The Journal of Open Source Software

TanML: Automated Model Validation Toolkit for Tabular Machine Learning Models

Tanmay Sah [1], **Dolly Sah** [2], **Harshul Jain** [3], **Tanya Sah** [3], **and Kayden Jordan**[1]

1 Harrisburg University of Science and Technology **2** University of Utah **3** New York University

DOI: 10.21105/joss.11253

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @crvernon

Submitted: 03 September 2026
Published: 04 September 2026

Summary

TanML (Sah et al., 2026) is an open-source toolkit for validating machine learning models built on tabular (structured) data through a user friendly Streamlit interface. It supports an end-to-end workflow that includes data profiling, preprocessing, feature ranking, model development, evaluation, and export of editable audit ready reports. The toolkit is designed to help users validate models built with common Python ML libraries such as scikit-learn (Pedregosa et al., 2011), XGBoost (Chen & Guestrin, 2016), statsmodels (Seabold & Perktold, 2010), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). TanML is especially useful in regulated and documentation heavy settings (Board of Governors of the Federal Reserve System, 2011) because it combines model validation, explainability, drift and stress analysis, and report generation in a single local first workflow. It is intended for data scientists, quantitative analysts, and model risk stakeholders who need a practical way to review and document ML models trained on structured datasets without building custom validation pipelines from scratch.

Statement of need

Machine learning models trained on structured, tabular data are widely used in high-stakes domains such as finance, insurance, and other regulated settings. In these environments, building a predictive model is only one part of the workflow, because practitioners must also assess data quality, model behavior, predictive performance, robustness, and interpretability while producing documentation that supports internal review, governance, and compliance processes. For example, banking institutions developing credit scoring or fraud detection models are required to produce comprehensive model validation reports documenting data integrity, performance benchmarking, and sensitivity analyses before regulatory approval (Board of Governors of the Federal Reserve System, 2011). Similarly, insurance firms building claims models must demonstrate that their automated systems are fair, explainable, and robust to distributional shifts (OECD, 2020).

In practice, these validation activities are often fragmented across notebooks, scripts, visualization libraries, and manually prepared reports, making workflows difficult to standardize, reproduce, and communicate, especially when results must be reviewed by stakeholders beyond the original model developer. TanML was developed to address this problem by providing a unified workflow for validating and documenting models trained on tabular data. Instead of requiring practitioners to assemble separate tools for analysis, evaluation, and reporting, TanML organizes these activities within a single interface intended to make validation outputs easier to reproduce, review, and communicate. The primary target audience includes data scientists, quantitative analysts, model validation teams, and other practitioners working in regulated or documentation heavy settings who need reviewable validation artifacts in addition

to numerical metrics or exploratory analysis.

State of the field

The open source ecosystem already includes several widely used tools that address important parts of the machine learning lifecycle for structured data. While tools exist to support individual components of this workflow, TanML brings all components together in a single software package. Data profiling packages such as YData Profiling (YData, 2026) emphasize exploratory analysis and dataset understanding. Monitoring frameworks such as Evidently (Evidently AI, 2026) focus on drift detection and production observability. Validation libraries such as Deepchecks (Deepchecks, 2024) provide programmable checks for data and model quality. AutoML frameworks such as PyCaret (PyCaret, 2024) and AutoGluon (Erickson et al., 2020) focus on model training, comparison, and selection. These tools are valuable within their intended scope, but their primary objectives differ from TanML's integrated validation-and-documentation focus. Profiling tools are centered on dataset exploration rather than end-to-end model validation. Monitoring tools are designed mainly for observability and post-deployment drift analysis rather than pre-deployment validation workflows. Validation libraries provide useful checks, but they are generally oriented toward developer-driven testing rather than stakeholder facing validation workflows. AutoML systems improve modeling efficiency, but they do not primarily address governance, documentation, or audit-ready reporting requirements.TanML was developed to fill this gap by integrating validation-oriented tasks into a single workflow for structured-data machine learning. Rather than replacing existing tools individually, it combines model evaluation, explainability, drift analysis, and stakeholder-ready reporting in one UI-driven system. This also motivates the build-versus-contribute decision. TanML's scholarly contribution lies at the workflow level rather than in a single isolated method. The software was built as a standalone package because existing profiling, monitoring, testing, and AutoML tools do not fully address the combined need for tabular model validation and stakeholder-ready documentation.

Table 1: Comparison of TanML with commonly used tools across primary focus, scope, outputs, and drift-analysis approach.

Tool	Primary Focus	Scope	Output	Drift Logic
TanML	Model Risk Management Governance	Data, Model,	Audit-ready report	PSI and KS
Evidently AI	Monitoring	Data, Model	Dashboards	Statistical tests
Deepchecks	Testing / CI	Data, Model	Reports	Multiple methods
AutoML	Model Training	Model building	Models	N/A
YData Profiling	Data EDA	Data only	HTML report	Warnings

Software design

TanML was explicitly designed as a modular, privacy-first desktop application that balances analytical rigor with non-developer accessibility. By rejecting a traditional web-based frontend (e.g., React/JS) and instead managing the entire application flow locally in pure Python via **Streamlit** (Streamlit, Inc., 2026), the toolkit lowers the barrier to entry for quantitative analysts and model risk practitioners. This trade-off allows users to inspect, extend, and deploy the UI code directly alongside their statistical models without requiring dedicated web engineering teams. Importantly, the local-first architecture ensures that sensitive financial and regulatory

data never leaves the user's machine. In domains such as banking and insurance, transmitting customer-level loan data or proprietary risk scores to cloud-hosted services may violate internal data governance policies or regulatory requirements. TanML avoids this risk entirely by keeping all computation and data in the user's local environment.

Core architecture

TanML is built on three foundational components that collectively enable its modular workflow: a session-based data management layer that preserves data privacy, a dynamic model registry for extensibility, and a document generation engine for automated reporting.

The first pillar is a centralized **Session State Manager** that handles data flow between modules without requiring a persistent database. This approach allows sensitive financial data to remain local to the user's environment, either held transiently in memory or stored in ephemeral local directories, rather than being transmitted to external hosted services. For example, when a practitioner uploads a loan-level dataset for credit model validation, the data persists only for the duration of the session and is never written to a remote store.

The second pillar is the **Model Registry**, a dynamic factory pattern (`tanml.models.registry`) that standardizes the instantiation of various estimators (XGBoost, CatBoost, LightGBM) with pre-configured hyperparameters. This decoupling allows researchers to easily inject novel models without altering the core validation engine UI.

The third pillar is the **Reporting Engine**, a `docx`-based generator that serializes analysis results into a structured document, mapping complex Python objects (such as **Matplotlib** (Hunter, 2007) figures and **Pandas** (McKinney, 2010) DataFrames) into native Open XML formats. The resulting Word document is immediately editable, allowing validation teams to annotate findings, append narrative commentary, and route the report through internal review processes without reformatting.

Modular workflow

The application is divided into distinct execution layers, designed to mirror the standard Model Risk Management lifecycle.

The **Data Profiling and Preprocessing** layer handles exploratory analysis, automated cleaning, high cardinality encoding, and missing value imputation. For instance, a risk analyst examining a credit scoring dataset can review distributional summaries, identify missing value patterns across borrower attributes, and apply encoding strategies for categorical features such as loan purpose or geographic region, all within a guided interface before any model is trained.

The **Feature Power Ranking** layer assesses predictive power and statistical significance (p-values) before modeling. This allows practitioners to identify which features contribute meaningfully to the target variable and to document the rationale for feature inclusion or exclusion, a common requirement in model validation reports.

The **Model Development** layer facilitates champion-challenger comparisons using K-Fold Cross-Validation. Users can train and compare multiple estimators side-by-side, selecting a champion model while retaining challenger results for documentation purposes.

The **Evaluation Suite** is the core validation engine. It calculates Population Stability Index (PSI) for drift analysis (Yurdakul, 2018), tree-based SHAP values (Lundberg & Lee, 2017) for explainability, and segmented performance metrics. For example, a model risk team validating an insurance claims model can use PSI to assess whether the production data has drifted from the training distribution, examine SHAP plots to verify that feature contributions align with domain expectations, and review performance breakdowns across customer segments to detect potential fairness or stability issues. These outputs are then serialized directly into the generated validation report.

To illustrate a complete workflow: a quantitative analyst at a financial institution receives a challenger credit model and its associated dataset. Using TanML, the analyst profiles the data to check for quality issues, ranks features by predictive power, retrains the model under cross-validation to verify reported performance, runs drift and SHAP analyses against a holdout or production sample, and exports a structured Word report. This report can then be circulated to model governance committees and regulators with minimal additional preparation. To illustrate the system flow, **Figure 1** provides a modular overview of the end-to-end pipeline and specific validation functionalities included in the toolkit.

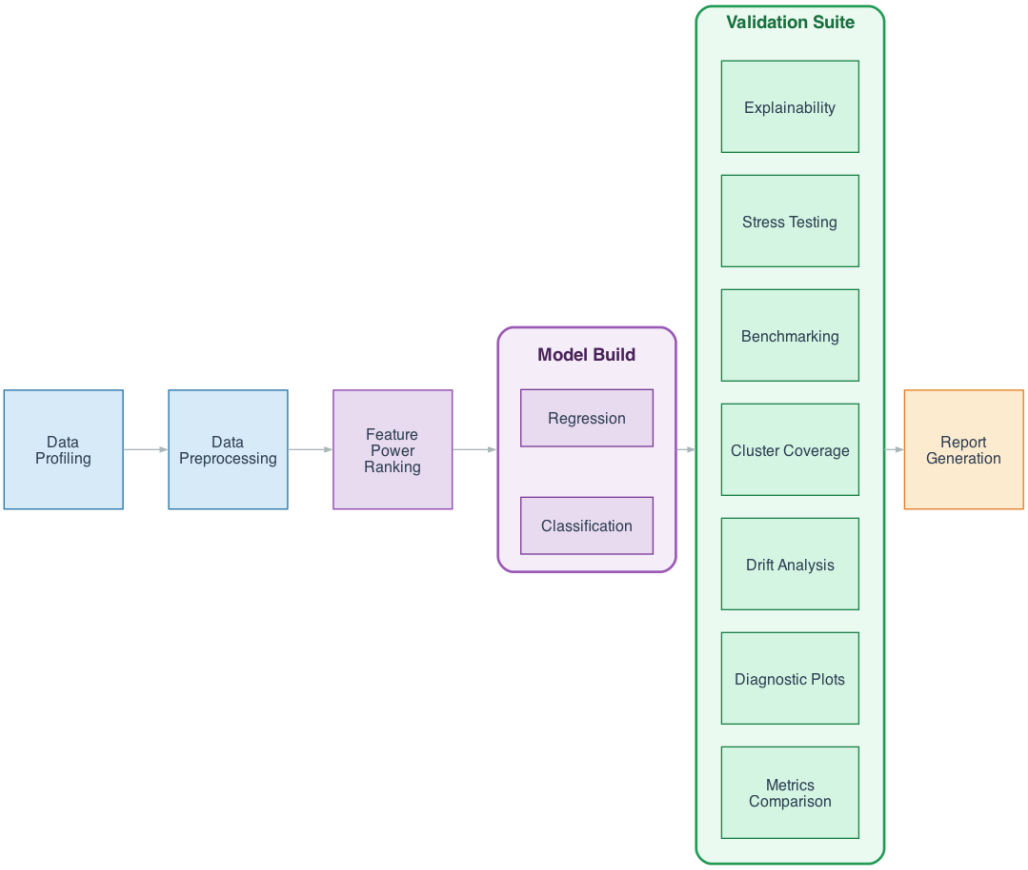

Figure 1: High-level modular workflow of TanML.

Research impact statement

TanML has accumulated over 5,000 downloads on PyPI since its initial release in July 2025. The toolkit is packaged and installable via `pip install tanml`, archived on Zenodo with a persistent DOI (Sah et al., 2026), and documented through a published MkDocs site. The repository includes an automated test suite run via GitHub Actions CI, a `CONTRIBUTING.md` guide, a code of conduct, and an open issue tracker to support community contributions. Reproducible demo datasets and example workflows are provided in the repository so that new users can evaluate the toolkit's end-to-end validation capabilities on representative tabular data without requiring proprietary datasets. The modular architecture, described in the Software design section, provides a clear extension pathway for contributors seeking to add new model families, evaluation metrics, or report templates.

AI usage disclosure

Portions of the TanML codebase, including specific unit tests and documentation templates, were refactored with the assistance of Large Language Models (LLMs). The human maintainers have reviewed and verified all AI-generated contributions to ensure technical accuracy.

Acknowledgements

We acknowledge the valuable feedback from the pyOpenSci editor and reviewers, whose insights significantly shaped and improved the toolkit. We also express our appreciation to the maintainers and contributors of the open-source libraries that TanML builds upon, including NumPy, pandas, SciPy, Numba, scikit-learn, statsmodels, XGBoost, matplotlib, seaborn, Pillow, python-docx, Streamlit, openpyxl, pyarrow, and pyreadstat. In addition, we are grateful to all those who shared ideas, suggestions, and feature requests that helped inform the development of TanML.

References

Board of Governors of the Federal Reserve System. (2011). *Supervisory letter SR 11-7: Guidance on model risk management*. https://www.federalreserve.gov/supervisionreg/srletters/sr1107.htm

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Deepchecks. (2024). *Deepchecks*. https://doi.org/10.5281/zenodo.6344777

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. J. (2020). AutoGluon-tabular: Robust and accurate AutoML for structured data. *arXiv Preprint arXiv:2003.06505*. https://doi.org/10.48550/arXiv.2003.06505

Evidently AI. (2026). *Evidently*. https://github.com/evidentlyai/evidently

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems 30*. https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems 30*. https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56. https://doi.org/10.25080/Majora-92bf1922-00a

OECD. (2020). *The impact of big data and artificial intelligence (AI) in the insurance sector*. OECD Publishing. https://doi.org/10.1787/c822ee53-en

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://www.jmlr.org/papers/v12/pedregosa11a.html

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems 31*. https://doi.org/10.5555/3327757.3327770

PyCaret. (2024). *PyCaret*. https://github.com/pycaret/pycaret

Sah, T., Sah, D., Jain, H., Sah, T., & Jordan, K. (2026). *TanML: Automated model validation toolkit for tabular machine learning*. Zenodo. https://doi.org/10.5281/zenodo.22285793

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. *Proceedings of the 9th Python in Science Conference*, 92–96. https://conference.scipy.org/proceedings/scipy2010/pdfs/statsmodels.pdf

Streamlit, Inc. (2026). *Streamlit*. https://github.com/streamlit/streamlit

YData. (2026). *YData Profiling*. https://doi.org/10.5281/zenodo.10123516

Yurdakul, B. (2018). *Statistical properties of the population stability index* [Doctoral dissertation, Western Michigan University]. https://scholarworks.wmich.edu/dissertations/3208