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: 10.21105/joss.09261

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @simonprovost
- @panagiotisanagnostou

Submitted: 24 July 2025
Published: 05 January 2026

Summary

Machine Learning (ML) workflows live or die by their data-preprocessing steps. In Python, these steps are often scattered across ad-hoc scripts or opaque scikit-learn snippets that are hard to read, audit, or reuse. `ReciPies` provides a concise, human-readable, and reproducible way to declare, execute, and share preprocessing pipelines following configuration-as-code principles. It lowers the cognitive load of feature engineering, improves reproducibility, and makes methodological choices explicit for researchers, engineering teams, and peer reviewers.

Statement of need

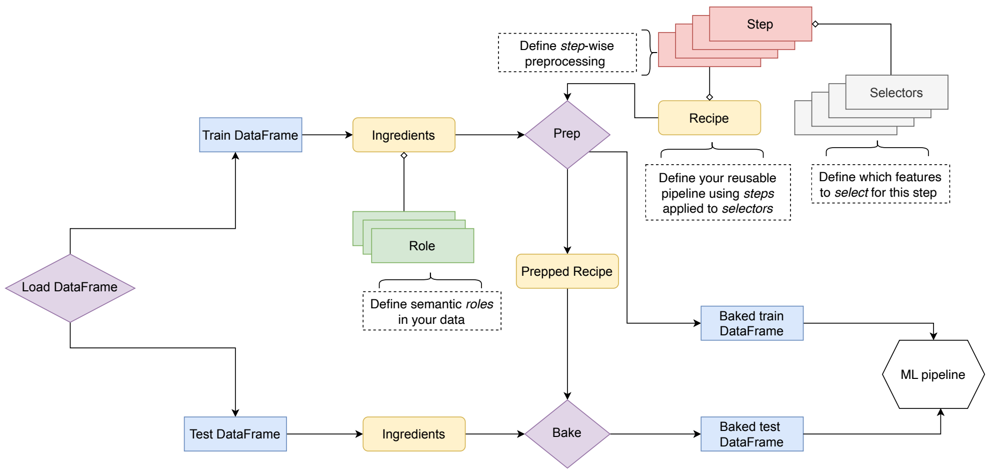
Transparent and reproducible preprocessing remains a weak link in many scientific ML studies. The consequences are (1) confounded research results (Gundersen & Kjensmo, 2018), (2) complicated peer review (Semmelrock et al., 2025), and (3) poor reuse (Samuel et al., 2021). Researchers and engineers working with longitudinal regulated data (e.g., in energy production, health, finance, or environmental monitoring) in particular need pipelines they can audit, serialize, and hand to collaborators without reverse-engineering a tangle of imperative code. The current lack of reproducibility has been documented extensively in the literature (Gundersen & Kjensmo, 2018; Johnson et al., 2017; Kelly et al., 2019; Raff, 2019; Semmelrock et al., 2025); moreover, scientific venues have begun to address this issue (Various, 2024).

Related work

Scikit-learn provides `Pipeline` and `ColumnTransformer`, along with a rich estimator ecosystem (Pedregosa et al., 2011), but lacks 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 and data-cleaning verbs. However, none provide a unified, role-centric abstraction with backend flexibility. The R `recipes` package established the prep/bake pattern and a clean grammar for preprocessing (Kuhn et al., 2024). `ReciPies` brings these ideas to Python, extends them with backend-agnostic execution on Pandas and Polars, and emphasizes configuration-as-code artifacts suitable for a wide range of machine learning pipelines.

Design and implementation

`ReciPies` adopts a tidy, stepwise *recipe* interface that emphasizes semantic roles over column names and a strict separation of fitting from application. Transformations are declared on roles such as predictor, outcome, identifier, or timestamp. Recipes are prepped on training data and baked on new data to prevent leakage. Each step is inspectable, versionable, and serializable to JSON or YAML for provenance and review. Steps are composable with explicit state and deterministic behavior given fixed inputs and seeds. `RecIPies` supports both Pandas (McKinney, 2010), which is widely adopted in the ML community, and the more recent Polars (Vink et al., 2024), which offers increased performance.



Step
Define *step*-wise preprocessing
Selectors
Train DataFrame
Ingredients
Prep
Recipe
Define your reusable pipeline using *steps* applied to *selectors*
Define which features to *select* for this step
Role
Define semantic *roles* in your data
Prepped Recipe
Load DataFrame
Baked train DataFrame
ML pipeline
Test DataFrame
Ingredients
Bake
Baked test DataFrame

A typical workflow 1) loads a Pandas or Polars training `DataFrame`, 2) wraps it as an `Ingredients` object that records role metadata, 3) defines a `Recipe` from `Steps` operating on columns selected based on roles by `Selectors`, 4) preps the recipe on the training split to estimate parameters, and 5) bakes it on the held-out split to apply those parameters without leakage. The baked outputs feed downstream modeling and evaluation. Figure 1 gives an overview of this workflow.

```
import polars as pl
from sklearn.model_selection import train_test_split
from sklearn.impute import MissingIndicator
from recipies import Ingredients, Recipe
from recipies.selector import has_role
from recipies.step import StepImputeFill, StepScale, StepSklearn

# Load and split Physionet Computing in Cardiology Challenge 2019 dataset
df = pl.read_csv("Physionet_CiCC_2019.csv", sep="|")
df_train, df_test = train_test_split(df, test_size=0.2, random_state=42)

# Define ingredients (training set + roles)
roles = {"outcomes": ["SepsisLabel"], "predictors": ["Age", "HR", "Temp"], "groups": ["PatientID"],
"sequences": ["ICULOS"]}
ing = Ingredients(df_train, roles=roles)

# Define the recipe for processing the ingredients (e.g., scaling and imputation)
rec = Recipe(ing)
rec.add_step(StepScale())
rec.add_step(StepSklearn(MissingIndicator(features="all"), sel=has_role("predictor")))
rec.add_step(StepImputeFill(strategy="forward"), sel=has_role("predictor"))

# Prepare (=fit) recipe on training data and apply the same recipe to the test set
df_train = rec.prep()
df_test = rec.bake(df_test)
```

Figure 2 demonstrates usage on the PhysioNet Computing in Cardiology 2019 dataset (Reyna et al., 2020), including role assignment, temporal imputation, and normalization. The prepped recipe serializes to JSON or YAML, and reloading the artifact reproduces the transforms across supported platforms.

Complete code and interactive notebooks are available in the project documentation. `RecIPies` also provides a benchmarking suite comparing the performance of different preprocessing steps on (generated) data. `RecIPies` is already used as the bedrock of reproducible pipelines in Yet Another ICU Benchmark (Van de Water et al., 2024) The adaptable, configurable code modules that make extensive use of `RecıPies` can be found here; this demonstrates that `RecıPies` can be used for arbitrary research domains. Our work shows that there is no need to sacrifice readability for performance, nor flexibility for simplicity. We encourage the development of domain-specific step libraries and integration patterns that can benefit the broader ecosystem.

Future steps

Our first step is 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. Second, we aim to integrate with ML versioning systems to streamline the transition from research to production.

Acknowledgements

Robin P. van de Water is funded by the European Commission in the Horizon 2020 project INTERVENE (Grant agreement ID: 101016775). This work has been edited with the help of Large Language Models (LLMs) to improve readability.

References

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

Gundersen, O. E., & Kjensmo, S. (2018). State of the art: Reproducibility in artificial intelligence. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 1644–1651. https://doi.org/10.1609/aaai.v32i1.11503

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/10.1101/2024.06.04.24308417

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

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://dl.acm.org/doi/10.5555/1953048.2078195

Raff, E. (2019). A step toward quantifying independently reproducible machine learning research. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 5485–5495. https://dl.acm.org/doi/10.5555/3454287.3454779

Reyna, M. A., Josef, C. S., Jeter, R., Shashikumar, S. P., Westover, M. B., Nemati, S., Clifford, G. D., & Sharma, A. (2020). Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019. *Critical Care Medicine*, *48*(2), 210. https://doi.org/10.1097/CCM.0000000000004145

Samuel, S., Löffler, F., & König-Ries, B. (2021). Machine Learning Pipelines: Provenance, Reproducibility and FAIR Data Principles. In B. Glavic, V. Braganholo, & D. Koop (Eds.), *Provenance and Annotation of Data and Processes* (pp. 226–230). Springer International Publishing. https://doi.org/10.1007/978-3-030-80960-7_17

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, V. 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