





arfp: A Python package for adversarial random forests

Kristin Blesch ^{1,2} and Marvin N. Wright ^{1,2,3}

¹ Leibniz Institute for Prevention Research and Epidemiology – BIPS, Bremen, Germany ² Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany ³ Department of Public Health, University of Copenhagen, Copenhagen, Denmark  Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

In partnership with



AMERICAN
ASTRONOMICAL
SOCIETY

This article and software are linked with research article DOI [10.3847/xxxxx](https://doi.org/10.3847/xxxxx) <- [update this with the DOI from AAS once you know it.](#), published in the .

Summary

Generative modeling is a challenging task in machine learning that aims to synthesize new data which is similar to a set of given data. State of the art are computationally intense and tuning-heavy algorithms such as generative adversarial networks ([Goodfellow et al., 2014](#); [Xu et al., 2019](#)), variational autoencoders ([Kingma & Welling, 2014](#)), normalizing flows ([Rezende & Mohamed, 2015](#)), diffusion models ([Ho et al., 2020](#)) or transformers ([Vaswani et al., 2017](#)). A much more lightweight procedure is to use an Adversarial Random Forest (ARF) ([Watson et al., 2023](#)). ARFs achieve competitive performance in generative modeling in much faster runtime ([Watson et al., 2023](#)) and are especially useful for data that comes in a table format, i.e., tabular data. That is because ARFs are based on random forests ([Breiman, 2001](#)) that leverage the advantages that tree-based methods have over neural networks on tabular data (see ([Grinsztajn et al., 2022](#))) for generative modeling. Further, as part of the procedure, ARFs give access to the estimated joint density, which is useful for several other fields of research, e.g., unsupervised machine learning. For the task of density estimation, ARFs have been demonstrated to yield remarkable results as well ([Watson et al., 2023](#)). Hence, ARFs are a promising methodological contribution to the field of generative modeling and density estimation. To reach scholars in these fields that are predominantly based in python, and a broad audience more generally, a fast and userfriendly implementation of ARFs in python is highly desirable, which is provided by the software package arfp.

Statement of need

The package arfp implements density estimation and generative modeling with ARFs in python. ARFs have been introduced with a solid theoretical background, yet do not have to compromise on a complex algorithmic structure and instead are a low-key algorithm that does not require extensive hyperparameter tuning ([Watson et al., 2023](#)). This makes the methodology attractive for both scholars conducting rather theoretical research in statistics, e.g., density estimation, as well as practitioners from other fields that need to generate new data samples. Typical use cases of such synthesized data samples are, for example, the imputation of missing values, data augmentation or the conduct of analyses that respect data protection rules. With the speciality of ARFs being particularly suitable for tabular data, including a natural incorporation of both continuous and categorical features, the straightforward python implementation of ARFs provides a convenient algorithm to a broad audience from different fields.

ARFs have already gained some attention in the scientific community ([Nock & Guillaume-Bert, 2023](#)), however, the paper by Watson et al. ([2023](#)) provides the audience with only an R software package. The machine learning and generative modeling community however is mostly using python as a programming language. We aim to fill this gap with the presentend

python implementation of ARFs. arfpy is inspired by the R implementation arf (Wright & Watson, 2023), but transfers the algorithmic structure to match the class-based structure of python code and takes advantage of computationally efficient python functions. This is more robust and convenient to users than calling fragile wrappers like rpy2 Laurent Gautier et al. (2023) that attempt to make R code running in python. The benefits of a direct python implementation of ARFs for the generative modeling community have already been recognized by now. For example, arfpy is integrated in the data synthesizing framework synthcity by Schaar Lab (2023).

For interested readers, we briefly describe the ARF algorithm below, but refer to (Watson et al., 2023) for details. First, naive synthetic data is generated (initial generation step) by sampling from the marginal distributions of the features. Then, an unsupervised random forest (Shi & Horvath, 2006) is fit to distinguish this synthetic from real data (initial discrimination step). By doing so, the unsupervised random forest learns the dependency structure in the data. Using this forest, we can sample observations from the leaves to generate updated synthetic data (generation step). Subsequently, a new unsupervised random forest is fit to differentiate between synthetic and real data (discrimination step). Drawing on the adversarial idea of GANs, this iterative procedure of data generation and discrimination will be repeated until the discriminator cannot distinguish between generated and real data anymore. At this stage, the accuracy of the forest will be ≤ 0.5 and the forest is assumed to have converged, which implies mutually independent features in the terminal nodes. This facilitates the endeavor of density estimation and generative modeling drastically, as it allows us to formulate the univariate density for each feature separately and then combine them to the joint density, instead of having to model multivariate densities. For data generation, we can use this trait to sample a new observation by drawing a leaf from the forest of the last iteration step and use the data distributions with parameters estimated from that leaf to sample each feature separately.

Summarizing our contribution, arfpy introduces density estimation and generative modeling with ARFs to python. This enables practitioners from a wide variety of fields to generate fast and reliable synthetic data and density estimates using python as a programming language.

Acknowledgements

This work was supported by the German Research Foundation (DFG), Emmy Noether Grant 437611051. We thank David S. Watson and Jan Kapar for their contributions to establishing the theoretical groundwork of adversarial random forests.

References

- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. <https://doi.org/https://doi.org/10.1023/A:1010933404324>
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27.
- Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35, 507–520.
- Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33, 6840–6851.
- Kingma, D., & Welling, M. (2014). Auto-encoding variational bayes international. *Proceedings of the International Conference on Learning Representations (ICLR)*.

- 88 Laurent Gautier et al. (2023). rpy2: Python-r bridge. In *GitHub repository*. GitHub.
89 <https://github.com/rpy2/rpy2>
- 90 Nock, R., & Guillaume-Bert, M. (2023). *Generative forests*. <https://arxiv.org/abs/2308.03648>
- 91 Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In F. Bach
92 & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning*
93 (Vol. 37, pp. 1530–1538). PMLR. <https://proceedings.mlr.press/v37/rezende15.html>
- 94 Schaar Lab, van der. (2023). Synthcity: A library for generating and evaluating synthetic
95 tabular data. In *GitHub repository*. GitHub. <https://github.com/vanderschaarlab/synthcity>
- 96 Shi, T., & Horvath, S. (2006). Unsupervised learning with random forest predictors. *Journal*
97 *of Computational and Graphical Statistics*, 15(1), 118–138.
- 98 Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., &
99 Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing*
100 *Systems*, 30.
- 101 Watson, D. S., Blesch, K., Kapar, J., & Wright, M. N. (2023). Adversarial random forests
102 for density estimation and generative modeling. *International Conference on Artificial*
103 *Intelligence and Statistics*, 5357–5375.
- 104 Wright, M. N., & Watson, D. S. (2023). arf: Adversarial random forests. [https://CRAN.](https://CRAN.R-project.org/package=arf)
105 [R-project.org/package=arf](https://CRAN.R-project.org/package=arf)
- 106 Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular
107 data using conditional gan. *Advances in Neural Information Processing Systems*, 32.