Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SSLearn: A Semi-Supervised Learning library for Python

José L. Garrido-Labrador [*], Jesús M. Maudes-Raedo, Juan J. Rodríguez, César I. García-Osorio

Universidad de Burgos, Escuela Politécnica Superior, Avd. Cantabria s/n, 09006, Burgos, Spain

ARTICLE INFO

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

ABSTRACT

SSLearn is an open-source Python-based library that advances semi-supervised learning (SSL) with a focus on wrapper algorithms and restricted set classification (RSC), a novel paradigm. It fosters innovation by allowing researchers to modify methods or create new ones, facilitating access to state-of-the-art algorithms and comparative studies. As the only library incorporating RSC for constrained classification, SSLearn fills an important gap in SSL tools. Fully compatible with Scikit-Learn, it integrates seamlessly into research workflows, lowering the barrier to entry to SSL and catalyzing its adoption in diverse domains. This makes SSLearn a critical resource for advancing SSL research and applications.

Code metadata

Current code version	1.0.5.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00323
Permanent link to Reproducible Capsule	https://colab.research.google.com/drive/1wKSz-f7N4elqQwz_phrWXDrf3lRqaD6s#offline=true&sandboxMode=true
Legal Code License	BSD 3-Clause
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python $\geq$ 3.8, joblib $\geq$ 1.2.0, numpy $\geq$ 1.23.3, pandas $\geq$ 1.4.3, scikit-learn $\geq$ 1.2.0, scipy $\geq$ 1.10.1, statsmodels $\geq$ 0.13.2
If available Link to developer documentation/manual	https://jlgarridol.github.io/sslearn/sslearn.html
Support email for questions	jlgarrido@ubu.es

Software metadata

Current software version	1.0.5.3
Permanent link to executables of this version	https://pypi.org/project/sslearn/
Permanent link to Reproducible Capsule	https://colab.research.google.com/drive/1wKSz-f7N4elqQwz_phrWXDrf3lRqaD6s#offline=true&sandboxMode=true
Legal Software License	BSD 3-Clause
Computing platforms/Operating Systems	GNU/Linux, Windows, macOS
Installation requirements & dependencies	numpy, pandas, scikit-learn, scipy, statsmodels, joblib
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://jlgarridol.github.io/sslearn/sslearn.html
Support email for questions	jlgarrido@ubu.es

1. Motivation and significance

Semi-supervised machine learning (SSL) is a data mining technique that has been with us for almost three decades [1]. Despite its age, it has only recently become more accessible due to the availability of code libraries containing a variety of SSL algorithms.

This type of machine learning is concerned with the combination of labeled (supervised) instances with unlabeled (unsupervised) instances.

* Corresponding author.
E-mail address: jlgarrido@ubu.es (José L. Garrido-Labrador).

https://doi.org/10.1016/j.softx.2024.102024
Received 21 June 2024; Received in revised form 2 December 2024; Accepted 17 December 2024
Available online 7 January 2025

This can be done either to create predictive models (inductive) or to extend the labels from the labeled set to the unlabeled set (transductive) [2]. Combining both types of data can improve models by exploiting the patterns of all data combined with the classes of a few instances [3]. Furthermore, this approach can reduce the cost of the process of labeling all data [4].

Faced with the challenge of embarking on an SSL research project, we encountered the difficulty of lacking the tools to experiment with state-of-the-art SSL algorithms. For this reason, SSLearn (semi-super vised learning library), a Python-based open-source code library inspired by Scikit-Learn [5,6], has been designed with the aim of contributing to the scientific community by offering a collection of already implemented algorithms. This allows for the comparison of new algorithms as they emerge with those representing the current state-of-the-art. Moreover, being an open-source library, it promotes the modification of existing algorithms to create new ones without the need to start from scratch.

The initial versions of SSLearn have concentrated on wrapper algorithms, which are a type of semi-supervised learning method. These algorithms build models using a supervised learning algorithm as a starting point and then incorporate new labeled instances from the unlabeled set in an iterative process. The most relevant methods from the literature have been incorporated into SSLearn [7,8].

Finally, this library incorporates algorithms for restricted set classification (RSC) problems, which represent a novel data mining paradigm introduced by Kuncheva et al. [9]. This paradigm is particularly relevant in situations where the objects to be classified are neither independent nor identically distributed, which is a common assumption in traditional classification tasks. In contrast, RSC addresses specific restrictions on the number of objects that can belong to each class. This paradigm, although novel, has already been the subject of previous research, resulting in the development of semi-supervised algorithms for this type of problem [10]. In order to support these new methods presented in a library, they have been incorporated into SSLearn.

1.1. Related works

There are other data mining software solutions that employ SSL. The most notable include three: two Python libraries and one desktop software in Java. LAMDA-SSL [11] is arguably the most comprehensive Python library for SSL, with a primary focus on image processing. It encompasses a total of 30 algorithms, the majority of which are intrinsically semi-supervised or related to semi-supervised preprocessing. The primary distinction between the two is that the work presented here includes only four wrapper algorithms, whereas SSLearn is currently focused exclusively on wrapper algorithms and includes a total of ten different algorithms. It is important to note that wrapper algorithms support any type of supervised classifier, including trees, neighbors, deep learning, etc.

On the other hand, Scikit-Learn [5] stands out as the main Python data mining library, although it only includes three SSL algorithms. The Scikit-Learn API [6] has served as a model for the development of SSLearn. And finally there is KEEL [12], which more than a library is a data mining tool similar to Weka [13] or Orange [14]. KEEL is written in Java and supports a wide variety of machine learning algorithms, including some specific wrapper SSL algorithms. This was the tool used by Triguero et al. [7] in a review paper about SSL.

A comparison between all of these software products is in the Table 1

Table 1
Comparison between the related works and SSLearn. Wrappers and RSC are the previously mentioned, where SSLearn focuses. Deep learning SSL refers to algorithms that only use deep learning techniques. Other refers to a wide range of semi-supervised algorithms (graph-based, generative, maxim-margin, etc.).

	SSLearn	LAMDA-SSL	Scikit-Learn	KEEL
Year	2024	2022	2024	2018
Type	Library	Library	Library	GUI App
License	BSD 3-Clause	MIT	BSD 3-Clause	GPLv3
Language	Python	Python	Python	Java
Wrapper algorithms	10	4	1	12
RSC for SSL	2	0	0	0
Deep learning SSL	0	18	0	0
Other SSL	0	8	2	0

2. Software description

Designed as a free Python library (open source under the 3-Clause BSD license), SSLearn provides a variety of SSL algorithms as well as a set of tools for data set manipulation. It is a tool designed for machine learning researchers engaged in semi-supervised data analysis and for the practitioners who need a set of semi-supervised learning algorithms ready to be used in Python.

In the context of knowledge discovery and data mining (KDD), the modules and functionalities of SSLearn are distributed across multiple steps of the mentioned process. Fig. 1 illustrates the location of the primary algorithms. All these algorithms and modules will be better explained in their respective sections.

The current version of the library provides a solid foundation for semi-supervised learning by including some of the most cited and influential algorithms in the field. At present, these methods are limited to those that are classification oriented and come from research that is more than a decade old. This time lag is deliberately intended to focus on well-established techniques, and therefore does not include semi-supervised regression, clustering, unbalanced or multi-label methods, or deep learning methods. Although tools such as grid search and multi-class methods are not yet implemented, these features are planned for future updates. The library does not currently include example datasets, but it does include sufficient functionality to import semi-supervised datasets from the main SSL tools.

2.1. Data sets manipulation

The library includes tools for reading and writing SSL data sets, focusing on CSV and `.dat` files [15] (a version of KEEL's `.arff` files[1]), along with a couple of functions for randomly converting supervised data sets into SSL data sets. These algorithms are particularly useful for experimentation. They are included in the submodules `sslearn.datasets` and `sslearn.model_selection`.

Fig. 2 shows the KEEL `.dat` data file and Fig. 3 shows the same file in CSV format. In extending the indications for SSL provided by Scikit-Learn, instances whose label is shown as -1 are considered unlabeled. In KEEL, this is designated as "unlabeled''. The data loading and manipulation functions provided by SSLearn are compatible with -1.

2.2. Wrappers algorithms

The library includes ten wrapper-type algorithms for SSL, two self-training algorithms, and eight co-training algorithms. All algorithms follow the same design as those in Scikit-Learn, with default settings based on their original academic publications. These algorithms are grouped under the `sslearn.wrapper` module and are designed to extend the Scikit-Learn API [6]. The class diagrams showing inheritance relationships for these algorithms are shown in Figs. 4 and 5.

A brief summary of each of the wrapper algorithms implemented in the library is provided below:

[1] So it is possible to access all the data sets already prepared for SSL experimentation available at https://sci2s.ugr.es/SelfLabeled.

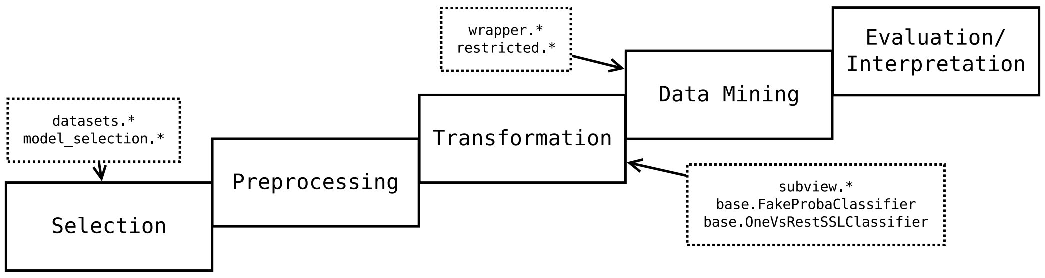

Fig. 1. KDD workflow comprises the principal algorithms and modules. The `datasets` module and the `model_selection` module are included in the data selection step, while the `subview` module and the `FakeProbaClassifier` and `OneVsRestSSLClassifier` classes are part of the transformation step. Finally, the `restricted` and `wrapper` modules are included in the data mining step.

```
@relation iris
@attribute SepalLength real [4.3, 7.9]
@attribute SepalWidth real [2.0, 4.4]
@attribute PetalLength real [1.0, 6.9]
@attribute PetalWidth real [0.1, 2.5]
@attribute Class {Iris-setosa, Iris-versicolor, Iris-virginica,
    unlabeled}
@inputs SepalLength, SepalWidth, PetalLength, PetalWidth
@outputs Class
@data
6.8, 3.2, 5.9, 2.3, Iris-virginica
6.1, 3.0, 4.6, 1.4, Iris-versicolor
4.6, 3.2, 1.4, 0.2, Iris-setosa
...
5.0, 3.6, 1.4, 0.2, unlabeled
4.9, 3.1, 1.5, 0.1, unlabeled
6.9, 3.1, 5.1, 2.3, unlabeled
5.8, 2.7, 3.9, 1.2, unlabeled
6.4, 2.8, 5.6, 2.1, unlabeled
...
```

Fig. 2. KEEL SSL data set example.

```
SepalLenght,SepalWidth,PetalLength,PetalWidth,Class
6.8,3.2,5.9,2.3, Iris-virginica
6.1,3,4.6,1.4, Iris-versicolor
4.6,3.2,1.4,0.2, Iris-setosa
...
5,3.6,1.4,0.2,-1
4.9,3.1,1.5,0.1,-1
6.9,3.1,5.1,2.3,-1
5.8,2.7,3.9,1.2,-1
6.4,2.8,5.6,2.1,-1
...
```

Fig. 3. CSV SSL data set example.

- **SelfTraining** [16] iteratively retrains a classifier, adding at each iteration those instances for which the confidence in the assigned label is above a given threshold, until the specified iteration limit is reached.
- **Setred** [17] is similar to SelfTraining but with a rejection step, filtering out potentially noisy predictions based on the neighborhood of the instances and statistical significance.
- **CoTraining** [18,19] utilizes two classifiers on different views of the dataset, enlarging each other's labeled set based on acceptance criteria over several iterations.
- **CoTrainingByCommittee** [20] uses a committee of classifiers to label new instances by ensemble voting, ensuring class balance across multiple iterations.
  It uses multiple classifiers to label instances where there is majority agreement, updating labels only if no increase in errors is detected.
- **DemocraticCoLearning** [21] uses multiple classifiers to label instances where there is majority agreement, updating labels only if no increase in errors is detected.

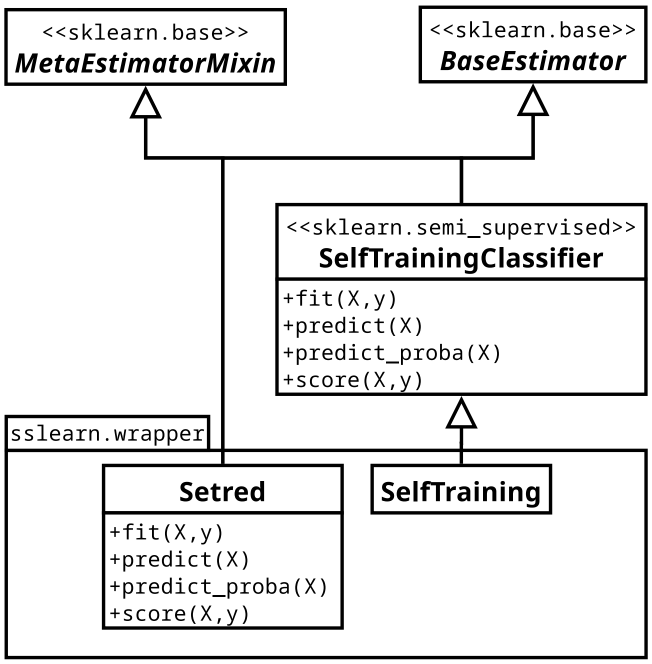

Fig. 4. Class diagram showing inheritance relationships diagram for the Self-Training algorithms in the `wrapper` module.

- **Rasco** [22] trains classifiers on random subspaces, updating models with the most probable classes while keeping class ratios consistent.
- **RelRasco** [23], which is similar to Rasco, but subspaces compete for the best mutual information [24], selecting those with the highest score.
- **CoForest** [25] is an adaptation of Random Forest for SSL which trains trees with bootstrapping, using thresholds to maintain prediction consistency.
- **TriTraining** [26] uses the predictions of three classifiers, adding to each classifier's training set those instances where the other two classifiers agree on the predicted labels (but also considering whether adding the new instances offsets any noise that may be added).
- **DeTriTraining** [27] is like TriTraining but uses clustering and nearest neighbors to refine labels, reassigning clusters based on majority class..

2.3. Restricted set classification

Classification algorithms with constraints have also been incorporated into the `sslearn.restricted` module. This type of classification is based on the relationships between instances, such that there are instances that must share the same class, known as "must-link" constraints, and instances that must not share the same class, known as "cannot-link" constraints [9].

The algorithms are the following:

- `WhoIsWhoClassifier` [28] uses "cannot-link" constraints to prevent instances that cannot share a label from being assigned the same class.
- `probability_fusion` [10] trains a classifier with "must-link" and "cannot-link" constraints, averaging probabilities for linked instances and preventing shared labels for non-linkable ones.
- `feature_fusion` [10], similar to probability_fusion, but averages features of linked instances instead of probabilities to ensure shared class assignments.

Constraints are represented as lists or dictionaries, where "must-link" and "cannot-link" are set. In the case of lists, each value represents a group (i.e., "must-link" and "cannot-link" sets of instances) of instances and the index in it, to which instance it belongs. In the case of dictionaries, each key represents the group, and the value, which would be a collection, represents the indexes of the instances it affects.

The nature of the constraints depends on the problem, using as example the work from Kuncheva et al. [10], focused on re-identifying animals in video, "must-link" constraints represent instances that have an important overlap to each other in consecutive frames, while "cannot-link" on instances that belong to the same frame. A detailed example can be found in the "Reproductive capsule".

2.4. Utilities

In addition to what has been presented, various utility tools are also included.

Within `sslearn.base`, numerous classes and functions are designed to serve as the solid found a limitation is that there are few methods and they are all classification-oriented, but this is actually a consequence of having focused on the most relevant methods in the literature (the papers where these methods were presented are among the first published, that is why they are the papers with the most citations). Example datasets are not included either, tion of the models. The function `get_dataset` separates the labeled from the unlabeled set. The class `FakedProbaClassifier` implements the `predict_proba` function for algorithms that do not have this implementation, assigning a probability of 1 to the predicted class, thus allowing its use by wrapper algorithms that require probabilities. Moreover, the `OneVsRestSSLClassifier` adapts the Scikit-Learn class for SSL data sets, ensuring that class `-1` are not consider in the distribution.

In `sslearn.utils` there are several functions available that are used by the implemented algorithms and may also be useful to others. `safe_division` prevents division by zero, `confidence_interval` calculates confidence intervals as its name suggests, `choice_with_proportion` ensures class ratio, `calculate_prior_probability` computes the distribution of classes, and `mode` identifies the majority class.

In `sslearn.datasets` there are functions to read and write `csv` and `dat`.[2]

Lastly, `sslearn.subview` includes two classes, `SubViewClassifier` and `SubViewRegressor`, which are model decorators that allow training a base classifier with a particular view of the data defined during training.

3. Illustrative examples

In this section, we present two examples that demonstrate the main features. In addition, the library documentation includes an example for each classifier and algorithm.

In the first example, we show how to use a wrapper classifier, the `CoForest`, trained with the Iris data set and manipulated to only have a 10% of labeled instances. Initially, a train/test split is performed using the `train_test_split` function from Scikit-Learn, and the `StratifiedKFoldSS` class is used to generate SSL data sets from the training partition. Triguero et al. [7] introduced a distinction in the calculation of metrics, referring to metrics on the unlabeled training set as transductive and on new data as inductive. This distinction is evident in the two score calculations, where in Scikit-Learn the score corresponds to accuracy.

[2] KEEL format based on Weka `arff` format.

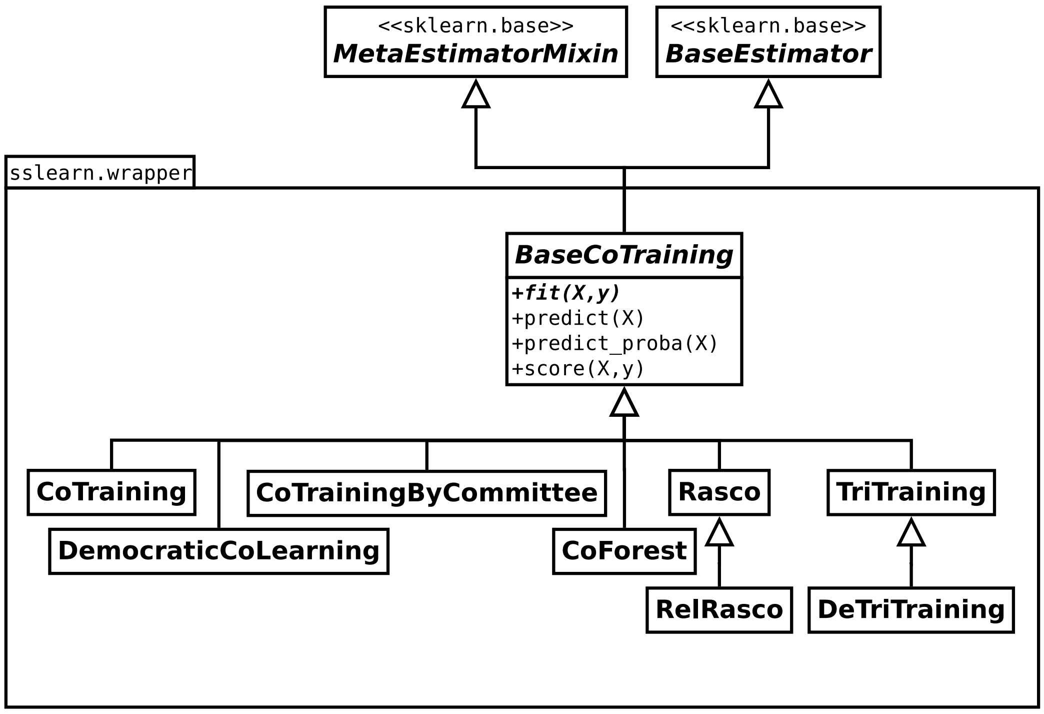

Fig. 5. Class diagram showing inheritance relationships diagram for the Co-Training algorithms in the `wrapper` module.

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import
    train_test_split
from sslearn.wrapper import CoForest
from sslearn.model_selection import
    StratifiedKFoldSS

X, y = load_iris(return_X_y=True)
skf = StratifiedKFoldSS(n_splits=10)
# 10
X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=.30)

transductive_acc = list()
inductive_acc = list()

for X_t, y_t, l_index, u_index in skf.split(
    X_train, y_train):
    coforest = CoForest(n_estimators=10).fit
        (X_t, y_t)
    X_u = X_train[u_index]
    y_u = y_train[u_index]
    transductive_acc.append(coforest.score(
        X_u, y_u))
    inductive_acc.append(coforest.score(
        X_test, y_test))
```

In the second example, a `DemocraticCoLearning` classifier is featured, unique in that each base classifier is trained using a different view of data via the `SubViewClassifier` class. This enables the user to select which features each algorithm utilizes, with the understanding that the wrapper will disseminate the same feature space to all of its base learners,[3] in this case three.

In addition, data is loaded directly from a KEEL data set file,[4] incorporating the training set which already includes unlabeled instances, and the test set for evaluation.

```python
from sklearn.tree import
    DecisionTreeClassifier as DT
from sslearn.datasets import read_keel
from sslearn.subview import
    SubViewClassifier
from sslearn.wrapper import
    DemocraticCoLearning

X_train, y_train = read_keel("
    movement_libras-ssl10-10-1tra.dat")
X_test, y_test = read_keel("movement_libras-
    ssl10-10-1tst.dat")

view1 = SubViewClassifier(DT(), "abcissa",
    mode="include")
# Only the features which has abcissa in the
    name
view2 = SubViewClassifier(DT(), "ordinate",
    mode="include")
# Only the features which has ordinate in
    the name
view3 = SubViewClassifier(DT(), "
    ^[2-4][0-3].*", mode="regex")
# The features which start with 20 to 23, 30
    to 33 and 40 to 43

tri = DemocraticCoLearning(base_estimator=[
    view1,view2,view3])
tri.fit(X_train, y_train)
tri.score(X_test, y_test)
```

Finally, we have included a comparative example of the different wrappers included in the library using the breast-cancer dataset.[5] All of them trained with 10, 20, 30 and 40 percent labeling, using $k = 10$ cross-validation to evaluate performance. The accuracy results are in

[3] It is crucial to highlight that numerous wrappers generate their own subviews, which impairs the functionality of the `SubView` classes.

[4] https://sci2s.ugr.es/keel/datasets.php

[5] The code is available in the reproducible capsule.

Table 2
Mean accuracy of a 10-fold cross validation with breast-cancer dataset for 10, 20, 30 and 40 percent of label rate.

	10%	20%	30%	40%
Self-Training	89.99%	90.52%	91.05%	89.81%
Setred	88.76%	90.86%	90.86%	91.04%
Co-Training	90.69%	92.27%	91.75%	91.75%
Co-Training by Committee	91.22%	92.62%	91.75%	91.75%
Democratic Co-Learning	91.92%	92.80%	93.68%	94.20%
RASCO	90.86%	91.74%	94.38%	93.33%
RelRASCO	90.87%	93.15%	92.45%	93.50%
CoForest	91.39%	92.80%	92.63%	92.45%
TriTraining	91.56%	90.68%	91.04%	91.75%
DeTriTraining	85.24%	85.06%	85.06%	85.24%

Table 2.

4. Impact

Semi-supervised learning (SSL) is a powerful paradigm within machine learning [1,2], bridging the gap between supervised and unsupervised learning by leveraging labeled and unlabeled data. One of the main advantages of SSL is its ability to significantly reduce the dependence on large labeled data sets, which are often expensive, time-consuming and labor-intensive to obtain. In many real-world scenarios, acquiring a large amount of labeled data is impractical, while unlabeled data is abundant and easily accessible [4]. SSL algorithms make effective use of this unlabeled data, allowing us to obtain models that in many cases can improve the performance of models that only use labeled data.

Another important advantage of SSL is its potential to improve the generalization and robustness of models. By incorporating unlabeled data into the learning process, SSL can help models capture a more complete understanding of the underlying data distribution.

The library presented here makes it easy to quickly exploit all these advantages by providing a rich set of wrapper algorithms specifically designed to take advantage of the full potential of SSL, eliminating the complex task of having to understand the details of the algorithms in the literature and avoiding having to implement them. Furthermore, since the library is fully compatible with Scikit-Learn, it is easier to incorporate the algorithms provided into the search workflows for the best model, semi-supervised or supervised, to solve the problems at hand.

To illustrate the potential impact of such a library, it is noteworthy that the number of proposed methods has increased substantially in ensembles alone over the past decade. Indeed, more than 120 new techniques have been introduced in the literature. Additionally, more than 400 articles have been published in indexed journals exploring the use of ensembles in semi-supervised learning [29]. The majority of these novel methodologies lack both code availability and incorporation into open-source libraries. In addition, SSLearn is presented as a collaborative platform that adheres to the principles of free software [30]. It encourages the incorporation of new methods into its library.

Being open source, it facilitates the addition of new features, allows researchers to study the code of the algorithms they wish to use, and allows adaptation to their specific needs. On this last point, this software can serve as the foundation for new SSL algorithms, inspired by those already implemented, as exemplified by Barbero-Aparicio et al. [31], who modified the `TriTraining` algorithm to function in regression contexts in a work on protein fitness landscape analysis.

It should be noted that while there are other open source libraries and desktop software for semi-supervised learning, SSLearn incorporates a number of methods into the Scikit-learn ecosystem that are not grouped in another library and highlight again that this is the only library that incorporates restricted set classification techniques for SSL.

Moreover, this library has already been utilized in other studies such as Stress Stimuli in Learning Contexts [32] and the re-identification of animals by video [10]. Despite the library's recent inception, its influence is already evident, although its use is not yet sufficiently widespread.

5. Conclusions

The SSLearn library, presented in this article, has been designed to provide tools for research in SSL within the Python ecosystem. Following the design guidelines of Scikit-Learn [6] and the philosophy of open-source software, it offers the research community the opportunity to review the code itself and contribute new algorithms, suggestions, bug fixes, etc.

Deployment is available on the Python Package Index (PyPI) and includes documentation for all algorithms.

While it is not the first Python library for SSL, most algorithms are not available in any other library compatible with Scikit-Learn. Additionally, various tools are offered to manipulate data, generate SSL sets from supervised data sets, support for KEEL data set format, and all algorithms are compatible with both Numpy and Pandas.

As future lines of development, the objective is to continue the expansion of this library by incorporating new semi-supervised wrapper methods, in addition to those employed in other semi-supervised algorithmic families. Furthermore, the incorporation of tools for the manipulation of semi-supervised sets, such as the adaptation of multi-class strategies (One-vs-One or Error-Correcting Output-Code), and the support of new file formats, is also envisaged.

CRediT authorship contribution statement

José L. Garrido-Labrador: Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization. **Jesús M. Maudes-Raedo:** Writing – review & editing, Validation, Supervision, Software. **Juan J. Rodríguez:** Writing – review & editing, Software, Resources, Project administration, Funding acquisition, Conceptualization. **César I. García-Osorio:** Writing – review & editing, Supervision, Software, Resources, Project administration, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported through the *Junta de Castilla y León* (JCyL) (regional government) under project BU055P20 (JCyL/FEDER, UE), the Spanish Ministry of Science and Innovation under project PID2020-119894GB-I00 co-financed through European Union FEDER funds, and project TED2021-129485B-C43 funded by MCIN/AEI/ 10.13039/501100011033 and the European Union NextGenerationEU/PRTR. J.L. Garrido-Labrador is supported through *Consejería de Educación* of the *Junta de Castilla y León* and the European Social Fund through a pre-doctoral grant EDU/875/2021 (Spain). We express our gratitude to G. Lucas-Pérez for his assistance with the documentation, to R. Marticorena-Sanchez, J.A. Barbero-Aparicio, A. Olivares-Gil, and J.M. Ramírez-Sanz for their bug reports and suggestions for functionality, and to C. López-Nozal for his advice on this article.

References

[1] van Engelen JE, Hoos HH. A survey on semi-supervised learning. Mach Learn 2020;109(2):373–440. http://dx.doi.org/10.1007/s10994-019-05855-6.

[2] Chapelle O, Schölkopf B, Zien A, editors. Semi-Supervised Learning. The MIT Press; 2006, http://dx.doi.org/10.7551/mitpress/9780262033589.001.0001.

[3] Singh A, Nowak R, Zhu J. Unlabeled data: Now it helps, now it doesn't. In: Koller D, Schuurmans D, Bengio Y, Bottou L, editors. Advances in neural information processing systems, vol. 21. Curran Associates, Inc.; 2008, p. 1513–20, URL https://proceedings.neurips.cc/paper_files/paper/2008/file/07871915a8107172b3b5dc15a6574ad3-Paper.pdf.

[4] Rodríguez Díez JJ. Aprendizaje Automático en ciencia de datos. 2023, URL https://www.ubu.es/sites/default/files/portal_page/files/leccion_inaugural_2023-2024.pdf.

[5] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[6] Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, et al. API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: languages for data mining and machine learning. 2013, p. 108–22.

[7] Triguero I, García S, Herrera F. Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. Knowl Inf Syst 2013;42(2):245–84. http://dx.doi.org/10.1007/s10115-013-0706-y.

[8] Ning X, Wang X, Xu S, Cai W, Zhang L, Yu L, et al. A review of research on co-training. In: Concurrency and computation: practice and experience. Wiley Online Library; 2021, e6276. http://dx.doi.org/10.1002/cpe.6276.

[9] Kuncheva LI, Rodríguez JJ, Jackson AS. Restricted set classification: Who is there? Pattern Recognit 2017;63. http://dx.doi.org/10.1016/j.patcog.2016.08.028.

[10] Kuncheva LI, Garrido-Labrador JL, Ramos-Pérez I, Hennessey SL, Rodríguez JJ. Semi-supervised classification with pairwise constraints: A case study on animal identification from video. Inf Fusion 2024;74:101994. http://dx.doi.org/10.1016/j.inffus.2023.102188.

[11] Jia L-H, Guo L-Z, Zhou Z, Li Y-F. LAMDA-SSL: Semi-supervised learning in Python. 2022, http://dx.doi.org/10.48550/arXiv.2208.04610, arXiv preprint arXiv:2208.04610.

[12] Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC, Herrera F. KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 2008;13(3):307–18. http://dx.doi.org/10.1007/s00500-008-0323-y.

[13] Frank E, Hall MA, Holmes G, Kirkby R, Pfahringer B, Witten IH. Weka: A machine learning workbench for data mining.. In: Data mining and knowledge discovery handbook: a complete guide for practitioners and researchers. Berlin: Springer; 2005, p. 1305–14.

[14] Demšar J, Curk T, Erjavec A, Gorup Č, Hočevar T, Milutinovič M, et al. Orange: Data mining toolbox in Python. J Mach Learn Res 2013;14:2349–53, URL http://jmlr.org/papers/v14/demsar13a.html.

[15] Derrac J, Garcia S, Sanchez L, Herrera F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J Mult Valued Logic Soft Comput 2015;17:255–87.

[16] Yarowsky D. Unsupervised word sense disambiguation rivaling supervised methods. In: Proceedings of the 33rd annual meeting on association for computational linguistics. Association for Computational Linguistics; 1995, p. 189–96. http://dx.doi.org/10.3115/981658.981684.

[17] Li M, Zhou Z-H. SETRED: Self-training with editing. In: Ho TB, Cheung D, Liu H, editors. Advances in knowledge discovery and data mining. Lecture notes in computer science, Springer; 2005, p. 611–21. http://dx.doi.org/10.1007/11430919_71.

[18] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on computational learning theory. 1998, p. 92–100. http://dx.doi.org/10.1145/279943.279962.

[19] Han X-H, Chen Y-W, Ruan X. Multi-class co-training learning for object and scene recognition. In: MVA. 2011, p. 67–70.

[20] Hady MFA, Schwenker F. Co-training by committee: A new semi-supervised learning framework. In: 2008 IEEE international conference on data mining workshops. 2008, p. 563–72. http://dx.doi.org/10.1109/ICDMW.2008.27.

[21] Zhou Y, Goldman S. Democratic co-learning. In: 16th IEEE international conference on tools with artificial intelligence. IEEE; 2004, p. 594–602. http://dx.doi.org/10.1109/ICTAI.2004.48.

[22] Wang J, Luo S-w, Zeng X-h. A random subspace method for co-training. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence). IEEE; 2008, p. 195–200. http://dx.doi.org/10.1109/IJCNN.2008.4633789.

[23] Yaslan Y, Cataltepe Z. Co-training with relevant random subspaces. Neurocomputing 2010;73(10–12):1652–61. http://dx.doi.org/10.1016/j.neucom.2010.01.018.

[24] Kraskov A, Stögbauer H, Grassberger P. Estimating mutual information. Phys Rev E 2004;69:066138. http://dx.doi.org/10.1103/PhysRevE.69.066138.

[25] Li M, Zhou Z-H. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Trans Syst Man Cybern A Syst Hum 2007;37(6):1088–98. http://dx.doi.org/10.1109/TSMCA.2007.904745, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans.

[26] Zhou Z-H, Li M. Tri-training: exploiting unlabeled data using three classifiers. IEEE Trans Knowl Data Eng 2005;17(11):1529–41. http://dx.doi.org/10.1109/TKDE.2005.186.

[27] Deng C, Guo MZ. Tri-training and data editing based semi-supervised clustering algorithm. In: MICAI 2006: Advances in artificial intelligence. Springer Berlin Heidelberg; 2006, p. 641–51. http://dx.doi.org/10.1007/11925231_61.

[28] Kuncheva LI. Full-class Set classification using the Hungarian algorithm. Int J Mach Learn Cybern 2010;1(1):53–61. http://dx.doi.org/10.1007/s13042-010-0002-z.

[29] Garrido-Labrador JL, Serrano-Mamolar A, Maudes-Raedo J, Rodríguez JJ, García-Osorio C. Ensemble methods and semi-supervised learning for information fusion: A review and future research directions. Inf Fusion 2024;107:102310. http://dx.doi.org/10.1016/j.inffus.2024.102310.

[30] Raymond ES. The Cathedral and the Bazaar. Sebastopol, CA: O'Reilly Media; 1999.

[31] Barbero-Aparicio JA, Olivares-Gil A, Rodríguez JJ, García-Osorio C, Díez-Pastor JF. Addressing data scarcity in protein fitness landscape analysis: A study on semi-supervised and deep transfer learning techniques. Inf Fusion 2024;102:102035. http://dx.doi.org/10.1016/j.inffus.2023.102035.

[32] Ramírez-Sanz JM, Peña-Alonso HM, Serrano-Mamolar A, Arnaiz-González Á, Bustillo A. Detection of stress stimuli in learning contexts of iVR environments. In: De Paolis LT, Arpaia P, Sacco M, editors. Extended reality. Cham: Springer Nature Switzerland; 2023, p. 427–40. http://dx.doi.org/10.1007/978-3-031-43404-4_29.