The Journal of Open Source Software

# Imbalance: A comprehensive multi-interface Julia toolbox to address class imbalance

**Essam Wisam** [1*] **and Anthony Blaom** [2*]

**1** Cairo University, Egypt **2** University of Auckland, New Zealand * These authors contributed equally.

**DOI:** 10.21105/joss.06310

**Software**

- Review
- Repository
- Archive

**Editor:** Mehmet Hakan Satman

**Reviewers:**

- @sylvaticus
- @ArneTillmann

**Submitted:** 17 October 2023
**Published:** 18 March 2024



## Summary

Given a set of observations that each belong to a certain class, supervised classification aims to learn a classification model that can predict the class of a new, unlabeled observation (Cunningham et al., 2008). This modeling process finds extensive application in real-life scenarios, including but not limited to medical diagnostics, recommendation systems, credit scoring, and sentiment analysis.

In various real-world scenarios where supervised classification is employed, such as those pertaining to the detection of particular conditions like fraud, faults, pollution, or rare diseases, a severe discrepancy between the number of observations in each class can occur. This is known as class imbalance. This poses a problem if assumptions inherent in the classification model imply hindered performance when the model is trained on imbalanced data as is commonly the case (Ali et al., 2015). Two prevalent strategies for mitigating class imbalance, when it poses a problem to the classification model, involve either increasing the representation of less frequently occurring classes through oversampling or reducing instances of more frequently occurring classes through undersampling. It may be also possible to achieve even greater performance by combining both approaches in a sequential pipeline (Zeng et al., 2016) or by undersampling the data multiple times and training the classification model on each resampled dataset to form an ensemble model that aggregates results from different model instances (Liu et al., 2009). Contrary to undersampling, oversampling, or their combination, the ensemble approach possesses the ability to address class imbalance while making use of the entire dataset and without generating synthetic data.

## Statement of Need

A substantial body of literature in the field of machine learning and statistics is devoted to addressing the class imbalance issue. This predicament has often been aptly labeled the "curse of class imbalance," as noted in (Picek et al., 2018) and (Kubát & Matwin, 1997) which follows from the pervasive nature of the issue across diverse real-world applications and its pronounced severity; a classifier may incur an extraordinarily large performance penalty in response to training on imbalanced data.

The literature encompasses a myriad of oversampling and undersampling techniques to approach the class imbalance issue. These include SMOTE (Chawla et al., 2002) which operates by generating synthetic examples along the lines joining existing ones, SMOTE-N and SMOTE-NC (Chawla et al., 2002) which are variants of SMOTE that can handle categorical data. The sheer number of SMOTE variants makes them a body of literature on their own. Notably, the most widely cited variant of SMOTE is BorderlineSMOTE (Han et al., 2005). Other well-established oversampling techniques include RWO (Zhang & Li, 2014) and ROSE (Menardi & Torelli, 2012) which operate by estimating probability densities and sampling from them to generate synthetic points. On the other hand, the literature also encompasses many undersampling techniques.

Cluster undersampling (Lin et al., 2016) and condensed nearest neighbors (Hart, 1968) are two prominent examples that attempt to reduce the number of points while preserving the structure or classification boundary of the data. Furthermore, methods that combine oversampling and undersampling such as SMOTETomek (Zeng et al., 2016) are also present. The motivation behind these methods is that when undersampling is not random, it can filter out noisy or irrelevant oversampled data. Lastly, resampling with ensemble learning has also been presented in the literature with EasyEnsemble being the most well-known approach of that type (Liu et al., 2009).

The existence of a toolbox with techniques that harness this wealth of research is imperative to the development of novel approaches to the class imbalance problem and for machine learning research broadly. Aside from addressing class imbalance in a general machine learning research setting, such a toolbox can help in class imbalance research settings by making it possible to juxtapose different methods, compose them together, or form variants of them without having to reimplement them from scratch. In prevalent programming languages, such as Python, a variety of such toolboxes already exist, such as imbalanced-learn (Lemaître et al., 2016) and SMOTE-variants (Kovács, 2019). Meanwhile, Julia (Bezanson et al., 2017), a well-known programming language with over 40M downloads (Tuychiev, 2023), has been lacking a similar toolbox to address the class imbalance issue in general multi-class and heterogeneous data settings. This has served as the primary motivation for the creation of the `Imbalance.jl` toolbox, which we introduce in the subsequent section.

## Imbalance.jl

In this work, we present, `Imbalance.jl`, a software toolbox implemented in the Julia programming language that offers over 10 well-established techniques that help address the class imbalance issue. Additionally, we present a companion package, `MLJBalancing.jl`, which: (i) facilitates the inclusion of resampling methods in pipelines with classification models via the `BalancedModel` construct; and (ii) implements a general version of the EasyEnsemble algorithm presented in (Liu et al., 2009).

The toolbox offers a pure functional interface for each method implemented. For example, `SMOTE` can be used in the following fashion:

```
Xover, yover = smote(X, y)
```

Here `Xover, yover` are `X, y` after oversampling.

A `ratios` hyperparameter or similar is always present to control the degree of oversampling or undersampling to be done for each class. All hyperparameters for a resampling method have default values that can be overridden.

The set of resampling techniques implemented in either `Imbalance.jl` or `MLJBalancing.jl` are shown in the table below. Note that although no combination resampling techniques are explicitly presented, they are easy to form using the `BalancedModel` wrapper found in `MLJBalancing.jl` which can wrap an arbitrary number of resamplers in sequence.

**Table 1:** Resampling techniques implemented in `Imbalance.jl` and `MLJBalancing.jl`.

| Technique | Type | Supported Data Types |
|---|---|---|
| BalancedBaggingClassifier | Ensemble | Continuous and/or nominal |
| Borderline SMOTE1 | Oversampling | Continuous |
| Cluster Undersampler | Undersampling | Continuous |
| Edited Nearest Neighbors Undersampler | Undersampling | Continuous |
| Random Oversampler | Oversampling | Continuous and/or nominal |
| Random Undersampler | Undersampling | Continuous and/or nominal |

| Technique | Type | Supported Data Types |
|---|---|---|
| Random Walk Oversampler | Oversampling | Continuous and/or nominal |
| ROSE | Oversampling | Continuous |
| SMOTE | Oversampling | Continuous |
| SMOTE-N | Oversampling | Nominal |
| SMOTE-NC | Oversampling | Continuous and nominal |
| Tomek Links Undersampler | Undersampling | Continuous |

## Imbalance.jl Design Principles

The toolbox implementation follows a specific set of design principles in terms of the implemented techniques, interface support, developer experience and testing, and user experience.

### Implemented Techniques

- Should support all four major types of resampling approaches (oversampling, undersampling, combination, ensemble)
- Should be generally compatible with multi-class settings
- Should offer solutions to heterogeneous data settings (continuous and nominal data)
- When possible, preference should be given to techniques that are more common in the literature or industry

Methods implemented in the `Imbalance.jl` toolbox indeed meet all aforementioned design principles for the implemented techniques. The one-vs-rest scheme as proposed in (Fernández et al., 2013) was used to generalize binary technique to multi-class when needed.

### Interface Support

- Should support both matrix and table type inputs
- Target variable may or may not be given as a separate column
- Should expose a pure functional implementation, but also support popular Julia machine learning interfaces
- Should be possible to wrap an arbitrary number of resampler models with a classification model to behave as a unified model

Methods implemented in the `Imbalance.jl` toolbox meet all the interface design principles above. It particularly implements the `MLJ` (Blaom et al., 2020) and `TableTransforms` interface for each method. `BalancedModel` from `MLJBalancing.jl` also allows fusing an arbitrary number of resampling models and a classifier together to behave as one unified model.

### Developer Experience and Testing

- There should exist a developer guide to encourage and guide contribution
- Functions should be implemented in smaller units to aid in testing
- Testing coverage should be maximized; even the most basic functions should be tested
- Features commonly used by multiple resampling techniques should be implemented in a single function and reused
- Should document all functions, including internal ones
- Comments should be included to justify or simplify written implementations when needed

This set of design principles is also satisfied by `Imbalance.jl`. Implemented techniques are tested by testing smaller units that form them. Aside from that, end-to-end tests are performed for each technique by testing properties and characteristics of the technique or by using the `imbalanced-learn` toolbox (Lemaître et al., 2016) from Python and comparing outputs.

**User Experience**

- Functional documentation should be comprehensive and clear
- Examples (with shown output) that work after copy-pasting should accompany each method
- An illustrative visual example that presents a plot or animation should preferably accompany each method
- A practical example that uses the method with real data should preferably accompany each method
- If an implemented method lacks an online explanation, an article that explains the method after it is implemented should be preferably written

The `Imbalance.jl` documentation indeed satisfies this set of design principles. Methods are each associated with an example that can be copy-pasted, a visual example that demonstrates the operation of the technique, and possibly, an example that utilizes it with a real-world dataset to improve the performance of a classification model.

## Author Contributions

Design: E. Wisam, A. Blaom. Implementation, tests and documentation: E. Wisam. Code and documentation review: A. Blaom. The authors would like to acknowledge the financial support provided by the Google Summer of Code program, which made this project possible.

## References

Ali, A., Shamsuddin, S. M. Hj., & Ralescu, A. L. (2015). Classification with class imbalance problem: A review. *Soft Computing Models in Industrial and Environmental Applications*. https://api.semanticscholar.org/CorpusID:26644563

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Blaom, A. D., Király, F. J., Lienart, T., Simillides, Y., Arenas, D., & Vollmer, S. J. (2020). MLJ: A julia package for composable machine learning. *J. Open Source Softw.*, *5*, 2704. https://doi.org/10.21105/joss.02704

Chawla, N., Bowyer, K., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *ArXiv*, *abs/1106.1813*. https://doi.org/10.1613/jair.953

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. In M. Cord & P. Cunningham (Eds.), *Machine learning techniques for multimedia: Case studies on organization and retrieval* (pp. 21–49). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75171-7_2

Fernández, A., López, V., Galar, M., Jesús, M. J. del, & Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowl. Based Syst.*, *42*, 97–110. https://doi.org/10.1016/J.KNOSYS.2013.01.018

Han, H., Wang, W., & Mao, B. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. *International Conference on Intelligent Computing*. https://doi.org/10.1007/11538059_91

Hart, P. E. (1968). The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theory*, *14*, 515–516. https://doi.org/10.1109/TIT.1968.1054155

Kovács, G. (2019). Smote-variants: A Python implementation of 85 minority oversampling techniques. *Neurocomputing*, *366*, 352–354. https://doi.org/10.1016/j.neucom.2019.06.100

Kubát, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:18370956

Lemaître, G., Nogueira, F., & Aridas, C. K. (2016). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *ArXiv*, *abs/1609.06570*. https://api.semanticscholar.org/CorpusID:1426815

Lin, W.-C., Tsai, C.-F., Hu, Y.-H., & Jhang, J.-S. (2016). Clustering-based undersampling in class-imbalanced data. *Inf. Sci.*, *409*, 17–26. https://doi.org/10.1016/j.ins.2017.05.008

Liu, X.-Y., Wu, J., & Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*, 539–550. https://doi.org/10.1109/TSMCB.2008.2007853

Menardi, G., & Torelli, N. (2012). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, *28*, 92–122. https://doi.org/10.1007/s10618-012-0295-5

Picek, S., Heuser, A., Jović, A., Bhasin, S., & Regazzoni, F. (2018). The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, *2019*, 209–237. https://doi.org/10.13154/tches.v2019.i1.209-237

Tuychiev, B. (2023). *The rise of Julia*. https://www.datacamp.com/blog/the-rise-of-julia-is-it-worth-learn

Zeng, M., Zou, B., Wei, F., Liu, X., & Wang, L. (2016). Effective prediction of three common diseases by combining SMOTE with tomek links technique for imbalanced medical data. *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*, 225–228. https://doi.org/10.1109/ICOACS.2016.7563084

Zhang, H., & Li, M. (2014). RWO-sampling: A random walk over-sampling approach to imbalanced data classification. *Inf. Fusion*, *20*, 99–116. https://doi.org/10.1016/j.inffus.2013.12.003