averaging the probabilities of linked samples.
- `feature_fusion`: a method that uses both "cannot-link" and "must-link" constraints, averaging the features of linked samples.

3.4 Utilities

In addition, the `base`, `utils`, and `subview` modules provide a range of utilities for semi-supervised learning.

The `base` module contains useful base classes and tools for creating new algorithms, the `utils` module provides a set of generic utilities, and the `subview` module contains the classes `SubViewClassifier` and `SubViewRegressor` for training a classifier or regressor on a subset of features.

4 Impact

The first version of *SSLearn* was released on PyPI in December 2022 and later published in *SoftwareX* in January 2025.

Since its first release, the library has been downloaded more than 15,950 times, according to PyPI statistics collected by Pepy[3]. However, its impact is also reflected in its adoption by the research community.

Before the publication of the paper, the library had already been indexed in Zenodo and used in three research works. After the publication of the paper, it has been cited or used in at least four additional research works.

5 Conclusions

SSLearn provides a practical and extensible environment for semi-supervised learning in Python. By following the *scikit-learn* design philosophy, the library integrates naturally into existing machine learning workflows while offering a curated set of wrapper-based semi-supervised algorithms, restricted set classification methods, and supporting utilities.

One of the main strengths of the library is its focus on methods that are both useful in practice and easy to adapt for research purposes. The inclusion of utilities for dataset manipulation, the support for KEEL and CSV formats, and the availability of classes such as `SubViewClassifier` and `SubViewRegressor` make the library convenient for experimentation and algorithm development. In addition, the restricted set classification module addresses a niche but relevant problem that is not covered by other widely used Python libraries.

The adoption of *SSLearn* by the research community and its distribution through PyPI and GitHub show that the library is already contributing to semi-supervised learning research. Its open-source nature also facilitates code inspection, reproducibility, and the incorporation of future improvements. For these reasons, *SSLearn* is intended both as a usable software package and as a foundation for further work in semi-supervised learning.

Future developments will continue expanding the library with additional semi-supervised methods, broader support for related learning scenarios, and new utilities for data handling and model construction.

References

1. Garrido-Labrador, J.L., Maudes-Raedo, J.M., Rodríguez, J.J., García-Osorio, C.I.: Sslearn: A semi-supervised learning library for python. SoftwareX **29**, 102024 (Feb 2025). https://doi.org/10.1016/j.softx.2024.102024
2. Jia, L.H., Guo, L.Z., Zhou, Z., Li, Y.F.: Lamda-ssl: Semi-supervised learning in python. arXiv preprint arXiv:2208.04610 (2022). https://doi.org/10.48550/arXiv.2208.04610
3. Triguero, I., García, S., Herrera, F.: Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study **42**(2), 245–284 (2013). https://doi.org/10.1007/s10115-013-0706-y

[3] https://pepy.tech/projects/sslearn