

# VeridicalFlow: a python package for building build stable, trustworthy data-science pipelines

James Duncan<sup>\*1</sup>, Rush Kapoor<sup>†2</sup>, Abhineet Agarwal<sup>‡3</sup>, Chandan Singh<sup>§2</sup>, and Bin Yu<sup>1, 2</sup>

<sup>1</sup> Statistics Department, University of California, Berkeley <sup>2</sup> EECS Department, University of California, Berkeley <sup>3</sup> Physics Department, University of California, Berkeley

## Summary

VeridicalFlow is a Python package for simplifying building reproducible and trustworthy data-science pipelines. It provides users a simple interface for stability analysis, i.e. checking the robustness of results from a data-science pipeline to various judgement calls made during modeling. This ensures that arbitrary judgement calls made by data-practitioners (e.g. specifying a default imputation strategy) do not dramatically alter the final conclusions made in a modeling pipeline. In addition to wrappers facilitating stability analysis, VeridicalFlow also automates many cumbersome coding aspects of python pipelines, including experiment tracking and saving, parallelization, and caching, all through integrations with existing python packages.

## Statement of need

Stability is a central concern in modern statistical practice, as practitioners make many judgement calls during the data-science life cycle which often go unchecked (Yu & Kumbier, 2020). It is a common-sense principle related to notions of scientific reproducibility Ivie & Thain (2018) and sensitivity analysis (Saltelli, 2002). Moreover, stability serves as a prerequisite for understanding which parts of a model will generalize and can be interpreted (Murdoch et al., 2019).



Importantly, current software packages offer very little support to facilitate stability analyses. VeridicalFlow helps fill this gap by making stability analysis simple, reproducible, and computationally efficient. This enables a practitioner to represent a pipeline with many different perturbations in a simple-to-code way.

## Features

Using VeridicalFlows's simple wrappers easily enables many best practices for data science, and makes writing pipelines easy.

DOI: [DOIunavailable](#)

### Software

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

Editor: [Pending Editor](#) 

### Reviewers:

- [@Pending Reviewers](#)

Submitted: N/A

Published: N/A

### License

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

---

\*Equal contribution

†Equal contribution

‡Equal contribution

§Equal contribution

Stability	Computation	Reproducibility
Replace a single function (e.g. preprocessing) with a set of functions and easily assess the stability of downstream results	Automatic parallelization and caching throughout the pipeline	Automatic experiment tracking and saving

The main features of `VeridicalFlow` center around stability analysis. The central concept is to replace given functions with a set of functions subject to different pipeline perturbations. Then, a set of useful analysis functions and computations enable easily assessing the stability to these perturbations.

The package also helps users to improve the efficacy of their computational pipeline. Computation is (optionally) handled through Ray (Moritz et al., 2018), which easily facilitates parallelization across different machines and along different perturbations of the pipeline. Caching is handled via `jobjlib`, so that individual parts of the pipeline do not need to be rerun.

Experiment-tracking and saving are (optionally) handled via integration with MLFlow (Zaharia et al., 2018), which enables automatic experiment tracking and saving.

## Acknowledgements

The code here heavily derives from the wonderful work of previous projects. It hinges on the data-science infrastructure of python, including packages such as pandas (McKinney & others, 2011), numpy (Van Der Walt et al., 2011), and scikit-learn (Pedregosa et al., 2011) as well as newer projects such as imodels (Singh et al., 2021) and networkx (Hagberg & Conway, n.d.).

## References

- Fisher, R. A., & others. (1937). The design of experiments. *The Design of Experiments.*, 2nd Ed.
- Hagberg, A., & Conway, D. (n.d.). *NetworkX: Network analysis with python*.
- Ivie, P., & Thain, D. (2018). Reproducibility in scientific computing. *ACM Computing Surveys (CSUR)*, 51(3), 1–36.
- McKinney, W., & others. (2011). Pandas: A foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14(9), 1–9.
- Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & others. (2018). Ray: A distributed framework for emerging {AI} applications. *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 561–577.
- Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44), 22071–22080. <https://doi.org/10.1073/pnas.1900654116>
- Pedregosa, F., Varoquaux, G. ë. I., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12, 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>
- Saltelli, A. (2002). Sensitivity analysis for importance assessment. *Risk Analysis*, 22(3), 579–590.

- Singh, C., Nasser, K., Tan, Y. S., Tang, T., & Yu, B. (2021). Imodels: A python package for fitting interpretable models. *Journal of Open Source Software*, 6(61), 3192.
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22–30.
- Yu, B., & Kumbier, K. (2020). Veridical data science. *Proceedings of the National Academy of Sciences*, 117(8), 3920–3929.
- Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., & others. (2018). Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.*, 41(4), 39–45.