Original software publication

FAT Forensics: A Python toolbox for algorithmic fairness, accountability and transparency

Kacper Sokol [a,b,*], Raul Santos-Rodriguez [a], Peter Flach [a]

[a] *Intelligent Systems Laboratory, University of Bristol, United Kingdom*
[b] *ARC Centre of Excellence for Automated Decision-Making and Society, RMIT University, Australia*

ARTICLE INFO

ABSTRACT

Today, artificial intelligence systems driven by machine learning algorithms can be in a position to take important, and sometimes legally binding, decisions about our everyday lives. In many cases, however, these systems and their actions are neither regulated nor certified. To help counter the potential harm that such algorithms can cause we developed an open source toolbox that can analyse selected fairness, accountability and transparency aspects of the machine learning process: data (and their features), models and predictions, allowing to automatically and objectively report them to relevant stakeholders. In this paper we describe the design, scope, usage and impact of this Python package, which is published under the 3-Clause BSD open source licence.

Code metadata

Current code version	0.1.1
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-97
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/8437308/tree/v1
Legal Code Licence	3-Clause BSD Licence (New BSD)
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	https://fat-forensics.org/getting_started/install_deps_os.html#installation-instructions
If available Link to developer documentation/manual	https://fat-forensics.org/
Support email for questions	https://fat-forensics.org/#communication

1. Algorithmic fairness, accountability and transparency with FAT Forensics

Open source software is the backbone of reproducible research, especially so in Artificial Intelligence (AI) and Machine Learning (ML) where changing the seed of a random number generator may cause a state-of-the-art solution to become a subpar predictive system. Despite numerous efforts to ensure that publications are accompanied by code, both the AI and ML fields struggle with a reproducibility crisis [1]. One way to address this problem is to promote publishing high-quality software used for scientific experiments under an open source licence

or enforce it as part of the publishing process [2]. In spite of their importance, implementations are nonetheless commonly treated just as a research by-product and often abandoned after publishing the findings based upon them. We call this phenomenon *paperware*, i.e., code whose main purpose is to see a paper towards publication rather than implement any particular concept with thorough software engineering practice. Such attitude results in standalone packages that often prove difficult to use due to the lack of documentation, testing, usage examples and (post-publication) maintenance, therefore impacting their reach, usability and, more broadly, reproducibility of scientific

Table 1
FAT functionality implemented in the latest release – version 0.1.1 – of `FAT Forensics`.

	Fairness	Accountability	Transparency
Data & Features	• Systemic Bias • Sub-population Representation	• Sampling Bias • Data Density Checker	• Data Description • Summary Statistics
Models	• Group-based Fairness	• Group-based Performance Metrics • Systematic Performance Bias	• Global Surrogates (bLIMEy) • Partial Dependence • Submodular Pick
Predictions	• Counterfactual Fairness	• Prediction Confidence	• Model-agnostic Counterfactuals • Local Surrogates (bLIMEy) • LIME (bLIMEy implementation) • Individual Conditional Expectation

findings. This state of affairs is especially problematic for AI and ML research with its fast-paced environment, lack of standards and far-ranging social implications.

Widespread reliability issues with ML systems have inspired a range of frameworks to assess and document them as well as report their quality, robustness and other (technical) properties through standardised mechanisms. For example, researchers have suggested approaches to characterise data sets [3,4]; automated decision-making systems [5]; predictive models offered as a service accessible via an Application Programming Interface (API) [6]; ranking algorithms [7]; AI & ML explainability approaches [8]; and privacy aspects of applications that collect, process and share user data [9] to ensure their high quality, transparency, reliability and accountability. Such efforts are laudable, however they may require the authors to understand the investigated system in detail, suffer from limited scope or be subject to time- and labour-intensive creation process, all of which can hinder their uptake or slow down the ML research and development cycle. Moreover, self-reporting – and lack of external audits – means that some of their aspects may be subjective, hence misrepresent the true behaviour of the underlying system, whether done intentionally or not. Certification, on the other hand, creates a need for external bodies, which seems difficult to achieve for all ML systems that somehow affect humans.

To help address such shortcomings in the fields of AI & ML Fairness, Accountability and Transparency (FAT), we designed and developed an open source Python package called `FAT Forensics` [10] – Table 1 lists the algorithms distributed in its latest release (version 0.1.1). It is intended as an interoperable framework to *implement*, *test* and *deploy* novel algorithms proposed by the FAT community as well as facilitate their evaluation and comparison against state-of-the-art methods, therefore democratising access to these techniques. The toolbox is capable of analysing all facets of the data-driven predictive process – data (raw and their features), models and predictions – in view of their FAT aspects. The common interface layer of the software (described in Section 2) makes it flexible enough to support workflows typical of academics and practitioners alike, and enables two *modes of operation – research* and *deployment –* that span diverse use cases such as prototyping, exploratory analytics, (numerical or visual) reporting and dashboarding as well as inspection, monitoring and evaluation of FAT properties. Additionally, the package is backed by thorough and beginner-friendly documentation, which spans tutorials, examples, how-to manuals and a user guide. In the following section (Section 2) we introduce our software and describe its architecture. Next, we present a number of possible use cases and benefits of having various FAT algorithms under a shared roof (Section 3). We conclude the paper with an overview of the impact of our package to date and a discussion of the envisaged long-term benefits of `FAT Forensics` in view of our contributions (Section 4). While this paper focuses on the wide-reaching advantages of our software, a complementary publication [10] offers its high-level overview, implementation details and comparison to related packages.

2. Design and architecture

Systematic evaluation and comparison of AI & ML techniques is an active area of research across many different communities. In well-established research fields, such as supervised learning, we can observe convergence towards commonly accepted (predictive) performance metrics and evaluation software; their implementations often constitute a fundamental part of relevant packages, nonetheless the independence of many such metrics from the underlying predictive algorithms allows for standalone software dedicated to calculating them, e.g., *PyCM* [11]. In contrast, relatively young fields – such as algorithmic fairness, accountability (robustness, safety, security & privacy) and transparency (interpretability & explainability) – usually lack this type of evaluation strategies and software solutions, making them a welcome addition that has the potential to streamline research.

To address these challenges, we developed an open source Python framework for evaluating, comparing and deploying FAT algorithms. We chose Python because of its prevalence across different AI & ML research communities and overall simplicity. We opted for a minimal (required) dependency on NumPy and SciPy to facilitate easy deployment in a variety of settings. An optional dependency on Matplotlib, scikit-learn, Pillow and scikit-image enables access to basic visualisations, ML algorithms and image manipulations (needed by explainability functions). The toolbox is hosted on GitHub to facilitate community contributions, and released under the 3-Clause BSD licence to open it up for commercial applications. To encourage long-term sustainability it has been developed in accordance with the best software engineering practices such as: unit and integration testing; high code coverage; continuous integration; function- and module-level technical API documentation; task-focused code examples; narrative-driven tutorials; problem-oriented how-to guides; and a comprehensive user guide. The toolbox implements a number of popular FAT algorithms – with many more to come – under a coherent API, reusing many functional components across FAT tools and making them readily accessible to the community. The initial development is focused on tabular data and well-established predictive models (scikit-learn [12]), which will be followed by techniques capable of handling sensory data (images & text) and neural networks (TensorFlow [13] & PyTorch [14]). Additionally, we envisage that relevant software packages that are already prominent in the FAT community and that adhere to best software engineering practice can be "wrapped" by our toolbox under a common API to make them easily accessible and avoid re-implementing them.

Algorithms included in `FAT Forensics` are designed and engineered to support two main application areas. The *research mode*, characterised by "data in – visualisations out", envisages the toolbox being loaded into an interactive Python session (e.g., a Jupyter Notebook) to support exploratory analysis, prototyping, development, evaluation and testing. This mode is intended for researchers who could use it to propose new fairness metrics, compare them with existing solutions or inspect a new predictive system or data set (without the burden of setting up a dedicated software engineering workflow). Contributing these implementations of cutting-edge techniques to `FAT Forensics`

will in turn make the package attractive for monitoring and auditing of data-driven systems – the second intended application domain. More specifically, the *deployment mode*, characterised by "data in – data out", offers to incorporate the package into a data processing pipeline to provide a (numerical) analytics, hence support any kind of automated reporting, dashboarding or certification (thus partially alleviating the issues with manual, error-prone and subjective characterisation of AI & ML components). This mode is intended for ML practitioners who (by accessing the low-level API) may use it to monitor or evaluate a data-driven system; where continuous integration is used in software engineering to ensure high quality of the code, our toolbox could be employed to evaluate FAT of any component of an ML pipeline during its development and deployment.

A considerable portion of FAT software is developed to support research outputs, which often results in superfluous dependencies, data sets, predictive models and (interactive) visualisations being distributed with the code base that itself is accessible via a non-standard API. To mitigate these issues, `FAT Forensics` decouples the core FAT functionality from its possible presentation to the user and experiment-specific resources. This abstraction of the software infrastructure is achieved by making minimal assumptions about the operational setting of these algorithms, therefore facilitating a common interface layer for key FAT functionality, focusing only on the interactions between data, models, predictions and users [10]. In this purview a predictive model is assumed to be a plain Python object with `fit`, `predict` and, optionally, `predict_proba` methods, which offers compatibility with scikit-learn [12] – the most popular Python ML toolbox – without explicitly depending on it, in addition to supporting any other predictor that can be represented in this way, e.g., TensorFlow, PyTorch or even one hosted on the Internet and accessible via a web API. Similarly, a data set is assumed to be a two-dimensional NumPy array: either a classic or a structured array, with the latter bringing support for (string-based) categorical attributes. Since visualisations are a vital part of our first application mode (research), the software provides basic plotting functionality that is only enabled when the *optional* Matplotlib dependency is installed. In addition to relaxed input requirements, all of the techniques incorporated into the package are split into interoperable algorithmic building blocks that can be easily reused, even across FAT borders, to create new functionality – the versatility of this atomic-level decomposition is demonstrated in the following section. More details about the technical aspects of the software can be found in the `FAT Forensics` technical paper [10].

3. Use cases

We present three distinct use cases to demonstrate how the software can be applied to analyse FAT aspects of real data, illustrating the diverse range of functionality enabled by its universal infrastructure. To this end, we employ the UCI Census Income (Adult) data set [15], which is popular in algorithmic fairness and transparency research. The data analysis that follows is representative of the *research mode* and is inspired by the tutorials included in the `FAT Forensics` documentation[1]; it can be reproduced with a dedicated Jupyter Notebook.[2] To demonstrate the *deployment mode*, we provide a dashboard based on Plotly Dash, which facilitates interactive analysis of the same data set using `FAT Forensics` as the back end.[3]

---

[1] https://fat-forensics.org/tutorials/index.html

[2] https://github.com/fat-forensics/resources/blob/master/fat_forensics_overview/FAT_Forensics.ipynb

[3] https://fatf.herokuapp.com/ (Source code available at: https://github.com/fat-forensics/fatf-dashboard/.)

Feature grouping. One of the core building blocks of `FAT Forensics` is a collection of functions to partition data based on (sets of) unique values for categorical features and threshold-based binning for numerical attributes. This algorithmic concept – in conjunction with any standard (predictive) performance metric derived from predicted and true labels – facilitates a number of FAT workflows. A variety of different group-based (pairwise) **fairness** criteria, not limited to the ones implemented in the package, can be computed in this way by conditioning on protected features (attributes that may be used for discriminatory treatment, e.g., gender), allowing us to investigate disparate impact of a predictive model based on group unaware, equal opportunity, equal accuracy or demographic parity metrics, among many others [16]. Since some of them are mutually incompatible [17], comparing them side-by-side can be beneficial. For example, the *Asian-Pac-Islander* (*Asi-Pac-Isl*) and *Other* groups are subject to fairness disparity when *equal accuracy* and *demographic parity* are considered; *Other* and *White* sub-populations are also treated unfairly according to *demographic parity*; whereas *equal opportunity* does not exhibit any signs of disparate impact as shown in Fig. 1.

The grouping functionality can also help to assess **accountability** of data and models in a similar fashion. For example, sample-size disparity across sub-populations in a given data set may cause a *systematic bias* in predictive performance of the resulting model over such (protected) groups since it is likely to under-perform for under-represented individuals. This effect can be observed when splitting Adult based on the *race* feature while measuring *accuracy* and *true negative rate*. As expected, the former (Fig. 2(a)) provides the same result as group-based fairness analysis under *equal accuracy* (Fig. 1(a)); the latter (Fig. 2(b)), on the other hand, reveals that four sub-population pairs exhibit significant performance differences, with *Other* affected the most by diverging from all the other groups except *Amer-Indian-Eskimo* (*Ame-Ind-Esk*). Partitioning is also useful for **transparency** analysis; for example, summary statistics such as the distribution of labels across sub-populations based on (protected) features can be generated for a data set prior to modelling to uncover any class imbalance. Studying the *race* attribute in this context – Fig. 3 – reveals that while the classes are skewed across all the splits, the strongest disproportion affects the *Other, Ame-Ind-Esk* and *Black* sub-populations.

Data density. Density estimate for a region in which a data point of interest is located (based on the distribution of training data) can be treated as a proxy for the confidence of its prediction [18], thus helping to judge its **accountability** and robustness as dense regions should offer more accurate modelling. To this end, `FAT Forensics` implements a bespoke neighbour-based density estimator – its scores are between 0 and 1, where high values are assigned to instances from sparse regions since their nth neighbour (a user-defined parameter) is relatively distant. As an illustration we estimate the density of Adult based on its first 1000 instances and select four data points – two from a dense and two from a sparse region – to assess robustness of their predictions. The former two receive density scores of 0 and are correctly predicted as *≤50K*; the latter two are assigned density scores of 1 with one predicted correctly and the other misclassified as *≤50K*. Upon closer inspection this data point has a relatively high value (99.99th percentile) of the *fnlwgt* feature (1,226,583), which is a clue to its high density score and incorrect prediction (see the aforementioned Jupyter Notebook for more details).

In addition to engendering trust in predictions, a density estimate can help to assess the quality of exemplar explanations and compute realistic counterfactuals [19], which can be used as a **transparency** tool and individual **fairness** mechanism (by conditioning on protected attributes). Sourcing counterfactuals from sparse regions may yield explanations based on instances that are unlikely to occur in the real life, e.g., prescribing a person to become 200 years old. Explaining the aforementioned misclassified data point taken from a sparse region provides explanations such as: (i) raising *capital-gain* from 0 to 25,000 predicts *>50K* (sparse region with 1 density score); and (ii) increasing

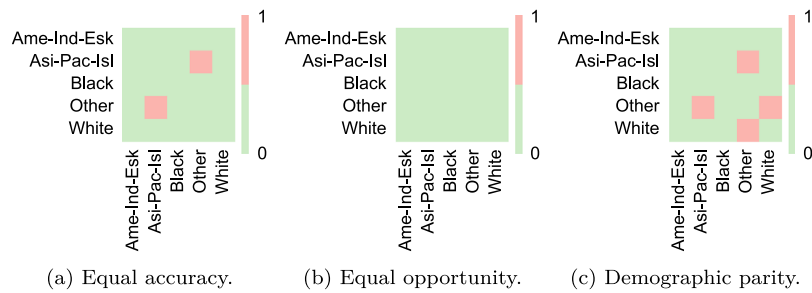(a) Equal accuracy.   (b) Equal opportunity.   (c) Demographic parity.

Fig. 1. Pairwise group-based **fairness** for the *race* feature of the Adult data set. Red (1) indicates disparate impact for a given pair of sub-populations and green (0) conveys that they are treated comparably.
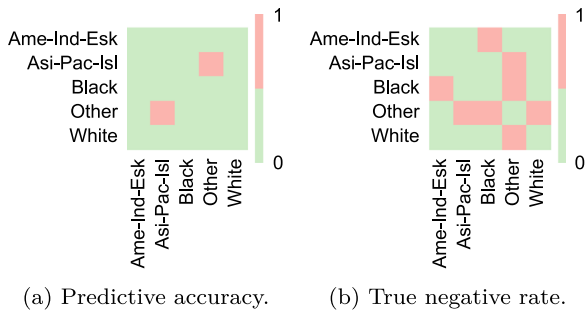

(a) Predictive accuracy.   (b) True negative rate.

Fig. 2. Pairwise group-based **performance** disparity for the *race* feature of Adult. Red (1) shows disparate performance and green (0) comparable treatment.

capital-loss from 0 to 4000 and decreasing *fnlwgt* from 1,226,583 to 430,985 predicts >*50K* (dense region with 0.02 density score). While (i) prescribes a sensible action, preserving the unusually high value of *fnlwgt* makes it unlikely; (ii), on the other hand, decreases the value of this attribute – therefore placing the counterfactual in a dense region – and shows that even with 4000 of *capital-loss* being classified as >*50K* is possible, casting even more suspicion on the unusually high original value of the former feature. Finally, no counterfactuals conditioned on protected attributes could be found for this instance, showing us that its prediction is fair (again, see the aforementioned Jupyter Notebook for more details).

Surrogate modularity. Surrogate explainers are a popular interpretability technique that fits a transparent model in a selected neighbourhood to approximate and explain the predictive behaviour of the underlying black box in said region [20–22]. Given their high modularity, FAT Forensics implements their core building blocks via the bLIMEy meta-algorithm[4] – consisting of interpretable representation composition, data sampling and explanation generation steps – which allows the user to easily construct a bespoke surrogate that is suitable for the problem at hand, thus considerably improving the quality and faithfulness of the resulting explanations [22,23]. For example, an interpretable representation of tabular data can be built with quartile-based discretisation or a feature space partition extracted from a decision tree (the latter is more faithful [24]); data can be augmented with Gaussian or mixup [25] sampling (the latter offers a diverse and local sample [22]); and an explanation can be generated with a linear model or a decision tree (the former is limited to feature influence, whereas the latter provides a diverse range of insights such as rules and counterfactuals [23,24,26]). Such a surrogate explainer can either be local – by sampling data in the neighbourhood of a selected instance – or global – when the sample covers the entire data space. Specifically, consider the two local surrogates shown in Fig. 4, where a tree-based explainer [24,26] is better able to approximate the decision boundary close to the selected instance.

4. Impact overview

While software is one of the primary drivers of progress in AI & ML research, its quality is often found lacking. FAT Forensics offers a possible solution in the space of algorithmic fairness, accountability and transparency by facilitating the development, evaluation, comparison and deployment of FAT tools. Sharing a common functional base between implementations of FAT algorithms is one of many advantages of such a comprehensive package. Its versatility as well as support for the *research* and *deployment* operation modes make it appealing to members of academia and industry, especially as it supports investigating FAT aspects of an entire predictive pipeline: data, models and predictions. This in turn ought to encourage the community to adopt the software and contribute their novel algorithms and bug fixes here (instead of releasing them as standalone code), thus exposing them to the wider audience in a robust and sustainable environment, enhancing reproducibility of research in this space and orienting the package towards real-world use cases. By developing FAT tools on a modular level from the ground up FAT Forensics ensures their robustness and accountability in addition to being shielded from any errors that otherwise could have been introduced downstream. For example, LIME [21] – which is "wrapped" by Microsoft's Interpret [27] and Oracle's Skater [28] libraries – has known issues with the locality and coherence of its explanations [22,29], which inadvertently affect both these packages. We therefore hope and expect that all the software engineering best practice followed during the initial development of FAT Forensics (and maintained carrying forwards) have helped us to create a sustainable package that is easy to extend and contribute to, serving the community for a long time to come.

Additionally, the modular design of the package facilitates conducting cutting-edge research. To date, the implementation of surrogate explainers available in FAT Forensics allowed us to carefully study their capabilities and failure modes, leading to new findings, theories and transparency tools. bLIMEy – the surrogate meta-algorithm – is a case in point; its inception was inspired by identifying independent algorithmic modules, whose further investigation showed the importance of local sampling for tabular data and effectiveness of decision trees as surrogate models [22,23]. One particular realisation of this explainer – LIMEtree – is based on multi-output regression trees and improves upon many shortcomings of surrogates by offering faithful, consistent, customisable and multi-class explanations of different types, including counterfactuals [26]. Diverse implementations of surrogate building blocks also helped us to analyse the role and parameterisation of interpretable representations and improve their robustness – they translate the low-level data representation used by predictive models into human-comprehensible concepts underlying explanations and are the backbone of surrogates [24]. FAT Forensics has also been the foundation of a hands-on conference tutorial on ML explainability [30] as well as numerous lectures, summer school sessions, educational events and learning resources.[5]

---

[4] https://fat-forensics.org/how_to/transparency/tabular-surrogates.html

[5] https://events.fat-forensics.org/

Fig. 3. Income distribution for each unique value of the *race* feature in the Adult data set.
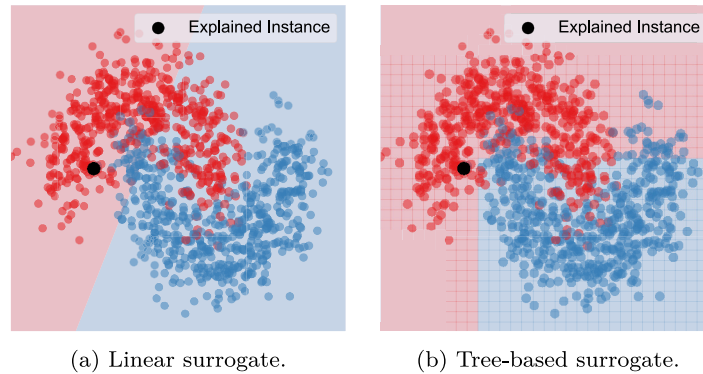

Fig. 4. Examples of (a) linear (LIME-like) and (b) tree-based local surrogates without an interpretable representation (enabling direct visualisation) built with the bLIMEy framework for the Two Moons data set.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Hutson, Artificial intelligence faces reproducibility crisis, Science (ISSN: 0036-8075) 359 (6377) (2018) 725–726, http://dx.doi.org/10.1126/science.359.6377.725.

[2] S. Sonnenburg, M.L. Braun, C.S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.-R. Müller, F. Pereira, C.E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent, J. Weston, R. Williamson, The need for open source software in machine learning, J. Mach. Learn. Res. 8 (Oct) (2007) 2443–2466.

[3] T. Gebru, J. Morgenstern, B. Vecchione, J.W. Vaughan, H. Wallach, H. Daumé III, K. Crawford, Datasheets for datasets, in: 5th Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2018) at the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 2018, arXiv:1803.09010.

[4] S. Holland, A. Hosny, S. Newman, J. Joseph, K. Chmielinski, The dataset nutrition label: A framework to drive higher data quality standards, Data Protection and Privacy, Volume 12: Data Protection and Democracy 12 (2020) 1.

[5] D. Reisman, J. Schultz, K. Crawford, M. Whittaker, Algorithmic impact assessments: A practical framework for public agency accountability, 2018, AI now Institute.

[6] M. Arnold, R.K. Bellamy, M. Hind, S. Houde, S. Mehta, A. Mojsilovic, R. Nair, K.N. Ramamurthy, A. Olteanu, D. Piorkowski, D. Reimer, J. Richards, J. Tsay, K.R. Varshney, FactSheets: Increasing trust in AI services through supplier's declarations of conformity, IBM J. Res. Dev. (ISSN: 0018-8646) 63 (4/5) (2019) 6:1–6:13, http://dx.doi.org/10.1147/JRD.2019.2942288.

[7] K. Yang, J. Stoyanovich, A. Asudeh, B. Howe, H. Jagadish, G. Miklau, A nutritional label for rankings, in: Proceedings of the 2018 International Conference on Management of Data, ACM, 2018, pp. 1773–1776.

[8] K. Sokol, P. Flach, Explainability fact sheets: A framework for systematic assessment of explainable approaches, in: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 2020, pp. 56–67.

[9] P.G. Kelley, J. Bresee, L.F. Cranor, R.W. Reeder, A nutrition label for privacy, in: Proceedings of the 5th Symposium on Usable Privacy and Security, ACM, 2009, p. 4.

[10] K. Sokol, A. Hepburn, R. Poyiadzi, M. Clifford, R. Santos-Rodriguez, P. Flach, FAT Forensics: A Python toolbox for implementing and deploying fairness, accountability and transparency algorithms in predictive systems, J. Open Source Softw. 5 (49) (2020) 1904, http://dx.doi.org/10.21105/joss.01904.

[11] S. Haghighi, M. Jasemi, S. Hessabi, A. Zolanvari, PyCM: Multiclass confusion matrix library in Python, J. Open Source Softw. 3 (25) (2018) 729, http://dx.doi.org/10.21105/joss.00729.

[12] 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, E. Duchesnay, scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[13] 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, X. Zheng, TensorFlow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, 2016, pp. 265–283.

[14] 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, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, pp. 8026–8037.

[15] R. Kohavi, B. Becker, Census income data set, 1996, URL https://archive.ics.uci.edu/ml/datasets/Census+Income.

[16] M. Hardt, E. Price, N. Srebro, Equality of opportunity in supervised learning, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., USA, ISBN: 978-1-5108-3881-9, 2016, pp. 3323–3331.

[17] T. Miconi, The impossibility of "fairness": A generalized impossibility result for decisions, 2017, arXiv:1707.01195.

[18] M. Perello-Nieto, E.S. Telmo De Menezes Filho, M. Kull, P. Flach, Background check: A general technique to build more reliable and versatile classifiers, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, IEEE, 2016, pp. 1143–1148.

[19] R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, P. Flach, FACE: Feasible and actionable counterfactual explanations, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 344–350.

[20] M. Craven, J.W. Shavlik, Extracting tree-structured representations of trained networks, in: Advances in Neural Information Processing Systems, 1996, pp. 24–30.

[21] M.T. Ribeiro, S. Singh, C. Guestrin, "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, San Francisco, CA, USA, August 13-17, 2016, 2016, pp. 1135–1144.

[22] K. Sokol, A. Hepburn, R. Santos-Rodriguez, P. Flach, bLIMEy: Surrogate prediction explanations beyond LIME, in: 2019 Workshop on Human-Centric Machine Learning (HCML 2019) at the 33rd Conference on Neural Information Processing Systems, NeurIPS 2019, Vancouver, Canada, 2019, arXiv:1910.13016.

[23] K. Sokol, Towards intelligible and robust surrogate explainers: A decision tree perspective (Ph.D. thesis), University of Bristol, 2021.

[24] K. Sokol, P. Flach, Towards faithful and meaningful interpretable representations, 2020, arXiv:2008.07007.

[25] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=r1Ddp1-Rb.

[26] K. Sokol, P. Flach, LIMEtree: Interactively customisable explanations based on local surrogate multi-output regression trees, 2020, arXiv:2005.01427.

[27] H. Nori, S. Jenkins, P. Koch, R. Caruana, InterpretML: A unified framework for machine learning interpretability, 2019, arXiv:1909.09223.

[28] A. Kramer, P. Choudhary, silversurfer84, B.V. Dyke, A. Thai, N. Pasumarthy, G. Lemaitre, D. Thompson, B. Cook, datascienceinc/Skater: 1.1.2, 2018, http://dx.doi.org/10.5281/zenodo.1423046, Zenodo.

[29] T. Laugel, X. Renard, M.-J. Lesot, C. Marsala, M. Detyniecki, Defining locality for surrogates in post-hoc interpretablity, ICML 2018, Stockholm, Sweden, 2018, arXiv:1806.07498,

[30] K. Sokol, A. Hepburn, R. Santos-Rodriguez, P. Flach, What and how of machine learning transparency: Building bespoke explainability tools with interoperable algorithmic components, in: Hands-on tutorial at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD, Ghent, Belgium, 2020, URL https://events.fat-forensics.org/2020_ecml-pkdd.