IMBENS: Ensemble Class-imbalanced Learning in Python

Zhining Liu[1] znliu19@mails.jlu.edu.cn
Zhepei Wei[1] weizp19@mails.jlu.edu.cn
Erxin Yu[1] yuex19@mails.jlu.edu.cn
Qiang Huang[1] huangqiang18@mails.jlu.edu.cn
Kai Guo[1] guokai20@mail.jlu.edu.cn
Boyang Yu[1] yuby19@mail.jlu.edu.cn
Zhaonian Cai[1] caizn19@mail.jlu.edu.cn
Hangting Ye[1] yeht2118@mails.jlu.edu.cn
Wei Cao[2] weicao@microsoft.com
Jiang Bian[2] jiang.bian@microsoft.com
Pengfei Wei[3] pengfei.wei@bytedance.com
Jing Jiang[4] jing.jiang@uts.edu.au
Yi Chang[1] yichang@jlu.edu.cn

[1] *School of Artificial Intelligence, Jilin University, China*
[2] *Microsoft Research Asia, China*
[3] *ByteDance AI Lab, Singapore*
[4] *University of Technology Sydney, Australia*

Abstract

`imbalanced-ensemble`, abbreviated as `imbens`, is an open-source Python toolbox for quick implementing and deploying ensemble learning algorithms on class-imbalanced data. It provides access to multiple state-of-art ensemble imbalanced learning (EIL) methods, visualizer, and utility functions for dealing with the class imbalance problem. These ensemble methods include resampling-based, e.g., under/over-sampling, and reweighting-based ones, e.g., cost-sensitive learning. Beyond the implementation, we also extend conventional binary EIL algorithms with new functionalities like multi-class support and resampling scheduler, thereby enabling them to handle more complex tasks. The package was developed under a simple, well-documented API design follows that of `scikit-learn` for increased ease of use. `imbens` is released under the MIT open-source license and can be installed from Python Package Index (PyPI). Source code, binaries, detailed documentation, and usage examples are available at `https://github.com/ZhiningLiu1998/imbalanced-ensemble`.

Keywords: ensemble learning, imbalanced learning, class-imbalanced data, long-tail problem, data mining, machine learning, Python

1. Introduction

Class-imbalance, also known as the long-tail problem, is the fact that the classes are not represented equally in a classification problem. Such issue widely exists in many real-world applications, such as click-through rate prediction (click/ignore), medical diagnosis (patient/non-patient), financial fraud detection (fraud/normal transaction), and network intrusion detection (malicious/normal request), etc (Haixiang et al., 2017). Imbalanced data often leads to degraded predictive performance of many standard machine learning algorithms since they assume a balanced class distribution and are directly optimized for global accuracy (He and Garcia, 2008; He and Ma, 2013). Imbalanced learning (IL) aims to tackle the class imbalance problem, i.e., learn an unbiased model from imbalanced data.

Most of the commonly used IL methods are based on resampling and reweighting, which are also the primary interest of existing open-source IL packages such as `imblearn` (Lemaître et al., 2017) and `smote-variants` (Kovács, 2019). Beyond them, ensemble imbalanced learning (EIL) further improves typical IL methods by combining the results of multiple independent resampling/reweighting and reducing variance (Galar et al., 2012). Recent studies have shown that the EIL solutions are highly competitive and gaining increasing popularity in IL (Haixiang et al., 2017). However, despite the success of EIL, only a handful of methods are available in existing open-source packages: only 4 basic EIL techniques are implemented in `imblearn`, while many important works like SmoteBoost (Chawla et al., 2003) and BalanceCascade (Liu et al., 2009) have no standard Python implementation.

To fill this gap, we propose a Python toolbox, namely `imbens` (imbalanced-ensemble), to *leverage the power of ensemble learning to address the class imbalance problem.* The following sections demonstrate the project vision, an overview of included EIL methods, a comparison with existing open-source packages, and the implementation design of `imbens`. Finally, we present our conclusion and future plans for the `imbens` package.

2. Project Focus

Documentation and examples. All EIL methods implemented in `imbens` share a unified API design similar to `scikit-learn` (Pedregosa et al., 2011). Detailed documentation is developed using `sphinx` and `numpydoc` and rendered using *ReadtheDocs*[1], including comprehensive API references, installation guideline, and code usage examples.

Extended functionalities. Most of exisiting EIL methods are designed for binary imbalanced classification. We extend their design in `imbens` to support multi-class imbalanced learning, allowing them to be employed in a wider range of applications. We also provide additional options such as customizable resampling scheduler for more fine-grained training control.

Customizable logging and visualization. `imbens` provides easy access to the status and statistics of the ensemble training process. With a few parameters, users can easily customize the information they want to monitor during training, including evaluation datasets, metrics, and log granularity. We also implement a general ensemble visualizer to provide further information and/or make comparison between multiple classifiers.

Performance and compatibility. Parallelization is enabled for both resampling and ensemble training when possible. The implemented EIL classifiers, visualizer, and utilities are also fully compatible with other popular packages like `scikit-learn` and `imblearn`.

3. Implemented methods

Currently (version 0.1.5), `imbens` have implemented 14 popular EIL methods, as summarized in Table 1. Their IL solutions can be divided into two main groups: resampling (under/over-sampling) and reweighting (boosting/cost-sensitive learning). Note that some of them combine resampling and reweighting, e.g., SmoteBoost. These methods also in-

1. `https://imbalanced-ensemble.readthedocs.io`

Method	Solution Type			Ensemble Type	Other Implementation	
	US	OS	RW		imbln	sv
SelfPacedEnsemble (Liu et al., 2020)	✓	✗	✗	Iterative	✗	✗
BalanceCascade (Liu et al., 2009)	✓	✗	✗	Iterative	✗	✗
BalancedRandomForest (Chen et al., 2004)	✓	✗	✗	Parallel	✓	✗
EasyEnsemble (Liu et al., 2009)	✓	✗	✗	Parallel	✓	✗
RusBoost (Seiffert et al., 2010)	✓	✗	✓	Iterative	✓	✗
UnderBagging (Barandela et al., 2003)	✓	✗	✗	Parallel	✓	✗
OverBoost (Galar et al., 2012)	✗	✓	✓	Iterative	✗	✗
SmoteBoost (Chawla et al., 2003)	✗	✓	✓	Iterative	✗	✗
KmeansSmoteBoost (Chawla et al., 2003)	✗	✓	✓	Iterative	✗	✗
OverBagging (Wang and Yao, 2009)	✗	✓	✗	Parallel	✓	✗
SmoteBagging (Wang and Yao, 2009)	✗	✓	✗	Parallel	✓	✗
AdaCost (Fan et al., 1999)	✗	✗	✓	Iterative	✗	✗
AdaUCost (Shawe-Taylor and Karakoulas, 1999)	✗	✗	✓	Iterative	✗	✗
AsymBoost (Viola and Jones, 2001)	✗	✗	✓	Iterative	✗	✗

* Abbreviations: under-sampling (US), over-sampling (OS), reweighting (RW), `imblearn` (`imbln`), `smote-variants` (`sv`).

Table 1: Ensemble imbalanced learning methods implemented in `imbens`.

Parameter	Data Type	Availability	Description
`target_label`	int	RS	Specify the class targeted by the resampling.
`n_target_samples`	int/dict	RS	Specify the desired number of samples (of each class).
`balancing_schedule`	str/callable	RS+IT	Scheduler that controls resampling during the training.
`cost_matrix`	str/array	CS	Specify (how to set) the misclassification cost matrix.
`eval_datasets`	dict	All	Dataset(s) used for evaluation during the training.
`eval_metrics`	dict	All	Metric(s) used for evaluation during the training.
`train_verbose`	bool/int/dict	All	Controls the verbosity during ensemble training.

* Abbreviations: resampling (RS), cost-sensitive (CS), iterative ensemble (IT).

Table 2: Additional key parameters of the `fit` method in `imbens`.

volve two different ensemble training manners: iterative (e.g., boosting) and parallel (e.g., bagging). Multi-core parallelization is enabled for all parallel EIL methods in `imbens`.

Up to our knowledge, we provide the first standard Python implementation for 10 of the 14 included EIL methods. Although the other 6 can be implemented with the `imblearn` package, they lack many of the useful features from `imbens` such as sampling scheduler and dynamic training logs. The `smote-variants` package focuses only on resampling techniques, especially oversampling, and does not involve any ensemble learning approaches.

4. Designs and implementation details

The implementation relies on `numpy`, `pandas`, `scipy`, and `scikit-learn` as well. We use `joblib` to implement multi-core execution for supported algorithms (with "parallel" ensemble type in Table 1) Inspired by `scikit-learn`'s API design, all EIL algorithms inherit from a base class (`ensemble.base.BaseImbalancedEnsemble`) and implement three main methods: (i) `fit` builds an ensemble classifier from the class-imbalanced training set (X, Y);

(ii) `predict` returns the predicted class labels corresponding to the given input samples; and (iii) `predict_proba` gives predicted class probabilities instead of labels.

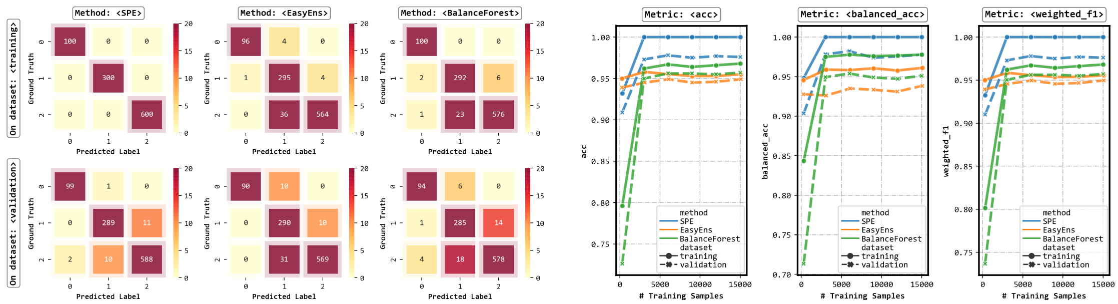
All EIL classifiers take two key parameters for initialization: `base_estimator` and `n_estimators`. The former can be any `scikit-learn`-style classifier instance, and the latter is an integer that specifies the size of the ensemble. To enable more precise control and detailed monitoring of the EIL training process, the `fit` function takes several additional arguments. `target_label`, `n_target_samples` and `balancing_schedule` can be used to dynamically adjust the sampling strategy during training, and `cost_matrix` allows the user to specify the mis-classification cost for each class. Besides, `eval_datasets`, `eval_metrics` and `train_verbose` control the content and granularity of the ensemble training log. Table 2 summarizes the data type, availability, and semantics of these keyword arguments.

We also make high-level abstractions for most of the included EIL methods based on the taxonomy in Table 1. For example, all EIL models that based on resampling and boosting (e.g., RusBoost, SmoteBoost) inherit the `ResampleBoostClassifier`, only with different samplers (e.g., `RandomUnderSampler`, `SMOTE`). New models can be easily implemented inside this framework by taking advantage of inheritance and polymorphism.

In addition to the EIL techniques, `imbens` also provides a versatile ensemble visualizer (`ImbalancedEnsembleVisualizer`) and a set of utility functions (`make_imbalance`, `generate_imbalance_data` and `evaluate_print`, etc.) to ease the EIL model exploration and evaluation. Code Snippet 1 is a demo showcasing how the deployment, evaluation, and visualization of EIL models can be conveniently conducted using the `imbens` API. Visualization examples provided by the `ImbalancedEnsembleVisualizer` are shown in Figure 1.

```
>>> from imbalanced_ensemble.ensemble import SelfPacedEnsembleClassifier
>>> from imbalanced_ensemble.datasets import generate_imbalance_data
>>> from imbalanced_ensemble.utils import evaluate_print
>>> from imbalanced_ensemble.visualizer import ImbalancedEnsembleVisualizer
>>>
>>> X_train, X_test, y_train, y_test = generate_imbalance_data(
...     n_samples=200, weights=[.9,.1], test_size=.5)
>>>
>>> clf = SelfPacedEnsembleClassifier()              # initialize ensemble
>>> clf.fit(X_train, y_train)
>>>
>>> y_test_pred = clf.predict(X_test)                # predict labels
>>> y_test_proba = clf.predict_proba(X_test)         # predict probabilities
>>>
>>> evaluate_print(y_test, y_test_pred, "SPE")       # performance evaluation
SPE balanced Acc: 0.972| macro Fscore: 0.886| macro Gmean: 0.972
>>>
>>> visualizer = ImbalancedEnsembleVisualizer()      # initialize visualizer
>>> visualizer.fit({'SPE': clf})
>>>
>>> visualizer.performance_lineplot()                # performance visualization
>>> visualizer.confusion_matrix_heatmap()            # prediction visualization
```

Code Snippet 1: Demo of `imbens` API with the SelfPacedEnsemble classifier.



(a) `confusion_matrix_heatmap()` (b) `performance_lineplot()`

Figure 1: Examples of using `ImbalancedEnsembleVisualizer` for visualization.

5. Conclusion and future plans

In this paper, we present `imbens`, a comprehensive Python toolbox for out-of-the-box ensemble class-imbalanced learning. As avenues for future work, we plan to include additional ensemble imbalanced learning methods that are based on evolutionary algorithm/meta-learning/hybrid-sampling, as well as more detailed examples, user guides and tutorials.

References

Ricardo Barandela, Rosa Maria Valdovinos, and José Salvador Sánchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3):245–256, 2003.

Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer, 2003.

Chao Chen, Andy Liaw, Leo Breiman, et al. Using random forest to learn imbalanced data. *University of California, Berkeley*, 110(1-12):24, 2004.

Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. Adacost: misclassification cost-sensitive boosting. In *Icml*, volume 99, pages 97–105. Citeseer, 1999.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

György Kovács. Smote-variants: A python implementation of 85 minority oversampling techniques. *Neurocomputing*, 366:352–354, 2019.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL `http://jmlr.org/papers/v18/16-365.html`.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

Zhining Liu, Wei Cao, Zhifeng Gao, Jiang Bian, Hechang Chen, Yi Chang, and Tie-Yan Liu. Self-paced ensemble for highly imbalanced massive data classification. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 841–852. IEEE, 2020.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1):185–197, 2010.

Grigoris Karakoulas John Shawe-Taylor and Grigoris Karakoulas. Optimizing classifiers for imbalanced training sets. *Advances in neural information processing systems*, 11(11):253, 1999.

Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System*, 14, 2001.

Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 324–331. IEEE, 2009.