Journal of Machine Learning Research 22 (2021) 1-7 Submitted 1/21; Revised 6/21; Published 6/21

Alibi Explain: Algorithms for Explaining Machine Learning Models

Janis Klaise jk@seldon.io
Arnaud Van Looveren avl@seldon.io
Giovanni Vacanti gv@seldon.io
Seldon Technologies Limited
Alexandru Coca[*] ac2123@cam.ac.uk
University of Cambridge

Editor: Alexandre Gramfort

Abstract

We introduce `Alibi Explain`, an open-source Python library for explaining predictions of machine learning models (`https://github.com/SeldonIO/alibi`). The library features state-of-the-art explainability algorithms for classification and regression models. The algorithms cover both the model-agnostic (black-box) and model-specific (white-box) setting, cater for multiple data types (tabular, text, images) and explanation scope (local and global explanations). The library exposes a unified API enabling users to work with explanations in a consistent way. `Alibi` adheres to best development practices featuring extensive testing of code correctness and algorithm convergence in a continuous integration environment. The library comes with extensive documentation of both usage and theoretical background of methods, and a suite of worked end-to-end use cases. `Alibi` aims to be a production-ready toolkit with integrations into machine learning deployment platforms such as `Seldon Core` and `KFServing`, and distributed explanation capabilities using `Ray`.

Keywords: Explainability, Open Source, Python

1. Introduction

Explainable AI, also known as model explainability, refers to techniques for elucidating the reasons behind predictions made by complex, opaque machine learning models in a format that is understandable to human observers (Molnar, 2019). The ability to explain predictions helps to build trust in the model's decision making process and is therefore an integral part of a robust machine learning system (Bhatt et al., 2020; Klaise et al., 2020).

The desired insights provided by explanations differ strongly dependent on the consumer of the explanations, ranging from data scientists debugging models to regulators auditing them. As a result, multiple methods are needed to cater to the needs of the target audience (ICO, 2019; Bhatt et al., 2020). Moreover, standalone explanation methods can generate non-informative or even misleading explanations (Heo et al., 2019). This means that a holistic approach to model explanations is required.

We present `Alibi` which aims to bridge the gap between the fast growing area of explainability research and industry. The goal of `Alibi` is to host reference implementations

[*]. Work done at Seldon.

of a broad range of production-ready model explanation algorithms. `Alibi` contains local, global, black- and white-box post-hoc explanation methods to cover a wide variety of use cases. Whilst there are a few contemporaneous explainability libraries (see Table 1), `Alibi` is uniquely focused on providing production level explanation methods with deployment platform integrations and a distributed backend.

2. Project Focus

Scope of applications. Model explainability often requires a holistic approach as there is no one-size-fits-all solution. This is reflected in the breadth of algorithms currently supported (Section 2.1) and the guidance of their applicability (Table 2).
Build robustness. Extensive testing of code correctness and algorithm convergence is done using `pytest` under various Python versions. Tests are executed on every pull request via a continuous integration setup using `Github Actions`.
Documentation and examples. The library features comprehensive documentation and extensive in-depth examples of use cases[1]. The documentation includes usage and theoretical background of each method. Furthermore, the scope and applicability of all methods is clearly documented to help practitioners quickly identify relevant algorithms (Table 2).
Industry relevance. `Alibi` is integrated into deployment platforms `Seldon Core` (Cox et al., 2018) and `KFServing` (KFServing, 2019) to enable deploying explanations into production. `Alibi` also features a distributed backend using `Ray` (Moritz et al., 2018) to enable large-scale parallel computation of batch explanations.

We also provide a more detailed feature comparison with other actively developed explanation libraries, see Table 1.

Library	Local post-hoc	Global post-hoc	Feature attributions	Anchors	Counter-factuals	Multiple data types	Deployment options
Alibi Explain	✓	✓	✓	✓	✓	✓	✓
AIX360	✓	✓	✓		✓	✓	✓
Interpret	✓	✓	✓				
Captum	✓		✓				
iNNvestigate	✓		✓				

Table 1: Comparison with related explanation libraries AIX360 (Arya et al., 2020), Interpret (Nori et al., 2019), Captum (Kokhlikyan et al., 2020), iNNvestigate (Alber et al., 2019). Libraries are selected and compared on the basis of providing post-hoc, black-box or white-box, local or global explanation techniques implemented in Python which have had some development activity in the past 12 months.

2.1 Algorithms

The current version of the library includes the following explanation algorithms (c.f. Table 2 for detailed capabilities): 1. *Accumulated Local Effects (ALE)*, Apley and Zhu (2016): calculate global feature effects on the model predictions. 2. *Anchor explanations*, Ribeiro

1. `https://docs.seldon.io/projects/alibi/en/latest/`

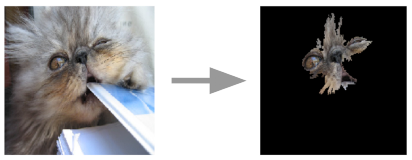
Method	Models	Explanations	Tasks	Data types	Train set req.
ALE	BB	global	C, R	tab*	✓
Anchors	BB	local	C	tab, text, img	For tabular
CEM	BB*, WB	local	C	tab*, img	Optional
Counterfactuals	BB*, WB	local	C	tab*, img	No
Prototype counterfactuals	BB*, WB	local	C	tab, img	Optional
Integrated Gradients	WB	local	C, R	tab, text, img	Optional
Kernel SHAP	BB	local, global	C, R	tab	✓
Tree SHAP	WB	local, global	C, R	tab	Optional

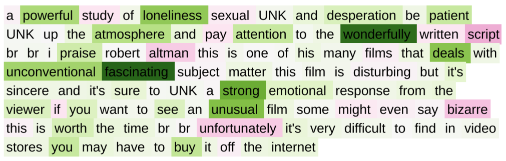
Table 2: Comparison of explanation methods in `Alibi`. **Models.** Type of model expected. BB: black-box (can call a model), BB*: differentiable black-box, WB: white-box (access to model internals). **Explanations.** *local:* explain single predictions, *global:* explain the model overall **Tasks.** Type of tasks supported. C: classification, R: regression. **Data types.** Type of data supported. tab: tabular, tab*: tabular without categorical variable support, img: images. **Train set req.** Whether a training set is required.

et al. (2018): find a minimal subset of features to guarantee (with high probability) the same prediction regardless of other features. 3. *Contrastive Explanation Method (CEM)*, Dhurandhar et al. (2018): find features which should be minimally and sufficiently present as well as features which should be necessarily absent to justify a prediction for a specific instance. 4. *Counterfactual explanations*, Wachter et al. (2018): find synthetic instances close to the original but resulting in a different prediction. 5. *Counterfactual explanations with prototypes*, Van Looveren and Klaise (2019): improves the counterfactual explanation method to result in more interpretable, in-distribution instances. 6. *Integrated Gradients*, Sundararajan et al. (2017): calculate feature attributions to the prediction by accumulating gradients along a path from a baseline instance to the instance of interest. 7. *Kernel Shapley Additive Values*, Lundberg and Lee (2017): calculate feature attributions to the prediction via a game theoretic approach by considering groups of features to be "uninformative". 8. *Tree Shapley Additive Values*, Lundberg et al. (2020): algorithmic improvement of Shapley additive values to tree ensemble models. Figure 1 shows outputs for a selection of supported explanation algorithms.

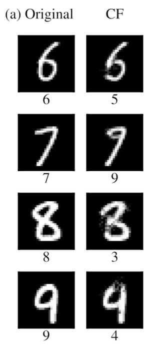
3. Library Design

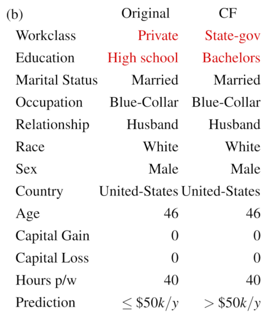
The user facing API of `Alibi` is designed to be consistent across algorithms and easy to use (Code Snippet 1). An explanation algorithm is initialized by passing either a prediction function (a Python `Callable` taking and returning `numpy` arrays) in the black-box case or a pre-trained model (e.g. `xgboost` for `TreeSHAP` or `TensorFlow`) in the white-box case. As detailed in Table 2, for methods where training data is required, the `fit` method must be called. Finally, an `explain` method is called to calculate an explanation on an instance or a set of instances. This returns an `Explanation` object containing dictionaries `meta` and `data` with the explanation metadata (e.g. hyperparameter settings, names) and the expla-





a powerful study of loneliness sexual UNK and desperation be patient UNK up the atmosphere and pay attention to the wonderfully written script br br i praise robert altman this is one of his many films that deals with unconventional fascinating subject matter this film is disturbing but it's sincere and it's sure to UNK a strong emotional response from the viewer if you want to see an unusual film some might even say bizarre this is worth the time br br unfortunately it's very difficult to find in video stores you may have to buy it off the internet





(b)

	Original	CF
Workclass	Private	State-gov
Education	High school	Bachelors
Marital Status	Married	Married
Occupation	Blue-Collar	Blue-Collar
Relationship	Husband	Husband
Race	White	White
Sex	Male	Male
Country	United-States	United-States
Age	46	46
Capital Gain	0	0
Capital Loss	0	0
Hours p/w	40	40
Prediction	$\leq \$50k/y$	$> \$50k/y$

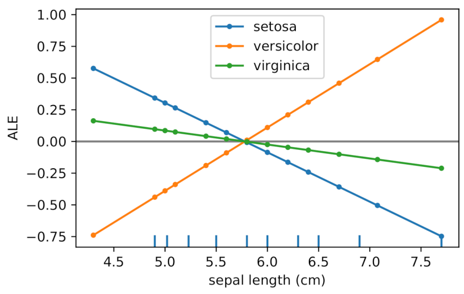


Figure 1: A selection of supported explanation algorithms. *Top left:* Anchor explanation on image classification explaining the prediction "Persian cat". *Top right:* Integrated Gradients attributions on a sentiment prediction task explaining the prediction "positive". *Bottom left:* Counterfactual explanations of (a) MNIST digit classification and (b) Income classification. *Bottom right:* ALE feature effects for a logistic regression model on the Iris dataset.

nation data respectively. The structure of the `Explanation` object enables easy serialization in production systems for further processing (e.g. logging, visualization). The metadata captures settings used to obtain each explanation and acts as an audit trail.

```
>>> from alibi.explainers import AnchorTabular
>>> explainer = AnchorTabular(predict_fn, feature_names)
>>> explainer.fit(X_train)
>>> explanation = explainer.explain(x)
>>> explanation.meta
{'name': 'AnchorTabular', 'type': ['blackbox'], 'explanations': ['local'],
 'params': {'seed': None, 'disc_perc': (25, 50, 75), 'threshold': 0.95}}
>>> explanation.data
{'anchor': ['petal width (cm) > 1.80', 'sepal width (cm) <= 2.80'],
 'precision': 0.98, 'coverage': 0.32}
```

Code Snippet 1: Demo of the `Alibi` API with the `AnchorTabular` explanation algorithm.

4. Outlook

The first phase of the development of `Alibi` has focused on creating a curated set of reference explanation algorithms with comprehensive guidance on typical use cases. While the work on white-box gradient based methods focused on supporting `TensorFlow` models (Abadi et al., 2016), achieving feature parity with `PyTorch` models (Paszke et al., 2019) in the near future is a key goal. Further, we plan to extend the use of the `Ray` project (Moritz et al., 2018) to enable parallelization for all explanation algorithms. The choice of `Ray` also enables the scaling of explanations beyond a single multi-core computation node (Coca, 2020).

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, Nov. 2016. USENIX Association. ISBN 978-1-931971-33-1. URL `https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi`.

M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K. T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, and P.-J. Kindermans. innvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019. URL `http://jmlr.org/papers/v20/18-540.html`.

D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *arXiv preprint arXiv:1612.08468*, 2016.

V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilovic, S. Mourad, P. Pedemonte, R. Raghavendra, J. T. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang. Ai explainability 360: An extensible toolkit for understanding data and machine learning models. *Journal of Machine Learning Research*, 21(130):1–6, 2020. URL `http://jmlr.org/papers/v21/19-1035.html`.

U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. F. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 648–657, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3375624. URL `https://doi.org/10.1145/3351095.3375624`.

A. Coca. Distributed black-box model explanation with Ray, 2020. URL `https://ray2020.sched.com/event/aWFq/distributed-black-box-model-explanation-with-ray-alexandru-coca-seldon`.

C. Cox, G. Sunner, A. Saucedo, R. Dawson, A. Gonzalez, and R. Skolasinski. Seldon Core: A framework to deploy, manage and scale your production machine learning to thousands of models., 2018. URL `https://github.com/SeldonIO/seldon-core`.

A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 592–603. Curran Associates, Inc.,

2018. URL http://papers.nips.cc/paper/7340-explanations-based-on-the-missing-towards-contrastive-explanations-with-pertinent-negatives.pdf.

J. Heo, S. Joo, and T. Moon. Fooling neural network interpretations via adversarial model manipulation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 2925–2936. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/7fea637fd6d02b8f0adf6f7dc36aed93-Paper.pdf.

ICO. Project explain interim report, 2019. URL https://ico.org.uk/about-the-ico/research-and-reports/project-explain-interim-report/.

KFServing. KFServing: Serverless inferencing on kubernetes, 2019. URL https://github.com/kubeflow/kfserving.

J. Klaise, A. Van Looveren, C. Cox, G. Vacanti, and A. Coca. Monitoring and explainability of models in production. *arXiv preprint arXiv:2007.06299*, 2020.

N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. 2017. URL https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.

S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, Jan 2020. ISSN 2522-5839. doi: 10.1038/s42256-019-0138-9. URL https://doi.org/10.1038/s42256-019-0138-9.

C. Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, et al. Ray: A distributed framework for emerging AI applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

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

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances* *in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982`.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

A. Van Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019. URL `https://arxiv.org/abs/1907.02584`.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard journal of law & technology*, 31:841–887, 04 2018. URL `https://arxiv.org/abs/1711.00399`.