PiML Toolbox for Interpretable Machine Learning Model Development and Diagnostics

Agus Sudjianto, Aijun Zhang*, Zebin Yang, Yu Su and Ningzhou Zeng

May 14, 2023

Abstract

PiML (read π-ML, /‘pai·‘em·‘el/) is an integrated and open-access Python toolbox for interpretable machine learning model development and model diagnostics. It is designed with machine learning workflows in both low-code and high-code modes, including data pipeline, model training, model interpretation and explanation, and model diagnostics and comparison. The toolbox supports a growing list of interpretable models (e.g. GAM, GAMI-Net, XGB2) with inherent local and/or global interpretability. It also supports model-agnostic explainability tools (e.g. PFI, PDP, LIME, SHAP) and a powerful suite of model-agnostic diagnostics (e.g. weakness, uncertainty, robustness, fairness). Integration of PiML models and tests to existing MLOps platforms for quality assurance are enabled by flexible high-code APIs. Furthermore, PiML toolbox comes with a comprehensive user guide and hands-on examples, including the applications for model development and validation in banking. The project is available at https://github.com/SelfExplainML/PiML-Toolbox.

Keywords: Inherent interpretability, Post-hoc explainability, Model diagnostics, Quality assurance.

1 Introduction

Supervised machine learning has being increasingly used in domains where decision making can have significant consequences. However, the lack of interpretability of many machine learning models makes it difficult to understand and trust the model-based decisions. This leads to growing interest in interpretable machine learning and model diagnostics. There emerge algorithms and packages for model-agnostic explainability, including the inspection module (including permutation feature importance, partial dependence) in scikit-learn (Pedregosa et al., 2011) and various others, e.g. Kokhlikyan et al. (2020); Klaise et al. (2021); Baniecki et al. (2021); Li et al. (2022).

Post-hoc explainability tools are useful for black-box models, but they are known to have general pitfalls (Rudin, 2019; Molnar et al., 2022). Inherently interpretable models are suggested for machine learning model development (Yang et al., 2020, 2021; Sudjianto et al., 2020). The InterpretML package (Nori et al., 2019) by Microsoft Research is such a package of promoting the use of inherently interpretable models, in particular their explainable boosting machine (EBM) based on the GA2M model (Lou et al., 2013). One may also refer to Sudjianto and Zhang (2021) for discussion about how to design inherently interpretable machine learning models.

In the meantime, model diagnostic tools become increasingly important for model validation and outcome testing. New tools and platforms are developed for model weakness detection and error analysis, e.g., Chung et al. (2019), PyCaret package, TensorFlow model analysis, FINRA's model validation toolkit, and Microsoft's responsible AI toolbox. They can be used for arbitrary pre-trained models, in the same way as the post-hoc explainability tools. Such type of model diagnostics or validation is sometimes referred to as black-box testing, and there is an increasing demand of diagnostic tests for quality assurance of machine learning models.

It is our goal to design an integrated Python toolbox for interpretable machine learning, for both model development and model diagnostics. This is particularly needed for model risk management in banking, where it is a routine exercise to run model validation including evaluation of model conceptual soundness and outcome testing from various angles. An inherently interpretable machine learning model tends to be more conceptually sound, while it is subject to model diagnostics in terms of accuracy, weakness detection, fairness, uncertainty, robustness and resilience. The PiML toolbox we develop is such a unique Python tool that supports not only a growing list of interpretable models, but also an enhanced suite of multiple diagnostic tests. It has been adopted by multiple banks since its first launch on May 4, 2022.

2 Toolbox Design

PiML toolbox is designed to support machine learning workflows by both low-code interface and high-code APIs; see Figure 1 for the overall design.

- Low-code panels: interactive widgets or dashboards are developed for Jupyter notebook or Jupyter lab users. A minimum level of Python coding is required. The data pipeline consists of convenient exp.data_load(), exp.data_summary(), exp.eda(), exp.data_quality(), exp.feature_select(), exp.data_prepare(), each calling an interactive panel with choices of parameterization and actions.

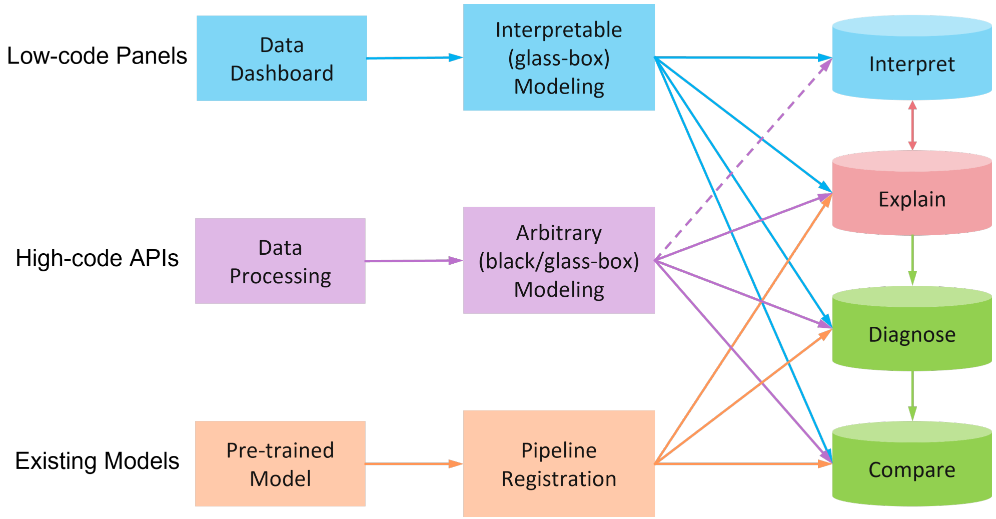

Figure 1: Design of PiML workflows with low-code interface and high-code APIs.

- High-code APIs: each low-code panel can be also executed through one or more Python functions with manually specified options and parameters. Such high-code APIs are flexible to be called both in Jupyter notebook cells and by Terminal command lines. High-code APIs usually provide more options than their default use in low-code panels. End-to-end pipeline automation can be enabled with appropriate high-code settings.

- Existing models: a pre-trained model can be loaded to PiML experimentation through pipeline registration. It is mandatory to include both training and testing datasets, in order for the model to take the full advantage of PiML explanation and diagnostic capabilities. It can be an arbitrary model in supervised learning settings, including regression and binary classification.

For PiML-trained models by either low-code interface or high-code APIs, there are four follow-up actions to be executed:

- model_interpret(): this unified API works only for inherently interpretable models (a.k.a. glass models) to be discussed in Section 3. It provides model-specific interpretation in both global and local ways. For example, a linear model is interpreted locally through model coefficients or marginal effects, while a decision tree is interpreted locally through the tree path.

- model_explain(): this unified API works for arbitrary models including black-box models and glass-box models. It provides post-hoc global explainability through permutation feature importance (PFI) and partial dependence plot (PDP) through sklearn.inspection module, accumulated local effects (Apley and Zhu, 2020), and post-hoc local explainability through LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017).

- model_diagnose(): this unified API works for arbitrary models and performs model diagnostics to be discussed in Section 4. It is designed to cover standardized general-purpose tests based on model data and predictions, i.e. model-agnostic tests. There is no need to access the model internals.

- model_compare(): this unified API is to compare two or three models at the same time, in terms of model performance and other diagnostic aspects. By inspecting the dashboard of graphical plots, one can easily rank models under comparison.

For registered models that are not trained from PiML, they are automatically treated as black-box models, even though such a model may be inherently interpretable (e.g. linear regression model). This is due to simplification of pipeline registration, where only the model prediction method is considered. For these models, model_interpret() is not valid, while the other three unified APIs are fully functional.

Regarding PiML high-code APIs, it is worthwhile to mention that these APIs are flexible enough for integration into existing MLOps platforms. After PiML installation to MLOps backend, the high-code APIs can be called not only to train interpretable models, but also to perform arbitrary model testing for quality assurance.

3 Interpretable Models

PiML supports a growing list of inherently interpretable models. For simplicity, we only list the models and the references. One may refer to the PiML user guide (PiML-Team, 2023) for details of each model use and hands-on examples. The following list of interpretable models are included PiML toolbox V0.5 (latest update: May 4, 2023).

1. GLM: Linear/logistic regression with ℓ_1 and/or ℓ_2 regularization (Hastie et al., 2015)

2. GAM: Generalized additive models using B-splines (Servén and Brummitt, 2018)

3. Tree: Decision tree for classification and regression (Pedregosa et al., 2011)

4. FIGS: Fast interpretable greedy-tree sums (Tan et al., 2022)

5. XGB1: Extreme gradient boosted trees of depth 1, using optimal binning (Chen et al., 2015; Navas-Palencia, 2020)

6. XGB2: Extreme gradient boosted trees of depth 2, with purified effects (Chen et al., 2015; Lengerich et al., 2020)

7. EBM: Explainable boosting machine (Lou et al., 2013; Nori et al., 2019)

8. GAMI-Net: Generalized additive model with structured interactions (Yang et al., 2021)

9. ReLU-DNN: Deep ReLU networks using Aletheia unwrapper and sparsification (Sudjianto et al., 2020)

4 Diagnostic Suite

PiML comes with a continuously enhanced suite of diagnostic tests for arbitrary supervised machine learning models under regression and binary classification settings. Below is a list of the supported general-purpose tests with brief descriptions. One may refer to the PiML user guide (PiML-Team, 2023) for details of each diagnostic test and hands-on examples.

1. Accuracy: popular metrics like MSE, MAE for regression tasks and ACC, AUC, Recall, Precision, F1-score for binary classification tasks.

2. WeakSpot: identification of weak regions with high magnitude of residuals by 1D and 2D slicing techniques.

3. Overfit/Underfit: identification of overfitting/underfitting regions according to train-test performance gap, also by 1D and 2D slicing techniques.

4. Reliability: quantification of prediction uncertainty by split conformal prediction techniques (Wang and et al., 2023).

5. Robustness: evaluation of performance degradation under different sizes of covariate noise perturbation (Cui et al., 2023).

6. Resilience: evaluation of performance degradation under different out-of-distribution scenarios.

7. Fairness: disparity test, segmented analysis and model de-bias through binning and thresholding techniques.

5 Future Plan

PiML toolbox is our new initiative of integrating state-of-the-art methods in interpretable machine learning and model diagnostics. It provides convenient user interfaces and flexible APIs for easy use of model interpretation, explanation, testing and comparison. Our future plan is to continuously improve the user experience, add new interpretable models, and expand the diagnostic suite. It is also our plan to enhance PiML experimentation with tracking and reporting.

Acknowledgements

We would like thank many people for their contributions and valuable comments to the development of PiML toolbox[1]. Our thanks also go to countless feedback from Model Risk Management communities of various financial institutions!

References

Apley, D. W. and Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086.

Baniecki, H., Kretowicz, W., Piatyszek, P., Wisniewski, J., and Biecek, P. (2021). Dalex: responsible machine learning with interactive explainability and fairness in python. *The Journal of Machine Learning Research*, 22(1):9759–9765.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4.

Chung, Y., Kraska, T., Polyzotis, N., Tae, K. H., and Whang, S. E. (2019). Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1550–1553. IEEE.

Cui, S., Sudjianto, A., Zhang, A., and Li, R. (2023). Enhancing robustness of gradient-boosted decision trees through one-hot encoding and regularization. *arXiv preprint arXiv:2304.13761*.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical learning with sparsity: the lasso and generalizations*. CRC press.

Klaise, J., Van Looveren, A., Vacanti, G., and Coca, A. (2021). Alibi explain: Algorithms for explaining machine learning models. *The Journal of Machine Learning Research*, 22(1):8194–8200.

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., et al. (2020). Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*.

[1] See the name list in https://github.com/SelfExplainML/PiML-Toolbox/blob/main/Acknowledgements.md

Lengerich, B., Tan, S., Chang, C.-H., Hooker, G., and Caruana, R. (2020). Purifying interaction effects with the functional ANOVA: An efficient algorithm for recovering identifiable additive models. In *International Conference on Artificial Intelligence and Statistics*, pages 2402–2412. PMLR.

Li, X., Xiong, H., Li, X., Wu, X., Chen, Z., and Dou, D. (2022). InterpretDL: explaining deep models in PaddlePaddle. *Journal of Machine Learning Research*, 23(197):1–6.

Lou, Y., Caruana, R., Gehrke, J., and Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 623–631. ACM.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Molnar, C., König, G., Herbinger, J., Freiesleben, T., Dandl, S., Scholbeck, C. A., Casalicchio, G., Grosse-Wentrup, M., and Bischl, B. (2022). General pitfalls of model-agnostic interpretation methods for machine learning models. In *xxAI-Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, pages 39–68. Springer.

Navas-Palencia, G. (2020). Optimal binning: mathematical programming formulation. *arXiv preprint arXiv:2001.08025*.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

PiML-Team (2023). *User Guide of PiML Toolbox. https://selfexplainml.github.io/PiML-Toolbox.*

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

Servén, D. and Brummitt, C. (2018). pygam: Generalized additive models in python. *Zenodo. doi*, 10.

Sudjianto, A., Knauth, W., Singh, R., Yang, Z., and Zhang, A. (2020). Unwrapping the black box of deep relu networks: interpretability, diagnostics, and simplification. *arXiv preprint:2011.04041*.

Sudjianto, A. and Zhang, A. (2021). Designing inherently interpretable machine learning models. *arXiv preprint arXiv:2111.01743*.

Tan, Y. S., Singh, C., Nasseri, K., Agarwal, A., and Yu, B. (2022). Fast interpretable greedy-tree sums (FIGS). *arXiv preprint arXiv:2201.11931*.

Wang, Y. and et al. (2023). Model weakness analysis via conformal prediction and uncertainty quantification. *arXiv preprint arXiv:TBD*.

Yang, Z., Zhang, A., and Sudjianto, A. (2020). Enhancing explainability of neural networks through architecture constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6):2610–2621.

Yang, Z., Zhang, A., and Sudjianto, A. (2021). GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. *Pattern Recognition*, 120:108192.