Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)
Demonstrations Track

PyXAI: An XAI Library for Tree-Based Models

Gilles Audemard[1] , **Jean-Marie Lagniez**[1] , **Pierre Marquis**[1,2] and **Nicolas Szczepanski**[1]
[1]Univ. Artois, CNRS, CRIL, F-62300 Lens, France
[2]Institut Universitaire de France
{audemard, lagniez, marquis, szczepanski}@cril.fr

Abstract

PyXAI (Python eXplainable AI) is a Python library designed for providing explanations and correcting tree-based Machine Learning (ML) models. It is suited to decision trees, random forests, and boosted trees, when used for regression or classification tasks. In contrast to many model-agnostic approaches to XAI, PyXAI exploits the model itself to generate explanations, ensuring them to be faithful. PyXAI includes several algorithms for the generation of explanations, which can be abductive or contrastive. PyXAI also includes algorithms for correcting tree-based models when their predictions conflict with pieces of user knowledge.

1 Introduction

The boom of Machine Learning (ML) through its numerous high-stake applications (medical diagnosis, image and voice recognition, autonomous driving, etc.) and the opacity of the most accurate ML models has led to the rapid development of eXplainable Artificial Intelligence (XAI) [Gunning, 2019] with the goal to make ML models more transparent and more trustable. More precisely, DARPA (Defense Advanced Research Projects Agency) put forward the following objective for XAI: *"to provide users with explanations that allow them to understand the forces and the overall weaknesses of the system in question, which allow them to understand how it will behave in the future, or even to correct the system's errors"*.

Several criteria can be used to characterize XAI systems, including (1) the family of ML models the system is suited to, (2) the nature of the explanations it delivers, and the properties such explanations satisfy, (3) the other functionalities the system provides, and (4) its target audience.

As to (1), unlike most existing approaches to explaining ML models that generate model-agnostic explanations, PyXAI is designed for *tree-based models*, i.e., decision trees (DTs), random forests (RFs), and boosted trees (BTs), when used for regression or classification tasks. Notably, BTs are among the state-of-the-art ML models when dealing with tabular data [Borisov *et al.*, 2021]. Though DTs, RFs and BTs are tree-based models, it turns out that DTs, RFs, and BTs do not behave equally from an XAI perspective. Indeed, when they are not too large, DTs are often viewed as interpretable by design [Molnar, 2019]. Furthermore, even when they are large enough, DTs are *computationally intelligible* [Audemard *et al.*, 2020] in the sense that many explanation / verification queries about DTs can be treated using polynomial-time algorithms [Audemard *et al.*, 2021]. Contrastingly, RFs and BTs are neither interpretable by design nor computationally intelligible [Audemard *et al.*, 2021].

PyXAI provides such XAI techniques for DTs, RFs, and BTs. Often, explanations that are irredundant, i.e., not containing characteristics of the input instance that are unnecessary for the explanation purpose, are looked for. One can even look for minimal explanations, i.e., explanations containing a minimal number of characteristics. Since computing explanations is NP-hard in the broad sense in general, scalability can be an issue for large datasets. Relaxing irredundancy or minimality conditions about explanations is a way to deal with this computational issue in practice.

As to (2), PyXAI generates explanations that are *post-hoc, local, and faithful*. The goal is to provide explanations for specific predictions, starting from a (tree-based) model that has been learned. Being faithful (aka sound or correct) [Nauta *et al.*, 2023; Vilone and Longo, 2021; Zhou *et al.*, 2021; Yang *et al.*, 2019] indicates that the explanations that are provided actually reflect the exact behaviour of the model. Faithfulness is paramount when dealing with high-risk or sensitive applications, which is the type of applications that are targeted by PyXAI. When faithfulness is not ensured, one can find "counterexamples" for the explanations that are generated, i.e., pairs of instances sharing an explanation but leading to distinct predictions [Ignatiev *et al.*, 2019b]. In particular, [Ignatiev, 2020] shows that the amount of "counterexamples" can be high when using some of the most popular approaches for computing model-agnostic explanations, namely LIME [Ribeiro *et al.*, 2016], Anchors [Ribeiro *et al.*, 2018], and SHAP [Lundberg and Lee, 2017]. In order to avoid the generation of unsound explanations, the approach followed by PyXAI is to map tree-based models to Boolean circuits (alias "transparent" or "white boxes"), exhibiting the same input-output behaviors [Narodytska *et al.*, 2018; Shih *et al.*, 2018a; 2019]. Thanks to such mappings, XAI queries about tree-based models can be delegated to the corresponding circuits.

As to (3), PyXAI provides *correction methods for tree-based models*. This more tricky facet of XAI is seldom offered by existing XAI systems. When some domain knowledge

is available and a prediction contradicts it, the model must be corrected. Rectification is a principled approach for such a correction operation, i.e., it is characterized by a set of rationality postulates [Coste-Marquis and Marquis, 2021].

As to (4), PyXAI is suited to *users that are not ML specialists*. The users of PyXAI may have pieces of knowledge about the domain that is targeted. Such pieces of knowledge can be used to make predictions for some instances (maybe only few of them when the user is more a layperson than an expert) and to correct the model when needed. The PyXAI library also deals with user preferences. Different kinds of preferences are handled [Audemard *et al.*, 2022a], and this is useful to provide explanations that fit the user expectations in the best way.

2 A General Overview

As illustrated in Figure 1, PyXAI is organized into three separate modules, dedicated to training, explaining, and rectifying (respectively). You produce, load, save or import ML models with the `learner` module, you retrieve explanations from these models using the `explainer` module, and you correct the models using the `rectifier` module.

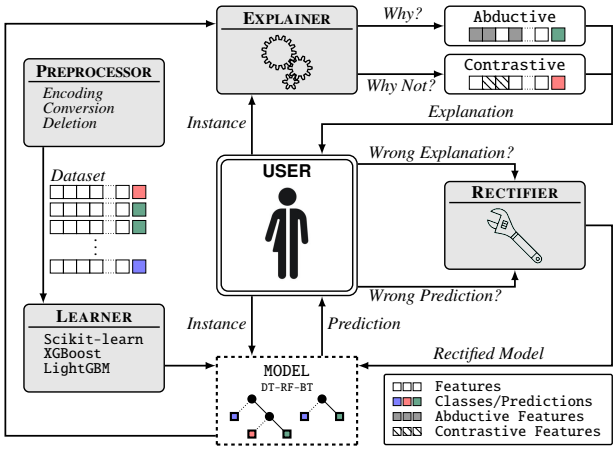

Figure 1: User interaction with PyXAI

Figure 2 shows that with few lines of code, PyXAI allows you to train a model, extract instances, and get explanations about the predictions made. A time limit can be given as a parameter to these methods.

In tree-based models, the conditions labelling the internal nodes of the trees are, in general, non-independent. For instance, one can find in a model a condition $age \geq 21$ and other conditions like $age \geq 18$ or of_age. Theories are representations of pieces of knowledge about the dataset that make precise how the conditions are connected (for instance, $(age \geq 21 \Rightarrow age \geq 18) \wedge (age \geq 18 \Leftrightarrow of\_age)$). PyXAI provides the option to take advantage of a theory. This is useful for deriving shorter abductive explanations [Gorji and Rubin, 2022], for avoiding to generate contrastive explanations that are impossible, and for simplifying both the explanations that are produced and the models obtained after a rectification step [Audemard *et al.*, 2023c].

```
from pyxai import Learning, Explainer

learner=Learning.Scikitlearn("iris.csv",
    learner_type=Learning.CLASSIFICATION)

model=learner.evaluate(method=Learning.HOLD_OUT,
                       output=Learning.DT)

instance, prediction = learner.get_instances(
    model, n=1, correct=True, predictions=[0])

explainer = Explainer.initialize(model, instance)
s_reason = explainer.sufficient_reason()
c_reason = explainer.contrastive_reason()

explainer.visualisation.notebook(instance, s_reason)
```

Figure 2: A simple example of Python code using PyXAI.

3 Computing Explanations

Using PyXAI, several types of (post-hoc, local) explanations (also called reasons) for a given instance $\mathbf{x}$ can be calculated. Among them, *abductive explanations* for $\mathbf{x}$ [Ignatiev *et al.*, 2019a] are intended to explain why the prediction made by the ML model about $\mathbf{x}$ has been obtained. Using PyXAI, it is possible to compute several types of abductive explanations, called direct, majoritary/tree-specific and sufficient reasons. Direct reasons are the easier to compute, but they are often very redundant (hence, they are quite large in general). Shorter abductive explanations correspond to other notions of reasons. Irredundant abductive explanations (i.e., explanations that are minimal w.r.t. set inclusion) are also called sufficient reasons [Darwiche and Hirth, 2020] or PI-explanations [Shih *et al.*, 2018b]. They can be computed in polynomial time for the DT model but not for the RF or the BT models (unless P = NP). For the DT model, PyXAI gives an algorithm for computing sufficient reasons [Audemard *et al.*, 2022b]. PyXAI also gives algorithms for generating majoritary reasons for the RF model [Audemard *et al.*, 2022c] and tree-specific reasons for the BT model [Audemard *et al.*, 2023b]. Majoritary reasons / tree-specific reasons are abductive explanations that can be computed in polynomial time. They can be redundant but in practice, their sizes are close to those of sufficient reasons. Regression models are also taken into account: PyXAI also includes an algorithm for deriving tree-specific reasons for the BT model [Audemard *et al.*, 2023a]. Note also that algorithms for computing minimal reasons (in terms of size) are also offered by the library.

Unlike abductive explanations, *contrastive explanations* [Miller, 2019] are intended to explain why $\mathbf{x}$ has not been classified by the ML model as the user expected it. Algorithms for the generation of contrastive explanations given a tree-based model are also provided in PyXAI [Audemard *et al.*, 2023c].

When computing explanations, PyXAI can also take advantage of *user preferences* [Audemard *et al.*, 2022a]. Dichotomous preference relations are handled, enabling to discard explanations containing characteristics that are not intelligible by a user from the explanations that are returned by PyXAI, or contrastive explanations containing characteristics that are not actionable (i.e., that cannot be changed). More gradual preference relations, modeled by utility or cost functions over

the features, can also be considered.

4 Correcting Tree-Based Models

Whenever the user disagrees with a prediction made by the ML model or with an explanation returned by PyXAI, she/he may provide PyXAI with a classification rule, supposed to be reliable enough and which can be used to rectify the model. This rule conflicts with the prediction / explanation that triggers the correction, in the sense that it has compatible premises, yet a distinct conclusion. It captures some domain knowledge that should be incorporated into the ML model in order to achieve better predictions, while preserving the explanation capacities of the model.

By construction, the rectification of an ML model by a classification rule is an ML model of the same kind, which makes the same predictions as those of the ML model at start, except for the instances for which other predictions are demanded by the rule, and for them, the resulting model provides the predictions that are required [Coste-Marquis and Marquis, 2023]. Thus, rectification is a conservative approach to the correction of a model (no retraining is made, and only the predictions that are questioned are modified). Notably, from a computation complexity point of view, rectifying a tree-based model can be achieved in time polynomial in the size of the input (the representation of the model and the classification rule used to correct it). This makes the approach practical enough.

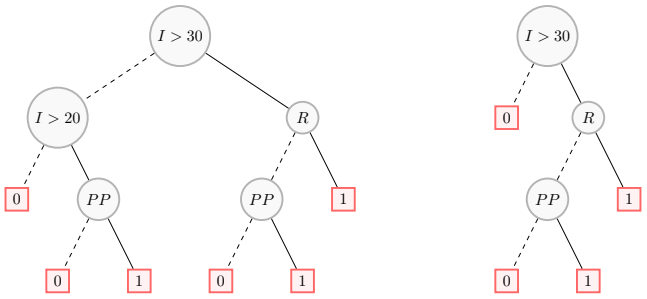

Figure 3: Rectifying a decision tree.

Figure 3 illustrates the rectification operation for a DT T used to address a problem of credit allocation to bank customers. Each customer is characterized by an annual income I (in $k\$$), the fact of having already reimbursed a previous loan (R) and, whether or not, she/he has a permanent position (PP). The Boolean conditions used in T are $I > 30$, $I > 20$, R, and PP. Consider the instance $\boldsymbol{x} = (I = 25, R = 1, PP = 1)$ corresponding to a customer applying for a loan. The DT T in Figure 3 (left) is such that $T(\boldsymbol{x}) = 1$: the loan should be granted. The user (a bank employee) disagrees with this prediction. For him/her, the following classification rule must be obeyed: whenever the annual income of the client is lower than 30, the demand should be rejected. Figure 3 (right) shows T after its rectification by this rule, once simplified.

5 Additional Facilities

Finally, PyXAI offers a few additional features that aim to make it a full-fledge integrated XAI framework. Those additional facilities are useful to ease the implementation of an ML protocol that includes the generation of explanations for the predictions made and the correction of the model.

Importing Models PyXAI provides an easy way to import models and is fully compatible with three ML libraries: Scikit-learn [Pedregosa *et al.*, 2011], `XGBoost` [Chen and Guestrin, 2016], and `LightGBM` [Ke *et al.*, 2017].

Visualizing Explanations By default, explanations can be shown in a notebook or saved in PNG format. PyXAI provides an optional Graphical User Interface (GUI) to display, save and load, instances and explanations for any dataset in a smart way. PyXAI supports multiple image formats for image datasets.

Figure 4 shows the graphical interface for an image dataset and the notebook display for a time series dataset.

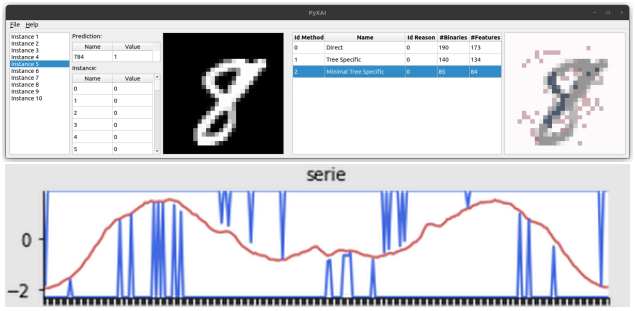

Figure 4: Visualizing explanations with PyXAI.

Saving/Loading Models and Instances The PyXAI library implements functions to save and load models and related hyper-parameters for training, and to save and load selected instances.

Installation and Documentation The PyXAI webpage[1] gives access to a number of resources, including a video presentation and more than 10 Jupyter notebooks. The source code of PyXAI is available on GitHub[2]. Installation is easy thanks to PyPi.

6 Related Work

Unlike model-agnostic explanations for which many popular libraries and toolkits exist (see [Le *et al.*, 2023] for a recent survey), as far as we know, there is no XAI library dedicated to tree-based models, but PyXAI.

To be more precise, two algorithms are available for the generation of abductive explanations suited to tree-based models. The first one is called `XPlainer`; it shows how to leverage an SMT solver to compute sufficient reasons [Ignatiev *et al.*, 2019c] given a BT. The second one is called `XReason`; it relies on another encoding scheme, based on MaxSAT [Ignatiev *et al.*, 2022]. Up to now, none of these algorithms has been integrated with training algorithms into a fully-developed XAI library for tree-based models. Finally, to the best of our knowledge, apart from PyXAI, there is no library that would provide algorithms for rectifying tree-based models.

[1] https://www.cril.univ-artois.fr/pyxai/
[2] https://github.com/crillab/pyxai

Acknowledgements

Many thanks to the anonymous reviewers for their comments and insights. This work has benefited from the support of the AI Chair EXPEKCTATION (ANR-19-CHIA-0005-01) and of the France 2030 MAIA Project (ANR-22-EXES-0009) of the French National Research Agency (ANR). This work also benefited from fundings from the French state under the Investments for the Future program within the framework of the SystemX Technological Research Institute. It was also partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

[Audemard *et al.*, 2020] G. Audemard, F. Koriche, and P. Marquis. On tractable XAI queries based on compiled representations. In *Proc. of KR'20*, pages 838–849, 2020.

[Audemard *et al.*, 2021] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. On the computational intelligibility of boolean classifiers. In *Proc. of KR'21*, pages 74–86, 2021.

[Audemard *et al.*, 2022a] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. On preferred abductive explanations for decision trees and random forests. In *Proc. of IJCAI'22*, pages 643–650, 2022.

[Audemard *et al.*, 2022b] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. On the explanatory power of boolean decision trees. *Data Knowl. Eng.*, 142:102088, 2022.

[Audemard *et al.*, 2022c] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. Trading complexity for sparsity in random forest explanations. In *Proc. of AAAI'22*, pages 5461–5469. AAAI Press, 2022.

[Audemard *et al.*, 2023a] G. Audemard, S. Bellart, J.-M. Lagniez, and P. Marquis. Computing abductive explanations for boosted regression trees. In *Proc. of IJCAI'23*, pages 3432–3441, 2023.

[Audemard *et al.*, 2023b] G. Audemard, J.-M. Lagniez, P. Marquis, and N. Szczepanski. Computing abductive explanations for boosted trees. In *Proc. of AISTATS'23*, volume 206 of *Proceedings of Machine Learning Research*, pages 4699–4711. PMLR, 2023.

[Audemard *et al.*, 2023c] G. Audemard, J.-M. Lagniez, P. Marquis, and N. Szczepanski. On contrastive explanations for tree-based classifiers. In *Proc. of ECAI'23*, pages 117–124, 2023.

[Borisov *et al.*, 2021] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci. Deep neural networks and tabular data: A survey. *CoRR*, abs/2110.01889, 2021.

[Chen and Guestrin, 2016] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proc. of KDD'16*, page 785–794, 2016.

[Coste-Marquis and Marquis, 2021] S. Coste-Marquis and P. Marquis. On belief change for multi-label classifier encodings. In *Proc. of IJCAI'21*, pages 1829–1836, 2021.

[Coste-Marquis and Marquis, 2023] S. Coste-Marquis and P. Marquis. Rectifying binary classifiers. In *Proc. of ECAI'23*, pages 485–492, 2023.

[Darwiche and Hirth, 2020] A. Darwiche and A. Hirth. On the reasons behind decisions. In *Proc. of ECAI'20*, pages 712–720, 2020.

[Gorji and Rubin, 2022] N. Gorji and S. Rubin. Sufficient reasons for classifier decisions in the presence of domain constraints. In *Proc. of AAAI'22*, pages 5660–5667, 2022.

[Gunning, 2019] D. Gunning. DARPA's explainable artificial intelligence (XAI) program. In *Proc. of IUI'19*, 2019.

[Ignatiev *et al.*, 2019a] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *Proc. of AAAI'19*, pages 1511–1519, 2019.

[Ignatiev *et al.*, 2019b] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019.

[Ignatiev *et al.*, 2019c] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019.

[Ignatiev *et al.*, 2022] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *Proc. of AAAI'22*, pages 3776–3785, 2022.

[Ignatiev, 2020] A. Ignatiev. Towards trustable explainable ai. In *Proc. of IJCAI'20*, pages 5154–5158, 2020.

[Ke *et al.*, 2017] G. Ke, Q. Meng, Th. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proc. of NeurIPS'17*, pages 3146–3154, 2017.

[Le *et al.*, 2023] P.Q. Le, M. Nauta, V. B. Nguyen, S. Pathak, J. Schlötterer, and Ch. Seifert. Benchmarking explainable ai - a survey on available toolkits and open challenges. In *Proc. of IJCAI'23*, pages 6665–6673, 8 2023. Survey Track.

[Lundberg and Lee, 2017] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[Miller, 2019] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[Molnar, 2019] Ch. Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. Leanpub, 2019.

[Narodytska *et al.*, 2018] N. Narodytska, S. Prasad Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh. Verifying properties of binarized deep neural networks. In *Proc. of AAAI'18*, pages 6615–6624, 2018.

[Nauta *et al.*, 2023] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van

Keulen, and Ch. Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Comput. Surv.*, 55(13s), 2023.

[Pedregosa *et al.*, 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Ribeiro *et al.*, 2016] M.T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[Ribeiro *et al.*, 2018] M.T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proc. of AAAI'18*, pages 1527–1535, Apr. 2018.

[Shih *et al.*, 2018a] A. Shih, A. Choi, and A. Darwiche. Formal verification of Bayesian network classifiers. In *Proc. of PGM'18*, pages 427–438, 2018.

[Shih *et al.*, 2018b] A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. In *Proc. of IJCAI'18*, pages 5103–5111, 2018.

[Shih *et al.*, 2019] A. Shih, A. Choi, and A. Darwiche. Compiling Bayesian networks into decision graphs. In *Proc. of AAAI'19*, pages 7966–7974, 2019.

[Vilone and Longo, 2021] G. Vilone and L. Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Inf. Fusion*, 76:89–106, 2021.

[Yang *et al.*, 2019] F. Yang, M. Du, and X. Hu. Evaluating explanation without ground truth in interpretable machine learning. *CoRR*, abs/1907.06831, 2019.

[Zhou *et al.*, 2021] J. Zhou, A.H. Gandomi, F. Chen, and A. Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10:593, 2021.