IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 5, May 2023

DOI: 10.17148/IJARCCE.2023.125225

InterpretML: A Unified Framework for Machine Learning Interpretability

Kiran Bandu Donge[1], Lovelesh N.Yadav[2], Neehal B.Jiwane[3]

Student, Computer Science & Engineering, Shri Sai College of Engineering &Technology, Bhdrawati, India[1]

Head Of Department, Computer Science & Engineering, Shri Sai College Of Engineering & Technology, Bhdrawati, India[2]

Asst.prof, Computer Science & Engineering, Shri Sai College Of Engineering & Technology, Bhdrawati, India[3]

Abstract: InterpretML is an open-source Python package which exposes machine learning interpretability algorithms to practitioners and researchers. InterpretML exposes two types of interpretability – glassbox, which are machine learning models designed for interpretability (ex: linear models, rule lists, generalized additive models), and blackbox explainability techniques for explaining existing systems (ex: Partial Dependence, LIME). The package enables practitioners to easily compare interpretability algorithms by exposing multiple methods under a unified API, and by having a built-in, extensible visualization platform. InterpretML also includes the first implementation of the Explainable Boosting Machine, a powerful, interpretable, glassbox model that can be as accurate as many blackbox models. The MIT licensed source code can be downloaded from github.com/microsoft/interpret.

Keywords: Interpretability, Explainable Boosting Machine, Glassbox, Blackbox

I. INTRODUCTION

As machine learning has matured into wide-spread adoption, building models that users can understand is becoming increasingly important. This can easily be observed in high-risk applications such as healthcare (Ahmad et al., 2018; Caruana et al., 2015), finance (Hajek, 2019; Chen et al., 2018) and judicial environments (Tan et al., 2018; Soundarajan and Clausen). Interpretability is also important in general applied machine learning problems such as model debugging, regulatory compliance, and human computer interaction.

We address these needs with InterpretML by exposing many state of the art interpretability algorithms under a unified API. This API covers two major interpretability forms: ”glassbox” models, which are inherently intelligible and explainable to the user, and ”blackbox” interpretability, methods that generate explanations for any machine learning pipeline, no matter how opaque it is. This is further supported with interactive visualizations and a built-in dashboard designed for interpretability algorithm comparison. InterpretML is MIT licensed, and emphasizes extensibility and compatibility with popular open-source projects such as scikit-learn (Pedregosa et al., 2011) and Jupyter Notebook environments (Kluyver et al., 2016).

II. PACKAGE DESIGN

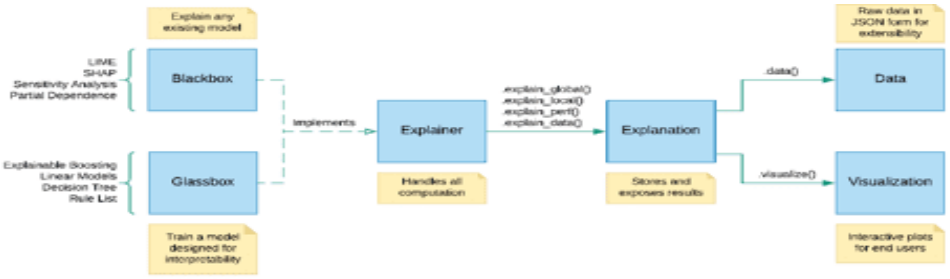
InterpretML follows four key design principles that influence its architecture and API. Ease of comparison. Make it as easy as possible to compare multiple algorithms. ML interpretability is in its infancy, and many algorithmic approaches have emerged from research, each of which has pros and cons. Comparison is critical to find the algorithm that best suits the users’ needs. InterpretML enables this by enforcing a scikit-learn style uniform API, and providing a visualization platform centered around algorithmic comparison. Stay true to the source. Use reference algorithms and visualizations as much as possible. Our goal is to expose interpretability algorithms to the world, in their most accurate form. Play nice with others. Leverage the open-source ecosystem, and don’t reinvent the wheel. InterpretML is highly compatible with popular projects like Jupyter Notebook and scikit-learn, and builds off of many libraries like plotly, lime, shap, and SALib. Take what you want. Use and extend any component of InterpretML without pulling in the whole framework. For example, it’s possible to produce a computationally intensive explanation on a server, without InterpretML’s visualization and its related dependencies. The code architecture and unified API is best expressed in Figure 1, providing an overview and relevant example code.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 5, May 2023

DOI: 10.17148/IJARCCE.2023.125225



Glassbox

```
from interpret import show
from interpret.glassbox import LogisticRegression


clf = LogisticRegression()
clf.fit(X, y)

global_exp = clf.explain_global()
local_exp = clf.explain_local(X, y)

show([global_exp, local_exp])
```

Blackbox

```
from interpret import show
from interpret.blackbox import PartialDependence
from sklearn.neural_network import MLPClassifier

blackbox = MLPClassifier()
blackbox.fit(X, y)

pdp = PartialDependence(blackbox.predict_proba, X)
global_exp = pdp.explain_global()

show(global_exp)
```

III. EXPLAINABLE BOOSTING MACHINE

As part of the framework, InterpretML also includes a new interpretability algorithm – the Explainable Boosting Machine (EBM). EBM is a glassbox model, designed to have accuracy comparable to state-of-the-art machine learning methods like Random Forest and Boosted Trees, while being highly intelligibile and explainable. EBM is a generalized additive model

(GAM) of the form:
$g(E[y]) = \beta 0 + X\, fj\, (xj\ )$

where g is the link function that adapts the GAM to different settings such as regression or classification. EBM has a few major improvements over traditional GAMs (Hastie and Tibshirani, 1987). First, EBM learns each feature function fj using modern machine learning techniques such as bagging and gradient boosting. The boosting procedure is carefully restricted to train on one feature at a time in round-robin fashion using a very low learning rate so that feature order does not matter. It round-robin cycles through features to mitigate the effects of co-linearity and to learn the best feature function fj for each feature to showhow each feature contributes to the model's prediction for the problem. Second, EBM canautomatically detect and include pairwise interaction terms of the form:

$g(E[y]) = \beta 0 + X\, fj\, (xj\ ) + X\, fij\, (xi\ , xj\ )$

which further increases accuracy while maintaining intelligibility. EBM is a fast implementation of the GA2M algorithm (Lou et al., 2013), written in C++ and Python. The implementation is parallelizable, and takes advantage of joblib to provide multi-core and multi-machine parallelization. The algorithmic details for the training procedure, selection of pairwise interaction terms, and case studies can be found in (Lou et al., 2012, 2013; Caruana et al., 2015).

EBMs are highly intelligible, because the contribution of each feature to a final prediction feature contributes to predictions in a modular way that makes it easy to reason about the contribution These term contributions are simply added up, and passed through the link function g to compute the final prediction. Because of the modularity (additivity), term contributions can be sorted and visualized to show which features had the most impact on any individual prediction.

Classification Performance (AUROC)					
Model	heart-disease (303, 13)	breast-cancer (569, 30)	telecom-churn (7043, 19)	adult-income (32561, 14)	credit-fraud (284807, 30)
EBM	**0.916**	**0.995**	**0.851**	**0.928**	0.975
LightGBM	0.864	0.992	0.835	**0.928**	0.685
Logistic Regression	0.895	**0.995**	0.804	0.907	0.979
Random Forest	0.89	0.992	0.824	0.903	0.95
XGBoost	0.87	**0.995**	0.85	0.922	**0.981**

Figure 3: Classification performance for models across datasets (rows, columns).

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 8.102 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 12, Issue 5, May 2023
DOI: 10.17148/IJARCCE.2023.125225

ACKNOWLEDGMENT

We would like to acknowledge everyone on our acknowledgements.md file for their support on this project. We also depend on many amazing software packages and research: scikit-learn 1. All models were trained with their default parameters. EBM's current default parameters are chosen for computational speed, to enable ease of experimentation. For the best accuracy and interpretability, we recommend using reference parameters: 100 inner bags, 100 outer bags, 5000 epochs, and a learning rate of 0.01.

REFERENCES

1.Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable machine learning in healthcare. In Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, pages 559–560. ACM, 2018.

2.Chaofan Chen, Kangcheng Lin, Cynthia Rudin, Yaron Shaposhnik, Sijia Wang, and Tong Wang. An interpretable model with globally consistent explanations for credit risk. arXiv preprint arXiv:1811.12615,

3.Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. Ann.Statist., 29(5):1189–1232, 10 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451

4.Trevor Hastie and Robert Tibshirani. Generalized additive models: some applications. Journal of the American Statistical Association, 82(398):371–386, 1987

5.Jon Herman and Will Usher. SALib: An open-source python library for sensitivity analysis. The Journal of Open Source Software, 2(9), jan 2017. doi: 10.21105/joss.00097. URL
https://doi.org/10.21105/joss.00097.

6.Sucheta Soundarajan and Daniel L Clausen. Equal protection under the algorithm: A legal-inspired framework for identifying discrimination in machine learning

7.Xuezhou Zhang, Sarah Tan, Paul Koch, Yin Lou, Urszula Chajewska, and Rich Caruana. Interpretability is harder in the multiclass setting: Axiomatic interpretability for multiclass additive models. CoRR, abs/1810.09092, 2018. URL http://arxiv.org/abs/
1810.09092

