The Journal of Open Source Software

`ReciPies`: A Lightweight Data Transformation Pipeline for Reproducible ML

Robin P. van de Water [1,2¶], **Hendrik Schmidt** [1], **and Patrick Rockenschaub** [3]

1 Hasso Plattner Institute, University of Potsdam, Potsdam, Germany **2** Hasso Plattner Institute for Digital Health at Mount Sinai, Icahn School of Medicine at Mount Sinai, New York City, NY, USA **3** Innsbruck Medical University, Innsbruck, Austria ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @simonprovost
- @panagiotisanagnostou

Submitted: 24 July 2025
Published: unpublished

Summary

Machine Learning (ML) workflows live or die by their data-preprocessing steps, yet in Python, these steps are often scattered across ad-hoc scripts or opaque Scikit-Learn (sklearn) snippets that are hard to read, audit, or reuse. `ReciPies` provides a concise, human-readable, and fully reproducible method to declare, execute, and share preprocessing pipelines, that adheres to Configuration as Code principles. It lets users describe transformations as a recipe made of ordered *steps* (e.g., imputing, encoding, normalizing) applied to variables identified by semantic roles (predictor, outcome, ID, time stamp, etc.). Recipes can be *prepped* once, *baked* many times, and separated between training and new data. `ReciPies` provides the choice of Pandas and Polars backends and is easily extensible. Data provenance can be tracked and published. Packaging preprocessing as clear, declarative objects, `ReciPies` lowers the cognitive load of feature engineering, improves reproducibility, and makes methodological choices explicit, benefiting individual researchers, engineering teams, and peer reviewers alike.

Statement of need

Robust machine-learning results in science hinge on transparent, reproducible data-preprocessing—yet in Python, these steps are typically spread across ad-hoc notebooks or are buried inside opaque scripts; that is, if this code is even made available. Additionally, most variable semantics are unclear (called *roles*). These problems confound research results, complicate peer review and hinder reuse. Researchers and engineers working with longitudinal regulated data (e.g., energy production, finance, and environmental monitoring) especially need pipelines they are able to audit, serialize, and hand to collaborators without reverse-engineering a tangle of imperative code (Various, 2024). The lack of reproducibility has been documented extensively in literature (Johnson et al., 2017; Kelly et al., 2019; Semmelrock et al., 2025).

`ReciPies` fills this gap by bringing a tidy, stepwise *recipe* interface to Python. Users declare transformations over variables selected by semantic roles; recipes are "prepped" once on training data and "baked" on new data to eliminate leakage; and every step is inspectable, versionable, and serializable (JSON/YAML). `Recipes` runs on Pandas (McKinney, 2010) and Polars (Vink et al., 2024) for interoperability and performance, and their Object-oriented abstractions enable users to implement custom steps. The framework is declarative and reproducible for data preprocessing, prioritizing human readability and methodological transparency. We demonstrate that there is no need to sacrifice readability for performance or flexibility for simplicity. By reducing the cognitive overhead of feature engineering and making methodological choices explicit, `ReciPies` enables researchers to focus on their core work. We hope this will broaden the reproducibility discussion in ML from hyperparameters to the entire experiment pipeline.

We encourage the development of domain-specific step libraries and integration patterns that can benefit the broader ecosystem.

Related Work

Our work brings the Recipes (Kuhn et al., 2024) framework to Python and extends it for the ML community. The design enables straightforward integration as part of a pipeline that includes ML libraries like sklearn (Pedregosa et al., 2011) and PyTorch(Paszke et al., 2019). To the best of our knowledge, no other packages comply with the flexibility and reproducibility of `ReciPies` and its Configuration as Code approach. Sklearn offers composable transformers, but no role-based variable grammar, limited human-readability, and awkward serialization. Feature-engine (Galli, 2021), pyjanitor (J. et al., 2019), or scikit-lego(warmerdam et al., 2025) add helpful transformers or cleaning verbs. However, none provide a unified, declarative recipe abstraction with a strict "prep/bake" split and backend flexibility. `ReciPies` provides a stepwise recipe that is easy to use and read, allowing users to readily preprocess data for a wide range of machine learning pipelines.

Usage

If we have a dataset, `df`, with a label `y`, some features `x1`, `x2`, `x3`, `x4`, an identifier `id`, and a sequential component `time`, we can build a preprocessing pipeline using `ReciPies`. We first do a train/test split:

```
df_train, df_test = train_test_split(df, test_size=0.2, random_state=42)
```

We then define the roles of the variables in this dataset:

```
roles = {outcomes:["y"], predictors=["x1", "x2", "x3", "x4"], groups=["id"],
sequences=["time"]}
```

Afterward, we create the `ingredients` which encapsulate the training data and its roles, and the `recipe` to preprocess the data:

```
ing = Ingredients(df_train, roles=roles)
rec = Recipe(ing)
```

We add preprocessing steps:

```
rec.add_step(StepScale())
rec.add_step(StepSklearn(MissingIndicator(features="all"),
  sel=has_role("predictor")))
rec.add_step(StepImputeFill(strategy="forward"))
rec.add_step(StepSklearn(LabelEncoder(), sel=has_type("categorical"),
  columnwise=True))
```

We can now fit the recipe and transform both the train and test set without leakage and in a transparent manner:

```
df_train = rec.prep()
df_test = rec.bake(df_test)
```

We can use the bake method on the training set to transform it again, e.g., to apply the same transformations to a new dataset. Complete code, benchmarks, and interactive notebooks are available in the project documentation. `ReciPies` also provides a benchmarking suite with results to compare the performance of different preprocessing steps on (generated) data.

`ReciPies` is used as the bedrock of reproducible pipelines of Yet Another ICU Benchmark (Van de Water et al., 2024) The adaptable, configurable code modules that make extensive use of RecipPies can be found here; this demonstrates that RecipPies can be used for arbitrary research domains.

Future steps

We plan to expand the library of Polars-native steps to fully leverage its columnar execution model, particularly for time-series operations and large-scale aggregations, where Polars shows significant performance advantages. We envision RecipPies recipes as portable preprocessing artifacts that can be versioned, tracked, and deployed across different environments. Tighter integration with experiment tracking and model registries would streamline the transition from research to production, a complex process in many application domains.

Galli, S. (2021). Feature-engine: A Python package for feature engineering for machine learning. *Journal of Open Source Software*, *6*(65), 3642. https://doi.org/10.21105/joss.03642

J., E., Barry, Z., Zuckerman, S., & Sailer, Z. (2019). Pyjanitor: A Cleaner API for Cleaning Data. *Proceedings of the Python in Science Conference*, 50–53. https://doi.org/10.25080/majora-7ddc1dd1-007

Johnson, A. E. W., Pollard, T. J., & Mark, R. G. (2017). Reproducibility in critical care: A mortality prediction case study. *Proceedings of the 2nd Machine Learning for Healthcare Conference*, 361–376. https://doi.org/10.1038/s41597-022-01899-x

Kelly, C. J., Karthikesalingam, A., Suleyman, M., Corrado, G., & King, D. (2019). Key challenges for delivering clinical impact with artificial intelligence. *BMC Medicine*, *17*(1), 195. https://doi.org/10.1186/s12916-019-1426-2

Kuhn, M., Wickham, H., Hvitfeldt, E., Software, P., & PBC. (2024). *Recipes: Preprocessing and Feature Engineering Steps for Modeling* (Version 1.1.0). https://doi.org/10.32614/CRAN.package.recipes

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, *32*. https://doi.org/10.48550/arXiv.1912.01703

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. https://doi.org/10.48550/arXiv.1201.0490

Semmelrock, H., Ross-Hellauer, T., Kopeinik, S., Theiler, D., Haberl, A., Thalmann, S., & Kowald, D. (2025). Reproducibility in machine-learning-based research: Overview, barriers, and drivers. *AI Magazine*, *46*(2), e70002. https://doi.org/10.1002/aaai.70002

Van de Water, R., Schmidt, H. N. A., Elbers, P., Thoral, P., Arnrich, B., & Rockenschaub, P. (2024, May 7). Yet Another ICU Benchmark: A Flexible Multi-Center Framework for Clinical ML. *Proceedings of The Twelfth International Conference on Learning Representations*. The Twelfth International Conference on Learning Representations. https://doi.org/10.48550/arXiv.2306.05109

Various, A. (2024). ACM REP '24: Proceedings of the 2nd ACM conference on reproducibility and replicability. *ACM REP '24: Proceedings of the 2nd ACM Conference on Reproducibility and Replicability*. https://doi.org/10.1145/3641525

Vink, R., Gooijer, S. de, Beedie, A., Gorelli, M. E., Guo, W., Zundert, J. van, Peters, O., Hulselmans, G., Grinstead, C., nameexhaustion, Marshall, chielP, Burghoorn, G., Turner-Trauring, I., Santamaria, M., Heres, D., Magarick, J., ibENPC, Wilksch, M., … Koutsouris, I. (2024). *Pola-rs/polars: Python Polars 1.0.0*. Zenodo. https://doi.org/10.5281/zenodo.12606903

warmerdam, vincent d, Bruzzesi, F., MBrouns, Collot, S., Boer, J. de, Kübler, R., pim-hoeven, mkalimeri, Paulino, A., Gorelli, M. E., Verheijen, P., Borry, M., Hoogland, K., Masip, D., Kowalczuk, M., ktiamur, AminaZ, Sharma, G., Lepelaars, C., … Payne, S. (2025). *Koaning/scikit-lego: V0.9.5*. Zenodo. https://doi.org/10.5281/zenodo.15313097